Tracts in Mathematics 14

Steffen Börm

Efficient Numerical Methods for Non-local Operators

$\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis

European Mathematical Society

EMS Tracts in Mathematics 14

EMS Tracts in Mathematics

Editorial Board:
Carlos E. Kenig (The University of Chicago, USA)
Andrew Ranicki (The University of Edinburgh, Great Britain)
Michael Röckner (Universität Bielefeld, Germany, and Purdue University, USA)
Vladimir Turaev (Indiana University, Bloomington, USA)
Alexander Varchenko (The University of North Carolina at Chapel Hill, USA)

This series includes advanced texts and monographs covering all fields in pure and applied mathematics. *Tracts* will give a reliable introduction and reference to special fields of current research. The books in the series will in most cases be authored monographs, although edited volumes may be published if appropriate. They are addressed to graduate students seeking access to research topics as well as to the experts in the field working at the frontier of research.

1 Panagiota Daskalopoulos and Carlos E. Kenig, *Degenerate Diffusions*
2 Karl H. Hofmann and Sidney A. Morris, *The Lie Theory of Connected Pro-Lie Groups*
3 Ralf Meyer, *Local and Analytic Cyclic Homology*
4 Gohar Harutyunyan and B.-Wolfgang Schulze, *Elliptic Mixed, Transmission and Singular Crack Problems*
5 Gennadiy Feldman, *Functional Equations and Characterization Problems on Locally Compact Abelian Groups*
6 Erich Novak and Henryk Woźniakowski, *Tractability of Multivariate Problems. Volume I: Linear Information*
7 Hans Triebel, *Function Spaces and Wavelets on Domains*
8 Sergio Albeverio et al., *The Statistical Mechanics of Quantum Lattice Systems*
9 Gebhard Böckle and Richard Pink, *Cohomological Theory of Crystals over Function Fields*
10 Vladimir Turaev, *Homotopy Quantum Field Theory*
11 Hans Triebel, *Bases in Function Spaces, Sampling, Discrepancy, Numerical Integration*
12 Erich Novak and Henryk Woźniakowski, *Tractability of Multivariate Problems. Volume II: Standard Information for Functionals*
13 Laurent Bessières et al., *Geometrisation of 3-Manifolds*

Steffen Börm

Efficient Numerical Methods for Non-local Operators

$\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis

European Mathematical Society

Author:

Prof. Dr. Steffen Börm
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
24118 Kiel
Germany

E-mail: sb@informatik.uni-kiel.de

2010 Mathematical Subject Classification: 65-02; 65F05, 65F30, 65N22, 65N38, 65R20

Key words: Hierarchical matrix, data-sparse approximation, boundary element method, preconditioner

ISBN 978-3-03719-091-3

The Swiss National Library lists this publication in The Swiss Book, the Swiss national bibliography, and the detailed bibliographic data are available on the Internet at http://www.helveticat.ch.

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. For any kind of use permission of the copyright owner must be obtained.

© 2010 European Mathematical Society

Contact address:

European Mathematical Society Publishing House
Seminar for Applied Mathematics
ETH-Zentrum FLI C4
CH-8092 Zürich
Switzerland

Phone: +41 (0)44 632 34 36
Email: info@ems-ph.org
Homepage: www.ems-ph.org

Typeset using the author's TEX files: I. Zimmermann, Freiburg
Printing and binding: Druckhaus Thomas Müntzer GmbH, Bad Langensalza, Germany
∞ Printed on acid free paper
9 8 7 6 5 4 3 2 1

Foreword

Non-local operators appear naturally in the field of scientific computing: non-local forces govern the movement of objects in gravitational or electromagnetic fields, non-local density functions describe jump processes used, e.g., to investigate stock prices, and non-local kernel functions play an important role when studying population dynamics.

Applying standard discretization schemes to non-local operators yields matrices that consist mostly of non-zero entries ("dense matrices") and therefore cannot be treated efficiently by standard sparse matrix techniques. They can, however, be approximated by *data-sparse* representations that significantly reduce the storage requirements.

Hierarchical matrices ($\mathcal{H}$-matrices) [62] are one of these data-sparse representations: $\mathcal{H}$-matrices not only approximate matrices arising in many important applications very well, they also offer a set of matrix arithmetic operations like evaluation, multiplication, factorization and inversion that can be used to construct efficient preconditioners or solve matrix equations. $\mathcal{H}^2$-matrices [70], [64] introduce an additional hierarchical structure to reduce the storage requirements and computational complexity of $\mathcal{H}$-matrices.

In this book, I focus on presenting an overview of theoretical results and practical algorithms for working with $\mathcal{H}^2$-matrices. I assume that the reader is familiar with basic techniques of numerical linear algebra, e.g., norm estimates, orthogonal transformations and factorizations. The error analysis of integral operators, particularly Section 4.7, requires some results from polynomial approximation theory, while the error analysis of differential operators, particularly Section 9.2, is aimed at readers familiar with standard finite element techniques and makes use of a number of fundamental properties of Sobolev spaces.

Different audiences will probably read the book in different ways. I would like to offer the following suggestions: Chapters 1–3 provide the basic concepts and definitions used in this book and any reader should at least be familiar with the terms $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix, cluster tree, block cluster tree, admissible and inadmissible blocks and cluster bases. After this introduction, different courses are possible:

- If you are a student of numerical mathematics, you should read Sections 4.1–4.4 on integral operators, Sections 5.1–5.5 on orthogonalization and truncation, Sections 6.1–6.4 on matrix compression, and maybe Sections 7.1, 7.2, 7.6 and 7.7 to get acquainted with the concepts of matrix arithmetic operations.

- If you are interested in using $\mathcal{H}^2$-matrices to treat integral equations, you should read Chapter 4 on basic approximation techniques and Chapters 5 and 6 in order to understand how the storage requirements can be reduced as far as possible. Remarks on practical applications can be found in Sections 10.1–10.4.

- If you are interested in applying $\mathcal{H}^2$-matrices to elliptic partial differential equations, you should consider Sections 5.1–5.5 on truncation, Chapter 6 on compression, Chapter 8 on adaptive matrix arithmetic operations and Sections 10.5 and 10.3. Convergence estimates can be found in Chapter 9.

If you would like to try the algorithms presented in this book, you can get the HLIB software package that I have used to provide the numerical experiments described in Chapters 4–10. Information on this package is available at http://www.hlib.org and it is provided free of charge for research purposes.

This book would not exist without the help and support of Wolfgang Hackbusch, whom I wish to thank for many fruitful discussions and insights and for the chance to work at the Max Planck Institute for Mathematics in the Sciences in Leipzig.

I am also indebted to my colleagues Stefan A. Sauter, Lars Grasedyck and J. Markus Melenk, who have helped me find answers to many questions arising during the course of this work.

Last, but not least, I thank Maike Löhndorf, Kai Helms and Jelena Djokić for their help with proofreading the (already quite extensive) drafts of this book and Irene Zimmermann for preparing the final version for publication.

Kiel, November 2010 Steffen Börm

Contents

1 Introduction 1
- 1.1 Origins of $\mathcal{H}^2$-matrix methods . . . 1
- 1.2 Which kinds of matrices can be compressed? . . . 3
- 1.3 Which kinds of operations can be performed efficiently? . . . 4
- 1.4 Which problems can be solved efficiently? . . . 6
- 1.5 Organization of the book . . . 6

2 Model problem 9
- 2.1 One-dimensional integral operator . . . 9
- 2.2 Low-rank approximation . . . 10
- 2.3 Error estimate . . . 11
- 2.4 Local approximation . . . 15
- 2.5 Cluster tree and block cluster tree . . . 16
- 2.6 Hierarchical matrix . . . 20
- 2.7 Matrix approximation error . . . 22
- 2.8 $\mathcal{H}^2$-matrix . . . 23
- 2.9 Numerical experiment . . . 26

3 Hierarchical matrices 28
- 3.1 Cluster tree . . . 29
- 3.2 Block cluster tree . . . 34
- 3.3 Construction of cluster trees and block cluster trees . . . 37
- 3.4 Hierarchical matrices . . . 47
- 3.5 Cluster bases . . . 53
- 3.6 $\mathcal{H}^2$-matrices . . . 56
- 3.7 Matrix-vector multiplication . . . 59
- 3.8 Complexity estimates for bounded rank distributions . . . 63
- 3.9 Technical lemmas . . . 70

4 Application to integral operators 74
- 4.1 Integral operators . . . 76
- 4.2 Low-rank approximation . . . 76
- 4.3 Approximation by Taylor expansion . . . 79
- 4.4 Approximation by interpolation . . . 87
- 4.5 Approximation of derivatives . . . 103
- 4.6 Matrix approximation . . . 114
- 4.7 Variable-order approximation . . . 125
- 4.8 Technical lemmas . . . 149
- 4.9 Numerical experiments . . . 155

5 Orthogonal cluster bases and matrix projections 163
5.1 Orthogonal cluster bases . . . 164
5.2 Projections into $\mathcal{H}^2$-matrix spaces . . . 166
5.3 Cluster operators . . . 175
5.4 Orthogonalization . . . 180
5.5 Truncation . . . 185
5.6 Computation of the Frobenius norm of the projection error . . . 200
5.7 Numerical experiments . . . 202

6 Compression 211
6.1 Semi-uniform matrices . . . 212
6.2 Total cluster bases . . . 218
6.3 Approximation by semi-uniform matrices . . . 222
6.4 General compression algorithm . . . 227
6.5 Compression of hierarchical matrices . . . 234
6.6 Recompression of $\mathcal{H}^2$-matrices . . . 239
6.7 Unification and hierarchical compression . . . 248
6.8 Refined error control and variable-rank approximation . . . 259
6.9 Numerical experiments . . . 271

7 A priori matrix arithmetic 280
7.1 Matrix forward transformation . . . 281
7.2 Matrix backward transformation . . . 287
7.3 Matrix addition . . . 292
7.4 Projected matrix-matrix addition . . . 293
7.5 Exact matrix-matrix addition . . . 297
7.6 Matrix multiplication . . . 301
7.7 Projected matrix-matrix multiplication . . . 302
7.8 Exact matrix-matrix multiplication . . . 319
7.9 Numerical experiments . . . 328

8 A posteriori matrix arithmetic 332
8.1 Semi-uniform matrices . . . 334
8.2 Intermediate representation . . . 336
8.3 Coarsening . . . 347
8.4 Construction of adaptive cluster bases . . . 359
8.5 Numerical experiments . . . 360

9 Application to elliptic partial differential operators 363
9.1 Model problem . . . 364
9.2 Approximation of the solution operator . . . 366
9.3 Approximation of matrix blocks . . . 376
9.4 Compression of the discrete solution operator . . . 384

10 Applications 387
10.1 Indirect boundary integral equation 388
10.2 Direct boundary integral equation 395
10.3 Preconditioners for integral equations 401
10.4 Application to realistic geometries 411
10.5 Solution operators of elliptic partial differential equations 413

Bibliography 421

Algorithm index 427

Subject index 429

Chapter 1
Introduction

The goal of this book is to describe a method for handling certain large dense matrices efficiently. The fundamental idea of the $\mathcal{H}^2$-matrix approach is to reduce the storage requirements by using an alternative multilevel representation of a dense matrix instead of the standard representation by a two-dimensional array.

1.1 Origins of $\mathcal{H}^2$-matrix methods

The need for efficient algorithms for handling dense matrices arises from several fields of applied mathematics: in the simulation of many-particle systems governed by the laws of gravitation or electrostatics, a fast method for computing the forces acting on the individual particles is required, and these forces can be expressed by large dense matrices.

Certain homogeneous partial differential equations can be reformulated as boundary integral equations, and compared to the standard approach, these formulations have the advantage that they reduce the spatial dimension, improve the convergence and can even simplify the handling of complicated geometries. The discretization of the boundary integral equations leads again to large dense matrices.

A number of models used in the fields of population dynamics or machine learning also lead to integral equations that, after discretization, yield large dense matrices.

Several approaches for handling these kinds of problems have been investigated: for special integral operators and special geometries, the corresponding dense matrices are of Toeplitz or circulant form, and the fast Fourier transform [37] can be used to compute the matrix-vector multiplication in $\mathcal{O}(n \log n)$ operations, where n is the matrix dimension. The restriction to special geometries limits the range of applications that can be treated by this approach.

The panel clustering method [71], [72], [91], [45] follows a different approach to handle arbitrary geometries: the matrix is not represented exactly, but approximated by a data-sparse matrix, i.e., by a matrix that is still dense, but can be represented in a compact form. This approximation is derived by splitting the domain of integration into a partition of subdomains and replacing the kernel function by local separable approximations. The resulting algorithms have a complexity of $\mathcal{O}(n m^\alpha \log^\beta n)$ for problem-dependent small exponents $\alpha, \beta > 0$ and a parameter m controlling the accuracy of the approximation.

The well-known multipole method [58], [60] is closely related and takes advantage of the special properties of certain kernel functions to improve the efficiency. It has

originally been introduced for the simulation of many-particle systems, but can also be applied to integral equations [88], [86], [85], [57]. "Multipole methods without multipoles" [2], [82], [108] replace the original multipole approximation by more general or computationally more efficient expansions while keeping the basic structure of the corresponding algorithms. Of particular interest is a fully adaptive approach [46] that constructs approximations based on singular value decompositions of polynomial interpolants and thus can automatically find efficient approximations for relatively general kernel functions.

It should be noted that the concept of separable approximations used in both the panel clustering and the multipole method is already present in the Ewald summation technique [44] introduced far earlier to evaluate Newton potentials in crystallographical research efficiently.

Wavelet techniques use a hierarchy of nested subspaces combined with a Galerkin method in order to approximate integral operators [94], [9], [41], [39], [73], [105], [102]. This approach reaches very good compression rates, but the construction of suitable subspaces on complicated geometries is significantly more complicated than for the techniques mentioned before.

Hierarchical matrices [62], [68], [67], [49], [52], [63] and the closely related mosaic skeleton matrices [103] are the algebraic counterparts of panel-clustering and multipole methods: a partition of the matrix takes the place of the partition of the domains of integration, and low-rank submatrices take the place of local separable expansions. Due to their algebraic structure, hierarchical matrices can be applied not only to integral equations and particle systems, but also to more general problems, e.g., partial differential equations [6], [56], [55], [54], [76], [77], [78] or matrix equations from control theory [53], [51]. Efficient approximations of densely populated matrices related to integral equations can be constructed by interpolation [16] or more efficient cross approximation schemes [5], [4], [7], [17].

$\mathcal{H}^2$-matrices [70], [64] combine the advantages of hierarchical matrices, i.e., their flexibility and wide range of applications, with those of wavelet and fast multipole techniques, i.e., the high compression rates achieved by using a multilevel basis. The construction of this *cluster basis* for different applications is one of the key challenges in the area of $\mathcal{H}^2$-matrices: it has to be efficient, i.e., it has to consist of a small number of vectors, but it also has to be accurate, i.e., it has to be able to approximate the original matrix up to a given tolerance. In some situations, an $\mathcal{H}^2$-matrix approximation can reach the *optimal* order $\mathcal{O}(n)$ of complexity while keeping the approximation error consistent with the requirements of the underlying discretization scheme [91], [23].

Obviously, we cannot hope to be able to approximate all dense matrices in this way: if a matrix contains only independent random values, the standard representation is already optimal and no compression scheme will be able to reduce the storage requirements. Therefore we have first to address the question "Which kinds of matrices can be compressed by $\mathcal{H}^2$-matrix methods?"

It is not sufficient to know that a matrix can be compressed, we also have to be able to find the compressed representation and to use it in applications, e.g., to perform

matrix-vector multiplications or solve systems of linear equations. Of course, we do not want to convert the $\mathcal{H}^2$-matrices back to the less efficient standard format, therefore we have to consider the question "Which kinds of operations can be performed efficiently with compressed matrices?"

Once these two theoretical questions have been answered, we can consider practical applications of the $\mathcal{H}^2$-matrix technique, i.e., try to answer the question "Which problems can be solved efficiently by $\mathcal{H}^2$-matrices?"

1.2 Which kinds of matrices can be compressed?

There are two answers to this question: in the introductory Chapter 2, a very simple one-dimensional integral equation is discussed, and it is demonstrated that its discrete counterpart can be handled by $\mathcal{H}^2$-matrices: if we replace the kernel function by a *separable approximation*, the resulting matrix will be an $\mathcal{H}^2$-matrix and can be treated efficiently. Chapter 4 generalizes this result to the more general setting of integral operators with asymptotically smooth kernel functions.

In Chapter 6, on the other hand, a relatively general characterization of $\mathcal{H}^2$-matrices is introduced. Using this characterization, we can determine whether arbitrary matrices can be approximated by $\mathcal{H}^2$-matrices. In this framework, the approximation of integral operators can be treated as a special case, but it is also possible to investigate more general applications, e.g., the approximation of solution operators of ordinary [59], [96] and elliptic partial differential equations by $\mathcal{H}^2$-matrices [6], [15]. The latter very important case is treated in Chapter 9.

Separable approximations

Constructing an $\mathcal{H}^2$-matrix based on separable approximations has the advantage that the problem is split into two relatively independent parts: the first task is to approximate the kernel function in suitable subdomains by separable kernel functions. This task can be handled by Taylor expansions [72], [100] or interpolation [45], [65], [23] if the kernel function is locally analytic. Both of these approaches are discussed in Chapter 4.

For special kernel functions, special approximations like the multipole expansion [58], [60] or its counterparts for the Helmholtz kernel [1], [3] can be used. The special techniques required by these methods are not covered here.

Once a good separable approximation of the kernel function has been found, we face the second task: the construction of an $\mathcal{H}^2$-matrix. This is accomplished by splitting the integral operator into a sum of local operators on suitably defined subsets and then replacing the original kernel function by its separable approximations. Discretizing the resulting perturbed integral operator by a standard scheme (e.g., Galerkin methods, collocation or Nystrøm techniques) then yields an $\mathcal{H}^2$-matrix approximation of the original matrix.

The challenge in this task is to ensure that the number of local operators is as small as possible: using one local operator for each matrix entry will not lead to a good compression ratio, therefore we are looking for methods that ensure that only a small number of local operators are required.

The standard approach in this context is to use *cluster trees*, i.e., to split the domains defining the integral operator into a hierarchy of subdomains and use an efficient recursive scheme to find an almost optimal decomposition of the original integral operator into local operators which can be approximated.

The efficiency of this technique depends on the properties of the discretization scheme. If the supports of the basis functions are local, i.e., if a neighborhood of the support of a basis function intersects only a small number of supports of other basis functions, it can be proven that the cluster trees will lead to efficient approximations of the matrix [52]. For complicated anisotropic meshes or higher-order basis functions, the situation becomes more complicated and special techniques have to be employed.

General characterization

Basing the construction of an $\mathcal{H}^2$-matrix on the general theory presented in Chapter 6 has the advantage that it allows us to treat arbitrary dense matrices. Whether a matrix can be approximated by an $\mathcal{H}^2$-matrix or not can be decided by investigating the effective ranks of two families of submatrices, the *total cluster bases*. If all of these submatrices can be approximated using low ranks, the matrix itself can be approximated by an $\mathcal{H}^2$-matrix.

Since this characterization relies only on low-rank approximations, but requires no additional properties, it can be applied in relatively general situations, e.g., to prove that solution operators of strongly elliptic partial differential operators with L^∞ coefficients can be approximated by $\mathcal{H}^2$-matrices. Chapter 9 gives the details of this result.

1.3 Which kinds of operations can be performed efficiently?

In this book, we consider three types of operations: first the construction of an approximation of the system matrix, then arithmetic operations like matrix-vector and matrix-matrix multiplications, and finally more complicated operations like matrix factorizations or matrix inversion, which can be constructed based on the elementary arithmetic operations.

Construction

An $\mathcal{H}^2$-matrix can be constructed in several ways: if it is the approximation of an explicitly given integral operator, we can proceed as described above and compute the

$\mathcal{H}^2$-matrix by discretizing a number of local separable approximations. For integral operators with locally smooth kernel functions, the implementation of this method is relatively straightforward and it performs well. This approach is described in Chapter 4.

If we want to approximate a given matrix, we can use the compression algorithms introduced in Chapter 6. These algorithms have the advantage that they construct quasi-optimal approximations, i.e., they will find an approximation that is almost as good as the best possible $\mathcal{H}^2$-matrix approximation. This property is very useful, since it allows us to use $\mathcal{H}^2$-matrices as a "black box" method.

It is even possible to combine both techniques: if we want to handle an integral operator, we can construct an initial approximation by using the general and simple interpolation scheme, and then improve this approximation by applying the appropriate compression algorithm. The experimental results in Chapter 6 indicate that this technique can reduce the storage requirements by large factors.

Arithmetic operations

If we want to solve a system of linear equations with a system matrix in $\mathcal{H}^2$-representation, we at least have to be able to evaluate the product of the matrix with a vector. This and related operations, like the product with the transposed matrix or forward and backward substitution steps for solving triangular systems, can be accomplished in optimal complexity for $\mathcal{H}^2$-matrices: not more than two operations are required per unit of storage.

Using Krylov subspace methods, it is even possible to construct solvers based exclusively on matrix-vector multiplications and a number of simple vector operations. This is the reason why most of today's schemes for solving dense systems of equations (e.g., based on panel clustering [72], [91] or multipole expansions [58], [60]) provide only efficient algorithms for matrix-vector multiplications, but not for more complicated operations.

Hierarchical matrices and $\mathcal{H}^2$-matrices, on the other hand, are purely algebraic objects, and since we have efficient compression algorithms at our disposal, we are able to approximate the results of complex operations like the matrix-matrix multiplication. In Chapters 7 and 8, two techniques for performing this fundamental computation are presented. The first one reaches the optimal order of complexity, but requires a priori knowledge of the structure of the result. The second one is slightly less efficient, but has the advantage that it is fully adaptive, i.e., that it is possible to guarantee a prescribed accuracy of the result.

Inversion and preconditioners

Using the matrix-matrix multiplication algorithms, we can perform more complicated arithmetic operations like the inversion or the LU factorization. The derivation of the

corresponding algorithms is straightforward: if we express the result in terms of block matrices, we see that it can be computed by a sequence of matrix-matrix multiplications. We replace each of these products by its $\mathcal{H}^2$-matrix approximation and combine all of the $\mathcal{H}^2$-submatrices to get an $\mathcal{H}^2$-matrix approximation of the result (cf. Section 6.7 and Chapter 10).

If we perform all operations with high accuracy, the resulting inverse or factorization can be used as a direct solver for the original system, although it may require a large amount of storage. If we use only a low accuracy, we can still expect to get a good preconditioner which can be used in an efficient iterative or semi-iterative scheme, e.g., a conjugate gradient or GMRES method.

1.4 Which problems can be solved efficiently?

In this book, we focus on dense matrices arising from the discretization of integral equations, especially those connected to solving homogeneous elliptic partial differential equations with the boundary integral method. For numerical experiments, these matrices offer the advantage that they are discretizations of a continuous problem, therefore we have a scale of discretizations of differing resolution at our disposal and can investigate the behavior of the methods for very large matrices and high condition numbers. The underlying continuous problem is relatively simple, so we can easily construct test cases and verify the correctness of an implementation.

We also consider the construction of approximate inverses for the stiffness matrices arising from finite element discretizations of elliptic partial differential operators. In the paper [6], it has been proven that these inverses can be approximated by hierarchical matrices [62], [52], [63], but the proof is based on a global approximation argument that does not carry over directly to the case of $\mathcal{H}^2$-matrices. Chapter 9 uses the localized approach presented in [15] to construct low-rank approximations of the total cluster bases, and applying the general results of Chapter 6 and [13] yields the existence of $\mathcal{H}^2$-matrix approximations.

$\mathcal{H}^2$-matrices have also been successfully applied to problems from the field of electromagnetism [24], heat radiation, and machine learning.

1.5 Organization of the book

In the following, I try to give an overview of the current state of the field of $\mathcal{H}^2$-matrices. The presentation is organized in nine chapters covering basic definitions, algorithms with corresponding complexity analysis, approximation schemes with corresponding error analysis, and a number of numerical experiments.

Chapter 2: Model problem This chapter introduces the basic concepts of $\mathcal{H}^2$-matrices for a one-dimensional model problem. In this simple setting, the construction of an $\mathcal{H}^2$-matrix and the analysis of its complexity and approximation properties is fairly straightforward.

Chapter 3: Hierarchical matrices This chapter considers the generalization of the definition of $\mathcal{H}^2$-matrices to the multi-dimensional setting. $\mathcal{H}^2$-matrices are defined based on a *block cluster tree* describing a partition of the matrix into a hierarchy of submatrices and *cluster bases* describing the form of these submatrices. If a number of relatively general conditions for the block cluster tree and the cluster bases are fulfilled, it is possible to derive optimal-order estimates for the storage requirements and the time needed to compute the matrix-vector multiplication

Chapter 4: Integral operators A typical application of $\mathcal{H}^2$-matrices is the approximation of matrices resulting from the finite element (or boundary element) discretization of integral operators. This chapter describes simple approximation schemes based on Taylor expansion and constant-order interpolation, but also more advanced approaches based on variable-order interpolation. The error of the resulting $\mathcal{H}^2$-matrices is estimated by using error bounds for the local approximants of the kernel function.

Chapter 5: Orthogonal cluster bases This chapter describes techniques for finding the optimal $\mathcal{H}^2$-matrix approximation of a given arbitrary matrix under the assumption that a suitable block cluster tree and good cluster bases are already known. If the cluster bases are *orthogonal*, the construction of the optimal approximation is straightforward, therefore this chapter contains two algorithms for converting arbitrary cluster bases into orthogonal cluster bases: the first algorithm yields an orthogonal cluster basis that is equivalent to the original one, the second algorithm constructs an approximation of lower complexity.

Chapter 6: Compression This chapter introduces the *total cluster bases* that allow us to give an alternative characterization of $\mathcal{H}^2$-matrices and to develop algorithms for approximating arbitrary matrices. The analysis of these algorithms relies on the results of Chapter 5 in order to establish quasi-optimal error estimates.

Chapter 7: A priori matrix arithmetic Once a matrix has been approximated by an $\mathcal{H}^2$-matrix, the question of solving corresponding systems of linear equations has to be answered. For dense matrices, the usual solution strategies require factorizations of the matrix or sometimes even its inverse. Since applying these techniques directly to $\mathcal{H}^2$-matrices would be very inefficient, this chapter introduces an alternative: factorization and inversion can be performed using matrix-matrix products, therefore finding an efficient algorithm for approximating these products is an important step towards solving linear systems. By using the orthogonal projections introduced in Chapter 5 and preparing suitable quantities in advance, the best approximation of a matrix-matrix product in a given $\mathcal{H}^2$-matrix space can be computed very efficiently.

Chapter 8: A posteriori matrix arithmetic The algorithms introduced in Chapter 7 compute the best approximation of the matrix-matrix product in a *given* matrix space, but if this space is not chosen correctly, the resulting error can be quite large. This chapter describes an alternative algorithm that constructs an $\mathcal{H}^2$-matrix approximation of the matrix-matrix product and chooses the cluster bases in such a way that a given precision can be guaranteed.

Chapter 9: Elliptic partial differential equations Based on the a posteriori arithmetic algorithms of Chapter 8, it is possible to compute approximate inverses of $\mathcal{H}^2$-matrices, but it is not clear whether these inverses can be represented efficiently by an $\mathcal{H}^2$-matrix. This chapter proves that the inverse of the stiffness matrix of an elliptic partial differential equation can indeed be approximated well in the compressed format, and due to the best-approximation property of the compression algorithm, this means that the computation can be carried out efficiently.

Chapter 10: Applications The final chapter considers a number of practical applications of $\mathcal{H}^2$-matrices. Most of the applications are related to boundary integral formulations for Laplace's equation, but there are also some examples related to more general elliptic partial differential equations.

In some chapters, I have collected technical lemmas in a separate section in the hope of focusing the attention on the important results, not on often rather technical proofs of auxiliary statements.

Chapter 2
Model problem

In this chapter, we introduce the basic concepts of hierarchical matrices and $\mathcal{H}^2$-matrices. Since the underlying ideas are closely related to panel-clustering techniques [72] for integral equations, we use a simple integral operator as a model problem.

2.1 One-dimensional integral operator

Let us consider the integral operator

$$\mathcal{G}[u](x) := -\int_0^1 \log|x-y|u(y)\,dy$$

for functions $u \in L^2[0,1]$. For $n \in \mathbb{N}$, we discretize it by Galerkin's method using the n-dimensional space spanned by the basis $(\varphi_i)_{i=1}^n$ of piecewise constant functions given by

$$\varphi_i(x) = \begin{cases} 1 & \text{if } x \in [(i-1)/n, i/n], \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \{1,\ldots,n\}$ and $x \in [0,1]$. This leads to a matrix $G \in \mathbb{R}^{n\times n}$ with entries

$$\begin{aligned} G_{ij} &:= \int_0^1 \varphi_i(x) \int_0^1 g(x,y)\varphi_j(y)\,dy\,dx \\ &= \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} g(x,y)\,dy\,dx, \end{aligned} \tag{2.1}$$

where the *kernel function* is given by

$$g(x,y) := \begin{cases} -\log|x-y| & \text{if } x \neq y, \\ 0 & \text{otherwise.} \end{cases}$$

Due to $\operatorname{supp} g = [0,1]^2$, all entries G_{ij} of the matrix are non-zero, therefore even a simple task like computing the matrix-vector product in the standard way requires at least n^2 operations.

Finite element techniques tend to require a large number of degrees of freedom in order to reach a suitable precision, and if n is large, a quadratic complexity means that the algorithm will take *very* long to complete. Therefore we have to look for more efficient techniques for handling the matrix G.

2.2 Low-rank approximation

Since typically all entries of G will be non-zero, the standard sparse matrix representations used in the context of partial differential equations will not by efficient. Therefore we have to settle for an approximation which is not sparse, but only *data-sparse*, i.e., which requires significantly less storage than the original matrix.

In order to derive a data-sparse approximation of G, we rely on an approximation of the kernel function g. By defining

$$f : \mathbb{R}_{>0} \to \mathbb{R}, \quad z \mapsto -\log z,$$

we have

$$g(x,y) = \begin{cases} f(x-y) & \text{if } x > y, \\ f(y-x) & \text{if } x < y, \\ 0 & \text{otherwise,} \end{cases}$$

for all $x, y \in \mathbb{R}$. Since g is symmetric, we only consider the case $x > y$ in the following. We approximate f by its m-th order Taylor expansion (see, e.g., Chapter XIII, §6 in [75]) around a center $z_0 \in \mathbb{R}_{>0}$, which is given by

$$\tilde{f}_{z_0,m}(z) := \sum_{\alpha=0}^{m-1} f^{(\alpha)}(z_0) \frac{(z-z_0)^\alpha}{\alpha!}.$$

We pick $x_0, y_0 \in \mathbb{R}$ with $z_0 = x_0 - y_0$ and notice that the Taylor expansion of f gives rise to an approximation of g,

$$\begin{aligned} \tilde{g}_{z_0,m}(x,y) := \tilde{f}_{z_0,m}(x-y) &= \sum_{\alpha=0}^{m-1} f^{(\alpha)}(z_0) \frac{(x-y-z_0)^\alpha}{\alpha!} \\ &= \sum_{\alpha=0}^{m-1} f^{(\alpha)}(x_0-y_0) \frac{(x-x_0+(y_0-y))^\alpha}{\alpha!} \\ &= \sum_{\alpha=0}^{m-1} \frac{f^{(\alpha)}(x_0-y_0)}{\alpha!} \sum_{\mu=0}^{\alpha} \binom{\alpha}{\mu} (x-x_0)^{\alpha-\mu} (y_0-y)^\mu \\ &= \sum_{\alpha=0}^{m-1} f^{(\alpha)}(x_0-y_0) \sum_{\mu=0}^{\alpha} \frac{(x-x_0)^{\alpha-\mu}}{(\alpha-\mu)!} (-1)^\mu \frac{(y-y_0)^\mu}{\mu!} \\ &= \sum_{\nu=0}^{m-1} \sum_{\mu=0}^{m-1-\nu} (-1)^\mu f^{(\nu+\mu)}(x_0-y_0) \frac{(x-x_0)^\nu}{\nu!} \frac{(y-y_0)^\mu}{\mu!}. \end{aligned} \tag{2.2}$$

The main property of $\tilde{g}_{z_0,m}$, as far as hierarchical matrices are concerned, is that the variables x and y are separated in each term of the sum. Expansions with this property,

no matter whether they are based on polynomials or more general functions, are called *degenerate*. A data-sparse approximation of G can be obtained by replacing the original kernel function g by $\tilde{g}_{z_0,m}$ in (2.1):

$$\begin{aligned}\widetilde{G}_{ij} &:= \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} \tilde{g}_{z_0,m}(x,y)\,dy\,dx \\ &= \sum_{\nu=0}^{m-1} \sum_{\mu=0}^{m-1-\nu} (-1)^{\mu} f^{(\nu+\mu)}(x_0-y_0) \int_{(i-1)/n}^{i/n} \frac{(x-x_0)^{\nu}}{\nu!}\,dx \int_{(j-1)/n}^{j/n} \frac{(y-y_0)^{\mu}}{\mu!}\,dy.\end{aligned}$$

We introduce $K := \{0,\ldots,m-1\}$ and $\mathcal{I} := \{1,\ldots,n\}$ and matrices $V, W \in \mathbb{R}^{\mathcal{I}\times K}$ and $S \in \mathbb{R}^{K\times K}$ with

$$\begin{aligned} V_{i\nu} &:= \int_{(i-1)/n}^{i/n} \frac{(x-x_0)^{\nu}}{\nu!}\,dx, \quad W_{j\nu} := \int_{(j-1)/n}^{j/n} \frac{(y-y_0)^{\mu}}{\mu!}\,dy, \\ S_{\nu\mu} &:= \begin{cases} (-1)^{\mu} f^{(\nu+\mu)}(x_0-y_0) & \text{if } x_0 > y_0 \text{and } \nu+\mu < m, \\ (-1)^{\nu} f^{(\nu+\mu)}(y_0-x_0) & \text{if } x_0 < y_0 \text{and } \nu+\mu < m, \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.3}$$

for $i, j \in \mathcal{I}$ and $\nu \in K$, and find

$$\widetilde{G} = VSW^*, \tag{2.4}$$

i.e., we can represent $\widetilde{G}$ in a factorized form which requires only $2nm + m^2$ units of storage instead of n^2.

The factorized representation (2.4) implies that the rank of $\widetilde{G}$ is bounded by m. Conversely, each rank-m-matrix can be expressed in this factorized form.

2.3 Error estimate

Replacing g by $\tilde{g}_{z_0,m}$ leads to an approximation error which we have to control in order to ensure that the matrix $\widetilde{G}$ is not only data-sparse, but also useful.

A simple first error estimate can be obtained by using the Lagrange representation of the remainder of the Taylor expansion: we consider the approximation of f in an interval $[a,b]$ with $0 < a \leq b$ using $z_0 = (a+b)/2$ as the center of expansion. We have

$$\begin{aligned} |f(z) - \tilde{f}_{z_0,m}(z)| &= \left| f^{(m)}(\zeta_0) \frac{(z-z_0)^m}{m!} \right| = \frac{(m-1)!}{m!} \left(\frac{|z-z_0|}{z_z} \right)^m \\ &\leq \frac{1}{m} \left(\frac{b-a}{2a} \right)^m \qquad \text{for all } z \in [a,b],\ m \in \mathbb{N}, \end{aligned}$$

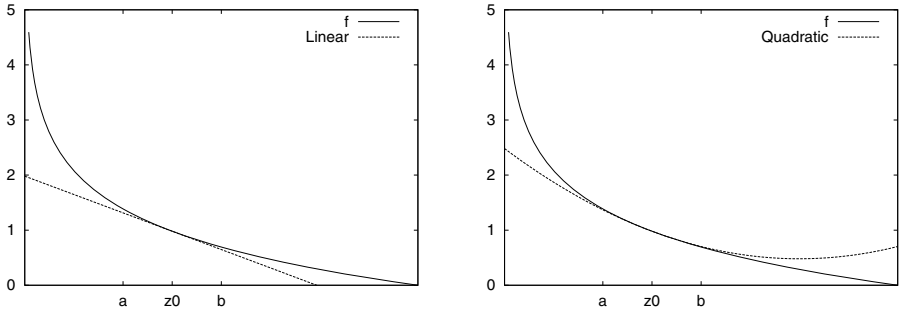


Figure 2.1. Approximation of f by linear and quadratic Taylor polynomials.

for points $z_z \in [a,b]$ depending on z and m. We can see that fast convergence is guaranteed if the diameter $b-a$ of the interval is less than its distance a from the singularity at zero. This simple error estimate is useful since it stresses the importance of controlling the ratio of diameter and distance, but it is not optimal: it suggests that we can only expect convergence if the diameter is not too large, while a more careful analysis reveals that the Taylor series will *always* converge as long as $a > 0$ holds. The refined proof is based on the Cauchy representation of the remainder:

Lemma 2.1 (Cauchy error representation). *Let $f \in C^\infty[a,b]$ and $z_0 \in [a,b]$. We have*

$$f(z) - \tilde{f}_{z_0,m}(z) = \int_0^1 (1-t)^{m-1} f^{(m)}(z_0 + t(z-z_0)) \frac{(z-z_0)^m}{(m-1)!}\, dt$$

for all $z \in [a,b]$ and all $m \in \mathbb{N}$.

Proof. See, e.g., [75], Chapter XIII, §6. □

Applying this error representation to the logarithmic function f and bounding the resulting terms carefully yields the following improved error estimate:

Lemma 2.2 (Error of the Taylor expansion). *Let $a,b \in \mathbb{R}$ with $0 < a < b$. Let $z_0 := (a+b)/2$ and $\zeta := 2a/(b-a)$. For all $z \in [a,b]$ and all $m \in \mathbb{N}$, we have*

$$|f(z) - \tilde{f}_{z_0,m}(z)| \leq \log\left(\frac{1}{\zeta} + 1\right)\left(\frac{1}{\zeta+1}\right)^{m-1}.$$

Proof. Let $z \in [a,b]$ and $m \in \mathbb{N}$. Due to Lemma 2.1, the Taylor expansion satisfies

$$f(z) - \tilde{f}_{z_0,m}(z) = \int_0^1 (1-t)^{m-1} f^{(m)}(z_0 + t(z-z_0)) \frac{(z-z_0)^m}{(m-1)!}\, dt. \tag{2.5}$$

For our special kernel function, we have

$$f^{(\nu)}(z) = \begin{cases} -\log(z) & \text{if } \nu = 0, \\ (\nu-1)!\,(-1)^{\nu}\,z^{-\nu} & \text{otherwise,} \end{cases}$$

and the remainder takes the form

$$\begin{aligned} |f(z) - \tilde{f}_{z_0,m}(z)| &\leq \int_0^1 (1-t)^{m-1}(m-1)!\,|z_0 + t(z-z_0)|^{-m}\,\frac{|z-z_0|^m}{(m-1)!}\,dt \\ &= \int_0^1 \frac{(1-t)^{m-1}|z-z_0|^m}{|z_0 + t(z-z_0)|^m}\,dt \\ &\leq \int_0^1 (1-t)^{m-1}\left(\frac{|z-z_0|}{|z_0| - t|z-z_0|}\right)^m dt. \end{aligned}$$

Due to

$$|z_0| = a + \frac{b-a}{2} = \zeta\frac{b-a}{2} + \frac{b-a}{2} = (\zeta+1)\frac{b-a}{2}, \quad |z-z_0| \leq \frac{b-a}{2},$$

we find

$$\frac{|z-z_0|}{|z_0| - t|z-z_0|} \leq \frac{(b-a)/2}{(\zeta+1)(b-a)/2 - t(b-a)/2} = \frac{1}{\zeta+1-t}$$

and observe

$$|f(z) - \tilde{f}_{z_0,m}(z)| \leq \int_0^1 \frac{(1-t)^{m-1}}{(\zeta+1-t)^m}\,dt = \int_0^1 \frac{1}{\zeta+s}\left(\frac{s}{\zeta+s}\right)^{m-1} ds$$

by substituting $s = 1-t$. Since elementary computations yield

$$\frac{s}{\zeta+s} \leq \frac{1}{\zeta+1} \quad \text{for all } s \in [0,1],$$

we can conclude

$$\begin{aligned} |f(z) - \tilde{f}_{z_0,m}(z)| &\leq \left(\frac{1}{\zeta+1}\right)^{m-1} \int_0^1 \frac{1}{\zeta+s}\,ds \\ &= \left(\frac{1}{\zeta+1}\right)^{m-1} (\log(1+\zeta) - \log\zeta) \\ &= \left(\frac{1}{\zeta+1}\right)^{m-1} \log\left(\frac{1+\zeta}{\zeta}\right), \end{aligned}$$

which is the desired result. □

The speed of convergence depends on the quantity ζ, the ratio between a and the radius of the interval $[a,b]$. In order to ensure uniform convergence of $\tilde{g}_{z_0,m}$, we have to assume a uniform lower bound of $|z| = |x-y|$.

Corollary 2.3. *Let $\eta \in \mathbb{R}_{>0}$. Let $t, s \subseteq \mathbb{R}$ be non-trivial intervals satisfying*

$$\operatorname{diam}(t) + \operatorname{diam}(s) \leq 2\eta \operatorname{dist}(t, s). \tag{2.6}$$

Let x_0 be the midpoint of t and let y_0 be the midpoint of s, and let $z_0 := x_0 - y_0$. Then the estimate

$$|g(x, y) - \tilde{g}_{z_0,m}(x, y)| \leq \log(\eta + 1) \left(\frac{\eta}{\eta + 1}\right)^{m-1}$$

holds for all $x \in t$, $y \in s$, and $m \in \mathbb{N}$.

Proof. Let $x \in t$, $y \in s$, and $m \in \mathbb{N}$. Since g and $\tilde{g}_{z_0,m}$ are symmetric, we can restrict our attention to the case $x > y$ without loss of generality.

For $a_t := \inf t$, $b_t := \sup t$, $a_s := \inf s$, $b_s := \sup s$ we have

$$\operatorname{diam}(t) = b_t - a_t, \quad \operatorname{diam}(s) = b_s - a_s, \quad \operatorname{dist}(t, s) = a_t - b_s,$$
$$x_0 = \frac{b_t + a_t}{2}, \quad y_0 = \frac{b_s + a_s}{2}, \quad z_0 = \frac{b_t + a_t - b_s - a_s}{2} = \frac{b + a}{2}$$

with $a := a_t - b_s = \operatorname{dist}(t, s)$ and $b := b_t - a_s$. We apply Lemma 2.2 to $z = x - y \in [a, b]$ and get

$$|g(x, y) - \tilde{g}_{z_0,m}(x, y)| \leq \log\left(1 + \frac{1}{\zeta}\right) \left(\frac{1}{1 + \zeta}\right)^{m-1}.$$

Now the *admissibility condition* (2.6) implies

$$\zeta = \frac{2a}{b - a} = \frac{2 \operatorname{dist}(t, s)}{\operatorname{diam}(t) + \operatorname{diam}(s)} \geq \frac{2 \operatorname{dist}(t, s)}{2\eta \operatorname{dist}(t, s)} = \frac{1}{\eta}$$

and we can conclude

$$|g(x, y) - \tilde{g}_{z_0,m}(x, y)| \leq \log(\eta + 1) \left(\frac{1}{1 + 1/\eta}\right)^{m-1} = \log(\eta + 1) \left(\frac{\eta}{\eta + 1}\right)^{m-1}$$

for all $x \in t$ and $y \in s$. □

This means that the Taylor expansion $\tilde{g}_{z_0,m}$ will converge exponentially in m, and that the speed of the convergence depends on the ratio of the distance and the diameter of the intervals. The assumption that the intervals containing x and y have positive distance is crucial: if x and y could come arbitrarily close, the resulting singularity in g could no longer be approximated by the polynomial $\tilde{g}_{z_0,m}$.

2.4 Local approximation

Corollary 2.3 implies that we cannot use a *global* Taylor expansion to store the entire matrix G in factorized form: at least the diagonal entries G_{ii} correspond to integrals with singular integrands which cannot be approximated efficiently by our approach.

Instead, we only use *local* Taylor expansions for subblocks of the matrix G. Let $\hat{t}, \hat{s} \subseteq \mathcal{I}$, and let $t, s \subseteq [0,1]$ be intervals satisfying

$$[(i-1)/n, i/n] \subseteq t \quad \text{and} \quad [(j-1)/n, j/n] \subseteq s \tag{2.7}$$

for all $i \in \hat{t}$ and $j \in \hat{s}$. Let $x_t \in t$ be the midpoint of t, and let $y_s \in s$ be the midpoint of s.

We assume that t and s satisfy the *admissibility condition* (2.6), so Corollary 2.3 implies that the Taylor expansion of g at the point $z_0 := x_t - y_s$ will converge exponentially for all $x \in t$ and $y \in s$.

Similar to the global case, we introduce the local approximation

$$\tilde{G}_{t,s} := V_t S_{t,s} W_s^* \tag{2.8}$$

with $S_{t,s} \in \mathbb{R}^{K \times K}$ and $V_t, W_s \in \mathbb{R}^{\mathcal{I} \times K}$ given by

$$\begin{aligned}
(V_t)_{iv} &:= \begin{cases} \int_{(i-1)/n}^{i/n} \frac{(x-x_t)^v}{v!}\, dx & \text{if } i \in \hat{t}, \\ 0 & \text{otherwise,} \end{cases} \\
(W_s)_{j\mu} &:= \begin{cases} \int_{(j-1)/n}^{j/n} \frac{(y-y_s)^\mu}{\mu!}\, dy & \text{if } j \in \hat{s}, \\ 0 & \text{otherwise,} \end{cases} \\
(S_{t,s})_{v\mu} &:= \begin{cases} (-1)^\mu f^{(v+\mu)}(x_t - y_s) & \text{if } x_t > y_s \text{ and } v+\mu < m, \\ (-1)^v f^{(v+\mu)}(y_s - x_t) & \text{if } x_t < y_s \text{ and } v+\mu < m, \\ 0 & \text{otherwise,} \end{cases}
\end{aligned} \tag{2.9}$$

for all $i, j \in \mathcal{I}$ and $v, \mu \in K$. The error estimate from Corollary 2.3 yields

$$\begin{aligned}
|G_{ij} - (\tilde{G}_{t,s})_{ij}| &\leq \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} |g(x,y) - \tilde{g}_{z_0,m}(x,y)|\, dy\, dx \\
&\leq \frac{1}{n^2} \max\{|g(x,y) - \tilde{g}_{z_0,m}(x,y)| \,:\, x \in [(i-1)/n, i/n], y \in [(j-1)/n, j/n]\} \\
&\leq \frac{1}{n^2} \max\{|g(x,y) - \tilde{g}_{z_0,m}(x,y)| \,:\, x \in t, y \in s\} \\
&\leq \frac{1}{n^2} \log(\eta+1) \left(\frac{\eta}{\eta+1}\right)^{m-1}
\end{aligned} \tag{2.10}$$

for all $m \in \mathbb{N}$, $i \in \hat{t}$ and $j \in \hat{s}$.

2.5 Cluster tree and block cluster tree

We have seen that local approximations $\widetilde{G}_{t,s}$ of subblocks $\hat{t} \times \hat{s}$ of the matrix G will converge exponentially if the intervals t and s satisfy the admissibility condition (2.6).

In order to approximate the entire matrix, we therefore have to cover it with subblocks that either satisfy the condition, and can therefore be stored efficiently in the factorized form (2.8), or which contain only a small number of entries, so that we can afford to store them in the standard way, i.e., as a two-dimensional array.

Techniques for constructing this covering for general matrices are discussed in Chapter 3, here we only consider a simple approach suitable for the model problem. We assume that there is a $q \in \mathbb{N}_0$ such that $n = 2^q$, and we define

$$t_{\ell,\alpha} := [\alpha 2^{-\ell}, (\alpha+1)2^{-\ell}], \quad \hat{t}_{\ell,\alpha} := \{\alpha 2^{q-\ell} + 1, \ldots, (\alpha+1)2^{q-\ell}\}$$

for all $\ell \in \{0, \ldots, q\}$ and all $\alpha \in \{0, \ldots, 2^\ell - 1\}$. By definition, we have

$$t_{\ell,\alpha} \subseteq [0,1], \quad \hat{t}_{\ell,\alpha} \subseteq \{1, \ldots, n\} = \mathcal{I},$$

and we observe

$$[(i-1)/n, i/n] = [(i-1)2^{-q}, i2^{-q}] \subseteq [\alpha 2^{q-\ell} 2^{-q}, (\alpha+1)2^{q-\ell}2^{-q}] = t_{\ell,\alpha}$$

for all $i \in \hat{t}_{\ell,\alpha}$. This means that the support of all basis functions φ_i with $i \in \hat{t}_{\ell,\alpha}$ is contained in $t_{\ell,\alpha}$.

We call the intervals $t_{\ell,\alpha}$ *clusters*, and we call $\hat{t}_{\ell,\alpha}$ the index set corresponding to $t_{\ell,\alpha}$. A very important property of clusters is that they can be arranged in a hierarchy: we have

$$\begin{aligned}
t_{\ell,\alpha} &= [\alpha 2^{-\ell}, (\alpha+1)2^{-\ell}] = [2\alpha 2^{-(\ell+1)}, (2\alpha+2)2^{-(\ell+1)}] \\
&= [2\alpha 2^{-(\ell+1)}, (2\alpha+1)2^{-(\ell+1)}] \cup [(2\alpha+1)2^{-(\ell+1)}, (2\alpha+2)2^{-(\ell+1)}] \\
&= t_{\ell+1,2\alpha} \cup t_{\ell+1,2\alpha+1}, \\
\hat{t}_{\ell,\alpha} &= \{\alpha 2^{q-\ell} + 1, \ldots, (\alpha+1)2^{q-\ell}\} = \{2\alpha 2^{q-\ell-1} + 1, \ldots, (2\alpha+2)2^{q-\ell-1}\} \\
&= \{2\alpha 2^{q-\ell-1} + 1, \ldots, (2\alpha+1)2^{q-\ell-1}\} \\
&\qquad \dot\cup \{(2\alpha+1)2^{q-\ell-1} + 1, \ldots, (2\alpha+2)2^{q-\ell-1}\} \\
&= \hat{t}_{\ell+1,2\alpha} \mathbin{\dot\cup} \hat{t}_{\ell+1,2\alpha+1}
\end{aligned}$$

for all $\ell \in \{0, \ldots, q-1\}$ and all $\alpha \in \{0, \ldots, 2^\ell - 1\}$. This observation suggests to organize the clusters in a tree. We pick the largest cluster $t_{0,0}$ as the root, and we define the father-son relationships by

$$\operatorname{sons}(t_{\ell,\alpha}) = \begin{cases} \{t_{\ell+1,2\alpha}, t_{\ell+1,2\alpha+1}\} & \text{if } \ell < q, \\ \emptyset & \text{otherwise,} \end{cases}$$

for all $\ell \in \{0, \ldots, q\}$ and $\alpha \in \{0, \ldots, 2^\ell - 1\}$. This defines a balanced binary tree of depth q with $2^{q+1} - 1$ nodes. Due to its relationship to the index set $\mathcal{I}$ we denote it by $\mathcal{T}_\mathcal{I}$, and since it consists of clusters, we call it a *cluster tree*.

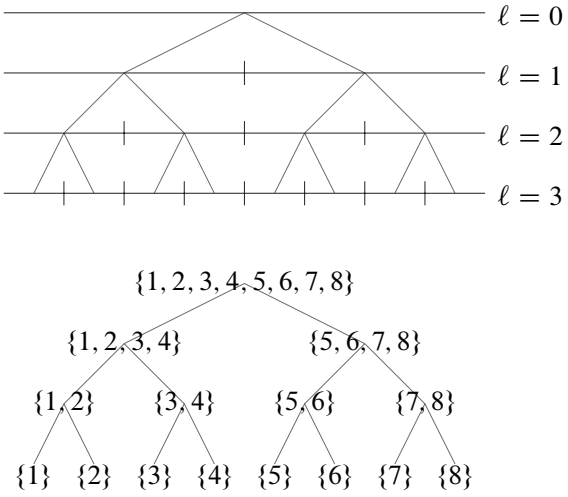


Figure 2.2. Cluster tree for $q = 3$.

The most important purpose of a cluster tree is to allow us to split the matrix G efficiently into submatrices $G|_{\hat{t}\times\hat{s}}$ that can be approximated efficiently, i.e., that correspond to intervals satisfying the admissibility condition. Using the cluster tree, we can organize the search for these submatrices by using a hierarchy: starting with $t = s = t_{0,0}$, we check whether a pair (t, s) of clusters is admissible. If it is, we can approximate the corresponding matrix block $G|_{\hat{t}\times\hat{s}}$. If the admissibility condition does not hold, we proceed to check all pairs consisting of sons of t and sons of s. The recursive procedure stops if a pair is admissible or if no more sons exist. In the latter case, our construction implies that $\hat{t}$ and $\hat{s}$ contain only a small number of indices (in our construction even only one), so we can afford to simply store all coefficients of the submatrix $G|_{\hat{t}\times\hat{s}}$.

The recursive search for admissible pairs of clusters suggests a second tree of pairs of clusters: its root is $(t_{0,0}, t_{0,0})$, and the father-son relationship is defined by

$$\operatorname{sons}(t,s) = \begin{cases} \emptyset & \text{if } (t,s) \text{ is admissible,} \\ \emptyset & \text{if } \operatorname{sons}(t) = \emptyset = \operatorname{sons}(s), \\ \{(t',s') \,:\, t' \in \operatorname{sons}(t),\ s' \in \operatorname{sons}(s)\} & \text{otherwise.} \end{cases}$$

The elements of this tree are called *blocks*, since each of them is a pair (t, s) corresponding to a block $G|_{\hat{t}\times\hat{s}}$ of the matrix G. The tree is called *block cluster tree* and

denoted by $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ since it describes a hierarchy of subsets of the index set $\mathcal{I}\times\mathcal{I}$ in the same way that the cluster tree $\mathcal{T}_{\mathcal{I}}$ describes a hierarchy of subsets of the index set $\mathcal{I}$.

Since the root of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ is the pair $(t_{0,0}, t_{0,0})$, the definition implies that each block $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ is of the form $b = (t_{\ell,\alpha}, t_{\ell,\beta})$ for $\ell \in \{0,\ldots,q\}$ and $\alpha,\beta \in \{0,\ldots,2^\ell - 1\}$. We use the short notation

$$b_{\ell,\alpha,\beta} := (t_{\ell,\alpha}, t_{\ell,\beta}).$$

In order to keep the presentation simple, we only consider the admissibility condition (2.6) with $\eta = 1$. A block $b_{\ell,\alpha,\beta}$ is admissible if and only if

$$\operatorname{diam}(t_{\ell,\alpha}) + \operatorname{diam}(t_{\ell,\beta}) \leq 2\operatorname{dist}(t_{\ell,\alpha}, t_{\ell,\beta})$$

holds. Due to the definition of the cluster tree, it is easy to see that

$$\operatorname{diam}(t_{\ell,\alpha}) = 2^{-\ell}, \quad \operatorname{dist}(t_{\ell,\alpha}, t_{\ell,\beta}) = 2^{-\ell}\max\{|\alpha-\beta|-1, 0\}$$

hold for all $\ell \in \{0,\ldots,q\}$ and $\alpha,\beta \in \{0,\ldots,2^\ell-1\}$, therefore a block $b_{\ell,\alpha,\beta}$ is admissible if and only if

$$1 \leq \max\{|\alpha-\beta|-1, 0\} \iff |\alpha-\beta| \geq 2$$

holds. This gives us a simple and practical alternative definition of the block cluster tree: its root is $b_{0,0,0}$, and we have

$$\operatorname{sons}(b_{\ell,\alpha,\beta}) = \begin{cases} \emptyset & \text{if } \ell = q \text{ or } |\alpha-\beta| \geq 2, \\ \{b_{\ell+1,2\alpha,2\beta}, b_{\ell+1,2\alpha+1,2\beta}, & \\ \quad b_{\ell+1,2\alpha,2\beta+1}, b_{\ell+1,2\alpha+1,2\beta+1}\} & \text{otherwise.} \end{cases}$$

Since we only have to store data for the leaves of the block cluster tree, we are interested in knowing how many leaves there are on the different levels ℓ.

Lemma 2.4 (Types of blocks). *A block $b_{\ell,\alpha,\beta}$ is called **diagonal** if $\alpha = \beta$ holds, it is called **neighbour** if $|\alpha-\beta| = 1$ holds, and it is called **admissible** otherwise.*

We denote the sets of diagonal, neighbour and admissible blocks on a level $\ell \in \{0,\ldots,q\}$ of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ by

$$\begin{aligned} \mathcal{D}_\ell &:= \{b_{\ell,\alpha,\beta} \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}} : \alpha = \beta\}, \\ \mathcal{N}_\ell &:= \{b_{\ell,\alpha,\beta} \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}} : |\alpha-\beta| = 1\}, \\ \mathcal{A}_\ell &:= \{b_{\ell,\alpha,\beta} \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}} : |\alpha-\beta| > 1\}. \end{aligned}$$

Then we have

$$\#\mathcal{D}_\ell = 2^\ell, \quad \#\mathcal{N}_\ell = 2^{\ell+1}-2, \quad \#\mathcal{A}_\ell = \begin{cases} 0 & \text{if } \ell = 0, \\ 3(2^\ell - 2) & \text{otherwise.} \end{cases} \tag{2.11}$$

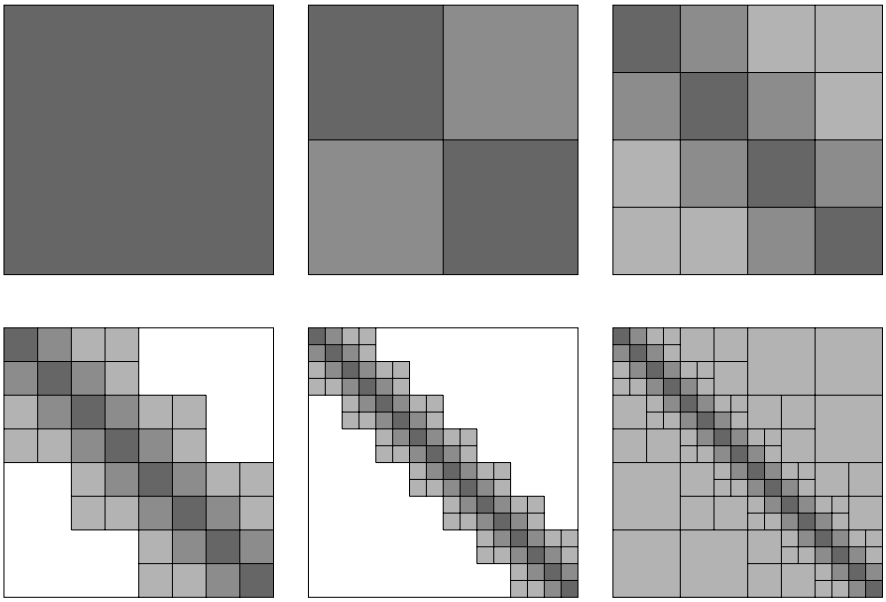


Figure 2.3. Diagonal, neighbour and admissible blocks on different levels of the block cluster tree. The bottom right picture shows all leaf blocks.

Proof. By induction on ℓ. On level $\ell = 0$ of the block cluster tree, there is exactly one block $b_{0,0,0}$, and it is obviously diagonal, so we have

$$\#\mathcal{D}_\ell = 1 = 2^\ell, \quad \#\mathcal{N}_\ell = 0 = 2^{\ell+1} - 2, \quad \#\mathcal{A}_\ell = 0.$$

Let now $\ell \in \{0, \dots, q-1\}$ be fixed such that (2.11) holds.

Let $b_{\ell,\alpha,\beta} \in \mathcal{D}_\ell$, i.e., we have $\alpha = \beta$. Since this block is not admissible, it is split into its sons $b_{\ell+1,2\alpha,2\alpha}$, $b_{\ell+1,2\alpha+1,2\alpha}$, $b_{\ell+1,2\alpha,2\alpha+1}$, $b_{\ell+1,2\alpha+1,2\alpha+1}$. The first and the last son are again diagonal, the others are neighbour blocks.

Let $b_{\ell,\alpha,\beta} \in \mathcal{N}_\ell$. Without loss of generality we assume $\alpha + 1 = \beta$. This block, too, is not admissible, so we split it into $b_{\ell+1,2\alpha,2\alpha+2}$, $b_{\ell+1,2\alpha+1,2\alpha+2}$, $b_{\ell+1,2\alpha,2\alpha+3}$, $b_{\ell+1,2\alpha+1,2\alpha+3}$. The second of these sons is a neighbour block, the others are admissible.

Let $b_{\ell,\alpha,\beta} \in \mathcal{A}_\ell$. Since admissible blocks are not split, this block contributes nothing to the next level.

Since each block has to be either diagonal, neighbour or admissible, we conclude

$$\begin{aligned}
\#\mathcal{D}_{\ell+1} &= 2\#\mathcal{D}_\ell = 2\,2^\ell = 2^{\ell+1},\\
\#\mathcal{N}_{\ell+1} &= 2\#\mathcal{D}_\ell + \#\mathcal{N}_\ell = 2\,2^\ell + 2^{\ell+1} - 2 = 2^{\ell+2} - 2,\\
\#\mathcal{A}_{\ell+1} &= 3\#\mathcal{N}_\ell = 3(2^{\ell+1} - 2),
\end{aligned}$$

and the induction is complete. □

The lemma implies that the number of blocks on a given level $\ell \in \{0, \ldots, q\}$ can be bounded by

$$\#\mathcal{D}_\ell + \#\mathcal{N}_\ell + \#\mathcal{A}_\ell < 2^\ell + 2\,2^\ell + 3\,2^\ell = 6\,2^\ell$$

and that the entire block cluster tree therefore contains not more than

$$\sum_{\ell=0}^{q} 6\,2^\ell = 6 \sum_{\ell=0}^{q} 2^\ell = 6(2^{q+1} - 1) \leq 12\,2^q = 12n$$

blocks, i.e., the number of blocks grows only linearly with the matrix dimension, not quadratically. This is a key property of the block cluster tree.

Since the sons of a block describe a disjoint partition of their father, a simple induction (cf. Corollary 3.15) yields that the set $\mathcal{L}_{\mathcal{I}\times\mathcal{I}}$ of leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ corresponds to a disjoint partition of $\mathcal{I} \times \mathcal{I}$, and this disjoint partition can be used to define an approximation of the matrix G.

2.6 Hierarchical matrix

Given a block cluster tree, we can now define an approximation of the matrix G. Until now we have denoted submatrices corresponding to blocks $b = (t, s)$ by $G|_{\hat{t}\times\hat{s}}$. In order to avoid technical complications dealing with different subsets of the index set $\mathcal{I}$, it is a good idea to use a different notation: we consider submatrices of G as elements of $\mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ which vanish outside of the index set $\hat{t} \times \hat{s}$. This does not increase the storage requirements, since only non-zero entries have to be stored, but it makes treating interactions between submatrices corresponding to different clusters easier.

For each cluster $t \in \mathcal{T}_\mathcal{I}$, we introduce the diagonal matrix $\chi_t \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ by

$$(\chi_t)_{ij} := \begin{cases} 1 & \text{if } i = j \in \hat{t}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i, j \in \mathcal{I}.$$

The restriction of G to a submatrix corresponding to a block $b = (t, s)$ is expressed by

$$\chi_t G \chi_s$$

in the new notation: all coefficients outside the rows in $\hat{t}$ are eliminated, and also all outside the columns of $\hat{s}$.

Since the leaves $\mathcal{L}_{\mathcal{I}\times\mathcal{I}}$ of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ form a disjoint partition of $\mathcal{I} \times \mathcal{I}$, we get

$$G = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{I}}} \chi_t G \chi_s. \tag{2.12}$$

We split the leaves into admissible and inadmissible ones using

$$\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}} := \{b \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}} \,:\, b \text{ is admissible}\}, \quad \mathcal{L}^-_{\mathcal{I}\times\mathcal{I}} := \mathcal{L}_{\mathcal{I}\times\mathcal{I}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}$$

and note that for admissible leaves the submatrix $\chi_t G \chi_s$ can be replaced by the approximation (2.8). This yields the approximation of G by a hierarchical matrix

$$\begin{aligned} \widetilde{G} &:= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}} \widetilde{G}_{t,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}} \chi_t G \chi_s \\ &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}} V_t S_b W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}} \chi_t G \chi_s. \end{aligned} \tag{2.13}$$

The submatrices corresponding to admissible blocks have the form

$$\widetilde{G}_{t,s} = V_t S_b W_s^*$$

and are typically stored in matrices $A_b = V_t S_b \in \mathbb{R}^{\hat{t}\times K}$ and $B_b = W_s \in \mathbb{R}^{\hat{s}\times K}$ in the simple product form

$$\widetilde{G}_{t,s} = A_b B_b^* \tag{2.14}$$

requiring $(\#\hat{t} + \#\hat{s})m$ units of storage. This representation is called the *hierarchical matrix representation* or *$\mathcal{H}$-matrix representation*.

Let us take a look at the storage requirements of this representation. Since the factorized representation is only useful if

$$(\#\hat{t})(\#\hat{s}) \geq (\#\hat{t} + \#\hat{s})m$$

holds, it makes sense to stop subdividing the block cluster tree before the blocks become too small. On level $\ell \in \{0,\ldots,q\}$, we have $\#\hat{t}_{\ell,\alpha} = 2^{q-\ell}$ for all $\alpha \in \{0,\ldots,2^\ell - 1\}$ and get

$$2^{q-\ell}2^{q-\ell} = (\#\hat{t}_{\ell,\alpha})(\#\hat{t}_{\ell,\beta}) \geq (\#\hat{t} + \#\hat{s})m = 2\,2^{q-\ell}m \iff 2^{q-\ell} \geq 2m.$$

We let

$$p_0 := \lceil \log_2 m \rceil + 1, \quad 2m = 2^{\log_2 m + 1} \leq 2^{p_0} < 2^{\log_2 m + 2} = 4m \tag{2.15}$$

and choose to stop subdividing blocks at level $p := q - p_0$. This implies

$$2^{q-\ell} \geq 2^{q-p} = 2^{p_0} \geq 2m$$

and therefore ensures that we apply the approximation only to submatrices that are sufficiently large. From now on we assume that the cluster tree $\mathcal{T}_\mathcal{I}$ and the block cluster tree stop at level p and that the hierarchical matrix $\widetilde{G}$ is defined accordingly.

By definition, admissible blocks $b_{\ell,\alpha,\beta} \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ are stored in the factorized form (2.14) requiring

$$(\#\hat{t}_{\ell,\alpha} + \#\hat{t}_{\ell,\beta})m = (2^{q-\ell} + 2^{q-\ell})m = 2m2^{q-\ell}$$

units of storage. Using Lemma 2.4 yields that *all* admissible blocks require

$$\begin{aligned}\sum_{\ell=0}^{p}(\#\mathcal{A}_\ell)2m2^{q-\ell} &= \sum_{\ell=1}^{p}3(2^\ell-2)2m2^{q-\ell} = 6m\sum_{\ell=1}^{p}2^q - 2^{q-\ell+1}\\ &= 6mpn - 12mn + 6m2^{p_0+1} = 6m(p-2)n + 6m2^{p_0+1}\end{aligned}$$

units of storage if $p \geq 2$. Due to $2^{p_0} \leq n$ we get the upper bound

$$6m(p-2)n + 6m2^{p_0+1} \leq 6m(p-2)n + 6m2n = 6mpn, \tag{2.16}$$

i.e., the storage requirements for the admissible blocks are in $\mathcal{O}(nm\log_2 n)$.

The inadmissible blocks occurring on the maximal level p are stored as two-dimensional arrays requiring

$$(\#\hat{t}_{\ell,\alpha})(\#\hat{t}_{\ell,\beta}) = 2^{q-\ell}2^{q-\ell} = 2^{2p_0}$$

units of storage per block, and using Lemma 2.4 shows that *all* inadmissible blocks require

$$\begin{aligned}(\#\mathcal{D}_p + \#\mathcal{N}_p)2^{2p_0} &= (2^p + 2^{p+1} - 2)2^{2p_0} = (3\,2^p - 2)2^{2p_0}\\ &= 3\,2^{p+p_0}2^{p_0} - 2^{2p_0+1} = 3n2^{p_0} - 2^{2p_0+1}\end{aligned}$$

units of storage. Due to $2^{p_0} \leq 4m$ we get the upper bound

$$3n2^{p_0} - 2^{2p_0+1} \leq 3n4m = 12nm, \tag{2.17}$$

i.e., the storage requirements for the inadmissible blocks are in $\mathcal{O}(nm)$.

We can conclude that the representation of $\widetilde{G}$ by a hierarchical matrix requires not more than

$$6m(p+2)n \tag{2.18}$$

units of storage, therefore the storage complexity is in $\mathcal{O}(nm\log_2 n)$, a major improvement over the quadratic complexity of the traditional representation by a two-dimensional array.

2.7 Matrix approximation error

The approximation error is given by

$$G - \widetilde{G} = \sum_{b=t\times s\in\mathcal{P}_{\text{far}}} (\chi_t G \chi_s - G_{t,s}),$$

i.e., we can construct global error estimates by combining local ones.

For the Frobenius norm, we can use (2.10) to find the following simple estimate:

$$\begin{aligned}\|G-\widetilde{G}\|_F &= \Big(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}(G_{ij}-\widetilde{G}_{ij})^2\Big)^{1/2}\\ &\leq \left(\frac{1}{n^2}\log^2(\eta+1)\left(\frac{\eta}{\eta+1}\right)^{2m-2}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}\frac{1}{n^2}\right)^{1/2}\\ &= \frac{1}{n}\log(\eta+1)\left(\frac{\eta}{\eta+1}\right)^{m-1}.\end{aligned} \tag{2.19}$$

For the spectral norm, we can derive a simple global error estimate by using the Cauchy–Schwarz inequality: for all vectors $x\in\mathbb{R}^{\mathcal{I}}$ we have

$$\|Gx\|_2 = \Big(\sum_{i\in\mathcal{I}}\Big(\sum_{j\in\mathcal{I}}G_{ij}x_j\Big)^2\Big)^{1/2} \leq \Big(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}G_{ij}^2\sum_{k\in\mathcal{I}}x_k^2\Big)^{1/2} = \|G\|_F\|x\|_2,$$

and this implies $\|G\|_2\leq\|G\|_F$, so we conclude

$$\|G-\widetilde{G}\|_2\leq\|G-\widetilde{G}\|_F\leq\frac{1}{n}\log(\eta+1)\left(\frac{\eta}{\eta+1}\right)^{m-1}. \tag{2.20}$$

The storage requirements of the $\mathcal{H}$-matrix are bounded by $6mn(p+2)$, therefore they only increase linearly with respect to m, while (2.19) and (2.20) show that the error decreases exponentially (for our choice $\eta=1$, each additional Taylor term at least halves the error).

2.8 $\mathcal{H}^2$-matrix

We have seen that the nearfield matrices require not more than $12nm$ units of storage, i.e., the storage requirements grow only linearly in the number n of degrees of freedom. The farfield representation of an $\mathcal{H}$-matrix, on the other hand, requires roughly $6mnp$ units of storage, i.e., it scales like $n\log_2 n$ if the matrix dimension grows.

For large problems, the logarithmic factor is undesirable, therefore we are interested in finding a representation of $\widetilde{G}$ that scales linearly with n.

The key idea is to consider the matrices V_t and W_s appearing in (2.8) not independently, but to treat the entire families $(V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(W_s)_{s\in\mathcal{T}_{\mathcal{I}}}$ and take advantage of relationships between "family members". These families are referred to as *cluster bases*, since the columns of V_t form a generating set for the range of $\widetilde{G}_{t,s}$, while the columns of W_s form a generating set for the range of $\widetilde{G}_{t,s}^*$. The columns do not have to be linearly independent, although they usually will be (e.g., for the orthogonal cluster bases introduced in Chapter 5).

Let us take a look at a matrix V_t for $t \in \mathcal{T}_I$: it is defined by

$$(V_t)_{iv} = \int_{(i-1)/n}^{i/n} \frac{(x - x_t)^v}{v!}\, dx$$

for the midpoint x_t of t, $i \in \hat{t}$ and $v \in K$.

We consider the case that t is not a leaf. Let $t' \in \operatorname{sons}(t)$, and let $x_{t'}$ be the midpoint of t'. In this case, we can use

$$\begin{aligned}\frac{(x - x_t)^v}{v!} &= \frac{1}{v!}(x - x_{t'} + x_{t'} - x_t)^v = \frac{1}{v!}\sum_{\mu=0}^{v} \binom{v}{\mu}(x - x_{t'})^\mu (x_{t'} - x_t)^{v-\mu} \\ &= \sum_{\mu=0}^{v} \frac{(x - x_{t'})^\mu}{\mu!}\frac{(x_{t'} - x_t)^{v-\mu}}{(v-\mu)!}\end{aligned}$$

to express the integrand in terms of Taylor monomials centered at $x_{t'}$ of t' instead of x_t and get

$$(V_t)_{iv} = \int_{(i-1)/n}^{i/n} \frac{(x - x_t)^v}{v!}\, dx = \sum_{\mu=0}^{v} \int_{(i-1)/n}^{i/n} \frac{(x - x_{t'})^\mu}{\mu!}\, dx \frac{(x_{t'} - x_t)^{v-\mu}}{(v-\mu)!} \tag{2.21}$$

for all $i \in \hat{t}'$ and all $v \in K$. The first factor is just the coefficient $(V_{t'})_{i\mu}$ of the matrix corresponding to the cluster t'. The second factor describes the change of basis from the center x_t to the center $x_{t'}$, and we collect its coefficients in a *transfer matrix* $E_{t'} \in \mathbb{R}^{K\times K}$ given by

$$(E_{t'})_{\mu v} := \begin{cases} \frac{(x_{t'} - x_t)^{v-\mu}}{(v-\mu)!} & \text{if } \mu \le v, \\ 0 & \text{otherwise,} \end{cases}$$

for all $v, \mu \in K$. Using this matrix, the equation (2.21) can be expressed by

$$(V_t)_{iv} = \sum_{\mu\in K} (V_{t'})_{i\mu}(E_{t'})_{\mu v} = (V_{t'}E_{t'})_{iv} \tag{2.22}$$

for all $i \in \hat{t}'$ and all $v \in K$.

Let now $i \in \hat{t}$ and $v \in K$. Due to our definition of the cluster tree, there is exactly one $t_i \in \operatorname{sons}(t)$ with $i \in \hat{t}_i$. Combining the equations (2.9) and (2.22) yields

$$\sum_{t'\in\operatorname{sons}(t)} (V_{t'}E_{t'})_{iv} = (V_{t_i}E_{t_i})_{iv} = (V_t)_{iv},$$

since the i-th row of $V_{t'}$ can only differ from zero if $i \in \hat{t}'$, and we have already established that this only happens for $t' = t_i$.

We can conclude that

$$V_t = \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'}$$

holds for all cluster $t \in \mathcal{T}_{\mathcal{I}}$ that are not leaves, i.e., the matrix V_t corresponding to any non-leaf cluster t can be expressed in terms of the matrices $V_{t'}$ corresponding to the sons of t. Cluster bases with this property are called *nested*.

This property suggests a data-sparse, hierarchical representation for $(V_t)_{t \in \mathcal{T}_{\mathcal{I}}}$: we store V_t only for leaf clusters and use the expansion matrices E_t to express all other cluster basis matrices. Since the expansion matrices require only m^2 units of storage, this representation is far more efficient than the original one.

Let us consider the storage requirements for a cluster basis $(V_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ represented in this form. With q, p and p_0 as in Section 2.6, we see that the cluster tree $\mathcal{T}_{\mathcal{I}}$ consists of

$$\sum_{\ell=0}^{p} 2^{\ell} = 2^{p+1} - 1 \tag{2.23}$$

clusters. We have to store a transfer matrix E_t for all clusters except for the root, and since each of these matrices is given by m^2 coefficients, we need $m^2(2^{p+1} - 2)$ units of storage.

We also have to store the matrices V_t for all *leaf* clusters. Due to our construction, all leaf clusters are on level p, and this level contains 2^p clusters t of size $\#\hat{t} = 2^{q-p}$. Since we only have to store the $\#\hat{t}$ non-zero rows of each V_t, the storage requirements for one matrix are $(\#\hat{t})m = 2^{q-p}m$, and for all matrices

$$2^p 2^{q-p} m = 2^q m = nm.$$

Therefore the entire cluster basis $(V_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ requires

$$nm + 2m^2(2^p - 1)$$

units of storage, and (2.15) yields the bound

$$nm + 2m^2(2^p - 1) \leq nm + 2^{p_0} m 2^p = nm + m2^{p+p_0} = nm + m2^q = 2nm \tag{2.24}$$

for the storage requirements of the cluster basis $(V_t)_{t \in \mathcal{T}_{\mathcal{I}}}$. Since $W_s = V_s$ holds for all $s \in \mathcal{T}_{\mathcal{I}}$, the cluster basis $(W_s)_{s \in \mathcal{T}_{\mathcal{I}}}$ requires no additional storage.

This leaves only the matrices $(S_b)_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}}$ describing the coupling coefficients of clusters t and s with respect to the cluster bases $(V_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ and $(W_s)_{s \in \mathcal{T}_{\mathcal{I}}}$. Each of these matrices requires m^2 units of storage, and we have to store one matrix for each admissible block. Using Lemma 2.4 we can compute that all the coupling matrices require

$$\sum_{\ell=0}^{p} (\#\mathcal{A}_\ell) m^2 = \sum_{\ell=1}^{p} 3(2^\ell - 2)m^2 = 6m^2 \sum_{\ell=0}^{p-1} (2^\ell - 1) = 6m^2(2^p - 1 - p)$$

units of storage, and due to (2.15) we can bound this by

$$6m^2(2^p - 1 - p) \leq 3m2m2^p \leq 3m2^{p_0}2^p = 3m2^q = 3mn. \tag{2.25}$$

We see that we have reached our goal: the storage requirements for the cluster basis and the coupling matrices, i.e., for the representation of all admissible blocks, are bounded by $5mn$, and therefore grow only linearly with n.

A matrix given in the form (2.13), where the cluster bases $(V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(W_s)_{s\in\mathcal{T}_\mathcal{I}}$ are nested and represented by expansion matrices, is called an $\mathcal{H}^2$*-matrix*. We have seen that the nearfield requires less than $12mn$ units of storage, the coupling matrices require less than $3mn$ units, and the cluster basis requires $2mn$, so we get a total of less than $17mn$ units of storage for the $\mathcal{H}^2$-matrix.

More precisely, the $\mathcal{H}$-matrix representation of the admissible blocks requires

$$6m((p-2)n + 2^{p_0+1}) = 6mn(p-2) + 12m2^{p_0}$$

units of storage according to (2.16), while the $\mathcal{H}^2$-matrix requires not more than

$$5mn$$

units of storage for the same purpose according to (2.24) and (2.25). We can see that the $\mathcal{H}^2$-matrix will be more efficient as soon as p grows larger than 3. The estimates even imply that the ratio of the storage requirements of $\mathcal{H}$- and $\mathcal{H}^2$-matrices is bounded from below by $p-2$, i.e., the $\mathcal{H}^2$-matrix representation will become more efficient as the matrix dimension grows.

2.9 Numerical experiment

We conclude this chapter with a simple numerical experiment: we compare the matrix G and its $\mathcal{H}^2$-matrix approximation $\tilde{G}$ for different problem dimensions n and different expansion orders m. In order to improve the storage efficiency, we stop the construction of the cluster tree as soon as the leaves contain not more than $4m$ indices.

Table 2.1 contains the absolute approximation errors $\|G-\tilde{G}\|_2$ in the spectral norm, estimated by a power iteration. For $\eta = 1$, our theory predicts that the approximation error will be proportional to $(1/2)^m$, and this behaviour is indeed visible. The theory also states that the error will be proportional to $1/n$, and this prediction also coincides with the experiment.

Table 2.2 contains the storage requirements in units of 1 KByte = 1024 Bytes. Our theoretical estimates predict that the storage requirements will be proportional to n, and this is clearly visible: $\mathcal{H}^2$-matrices indeed have linear complexity in the number n of degrees of freedom. The prediction of the dependence on m is far less accurate, since the expansion order influences the depth of the cluster tree and thereby the balance

Table 2.1. Approximation errors for the model problem.

n	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$
256	3.3_{-4}	7.0_{-5}	1.2_{-5}	3.8_{-6}	1.1_{-6}	4.2_{-7}	1.5_{-7}
512	1.7_{-4}	3.6_{-5}	6.0_{-6}	2.0_{-6}	5.6_{-7}	2.2_{-7}	7.5_{-8}
1024	8.4_{-5}	1.9_{-5}	3.0_{-6}	1.0_{-6}	2.9_{-7}	1.1_{-7}	3.8_{-8}
2048	4.2_{-5}	9.4_{-6}	1.5_{-6}	5.3_{-7}	1.4_{-7}	5.7_{-8}	1.9_{-8}
4096	2.1_{-5}	4.7_{-6}	7.6_{-7}	2.7_{-7}	7.2_{-8}	2.9_{-8}	9.5_{-9}
8192	1.1_{-5}	2.4_{-6}	3.8_{-7}	1.3_{-7}	3.6_{-8}	1.4_{-8}	4.8_{-9}

Table 2.2. Storage requirements in KB per degree of freedom for $\mathcal{H}^2$-matrix approximations and standard array representation for the model problem.

n	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	Array
256	0.45	0.39	0.43	0.51	0.55	0.59	0.64	2.0
512	0.46	0.40	0.45	0.53	0.58	0.62	0.68	4.0
1024	0.47	0.41	0.46	0.55	0.59	0.64	0.70	8.0
2048	0.47	0.41	0.46	0.56	0.60	0.65	0.71	16.0
4096	0.47	0.42	0.47	0.56	0.60	0.66	0.71	32.0
8192	0.48	0.42	0.47	0.56	0.61	0.66	0.72	64.0
1048576	0.48	0.42	0.47	0.56	0.61	0.66	0.72	8192.0

between storage required for the nearfield, coefficient, expansion and cluster basis matrices and storage required for internal bookkeeping of the program.

Still we can see that using an expansion order of $m = 7$ will yield a precision which should be sufficient for most practical applications at a cost of less than 1 KByte per degree of freedom. For $n = 8192 = 2^{13}$, this translates to a compression factor of 1.13%, and the compression factor will improve further as n grows larger. As an example, the case $n = 1048576 = 2^{20}$ has been included. The array representation would require more than 8 TBytes, while the $\mathcal{H}^2$-representation takes less than 1 GByte. This corresponds to a compression factor of 0.009%.

On a single 900 MHz UltraSPARC IIIcu processor of a SunFire 6800 computer, the setup of the standard representation for $n = 8192$ requires 118 seconds, while the setup of the $\mathcal{H}^2$-matrix approximation with $m = 7$ is accomplished in less than 0.5 seconds, so we can conclude that the compressed format saves not only storage, but also time.

Chapter 3
Hierarchical matrices

We have seen that $\mathcal{H}^2$-matrices can be used to represent the dense matrices corresponding to the model problem efficiently. In order to be able to treat more general problems, we require more general structures which still retain the important properties observed in the model problem.

We start by introducing general cluster trees and block cluster trees. The definitions can be kept relatively simple: the most important property of a cluster tree, as far as theoretical investigations are involved, is that the index set corresponding to a non-leaf cluster is the disjoint union of the index sets corresponding to its sons. The most important property of a block cluster tree is that it is a cluster tree and that the index sets corresponding to its nodes are of product structure.

Once we have general block cluster trees at our disposal, we can define general hierarchical matrices and prove bounds for their storage complexity. The definition of general $\mathcal{H}^2$-matrices is slightly more challenging, since we have to be able to handle simple constant-order approximation schemes as well as a variety of variable-order schemes, and we are looking for definitions that allow us to treat all of these methods in a unified way and still get optimal estimates. The key is the definition of a *bounded rank distribution* that covers all applications mentioned above and is still relatively simple.

The complexity estimate for $\mathcal{H}^2$-matrices given for the model problem in the previous chapter essentially relies on the fact that the number of matrix blocks is proportional to the number of clusters. We can use the same approach for general $\mathcal{H}^2$-matrices, i.e., we introduce the concept of *sparse block cluster trees* and prove that the storage requirements of an $\mathcal{H}^2$-matrix grow linearly with respect to the number of clusters if the rank distributions are bounded and the block cluster tree is sparse.

There is a price to pay for the optimal order of complexity: we cannot store cluster bases directly, but have to rely on transfer matrices. This implies that we cannot compute matrix-vector products directly, but require suitable recursive algorithms: the *forward* and *backward transformations*, which allow us to perform the computation in linear complexity.

This chapter is organized as follows:

- Section 3.1 introduces the general definition of a cluster tree and proves a number of its basic properties.
- Section 3.2 contains the general definition of a block cluster tree.
- Section 3.3 describes a simple geometrical construction for cluster and block cluster trees, which will be used in numerical examples presented in the other chapters.

- Section 3.4 gives the general definition of hierarchical matrices, a predecessor of $\mathcal{H}^2$-matrices (cf. [62], [68], [67], [52], [63]).
- Section 3.5 introduces general cluster bases and the basic concepts for estimating the complexity of algorithms for $\mathcal{H}^2$-matrices.
- Section 3.6 is devoted to the general definition of $\mathcal{H}^2$-matrices and $\mathcal{H}^2$-matrix spaces and to proving bounds for the storage complexity (cf. [70]).
- Section 3.7 presents the most important algorithm in the context of $\mathcal{H}^2$-matrices: the evaluation of the product of an $\mathcal{H}^2$-matrix and an arbitrary vector.
- Section 3.8 contains a number of definitions that allow us to express complexity estimates in terms of matrix dimensions instead of numbers of clusters.
- Section 3.9 contains a number of auxiliary results required in the other sections. The proofs are only included for the sake of completeness.

Assumptions in this chapter: We assume that finite index sets $\mathcal{I}$ and $\mathcal{J}$ are given and let $n_{\mathcal{I}} := \#\mathcal{I}$ and $n_{\mathcal{J}} := \#\mathcal{J}$ denote the cardinalities of these index sets.

3.1 Cluster tree

The construction of the $\mathcal{H}$- and $\mathcal{H}^2$-matrix approximations of the matrix G in Chapter 2 is based on a *cluster tree* $\mathcal{T}_{\mathcal{I}}$ and a *block cluster tree* $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$. The elements $t \in \mathcal{T}_{\mathcal{I}}$ of the cluster tree correspond to index sets $\hat{t} \subseteq \mathcal{I}$, and the elements $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ of the block cluster tree correspond to pairs of these index sets describing submatrices $G|_{\hat{t}\times\hat{s}}$ of the matrix G.

The cluster tree used in the model case has a very regular structure: all leaves are on the same level of the tree, and if a cluster is not a leaf, it has exactly two sons of exactly the same size. These properties severely limit the applicability of these regular cluster trees. In order to be able to handle more general matrices, we require more general cluster trees.

Definition 3.1 (Tree). Let $\mathcal{T} = (V, S, r)$ be a triple of a finite set V, a mapping

$$S\colon V \to \mathcal{P}(V)$$

of V into the subsets of V, and an element $r \in V$.

A tuple $v_0, v_1, \ldots, v_\ell \in V$ is called a *path* connecting v_0 and v_ℓ in $\mathcal{T}$ if

$$v_i \in S(v_{i-1}) \quad \text{holds for all } i \in \{1, \ldots, \ell\}.$$

The triple $\mathcal{T}$ is called a *tree* if for each $v \in \mathcal{T}$ there is a unique path connecting r to v in $\mathcal{T}$.

In this case, we call V the *set of nodes*, r the *root*, and $S(v)$ the *set of sons* of $v \in V$.

Definition 3.2 (Tree notations). Let $\mathcal{T} = (V, S, r)$ be a tree.

In order to avoid working with the triple (V, S, r) explicitly, we introduce the short notation $v \in \mathcal{T}$ for $v \in V$, $\operatorname{sons}(v) = S(v)$ (assuming that the tree $\mathcal{T}$ corresponding to v is clear from the context), and $\operatorname{root}(\mathcal{T}) = r$.

Due to Definition 3.1, for each $v \in \mathcal{T}$ there is a unique path $r = v_0, \ldots, v_\ell = v$ connecting the root to v. If $v \neq r$, we have $\ell > 0$, therefore $v_{\ell-1} \in \mathcal{T}$ exists, i.e., there is a unique $v_{\ell-1} \in \mathcal{T}$ with $v \in \operatorname{sons}(v_{\ell-1})$. We call this node the *father* of v and denote it by $\operatorname{father}(v)$.

In the model problem, there is an index set $\hat{t} \subseteq \mathcal{I}$ associated with each cluster $t \in \mathcal{T}_\mathcal{I}$. In the general case, we use *labels* to attach additional information to each node of a tree:

Definition 3.3 (Labeled tree). Let $\mathcal{T} = (V, S, r, \lambda, L)$. $\mathcal{T}$ is a *labeled tree* if (V, S, r) is a tree and if $\lambda\colon V \to L$ is a mapping from the set of nodes V into L.

The set L is called the *label set*, and for all $v \in V$, $\lambda(v)$ is called the *label of* v and denoted by $\hat{v}$. The notations for trees introduced in Definition 3.2 are also used for labeled trees.

Using the general labeled tree, we can define the general cluster tree that keeps the most important properties of the simple one introduced in Section 2.5, but can also be useful in far more general situations.

Definition 3.4 (Cluster tree). Let $\mathcal{T}_\mathcal{I}$ be a labeled tree. $\mathcal{T}_\mathcal{I}$ is a *cluster tree* for the index set $\mathcal{I}$ if the following conditions hold:

- The label of the root $r = \operatorname{root}(\mathcal{T}_\mathcal{I})$ is $\mathcal{I}$, i.e., $\hat{r} = \mathcal{I}$.
- If $t \in \mathcal{T}_\mathcal{I}$ has at least one son, then the labels of the sons form a disjoint partition of the label of the father, i.e., $\hat{t} = \dot{\bigcup}\{\hat{s} \,:\, s \in \operatorname{sons}(t)\}$.

The nodes $t \in \mathcal{T}_\mathcal{I}$ of a cluster tree $\mathcal{T}_\mathcal{I}$ are called *clusters*. A cluster $t \in \mathcal{T}_\mathcal{I}$ with $\operatorname{sons}(t) = \emptyset$ is called a *leaf*, and the set of all leaves is denoted by

$$\mathcal{L}_\mathcal{I} := \{t \in V \,:\, \operatorname{sons}(t) = \emptyset\}.$$

Differently from the special case considered for the model problem, we allow arbitrary partitions of the index set instead of the contiguous sets used in the one-dimensional case, see Figure 3.1.

Definition 3.5 (Cluster relationships). For all $t \in \mathcal{T}_\mathcal{I}$, we define the *set of descendants* inductively by

$$\operatorname{sons}^*(t) := \begin{cases} \{t\} \cup \bigcup_{t' \in \operatorname{sons}(t)} \operatorname{sons}^*(t') & \text{if } \operatorname{sons}(t) \neq \emptyset \\ \{t\} & \text{otherwise.} \end{cases}$$

The *set of predecessors* of a cluster $t \in \mathcal{T}_\mathcal{I}$ is given by

$$\operatorname{pred}(t) := \{t^* \in \mathcal{T}_\mathcal{I} \,:\, t \in \operatorname{sons}^*(t^*)\}.$$

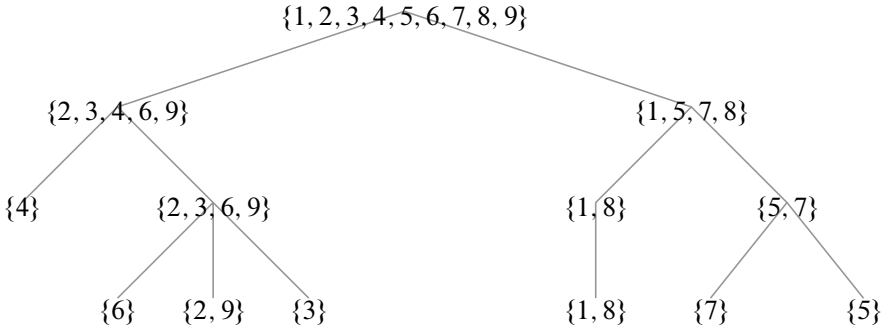


Figure 3.1. Simple cluster tree for $\mathcal{I} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

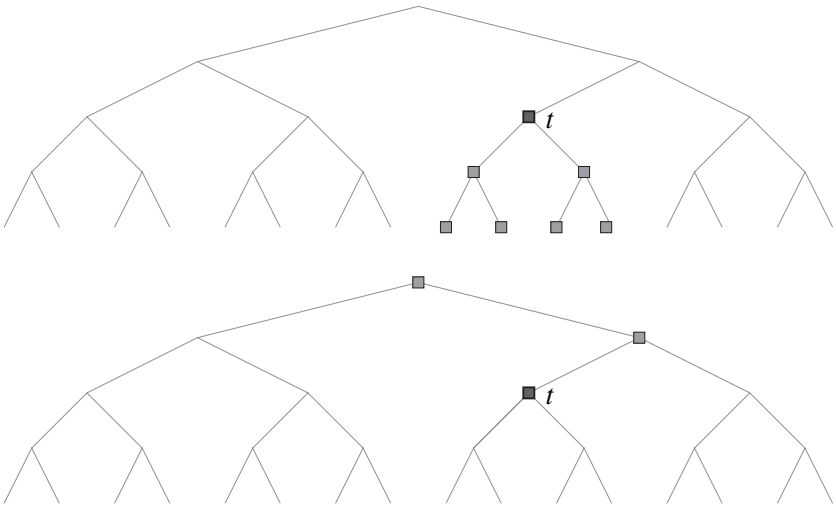


Figure 3.2. Descendants (top) and predecessors (bottom) of a cluster t.

It is frequently useful to split the cluster tree into *levels*, i.e., to assign each cluster an integer according to its distance from the root.

Definition 3.6 (Cluster level). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree. The *level* of a cluster $t \in \mathcal{T}_\mathcal{I}$ is defined inductively by

$$\operatorname{level}(t) = \begin{cases} \operatorname{level}(t^+) + 1 & \text{if there is a cluster } t^+ \in \mathcal{T}_\mathcal{I} \text{ with } t \in \operatorname{sons}(t^+) \\ 0 & \text{otherwise, i.e., if } t = \operatorname{root}(\mathcal{T}_\mathcal{I}). \end{cases}$$

The definition implies that the root will be assigned a level of zero and all other clusters

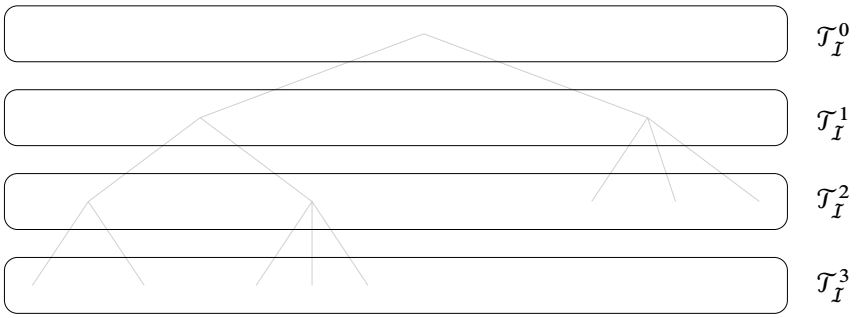


Figure 3.3. Levels in a cluster tree.

t will be assigned the level of their father t^+ plus one.

All clusters on a given level $\ell \in \mathbb{N}_0$ are collected in the set

$$\mathcal{T}_I^\ell := \{t \in \mathcal{T}_I \,:\, \operatorname{level}(t) = \ell\}$$

and all leaves on a given level $\ell \in \mathbb{N}_0$ in the set

$$\mathcal{L}_I^\ell := \mathcal{T}_I^\ell \cap \mathcal{L}_I = \{t \in \mathcal{L}_I \,:\, \operatorname{level}(t) = \ell\}.$$

As an example for the usefulness of the concept of cluster levels, let us consider the proof of the following lemma:

Lemma 3.7 (Transitivity). *Let $t, s, r \in \mathcal{T}_I$ with $s \in \operatorname{sons}^*(t)$ and $r \in \operatorname{sons}^*(s)$. Then we have $r \in \operatorname{sons}^*(t)$.*

Proof. By induction on $\operatorname{level}(r) - \operatorname{level}(t)$.

For $\operatorname{level}(r) - \operatorname{level}(t) = 0$, Definition 3.5 yields $r = s = t$, so our claim is trivially fulfilled.

Let now $n \in \mathbb{N}_0$, and let us assume that $r \in \operatorname{sons}^*(t)$ holds for all $t, s, r \in \mathcal{T}_I$ with $s \in \operatorname{sons}^*(t)$, $r \in \operatorname{sons}^*(s)$ and $\operatorname{level}(r) - \operatorname{level}(t) = n$.

Let $t, s, r \in \mathcal{T}_I$ with $s \in \operatorname{sons}^*(t)$, $r \in \operatorname{sons}^*(s)$ and $\operatorname{level}(r) - \operatorname{level}(t) = n + 1$.

Case 1: $t = s$. This case is trivial, since we have $r \in \operatorname{sons}^*(s) = \operatorname{sons}^*(t)$.

Case 2: $t \neq s$. Definition 3.5 implies that there is a cluster $t' \in \operatorname{sons}(t)$ with $s \in \operatorname{sons}^*(t')$. Since $\operatorname{level}(t') = \operatorname{level}(t) + 1$, we have $\operatorname{level}(r) - \operatorname{level}(t') = n$ and can apply the induction assumption in order to prove that $r \in \operatorname{sons}^*(t') \subseteq \operatorname{sons}^*(t)$ holds, which concludes the induction. □

For the simple cluster tree used in the one-dimensional example, the relationships between clusters are fairly simple: on each level, the clusters describe a disjoint partition of $\mathcal{I}$, all clusters on level p are leaves, while all clusters on all other levels have exactly two sons. Since all of these properties are lost in the general case, we have to base proofs of complexity or error estimates on the following result:

Lemma 3.8 (Intersecting clusters). *Let $t, s \in \mathcal{T}_\mathcal{I}$ with $\hat{t} \cap \hat{s} \neq \emptyset$. Then the following statements holds:*

1. $\operatorname{level}(t) = \operatorname{level}(s)$ *implies* $t = s$.
2. $\operatorname{level}(t) \leq \operatorname{level}(s)$ *implies* $s \in \operatorname{sons}^*(t)$.

Proof. We prove

$$\operatorname{level}(t) = \operatorname{level}(s) \wedge \hat{t} \cap \hat{s} \neq \emptyset \implies t = s \tag{3.1}$$

for all $t, s \in \mathcal{T}_\mathcal{I}$ by induction.

If $\operatorname{level}(t) = \operatorname{level}(s) = 0$ holds, t and s are both the root of $\mathcal{T}_\mathcal{I}$, so they have to be identical.

Let now $m \in \mathbb{N}_0$ be such that (3.1) holds for all $t, s \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(t) = \operatorname{level}(s) = m$. Let $t, s \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(t) = \operatorname{level}(s) = m+1$ and $\hat{t} \cap \hat{s} \neq \emptyset$. Due to the definition of the level, there are uniquely defined clusters $t^+, s^+ \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(t^+) = \operatorname{level}(s^+) = m$ and $t \in \operatorname{sons}(t^+)$, $s \in \operatorname{sons}(s^+)$. Since $\hat{t}^+ \cap \hat{s}^+ \supseteq \hat{t} \cap \hat{s} \neq \emptyset$ holds, we can apply the induction assumption to prove $t^+ = s^+$, i.e., t and s are sons of the same father t^+. Since all sons of t^+ are disjoint by definition, $\hat{t} \cap \hat{s} \neq \emptyset$ implies $t = s$, which concludes the induction.

Let $t, s \in \mathcal{T}_\mathcal{I}$ with $\hat{t} \cap \hat{s} \neq \emptyset$ and $\operatorname{level}(t) \leq \operatorname{level}(s)$. The definition of the level implies that we can find a cluster $s^+ \in \operatorname{pred}(s)$ with $\operatorname{level}(s^+) = \operatorname{level}(t)$. Due to $\hat{t} \cap \hat{s}^+ \supseteq \hat{t} \cap \hat{s} \neq \emptyset$, we can apply the first statement of this lemma to find $t = s^+$. □

This lemma states that if the index sets corresponding to two clusters intersect, one of the clusters has to be a descendant of the other. A simple consequence of this result is that the leaves of a cluster tree form a disjoint partition of $\mathcal{I}$.

Corollary 3.9 (Leaf clusters). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree, and let $r \in \mathcal{T}_\mathcal{I}$. The set $\{\hat{t} : t \in \mathcal{L}_\mathcal{I} \cap \operatorname{sons}^*(r)\}$ is a disjoint partition of $\hat{r}$. As a special case, the set $\{\hat{t} : t \in \mathcal{L}_\mathcal{I}\}$ is a disjoint partition of $\mathcal{I}$.*

Proof. Let $t, s \in \mathcal{L}_\mathcal{I}$ with $\hat{t} \cap \hat{s} \neq \emptyset$. Due to Lemma 3.8, we have $s \in \operatorname{sons}^*(t)$ or $t \in \operatorname{sons}^*(s)$. Since t and s are leaves, $\operatorname{sons}^*(t) = \{t\}$ and $\operatorname{sons}^*(s) = \{s\}$ hold and we conclude $t = s$, which means that the index sets corresponding to leaf clusters are pairwise disjoint.

Let $i \in \hat{r}$. We have to find $t \in \operatorname{sons}^*(r) \cap \mathcal{L}_\mathcal{I}$ with $i \in \hat{t}$. Let us consider the set $\mathcal{C} := \{t \in \operatorname{sons}^*(r) : i \in \hat{t}\}$. It contains r and is therefore not empty. Since the cluster tree $\mathcal{T}_\mathcal{I}$ is finite, $\mathcal{C}$ contains only a finite number of elements, so there has to be a cluster $t \in \mathcal{C}$ with

$$\operatorname{level}(t) = \max\{\operatorname{level}(s) : s \in \mathcal{C}\}.$$

If $\operatorname{sons}(t) \neq \emptyset$ would hold, Definition 3.4 implies that we could find $s \in \operatorname{sons}(t)$ with $i \in \hat{s}$, i.e., $s \in \mathcal{C}$ with $\operatorname{level}(s) = \operatorname{level}(t) + 1$, which would contradict the maximality property of t. Therefore t has to be a leaf. □

Corollary 3.10 (Level partitions). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree and let $\ell \in \mathbb{N}_0$. The set $\{\hat{t} : t \in \mathcal{T}_\mathcal{I}^\ell\}$ is a disjoint partition of a subset of $\mathcal{I}$.*

Proof. Let $t, s \in \mathcal{T}_\mathcal{I}^\ell$. If $\hat{t} \cap \hat{s} \neq \emptyset$ holds, Lemma 3.8 implies $t = s$, therefore all elements of $\{\hat{t} : t \in \mathcal{T}_\mathcal{I}^\ell\}$ are disjoint. □

Since most of the complexity estimates in the following sections depend on characteristic quantities of cluster trees, we introduce the following notations:

Definition 3.11 (Cluster tree quantities). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree for the index set $\mathcal{I}$. We denote the number of indices by

$$n_\mathcal{I} := \#\mathcal{I}$$

and the number of clusters by

$$c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}.$$

For some estimates we also require the *depth*

$$p_\mathcal{I} := \max\{\operatorname{level}(t) : t \in \mathcal{T}_\mathcal{I}\}$$

of a cluster tree.

In the model case, we have $p = p_\mathcal{I} \leq q = \log_2 n$, $c_\mathcal{I} = 2^{p+1} - 1$ and for any $\ell \in \{0, \ldots, p\}$, $t \in \mathcal{T}_\mathcal{I}^\ell$ implies that we can find an integer $\alpha \in \{0, \ldots, 2^\ell - 1\}$ with $\hat{t} = \{\alpha 2^{q-\ell} + 1, \ldots, (\alpha + 1)2^{q-\ell}\}$.

3.2 Block cluster tree

Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be cluster trees for finite index sets $\mathcal{I}$ and $\mathcal{J}$.

Using these cluster trees, we can now proceed to define the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ which gives rise to the hierarchical block partitions used in constructing $\mathcal{H}$- and $\mathcal{H}^2$-matrices.

In the model case, all blocks correspond to square subblocks of the matrix, and a block is either a leaf or has exactly four descendants. These properties are lost in the general case, since clusters on the same level in the cluster tree can correspond to index sets of different sizes and since not all leaves of the cluster tree are on the same level.

Definition 3.12 (Block cluster tree). Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a labeled tree. $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is a *block cluster tree* for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ if it satisfies the following conditions:

- $\operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}) = (\operatorname{root}(\mathcal{T}_\mathcal{I}), \operatorname{root}(\mathcal{T}_\mathcal{J}))$.
- Each node $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ has the form $b = (t, s)$ for $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$ and its label satisfies $\hat{b} = \hat{t} \times \hat{s}$.

- Let $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. If $\operatorname{sons}(b) \neq \emptyset$, we have

$$\operatorname{sons}(b) = \begin{cases} \{t\} \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) = \emptyset, \operatorname{sons}(s) \neq \emptyset, \\ \operatorname{sons}(t) \times \{s\} & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) = \emptyset, \\ \operatorname{sons}(t) \times \operatorname{sons}(s) & \text{otherwise.} \end{cases} \tag{3.2}$$

The nodes $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ of a block cluster tree are called *blocks*.

In order to handle the three cases of (3.2) in a unified fashion, we introduce the following notation:

Definition 3.13 (Extended set of sons). Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree. We let

$$\operatorname{sons}^+(t) := \begin{cases} \operatorname{sons}(t) & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ \{t\} & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

We can see that (3.2) takes the short form $\operatorname{sons}(b) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$.

Lemma 3.14 (Cluster tree of blocks). *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Then $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is a cluster tree for the product index set $\mathcal{I} \times \mathcal{J}$.*

Proof. Let r be the root of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, let $r_{\mathcal{I}}$ and $r_{\mathcal{J}}$ be the roots of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. By definition, we have $r = (r_{\mathcal{I}}, r_{\mathcal{J}})$ and $\hat{r} = \hat{r}_{\mathcal{I}} \times \hat{r}_{\mathcal{J}} = \mathcal{I} \times \mathcal{J}$.

Let $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{sons}(b) \neq \emptyset$. This implies $\operatorname{sons}(b) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$.

By definition, we have

$$\hat{t} = \dot{\bigcup}_{t' \in \operatorname{sons}^+(t)} \hat{t}', \quad \hat{s} = \dot{\bigcup}_{s' \in \operatorname{sons}^+(s)} \hat{s}',$$

therefore we can conclude

$$\begin{aligned} \hat{b} = \hat{t} \times \hat{s} &= \Big(\dot{\bigcup}_{t' \in \operatorname{sons}^+(t)} \hat{t}'\Big) \times \Big(\dot{\bigcup}_{s' \in \operatorname{sons}^+(s)} \hat{s}'\Big) \\ &= \dot{\bigcup}_{(t',s') \in \operatorname{sons}(b)} \hat{t}' \times \hat{s}' = \dot{\bigcup}_{b' \in \operatorname{sons}(b)} \hat{b}'. \end{aligned}$$

□

Combining this lemma with Corollary 3.9 yields the following useful observation:

Corollary 3.15 (Block partition). *Let $b^* = (t^*, s^*) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Then*

$$\{\hat{b} = \hat{t} \times \hat{s} \ : \ b = (t, s) \in \operatorname{sons}^*(b^*) \cap \mathcal{L}_{\mathcal{I}\times\mathcal{J}}\} \tag{3.3}$$

is a disjoint partition of $\hat{b}^*$. As a special case,

$$\{\hat{b} = \hat{t} \times \hat{s} \ : \ b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}\} \tag{3.4}$$

is a disjoint partition of $\mathcal{I} \times \mathcal{J}$.

Proof. Due to Lemma 3.14, $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is a cluster tree for the index set $\mathcal{I}\times\mathcal{J}$. Applying Corollary 3.9 concludes the proof of the first claim. In order to prove the second claim, we let $b^* = \operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}})$ and observe $\operatorname{sons}^*(b^*) \cap \mathcal{L}_{\mathcal{I}\times\mathcal{J}} = \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$. □

In short, the partition (3.4) corresponds to a decomposition of matrices in $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ into non-overlapping submatrices.

Among the elements of this partition, we now have to distinguish admissible and inadmissible subblocks, since we can apply low-rank approximation schemes only to admissible submatrices.

Definition 3.16 (Admissibility condition). Let $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ be cluster trees. A predicate

$$\mathcal{A} : \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{J}} \to \{\text{true}, \text{false}\}$$

is an *admissibility condition* for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ if

$$\mathcal{A}(t,s) \Rightarrow \mathcal{A}(t',s) \quad \text{holds for all } t \in \mathcal{T}_{\mathcal{I}}, s \in \mathcal{T}_{\mathcal{J}}, t' \in \operatorname{sons}(t)$$

and

$$\mathcal{A}(t,s) \Rightarrow \mathcal{A}(t,s') \quad \text{holds for all } t \in \mathcal{T}_{\mathcal{I}}, s \in \mathcal{T}_{\mathcal{J}}, s' \in \operatorname{sons}(s).$$

If $\mathcal{A}(t,s) = \text{true}$, the pair (t,s) is called *admissible*.

In the model case, the clusters $t, s \in \mathcal{T}_{\mathcal{I}}$ are intervals, and we can use (2.6) as an admissibility condition:

$$\mathcal{A}_{1d}(t,s) := \begin{cases} \text{true} & \text{if } \operatorname{diam}(t) + \operatorname{diam}(s) \leq 2\eta \operatorname{dist}(t,s), \\ \text{false} & \text{otherwise,} \end{cases} \tag{3.5}$$

for all intervals $t, s \in \mathcal{T}_{\mathcal{I}}$ and the admissibility parameter $\eta \in \mathbb{R}_{>0}$. Since $t' \subseteq t$ holds for all $t' \in \operatorname{sons}(t)$ and $s' \subseteq s$ holds for all $s' \in \operatorname{sons}(s)$, $\mathcal{A}_{1d}$ satisfies the conditions of Definition 3.16.

Remark 3.17 (Weak admissibility condition). In certain situations, we can replace (3.5) by the *weak admissibility condition*

$$\mathcal{A}_{1d,w}(t,s) := \begin{cases} \text{true} & \text{if } t \neq s, \\ \text{false} & \text{otherwise,} \end{cases}$$

investigated in [69]. This condition leads to the simple matrix structure investigated in the first paper [62] on hierarchical matrices. It can be proven that this structure is well-suited for treating certain one-dimensional problems [59], [96] and yields acceptable results for some essentially one-dimensional integral equations [84]. In the case of integral equations, experiments [69] indicate that the original admissibility condition leads to more robust approximation properties, while the weak admissibility condition offers algorithmic advantages. □

Using an admissibility condition, we can split the subblocks in the set (3.4) into admissible and inadmissible ones. The admissible leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ can be represented by a suitable factorization and can therefore be handled efficiently. The inadmissible leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ will be stored as dense matrices, therefore we have to ensure that these matrices are not too large. A simple way of doing this is to require all inadmissible leaves of the block cluster tree to be formed from leaves of the cluster tree, since these can be assumed to correspond only to small sets of indices.

Definition 3.18 (Admissible block cluster tree). Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, and let $\mathcal{A}$ be an admissibility condition. If for each $(t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ either $\operatorname{sons}(t) = \emptyset = \operatorname{sons}(s)$ or $\mathcal{A}(t,s) = \text{true}$ holds, the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called *$\mathcal{A}$-admissible*.

Definition 3.19 (Farfield and nearfield). Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, and let $\mathcal{A}$ be an admissibility condition for these cluster trees. The set

$$\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} := \{(t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \ : \ \mathcal{A}(t,s) = \text{true}\}$$

is called the set of *farfield* blocks. The set

$$\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} := \{(t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \ : \ \mathcal{A}(t,s) = \text{false}\}$$

is called the set of *nearfield* blocks. Obviously, the labels of the pairs in $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ form a disjoint partition of the labels of the pairs in $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.

From now on, we assume that each block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is implicitly associated with an admissibility condition $\mathcal{A}$ and that the sets $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ are defined with respect to this condition.

3.3 Construction of cluster trees and block cluster trees

Before we proceed to consider $\mathcal{H}$- and $\mathcal{H}^2$-matrices based on general cluster trees and block cluster trees, let us briefly discuss the construction of these trees in practical applications.

The problem of constructing a "good" cluster tree for an index set $\mathcal{I}$ can be approached in different ways: if the index set is already equipped with a hierarchy, e.g., if it is the result of the application of a refinement strategy to an initial coarse discretization, we can use the given hierarchy. If the index set corresponds to a quasi-uniform discretization, it is possible to construct cluster trees and block cluster trees by recursive binary splittings of the underlying domain, as demonstrated in [90], [91] and [92], Section 7.4.1. Even in the case of non-uniform discretizations, it is possible to find cluster trees and prove that algorithms based on these trees are efficient [52].

Binary space partitioning

Here we only introduce relatively simple, yet quite general, algorithms for the construction of suitable cluster and block cluster trees. The theoretical investigation of these algorithms is not the topic of this book, we only use them as the foundation for numerical experiments.

Our construction is based on the assumption that the index set $\mathcal{I}$ corresponds to a family of subsets of a domain or manifold $\Omega \subseteq \mathbb{R}^d$.

As a typical example let us consider the discretization of a differential or integral equation: each $i \in \mathcal{I}$ corresponds to a basis function or test functional, and, as in the model case, the success of an approximation is determined by the quality of the approximation on the *support* of this function or functional. As in the model case described in Chapter 2, we base general clustering strategies only on the supports of these functions or functionals and neglect their other features.

Definition 3.20 (Support). A family $(\Omega_i)_{i\in\mathcal{I}}$ of subsets of Ω is called a *family of supports* for $\mathcal{I}$. Given such a family, the set Ω_i is called the *support* of i for all $i \in \mathcal{I}$.

If a cluster tree $\mathcal{T}_\mathcal{I}$ for $\mathcal{I}$ is given, we define the *support* of a cluster $t \in \mathcal{T}_\mathcal{I}$ by

$$\Omega_t := \bigcup_{i\in\hat{t}} \Omega_i. \tag{3.6}$$

In standard applications, e.g., for finite element or boundary element discretizations, the supports Ω_i will be small, i.e., their diameters will be proportional to the meshwidth of the underlying grid. Therefore it makes sense to avoid the necessity of handling possibly complicated subdomains Ω_i by picking a single point $x_i \in \Omega_i$ for each $i \in \mathcal{I}$ and basing constructive algorithms on these points instead of the original subdomains Ω_i.

Definition 3.21 (Characteristic point). A family $(x_i)_{i\in\mathcal{I}}$ of points in Ω is called a *family of characteristic points* for $\mathcal{I}$. Given such a family, the point x_i is called the *characteristic point* of i for all $i \in \mathcal{I}$.

If a family of supports $(\Omega_i)_{i\in\mathcal{I}}$ satisfies $x_i \in \Omega_i$ for all $i \in \mathcal{I}$, we say that the characteristic points $(x_i)_{i\in\mathcal{I}}$ *match* the family of supports $(\Omega_i)_{i\in\mathcal{I}}$.

Using families of characteristic points, a wide variety of practical constructions for cluster trees can be investigated. A fairly general approach, which guarantees that the resulting cluster trees and block cluster trees lead to efficient algorithms for hierarchical matrices, is described in [52].

We restrict our attention to two simple constructions of cluster trees, which are both based on subdivisions of the "cloud" of characteristic points in d-dimensional space. The basic idea is to assign each cluster t an axis-parallel box

$$\mathcal{B}_t = [a_{t,1}, b_{t,1}] \times \cdots \times [a_{t,d}, b_{t,d}]$$

containing all corresponding characteristic points, i.e., satisfying

$$x_i \in \mathcal{B}_t \quad \text{for all } i \in \hat{t}. \tag{3.7}$$

If we decide that t should not be a leaf of the cluster tree, e.g., if the cardinality $\#\hat{t}$ is not small enough, we have to construct sons of t. We do this by picking a coordinate direction $\iota \in \{1, \ldots, d\}$, fixing the midpoint

$$m_\iota := \frac{b_{t,\iota} + a_{t,\iota}}{2}$$

of the corresponding interval $[a_{t,\iota}, b_{t,\iota}]$, and sorting the indices of $\hat{t}$ into "left" and "right" portions

$$\hat{t}_1 := \{i \in \hat{t} \,:\, x_{i,\iota} < m_\iota\}, \quad \hat{t}_2 := \{i \in \hat{t} \,:\, x_{i,\iota} \geq m_\iota\}.$$

Due to $\hat{t} = \hat{t}_1 \dot{\cup} \hat{t}_2$, we can use these sets to define sons t_1 and t_2 and use

$$\operatorname{sons}(t) := \begin{cases} \{t_1\} & \text{if } \hat{t}_1 \neq \emptyset, \hat{t}_2 = \emptyset, \\ \{t_2\} & \text{if } \hat{t}_1 = \emptyset, \hat{t}_2 \neq \emptyset, \\ \{t_1, t_2\} & \text{otherwise.} \end{cases}$$

In order to be able to proceed by recursion, we need boxes $\mathcal{B}_{t_1}$ and $\mathcal{B}_{t_2}$ for the new sons. Due to our construction, the boxes

$$\begin{aligned} \mathcal{B}_{t_1} &:= [a_{t,1}, b_{t,1}] \times \cdots \times [a_{t,\iota-1}, b_{t,\iota-1}] \\ &\quad \times [a_{t,\iota}, m_\iota] \times [a_{t,\iota+1}, b_{t,\iota+1}] \times \cdots \times [a_{t,d}, b_{t,d}], \\ \mathcal{B}_{t_2} &:= [a_{t,1}, b_{t,1}] \times \cdots \times [a_{t,\iota-1}, b_{t,\iota-1}] \\ &\quad \times [m_\iota, b_{t,\iota}] \times [a_{t,\iota+1}, b_{t,\iota+1}] \times \cdots \times [a_{t,d}, b_{t,d}] \end{aligned}$$

are a simple choice and satisfy (3.7). We also need coordinate directions $\iota_1, \iota_2 \in \{1, \ldots, d\}$ to be used for splitting these boxes. A simple approach is to cycle through all possible directions using

$$\iota_1 = \iota_2 = \begin{cases} \iota + 1 & \text{if } \iota < d, \\ 1 & \text{otherwise.} \end{cases}$$

This approach guarantees that the characteristic boxes on level $\ell + d$ of the cluster tree will be similar to the boxes on level ℓ, only scaled by $1/2$ and shifted. Due to this self-similarity property, we call this the "geometrically regular" clustering strategy. It is summarized in Algorithm 1. Figure 3.4 illustrates the procedure for the two-dimensional case: a (not necessarily optimal) axis-parallel box is split into two halves, and the set of indices is subdivided accordingly, creating two sons. The "left" son is not small enough, so it is split again.

Algorithm 1. Create a geometrically regular cluster tree.

function ConstructRegularTree($\hat{t}$, $\mathcal{B}_t$, ι) : **cluster**;
Create a new cluster t with index set $\hat{t}$; sons(t) $\leftarrow \emptyset$;
$[a_{t,1}, b_{t,1}] \times \cdots \times [a_{t,d}, b_{t,d}] \leftarrow \mathcal{B}_t$;
if $\#\hat{t}$ not small enough **then**
  $m_\iota \leftarrow (a_{t,\iota} + b_{t,\iota})/2$;
  $\hat{t}_1 \leftarrow \emptyset$; $\hat{t}_2 \leftarrow \emptyset$;
  for $i \in \hat{t}$ **do**
    if $x_{i,\iota} < m_\iota$ **then**
      $\hat{t}_1 \leftarrow \hat{t}_1 \cup \{i\}$
    else
      $\hat{t}_2 \leftarrow \hat{t}_2 \cup \{i\}$
    end if
  end for;
  $\mathcal{B}_{t_1} \leftarrow \{x \in \mathcal{B}_t \ : \ x_\iota \leq m\}$; $\mathcal{B}_{t_2} \leftarrow \{x \in \mathcal{B}_t \ : \ x_\iota \geq m\}$;
  if $\iota < d$ **then**
    $\iota' \leftarrow \iota + 1$
  else
    $\iota' \leftarrow 1$
  end if;
  if $\hat{t}_1 \neq \emptyset$ **then**
    $t_1 \leftarrow$ ConstructRegularTree($\hat{t}_1$, $\mathcal{B}_{t_1}$, ι');
    sons(t) $\leftarrow$ sons(t) $\cup \{t_1\}$
  end if
  if $\hat{t}_2 \neq \emptyset$ **then**
    $t_2 \leftarrow$ ConstructRegularTree($\hat{t}_2$, $\mathcal{B}_{t_2}$, ι');
    sons(t) $\leftarrow$ sons(t) $\cup \{t_2\}$
  end if
end if;
return t

For anisotropic situations, e.g., if the width and height of the boxes differ significantly, it is desirable to split the boxes in the direction of maximal extent, i.e., to choose the direction $\iota \in \{1, \ldots, d\}$ with

$$b_{t,\iota} - a_{t,\iota} = \max\{b_{t,\kappa} - a_{t,\kappa} \ : \ \kappa \in \{1, \ldots, d\}\}.$$

By this technique, we can ensure that the splitting strategy tries to equilibrate the dimensions of the boxes, i.e., to minimize their diameters. This is a very useful feature, since most admissibility criteria (cf. (2.6)) are based on diameters and distances, and clusters with small diameters are more likely to be admissible than those with large diameters. In order to reduce the diameters even further, we can recompute the boxes $\mathcal{B}_t$ in each step in order to ensure that they are minimal. As a desirable side-effect,

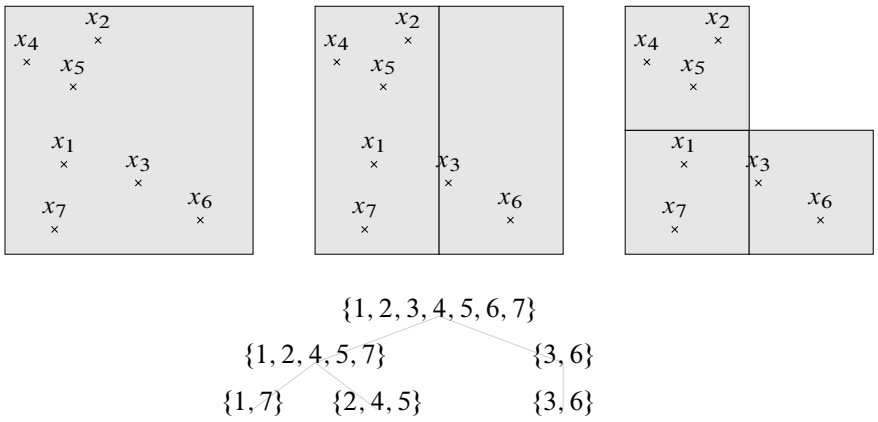


Figure 3.4. Two steps of the construction of a geometrically regular cluster tree for characteristic points in two spatial dimensions.

this will also ensure that splitting a cluster will create two sons as long as the cluster is not trivial, i.e., as long as we can find $i, j \in \hat{t}$ with $x_i \neq x_j$. This approach is called "geometrically balanced" clustering. The resulting procedure is given in Algorithm 2.

All clusters t in a cluster tree constructed by the geometrically balanced clustering Algorithm 2 will satisfy either $\#\operatorname{sons}(t) = 0$ or $\#\operatorname{sons}(t) = 2$. This fact allows us to bound the number of clusters:

Lemma 3.22 (Number of clusters). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree constructed by Algorithm 2. Then the number of clusters is bounded by*

$$c_\mathcal{I} \leq 2n_\mathcal{I} - 1,$$

with $n_\mathcal{I} = \#\mathcal{I}$ and $c_\mathcal{I} = \#\mathcal{T}_\mathcal{I}$ as in Definition 3.11.

Proof. Due to $\mathcal{I} \neq \emptyset$, Algorithm 2 ensures that $\hat{t} \neq \emptyset$ holds for all $t \in \mathcal{T}_\mathcal{I}$. According to Corollary 3.9, the index sets $\hat{t}$ of all leaf clusters $t \in \mathcal{L}_\mathcal{I}$ are disjoint, so there cannot be more than $n_\mathcal{I}$ leaf clusters. We can conclude the proof by using Lemma 3.52. □

Remark 3.23 (Improved bound). Under certain conditions we can ensure that all leaves of the cluster tree have a minimal size of m, i.e., we have

$$\#\hat{t} \geq m$$

for all leaf clusters $t \in \mathcal{L}_\mathcal{I}$. According to Corollary 3.9, all leaves correspond to disjoint index sets, and since all of these sets contain at least m indices, there cannot be more than $n_\mathcal{I}/m$ leaves. Applying Lemma 3.52 yields the bound

$$c_\mathcal{I} \leq 2\frac{n_\mathcal{I}}{m} - 1$$

for the number of clusters. This is similar to the result (2.23) for the model case. □

Algorithm 2. Create a geometrically balanced cluster tree.

```
function ConstructGeometricTree(t̂) : cluster;
Create a new cluster t;    sons(t) ← ∅;
if #t̂ not small enough then
  for ι ∈ {1, ..., d} do
    a_{t,ι} ← min{x_{i,ι} : i ∈ t̂};
    b_{t,ι} ← max{x_{i,ι} : i ∈ t̂};
  end for
  ι ← 1;
  for κ ∈ {2, ..., d} do
    if b_{t,κ} − a_{t,κ} > b_{t,ι} − a_{t,ι} then
      ι ← κ
    end if
  end for;
  if b_{t,ι} − a_{t,ι} > 0 then
    m_ι ← (a_{t,ι} + b_{t,ι})/2;
    t̂_1 ← ∅; t̂_2 ← ∅;
    for i ∈ t̂ do
      if x_{i,ι} < m then
        t̂_1 ← t̂_1 ∪ {i}
      else
        t̂_2 ← t̂_2 ∪ {i}
      end if
    end for;
    t_1 ← ConstructGeometricTree(t̂_1);
    t_2 ← ConstructGeometricTree(t̂_2);
    sons(t) ← {t_1, t_2}
  end if
end if;
return t
```

Block cluster tree

Once we have found cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ for the index sets $\mathcal{I}$ and $\mathcal{J}$, the next step is to construct an admissible block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

If an admissibility condition $\mathcal{A}$ is given, Definitions 3.12 and 3.18 suggest a simple recursive algorithm: according to Definition 3.12, the root of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is given by $(\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}))$. Due to Definition 3.18, a block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ can only be a leaf if it is admissible or if t and s are leaves of the respective cluster trees. If the block is not a leaf, Definition 3.12 uniquely defines its sons, and we can proceed by recursion. Algorithm 3 summarizes the resulting construction of a minimal $\mathcal{A}$-admissible block cluster tree: if called with $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$ and $s = \operatorname{root}(\mathcal{T}_{\mathcal{J}})$, it will construct the minimal

$\mathcal{A}$-admissible block cluster tree and return its root block.

Algorithm 3. Create a minimal $\mathcal{A}$-admissible block cluster tree.

```
function ConstructBlockClusterTree(t, s) : block;
Create a new block b := (t, s);
b̂ ← t̂ × ŝ;
sons(b) ← ∅;
if A(t, s) then
  L⁺_{I×J} ← L⁺_{I×J} ∪ {b}                          {Admissible leaf}
else if sons(t) = ∅ then
  if sons(s) = ∅ then
    L⁻_{I×J} ← L⁻_{I×J} ∪ {b}                        {Inadmissible leaf}
  else
    for s' ∈ sons(s) do
      b' ← ConstructBlockClusterTree(t, s');
      sons(b) ← sons(b) ∪ {b'}
    end for
  end if
else
  if sons(s) = ∅ then
    for t' ∈ sons(t) do
      b' ← ConstructBlockClusterTree(t', s);
      sons(b) ← sons(b) ∪ {b'}
    end for
  else
    for t' ∈ sons(t), s' ∈ sons(s) do
      b' ← ConstructBlockClusterTree(t', s');
      sons(b) ← sons(b) ∪ {b'}
    end for
  end if
end if;
return b
```

The efficiency of Algorithm 3 is determined by the algorithmic complexity of the test for $\mathcal{A}$-admissibility. We now examine two situations in which this test can be performed efficiently.

Admissibility condition for spherical domains

In applications based on Taylor expansions or spherical harmonics, we will usually encounter admissibility conditions of the type

$$\mathcal{A}(t,s) := \begin{cases} \text{true} & \text{if } \max\{\operatorname{diam}(\mathcal{K}_t), \operatorname{diam}(\mathcal{K}_s)\} \leq 2\eta \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s), \\ \text{false} & \text{otherwise,} \end{cases}$$

where $\mathcal{K}_t$ and $\mathcal{K}_s$ are d-dimensional balls satisfying $\Omega_t \subseteq \mathcal{K}_t$ and $\Omega_s \subseteq \mathcal{K}_s$ and $\eta \in \mathbb{R}_{>0}$ is a parameter. If the ball $\mathcal{K}_t$ corresponding to $t \in \mathcal{T}_{\mathcal{I}}$ is described by its center $c_t \in \mathbb{R}^d$ and its radius $r_t \in \mathbb{R}_{\geq 0}$, the test for admissibility is straightforward: we have $\operatorname{diam}(\mathcal{K}_t) = 2r_t$ and

$$\operatorname{diam}(\mathcal{K}_t) = 2r_t, \quad \operatorname{diam}(\mathcal{K}_s) = 2r_s,$$

$$\operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) = \begin{cases} \|c_t - c_s\|_2 - r_t - r_s & \text{if } \|c_t - c_s\|_2 \geq r_t + r_s, \\ 0 & \text{otherwise.} \end{cases}$$

If the cluster tree has been constructed by Algorithms 1 or 2, the construction of the balls $\mathcal{K}_t$ for all $t \in \mathcal{T}_{\mathcal{I}}$ can be handled by a simple recursion: if $\#\operatorname{sons}(t) = 0$, i.e., if t is a leaf, the computation of a suitable center c_t and radius r_t depends on the application, e.g., on the underlying discretization scheme. If $\#\operatorname{sons}(t) = 1$, we let $t' \in \operatorname{sons}(t)$ and observe $\hat{t} = \hat{t}'$, therefore we can use $\mathcal{K}_t := \mathcal{K}_{t'}$, i.e., $c_t := c_{t'}$ and $r_t := r_{t'}$.

Otherwise, i.e., if $\#\operatorname{sons}(t) = 2$, we pick $t_1, t_2 \in \operatorname{sons}(t)$ with $\operatorname{sons}(t) = \{t_1, t_2\}$. If $\mathcal{K}_{t_2} \subseteq \mathcal{K}_{t_1}$, we use $\mathcal{K}_t = \mathcal{K}_{t_1}$, i.e., $c_t := c_{t_1}$ and $r_t := r_{t_1}$. If $\mathcal{K}_{t_1} \subseteq \mathcal{K}_{t_2}$, we use $\mathcal{K}_t = \mathcal{K}_{t_2}$, i.e., $c_t := c_{t_2}$ and $r_t := r_{t_2}$. Otherwise, we are in the situation depicted in Figure 3.5.

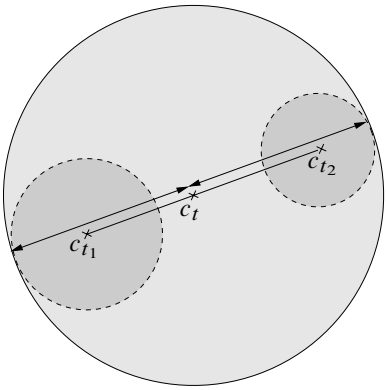


Figure 3.5. Covering two discs by one larger disc.

The optimal point c_t is characterized by the fact that it minimizes the radius r_t required to ensure that $\mathcal{K}_t$ covers $\mathcal{K}_{t_1}$ and $\mathcal{K}_{t_2}$, i.e., that the distance of c_t to the

boundaries of $\mathcal{K}_{t_1}$ and $\mathcal{K}_{t_2}$ is minimal. This leads to the problem of finding a point c_t which minimizes

$$r_t = \max\{\|c_t - c_{t_1}\|_2 + r_{t_1}, \|c_t - c_{t_2}\|_2 + r_{t_2}\}.$$

We can see that the center c_t of the optimal ball $\mathcal{K}_t$ has to be situated on the line connecting c_{t_1} and c_{t_2}, i.e.,

$$c_t = (1-\lambda)c_{t_1} + \lambda c_{t_2} \tag{3.8}$$

has to hold for a parameter $\lambda \in [0,1]$, so the minimization problem now takes the form

$$r_t = \max\{\lambda\|c_{t_2} - c_{t_1}\|_2 + r_{t_1}, (1-\lambda)\|c_{t_2} - c_{t_1}\|_2 + r_{t_2}\}$$

and we find that the minimum is determined by

$$\lambda\|c_{t_2} - c_{t_1}\|_2 + r_{t_1} = (1-\lambda)\|c_{t_2} - c_{t_1}\|_2 + r_{t_2}.$$

This equation implies

$$2\lambda\|c_{t_2} - c_{t_1}\|_2 = \|c_{t_2} - c_{t_1}\|_2 + r_{t_2} - r_{t_1}. \tag{3.9}$$

Since neither $\mathcal{K}_{t_2} \subseteq \mathcal{K}_{t_1}$ nor $\mathcal{K}_{t_1} \subseteq \mathcal{K}_{t_2}$ hold, we have $c_{t_2} \neq c_{t_1}$ and

$$\frac{|r_{t_2} - r_{t_1}|}{\|c_{t_2} - c_{t_1}\|_2} < 1,$$

so the equation (3.9) has the unique solution

$$\lambda := \frac{1}{2}\left(1 + \frac{r_{t_2} - r_{t_1}}{\|c_{t_2} - c_{t_1}\|_2}\right).$$

Combining this choice of λ with (3.8) yields the recursive Algorithm 4 for constructing $\mathcal{K}_t$ for all $t \in \mathcal{T}_I$.

If an efficient method for computing good covering balls $\mathcal{K}_t$ for the leaves $t \in \mathcal{L}_I$ is available (since $\#\hat{t}$ is assumed to be small in this case, only a small number of supports has to be taken into account), Algorithm 4 will also be very efficient.

Admissibility condition for rectangular domains

Let us now consider a second admissibility condition, typically used by applications based on tensor-product interpolation. Here, we assume that we have axis-parallel *bounding boxes* $\mathcal{Q}_t$ and $\mathcal{Q}_s$ satisfying

$$\Omega_t \subseteq \mathcal{Q}_t, \quad \Omega_s \subseteq \mathcal{Q}_s \quad \text{for all } t \in \mathcal{T}_I, s \in \mathcal{T}_J \tag{3.10}$$

Algorithm 4. Construct covering balls for cluster supports.

procedure ConstructCoveringBalls(t);
if $\#\operatorname{sons}(t) = 0$ **then**
  Compute c_t and r_t such that the corresponding ball $\mathcal{K}_t$ contains Ω_t
else if $\#\operatorname{sons}(t) = 1$ **then**
  Pick $t_1 \in \operatorname{sons}(t)$;
  ConstructCoveringBalls(t_1);
  $c_t \leftarrow c_{t_1}$; $r_t \leftarrow r_{t_1}$
else
  Pick $t_1, t_2 \in \operatorname{sons}(t)$ with $\operatorname{sons}(t) = \{t_1, t_2\}$;
  ConstructCoveringBalls(t_1); ConstructCoveringBalls(t_2);
  $\lambda \leftarrow \frac{1}{2}\left(1 + \frac{r_{t_2} - r_{t_1}}{\|c_{t_2} - c_{t_1}\|_2}\right)$;
  $c_t \leftarrow (1-\lambda)c_{t_1} + \lambda c_{t_2}$; $r_t \leftarrow \lambda\|c_{t_2} - c_{t_1}\|_2 + r_{t_1}$
end if

at our disposal, and use

$$\mathcal{A}(t,s) := \begin{cases} \text{true} & \text{if } \max\{\operatorname{diam}(\mathcal{Q}_t), \operatorname{diam}(\mathcal{Q}_s)\} \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s), \\ \text{false} & \text{otherwise} \end{cases}$$

to determine whether a block (t,s) is admissible. Here $\eta \in \mathbb{R}_{>0}$ is an additional parameter controlling the strictness of the admissibility condition (cf. (2.6)). If the boxes $\mathcal{Q}_t$ and $\mathcal{Q}_s$ are described by

$$\mathcal{Q}_t = [a_{t,1}, b_{t,1}] \times \cdots \times [a_{t,d}, b_{t,d}], \quad \mathcal{Q}_s = [a_{s,1}, b_{s,1}] \times \cdots \times [a_{s,d}, b_{s,d}],$$

the test for admissibility can be handled efficiently due to

$$\operatorname{diam}(\mathcal{Q}_t) = \left(\sum_{\iota=1}^{d} \operatorname{diam}^2([a_{t,\iota}, b_{t,\iota}])\right)^{1/2},$$

$$\operatorname{diam}(\mathcal{Q}_s) = \left(\sum_{\iota=1}^{d} \operatorname{diam}^2([a_{s,\iota}, b_{s,\iota}])\right)^{1/2},$$

$$\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s) = \left(\sum_{\iota=1}^{d} \operatorname{dist}^2([a_{t,\iota}, b_{t,\iota}], [a_{s,\iota}, b_{s,\iota}])\right)^{1/2}.$$

The parameters of the optimal bounding box $\mathcal{Q}_t$ are determined by

$$a_{t,\iota} := \inf\{x_\iota \,:\, x \in \Omega_t\}, \quad b_{t,\iota} := \sup\{x_\iota \,:\, x \in \Omega_t\} \quad \text{for } \iota \in \{1, \ldots, d\}.$$

As in the case of covering balls, we can compute the optimal bounding boxes by a recursion.

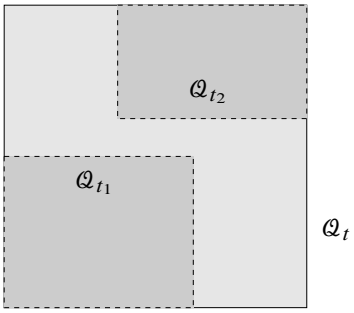


Figure 3.6. Covering two boxes by one large box.

The computation of $\mathcal{Q}_t$ for leaf clusters $t \in \mathcal{L}_\mathcal{I}$ depends on the application, but we can assume that any reasonable data structure allows us to find minimal and maximal coordinates efficiently, especially since only a small number of supports has to be considered due to our assumption that $\#\hat{t}$ is small for leaf clusters.

Let now $t \in \mathcal{T}_\mathcal{I}$ be a cluster with $\operatorname{sons}(t) \neq \emptyset$. Definitions 3.4 and 3.20 imply

$$\Omega_t = \bigcup_{t' \in \operatorname{sons}(t)} \Omega_{t'},$$

and we find

$$a_{t,\iota} = \inf\{x_\iota : x \in \Omega_t\} = \min_{t' \in \operatorname{sons}(t)} \inf\{x_\iota : x \in \Omega_{t'}\} = \min_{t' \in \operatorname{sons}(t)} a_{t',\iota},$$
$$b_{t,\iota} = \sup\{x_\iota : x \in \Omega_t\} = \max_{t' \in \operatorname{sons}(t)} \sup\{x_\iota : x \in \Omega_{t'}\} = \max_{t' \in \operatorname{sons}(t)} b_{t',\iota}.$$

This suggests the simple recursive Algorithm 5.

As in the case of Algorithm 4, the efficiency of Algorithm 5 depends mainly on the efficiency of the algorithm used to find good bounding boxes for leaf clusters.

3.4 Hierarchical matrices

Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{A}$-admissible block cluster tree. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. As in the model case, we use cut-off matrices in order to restrict matrices to submatrices related to single clusters or blocks:

Definition 3.24 (Cut-off matrices). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree. For all $t \in \mathcal{T}_\mathcal{I}$, the *cut-off matrix* $\chi_t \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ corresponding to t is defined by

$$(\chi_t)_{ij} := \begin{cases} 1 & \text{if } i = j \in \hat{t}, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for all } i, j \in \mathcal{I}.$$

Algorithm 5. Construct bounding boxes for cluster supports.

```
procedure ConstructBoundingBoxes(t);
if #sons(t) = 0 then
  for ι ∈ {1,...,d} do
    a_{t,ι} ← inf{x_ι : x ∈ Ω_t}; b_{t,ι} ← sup{x_ι : x ∈ Ω_t}
  end for
else
  for t' ∈ sons(t) do
    ConstructBoundingBoxes(t')
  end for;
  for ι ∈ {1,...,d} do
    a_{t,ι} ← min{a_{t',ι} : t' ∈ sons(t)}; b_{t,ι} ← max{b_{t',ι} : t' ∈ sons(t)}
  end for
end if
```

The cut-off matrices correspond to subspaces of vectors and matrices which vanish on subsets of the index sets.

Definition 3.25 (Restricted spaces). Let $\mathcal{I}' \subseteq \mathcal{I}$, let $\mathcal{J}' \subseteq \mathcal{J}$, let K be a finite index set. We define

$$\mathbb{R}^{\mathcal{I}\times K}_{\mathcal{I}'} := \{X \in \mathbb{R}^{\mathcal{I}\times K} \ : \ X_{ik} = 0 \text{ for all } i \in \mathcal{I} \setminus \mathcal{I}' \text{ and } k \in K\},$$
$$\mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}',\mathcal{J}'} := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \ : \ X_{ij} = 0 \text{ for all } i \in \mathcal{I} \setminus \mathcal{I}' \text{ and } j \in \mathcal{J} \setminus \mathcal{J}'\},$$

i.e., $\mathbb{R}^{\mathcal{I}\times K}_{\mathcal{I}'}$ contains all matrices which are zero outside of the rows corresponding to $\mathcal{I}'$, and $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}',\mathcal{J}'}$ contains all matrices which are zero outside of the block $\mathcal{I}' \times \mathcal{J}'$.

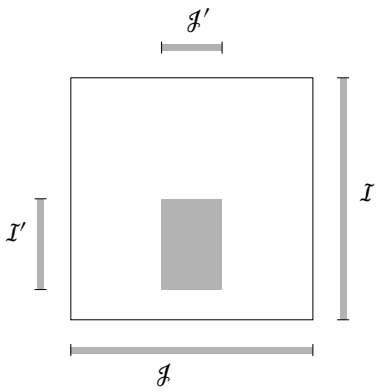


Figure 3.7. Matrix in $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}',\mathcal{J}'}$.

The cut-off matrices χ_t provide us with a purely algebraic characterization of matrices in the restricted spaces:

Remark 3.26 (Restricted spaces). Let K be a finite index set, let $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$. Let $X \in \mathbb{R}^{\mathcal{I}\times K}$ and $N \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. We have $X \in \mathbb{R}^{\mathcal{I}\times K}_{\hat{t}}$ if and only if $X = \chi_t X$ holds, and we have $N \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\hat{t},\hat{s}}$ if and only if $N = \chi_t N \chi_s$ holds.

Proof. Combine Definitions 3.25 and 3.24. □

A generalization of (2.12) is provided by Corollary 3.15: we can split the matrix X into submatrices

$$X = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{3.11}$$

Remark 3.26 implies $\chi_t X \chi_s \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\hat{t},\hat{s}}$ for all blocks $b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, therefore each of these blocks requires only $(\#\hat{t})(\#\hat{s})$ units of storage. As in the model case, we are looking for a more efficient representation of the admissible leaves.

In order to define a general hierarchical matrix, we follow the approach suggested by (2.14) and represent each admissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ in a factorized form using "slim" matrices, i.e., matrices with only a small number of columns.

Definition 3.27 (Hierarchical matrix). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree, and let $(K_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be a family of finite index sets. X is a *hierarchical matrix* (or short $\mathcal{H}$*-matrix*) if for each $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ there are $A_b \in \mathbb{R}^{\mathcal{I}\times K_b}_{\hat{t}}$ and $B_b \in \mathbb{R}^{\mathcal{J}\times K_b}_{\hat{s}}$ with

$$\chi_t X \chi_s = A_b B_b^*.$$

The quantity

$$k := \max\{\#K_b \;:\; b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}$$

is called the *local rank* of the $\mathcal{H}$-matrix.

For an $\mathcal{H}$-matrix, equation (3.11) implies

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} A_b B_b^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s, \tag{3.12}$$

and we call this the $\mathcal{H}$*-matrix representation* of X.

Usually hierarchical matrices [62], [19], [18], [63] are defined by requiring that the ranks of submatrices corresponding to admissible blocks are bounded. This approach coincides with Definition 3.27:

Remark 3.28 (Alternative definition). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree, and let

$$\operatorname{rank}(\chi_t X \chi_s) \leq k$$

hold for all $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $X_b := \chi_t X \chi_s$. The definition of the rank of a matrix implies that the dimension k_b of the range of X_b is bounded by k, therefore we can find an orthonormal basis of range(X_b) consisting of k_b vectors. We use these vectors as columns of a matrix $V_b \in \mathbb{R}^{\hat{t}\times k_b}$ satisfying

$$V_b^* V_b = I.$$

Let now $x \in \operatorname{range}(X_b)$. Due to $\operatorname{range}(X_b) = \operatorname{range}(V_b)$ we can find $y \in \mathbb{R}^{k_b}$ with $x = V_b y$ and get

$$V_b V_b^* x = V_b V_b^* V_b y = V_b y = x,$$

i.e., $V_b V_b^*$ is a projection into range(X_b). This means

$$X_b = V_b V_b^* X_b = V_b (X_b^* V_b)^* = A_b B_b^*$$

for $A_b = V_b$ and $B_b = X_b^* V_b$. This is the factorized representation required by Definition 3.27, therefore X is an $\mathcal{H}$-matrix. □

Now let us consider the storage requirements of hierarchical matrices. In the model case, we can compute the numbers of admissible and inadmissible blocks per level explicitly, and this makes computing the storage requirements a simple task.

In the general case, the level of the block cluster tree is no longer connected to the size of its blocks, therefore we need an approach that allows us to take care of different block sizes. Although the size of a block $b = (t, s)$ is not connected to its level, it is connected to the sizes of its row cluster t and its column cluster s. This observation leads to one of the central ideas of the complexity analysis of $\mathcal{H}$- and $\mathcal{H}^2$-matrices: by bounding the number of blocks b connected to a row cluster t or a column cluster s, we can derive bounds for the number of blocks and for the complexity of many important algorithms [49], [52].

Definition 3.29 (Block rows and columns). For each $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$, we define

$$\operatorname{row}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t) := \{s' \in \mathcal{T}_\mathcal{J} \ : \ (t, s') \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\}$$

and

$$\operatorname{col}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s) := \{t' \in \mathcal{T}_\mathcal{I} \ : \ (t', s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\}.$$

The set $\operatorname{row}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ is called the *block row* corresponding to t, the set $\operatorname{col}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s)$ is called the *block column* corresponding to s (cf. Figure 3.8).

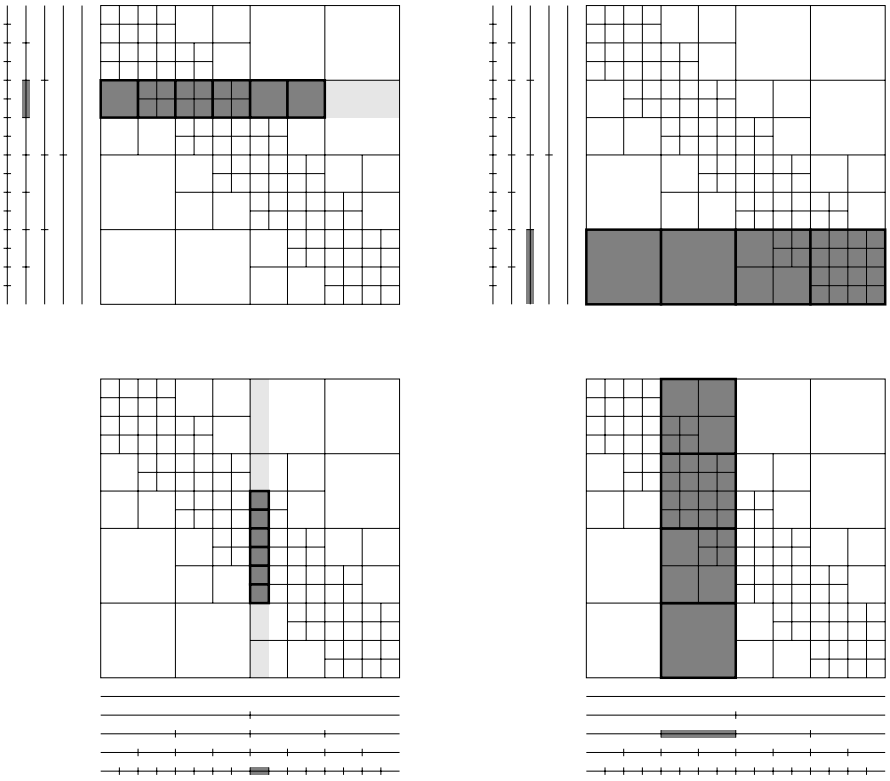


Figure 3.8. Block rows (top) and columns (bottom) of clusters in the model case.

If this does not lead to ambiguity, we use $\operatorname{row}(t)$ instead of $\operatorname{row}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ and $\operatorname{col}(s)$ instead of $\operatorname{col}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s)$.

A block cluster tree is called *sparse* if block rows and columns contain only a bounded number of elements.

Definition 3.30 (Sparsity). Let $C_{\mathrm{sp}} \in \mathbb{N}$. $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called C_{sp}-*sparse* if

$$\#\operatorname{row}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t) \leq C_{\mathrm{sp}} \quad \text{and} \quad \#\operatorname{col}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s) \leq C_{\mathrm{sp}} \tag{3.13}$$

hold for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$.

In the case of the one-dimensional model problem of Chapter 2, we can establish sparsity by the following argument: let $t \in \mathcal{T}_{\mathcal{I}}$. Due to the definition of the cluster tree, there are $\ell \in \{0, \ldots, p\}$ and $\alpha \in \{0, \ldots, 2^\ell - 1\}$ with $t = t_{\ell,\alpha}$. Let $s \in \operatorname{row}(t)$ and $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$. Due to the definition of the block cluster tree, there is a $\beta \in \{0, \ldots, 2^\ell - 1\}$ with $b = b_{\ell,\alpha,\beta}$. If $\ell = 0$, we have $t = t_{0,0}$ and therefore $\operatorname{row}(t) = \{b_{0,0,0}\}$ and $\#\operatorname{row}(t) = 1$.

If $\ell > 0$, the construction of the block cluster tree implies that the father $b_{\ell-1,\alpha^+,\beta^+}$ of b, given by

$$\alpha^+ := \lfloor \alpha/2 \rfloor, \quad \beta^+ := \lfloor \beta/2 \rfloor,$$

cannot have been admissible, since admissible blocks are not subdivided by our algorithm. Due to our admissibility condition, this means $|\alpha^+ - \beta^+| \leq 1$ and $\beta^+ \in \{\alpha^+ - 1, \alpha^+, \alpha^+ + 1\}$. We conclude

$$\begin{aligned} 2\beta^+ &\in \{2\alpha^+ - 2, 2\alpha^+, 2\alpha^+ + 2\}, \\ 2\beta^+ + 1 &\in \{2\alpha^+ - 1, 2\alpha^+ + 1, 2\alpha^+ + 3\}, \\ \beta &\in \{2\alpha^+ - 2, \ldots, 2\alpha^+ + 3\}. \end{aligned}$$

This means that for each $t_{\ell,\alpha} \in \mathcal{T}_I$ there cannot be more than six blocks of the form $b_{\ell,\alpha,\beta} \in \mathcal{T}_{I\times I}$ in the block cluster tree, i.e., $\#\operatorname{row}(t_{\ell,\alpha}) \leq 6$. Due to symmetry, this means that the block cluster tree $\mathcal{T}_{I\times I}$ used for the model problem is 6-sparse.

Lemma 3.31 (Storage requirements). *Let the block cluster tree $\mathcal{T}_{I\times J}$ be C_{sp}-sparse and admissible. Let $p_{I\times J}$ be its depth. Let $X \in \mathbb{R}^{I\times J}$ be a hierarchical matrix with local rank $k \in \mathbb{N}$ and let $r \in \mathbb{N}$ satisfy*

$$\#\hat{t} \leq r, \quad \#\hat{s} \leq r \quad \text{hold for all leaves } t \in \mathcal{L}_I,\ s \in \mathcal{L}_J. \tag{3.14}$$

Then the representation (3.12) requires not more than

$$C_{\mathrm{sp}} \max\{k, r/2\}(p_{I\times J} + 1)(n_I + n_J)$$

units of storage.

Proof. We consider first the storage requirements for one block $b = (t, s) \in \mathcal{T}_{I\times J}$. If b is not a leaf, no storage is required (at least not for the matrix coefficients). If b is an admissible leaf, the matrix block $X_b = \chi_t X \chi_s$ is represented as $X_b = A_b B_b^*$ with $A_b \in \mathbb{R}^{I\times k}_{\hat{t}}$ and $B_b \in \mathbb{R}^{J\times k}_{\hat{s}}$. This requires $k(\#\hat{t} + \#\hat{s})$ units of storage.

If b is an inadmissible leaf, Definition 3.18 implies that t and s are leaves of $\mathcal{T}_I$ and $\mathcal{T}_J$, respectively. Due to the assumption (3.14), we get $\#\hat{t} \leq r$ and $\#\hat{s} \leq r$. For inadmissible blocks, we store all coefficients of the submatrices $X_b = \chi_t X \chi_s$, and this requires

$$(\#\hat{t})(\#\hat{s}) \leq r\#\hat{s}, \quad (\#\hat{t})(\#\hat{s}) \leq r\#\hat{t}$$

units of storage. Adding both estimates and dividing by two yields

$$(\#\hat{t})(\#\hat{s}) \leq \frac{r}{2}(\#\hat{t} + \#\hat{s}),$$

and we conclude that the storage requirements for each individual block are bounded by

$$\max\{k, r/2\}(\#\hat{t} + \#\hat{s}).$$

In order to get the total storage requirements, we have to sum over all blocks:

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \max\{k,r/2\}(\#\hat{t}+\#\hat{s}) = \max\{k,r/2\}\Big(\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \#\hat{t} + \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \#\hat{s}\Big).$$

Due to Definition 3.12, we have $\operatorname{level}(t), \operatorname{level}(s) \leq \operatorname{level}(b)$ for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and the sparsity assumption (3.13) yields

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \#\hat{t} = \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{s\in\operatorname{row}(t)} \#\hat{t} \leq C_{\mathrm{sp}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \#\hat{t}.$$

In order to bound this sum, we have to know how often an index $i \in \mathcal{I}$ can appear in clusters $t \in \mathcal{T}_{\mathcal{I}}$. Due to Corollary 3.10, it can only be part of at most one cluster per level, and we find

$$\sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \#\hat{t} = \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} \sum_{t\in\mathcal{T}_{\mathcal{I}}^{\ell}} \#\hat{t} = \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} \#\bigcup_{t\in\mathcal{T}_{\mathcal{I}}^{\ell}} \hat{t} \leq \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} \#\mathcal{I} = (p_{\mathcal{I}\times\mathcal{J}}+1)n_{\mathcal{I}}.$$

The same arguments can be applied to the second sum and yield

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \#\hat{s} \leq C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1)n_{\mathcal{J}}.$$

We combine both bounds to conclude that the $\mathcal{H}$-matrix requires not more than

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \max\{k,r/2\}(\#\hat{t}+\#\hat{s}) \leq C_{\mathrm{sp}} \max\{k,r/2\}(p_{\mathcal{I}\times\mathcal{J}}+1)(n_{\mathcal{I}}+n_{\mathcal{J}})$$

units of storage. □

In the model case, we have $k = m$, $r = 4m$ and $C_{\mathrm{sp}} = 6$, and Lemma 3.31 yields a bound of $24m(p+1)n$, approximately four times the bound (2.18) we derived in Chapter 2. The additional factor is due to the fact that the proof does not distinguish between admissible and inadmissible blocks.

3.5 Cluster bases

As we have seen in the model case, $\mathcal{H}^2$-matrices are based on cluster bases, which we now introduce in a general setting required for our applications.

Definition 3.32 (Rank distribution). Let $K = (K_t)_{t\in\mathcal{T}_I}$ be a family of finite index sets. Then K is called a *rank distribution* for $\mathcal{T}_I$.

Definition 3.33 (Cluster basis). Let $K = (K_t)_{t\in\mathcal{T}_I}$ be a rank distribution. Let $V = (V_t)_{t\in\mathcal{T}_I}$ be a family of matrices satisfying $V_t \in \mathbb{R}^{\hat{t}\times K_t}$ for all $t \in \mathcal{T}_I$. Then V is called a *cluster basis* with rank distribution K and the matrices V_t are called *cluster basis matrices*.

Due to Remark 3.26, we have $V_t = \chi_t V_t$ for each $t \in \mathcal{T}_I$. In a matrix V_t, only rows corresponding to indices $i \in \hat{t}$ can have non-zero entries. This implies that we need only $(\#K_t)(\#\hat{t})$ units of memory to store the matrix V_t (cf. Figure 3.9).

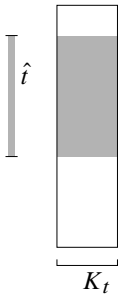


Figure 3.9. Cluster basis.

In order to store a general cluster basis, even one with constant rank $k \equiv 1$, we have to handle the matrices V_t for all $t \in \mathcal{T}_I$, and since each of these matrices requires $\#\hat{t}$ units of storage, we will need a total of $\mathcal{O}(n_I(p_I + 1))$ units of storage.

As in the case of the model problem, we can reduce the storage requirements significantly if we assume that the matrices V_t are connected to each other.

Definition 3.34 (Nested cluster basis). Let V be a cluster basis with rank distribution K. V is called *nested* if there is a family $(E_t)_{t\in\mathcal{T}_I}$ of matrices satisfying the following conditions:

- For all $t \in \mathcal{T}_I$ and all $t' \in \operatorname{sons}(t)$, we have $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$.
- For all $t \in \mathcal{T}_I$ with $\operatorname{sons}(t) \neq \emptyset$, the following equation holds:

$$V_t = \sum_{t'\in\operatorname{sons}(t)} V_{t'} E_{t'}. \tag{3.15}$$

The matrices E_t are called *transfer matrices* or *expansion matrices*.

V is given in *nested representation* if the cluster basis matrices V_t are only stored for leaf clusters t and expressed by transfer matrices for all other clusters.

Due to equation (3.15), we do not have to store the matrices V_t for all $t \in \mathcal{T}_\mathcal{I}$: if $\operatorname{sons}(t) \neq \emptyset$, we only have to store the, typically small, transfer matrices for each of the sons and derive quantities corresponding to V_t by a recursion (cf. Figure 3.10) whenever necessary.

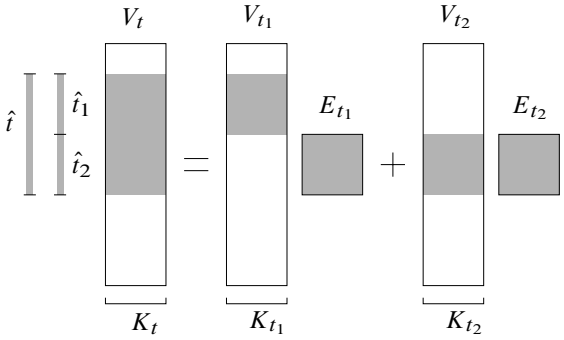


Figure 3.10. Nested cluster basis.

In the model case, we have seen that this strategy leads to the optimal order of complexity, i.e., we can find a bound for the amount of storage which is proportional to the number of degrees of freedom.

In more general situations, especially in the case of variable-order approximations (cf. Section 4.7), we require more flexible methods for controlling the complexity. We base all of our complexity estimates on the rank distribution $K = (K_t)_{t \in \mathcal{T}_\mathcal{I}}$. A typical $\mathcal{H}^2$-matrix algorithm performs a number of operations for each cluster $t \in \mathcal{T}_\mathcal{I}$, and the number of operations depends on the cluster: if t is a leaf, the dimensions of the matrix V_t, i.e., $\#\hat{t}$ and $\#K_t$ are important, if t is not a leaf, the dimensions of the transfer matrices $E_{t'}$ for the sons $t' \in \operatorname{sons}(t)$ are relevant. In summary, the complexity of most algorithms is determined by the quantities defined by

$$k_t := \begin{cases} \max\{\#K_t, \#\hat{t}\} & \text{if } \operatorname{sons}(t) = \emptyset, \\ \max\left\{\#K_t, \sum_{t' \in \operatorname{sons}(t)} \#K_{t'}\right\} & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}. \tag{3.16}$$

As an example, we can use this definition to prove a simple bound for the storage requirements of a cluster basis:

Lemma 3.35 (Storage complexity). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree, and let V be a nested cluster basis for $\mathcal{T}_\mathcal{I}$ with rank distribution K. Then the nested representation of V requires not more than*

$$\sum_{t \in \mathcal{T}_\mathcal{I}} k_t \#K_t \leq \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2$$

units of storage.

Proof. Let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ be defined by (3.16).

Let $t \in \mathcal{T}_\mathcal{I}$. If t is a leaf, we have to store $V_t \in \mathbb{R}^{\mathcal{I}\times K_t}_{\hat{t}}$, and this takes $(\#\hat{t})(\#K_t) \leq k_t(\#K_t)$ units of storage. If t is not a leaf, we have to store $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$ for all $t' \in \operatorname{sons}(t)$, and this takes

$$\sum_{t'\in\operatorname{sons}(t)} (\#K_{t'})(\#K_t) = \Big(\sum_{t'\in\operatorname{sons}(t)} \#K_{t'}\Big)(\#K_t) \leq k_t(\#K_t)$$

units of storage. For the total storage requirements, we sum over all $t \in \mathcal{T}_\mathcal{I}$ and can use $\#K_t \leq k_t$ to get the bound

$$\sum_{t\in\mathcal{T}_\mathcal{I}} k_t(\#K_t) \leq \sum_{t\in\mathcal{T}_\mathcal{I}} k_t^2. \qquad \square$$

In the model case, we have $\#K_t = m$ and $\#\mathcal{T}_\mathcal{I} = 2^{p+1} - 1$, and we can bound k_t by $2m$ for the $2^p - 1$ non-leaf clusters $t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}$ and by $\#\hat{t}$ for the leaf clusters $t \in \mathcal{L}_\mathcal{I}$, so Lemma 3.35 combined with Corollary 3.9 yields the bound

$$2m^2(2^p - 1) + m\sum_{t\in\mathcal{L}_\mathcal{I}} \#\hat{t} < 2m^2 2^p + m\#\bigcup_{t\in\mathcal{L}_\mathcal{I}} \hat{t} \leq m2^{p+p_0} + mn = 2mn,$$

and this is exactly the estimate given in (2.24).

3.6 $\mathcal{H}^2$-matrices

Let V be a nested cluster basis for $\mathcal{T}_\mathcal{I}$ with rank distribution K, and let W be a nested cluster basis for $\mathcal{T}_\mathcal{J}$ with rank distribution L. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$.

Based on V, W and $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we can introduce the corresponding space of $\mathcal{H}^2$-matrices.

Definition 3.36 ($\mathcal{H}^2$-matrix). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. X is an *$\mathcal{H}^2$-matrix* for the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, the *row cluster basis* V and the *column cluster basis* W if there is a family $S = (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of matrices satisfying $S_b \in \mathbb{R}^{K_t\times L_s}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_b W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{3.17}$$

The elements S_b of the family S are called *coupling matrices* and are sometimes also referred to by $S_{t,s} = S_b$ for $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Before we investigate the storage requirements of $\mathcal{H}^2$-matrices, we observe that the set of all $\mathcal{H}^2$-matrices for the same block cluster tree and cluster bases is a matrix subspace. This is important since it not only means that we can add and scale $\mathcal{H}^2$-matrices efficiently, but also that we can construct approximations of general matrices by projecting into the $\mathcal{H}^2$-matrix subspace (cf. Chapter 5 for optimal projections into this space and Chapter 7 for an application).

Remark 3.37 (Matrix subspace). The set

$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W) := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \ : \ X \text{ is an } \mathcal{H}^2\text{-matrix for } \mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V \text{ and } W\}$$

is a subspace of $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

Proof. Let $A, B \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$, and let $S_A = (S_{A,b})_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$, $S_B = (S_{B,b})_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be the corresponding coupling matrices. Let $\lambda \in \mathbb{R}$ and $C := A + \lambda B$. We have

$$\begin{aligned}
C &= A + \lambda B \\
&= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_{A,t,s} W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t A \chi_s \\
&\quad + \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \lambda V_t S_{B,t,s} W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \lambda \chi_t B \chi_s \\
&= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t (S_{A,t,s} + \lambda S_{B,t,s}) W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t (A + \lambda B) \chi_s \\
&= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_{C,t,s} W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t C \chi_s,
\end{aligned}$$

where we let $S^C_{t,s} := S^A_{t,s} + \lambda S^B_{t,s}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Therefore C can be represented in the form (3.17), which implies $C \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Observing $0 \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ concludes the proof. □

We have already seen that the storage requirements for the cluster basis V grow at most proportionally to the number of clusters $c_{\mathcal{I}}$ in the corresponding cluster tree $\mathcal{T}_{\mathcal{I}}$. The same holds for the cluster basis W and the cluster tree $\mathcal{T}_{\mathcal{J}}$. In order to establish an upper bound for the storage requirements of an $\mathcal{H}^2$-matrix, we therefore only have to consider the nearfield and coupling matrices.

Lemma 3.38 (Storage). *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible C_{sp}-sparse block cluster tree for the cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Let V and W be nested cluster bases with rank distributions K and L, let $(k_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be defined as in (3.16) and define*

$$l_s := \begin{cases} \max\{\#L_s, \#\hat{t}\} & \text{if } \operatorname{sons}(s) = \emptyset, \\ \max\left\{\#L_s, \sum_{s'\in\operatorname{sons}(s)} \#L_{s'}\right\} & \text{otherwise,} \end{cases} \qquad \text{for all } s \in \mathcal{T}_{\mathcal{J}} \tag{3.18}$$

correspondingly. The $\mathcal{H}^2$-matrix representation of a matrix in $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ requires not more than

$$\frac{C_{\mathrm{sp}}}{2}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2\Big)$$

units of storage. If we also include the representations of the cluster bases V and W, a total of

$$\frac{C_{\mathrm{sp}}+2}{2}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2\Big)$$

units of storage is sufficient.

Proof. Let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Let $b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$. If b is admissible, we store the coupling matrix $S_b \in \mathbb{R}^{K_t\times L_s}$, and this takes

$$(\#K_t)(\#L_s) \leq k_t l_s \leq \frac{1}{2}(k_t^2 + l_s^2)$$

units of storage. If b is not admissible, we store the matrix $\chi_t X \chi_s \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\hat{t}\times\hat{s}}$, and this requires

$$(\#\hat{t})(\#\hat{s}) \leq k_t l_s \leq \frac{1}{2}(k_t^2 + l_s^2)$$

units of storage. For the total storage requirements we add the results for all blocks and use (3.13) to get

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \frac{1}{2}(k_t^2 + l_s^2) &\leq \frac{1}{2}\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} k_t^2 + \frac{1}{2}\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} l_s^2 \\
&= \frac{1}{2}\sum_{t\in\mathcal{T}_{\mathcal{I}}}\sum_{s\in\mathrm{row}(t)} k_t^2 + \frac{1}{2}\sum_{s\in\mathcal{T}_{\mathcal{J}}}\sum_{t\in\mathrm{col}(s)} l_s^2 \\
&\overset{(3.13)}{\leq} \frac{C_{\mathrm{sp}}}{2}\sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + \frac{C_{\mathrm{sp}}}{2}\sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2.
\end{aligned}$$

Lemma 3.35 implies that V and W require not more than

$$\sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2$$

units of storage, and adding both estimates completes the proof. □

In the model case, we have $C_{\mathrm{sp}} = 6$, $k_t \leq \#\hat{t} \leq 4m$ for leaf clusters and $k_t = 2m$ for the $2^p - 1$ non-leaf clusters, so Lemma 3.38 yields a bound of

$$8\Big(\sum_{t\in\mathcal{L}_{\mathcal{I}}} (\#\hat{t})^2 + (2^p - 1)(2m)^2\Big) \leq 8(4mn + 2mn) = 48mn.$$

The result in Chapter 2 is better by a factor of almost 3 because it distinguishes admissible and inadmissible blocks, while we avoided this for the sake of simplicity in the proof of the general estimate.

Remark 3.39 (HSS-matrices). A quadratic $\mathcal{H}^2$-matrix based on a balanced binary cluster tree $\mathcal{T}_\mathcal{I}$ and a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ using the weak admissibility condition (cf. Remark 3.17) is called a *hierarchically semi-separable matrix* (or short an *HSS-matrix*) [30], [106]. Matrices of this type can be handled very efficiently, even robust direct solvers of linear complexity are available [29], [31].

Since the weak admissibility condition is only adequate for essentially one-dimensional problems, the range of applications of HSS-matrices is limited to problems of this type, e.g., there are exact solvers for one-dimensional boundary value problems [59], [96], approximate solvers for Toeplitz matrices [32], and it is possible to use nested dissection techniques to reduce two-dimensional sparse problems to one-dimensional problems that can be treated by HSS-matrices [83], [107].

Hierarchically semi-separable matrices are closely related to the matrix set $\mathcal{M}_k(\epsilon)$ introduced and analyzed in Section 4 of [69]. □

3.7 Matrix-vector multiplication

Merely storing a matrix in the efficient $\mathcal{H}^2$-matrix representation (3.17) is not sufficient for the majority of applications, since these will require a way of evaluating the operator associated with the matrix, i.e., of multiplying the matrix with a vector.

Since the $\mathcal{H}^2$-matrix is given in factorized form, and since we do not intend to convert the efficient representation back into a less efficient one, we have to find a way of performing the necessary operations based on the available factorized representation.

Let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ be given in the form (3.17), and let the nested cluster bases V and W be given in nested representation.

Let $x \in \mathbb{R}^\mathcal{J}$. We are looking for a way of computing $y := Xx$ efficiently. Due to (3.17), y is given in the form

$$y = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_{t,s} W_s^* x + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s x.$$

In order to be able to rewrite this equation in a form more suitable for our algorithm, we introduce the following subsets of block rows and columns (cf. Definition 3.29).

Definition 3.40 (Admissible block rows and columns). For each $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$, we define

$$\operatorname{row}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t) := \{s' \in \mathcal{T}_\mathcal{J} \ : \ (t,s') \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}$$

and

$$\operatorname{col}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s) := \{t' \in \mathcal{T}_{\mathcal{I}} : (t', s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}.$$

The set $\operatorname{row}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ is called the *admissible block row* corresponding to t, the set $\operatorname{col}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s)$ is called the *admissible block column* corresponding to s.

If the block cluster tree is implied by the context, we use $\operatorname{row}^+(t)$ instead of $\operatorname{row}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ and $\operatorname{col}^+(s)$ instead of $\operatorname{col}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, s)$.

Using the admissible block rows, we find

$$y = \sum_{t\in\mathcal{T}_{\mathcal{I}}} V_t \sum_{s\in\operatorname{row}^+(t)} S_{t,s} W_s^* x + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s x,$$

therefore the task of computing y can be split into four subtasks:

- Compute $\hat{x}_s := W_s^* x$ for all $s \in \mathcal{T}_{\mathcal{J}}$. This is called the *forward transformation*.
- Compute $\hat{y}_t := \sum_{s\in\operatorname{row}^+(t)} S_{t,s}\hat{x}_s$. This is called the *multiplication step*.
- Compute $y_{\text{far}} := \sum_{t\in\mathcal{T}_{\mathcal{I}}} V_t \hat{y}_t$. This is called the *backward transformation*.
- Compute $y = y_{\text{far}} + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s x$. This is called the *nearfield step*.

The multiplication and nearfield steps do not present us with a challenge, since they only involve the "small" matrices $S_{t,s}$ and $\chi_t X \chi_s$.

The forward transformation requires a close investigation, since it involves the matrix W_s, which may have a large number of rows and is only implicitly given by equation (3.15). Fortunately, the nested structure implies

$$\hat{x}_s = W_s^* x = \Bigl(\sum_{s'\in\operatorname{sons}(s)} W_{s'} F_{s'}\Bigr)^* x = \sum_{s'\in\operatorname{sons}(s)} F_{s'}^* W_{s'}^* x = \sum_{s'\in\operatorname{sons}(s)} F_{s'}^* \hat{x}_{s'}$$

for all $s \in \mathcal{T}_{\mathcal{J}}$ with $\operatorname{sons}(s) \neq \emptyset$, i.e., we require the matrices W_s only for leaves of the cluster tree and can use the transfer matrices for all other clusters. Applying the transfer matrices recursively leads to Algorithm 6.

The backward transformation can be handled in a similar fashion: let $t \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{sons}(t) \neq \emptyset$. Then (3.15) implies

$$V_t \hat{y}_t + \sum_{t'\in\operatorname{sons}(t)} V_{t'} \hat{y}_{t'} = \sum_{t'\in\operatorname{sons}(t)} V_{t'} E_{t'} \hat{y} + \sum_{t'\in\operatorname{sons}(t)} V_{t'} \hat{y}_{t'} = \sum_{t'\in\operatorname{sons}(t)} V_{t'} (\hat{y}_{t'} + E_{t'} \hat{y}_t).$$

Applying this equation recursively allows us to reduce the original sum over $\mathcal{T}_{\mathcal{I}}$ to a sum over $\mathcal{L}_{\mathcal{I}}$, which can be easily computed. The result is Algorithm 7.

Algorithm 6. Forward transformation.

```
procedure ForwardTransformation(t, V, x, var x̂);
if sons(t) = ∅ then
  x̂_t ← V_t^* x
else
  x̂_t ← 0;
  for t' ∈ sons(t) do
    ForwardTransformation(t', V, x, x̂);
    x̂_t ← x̂_t + E_{t'}^* x̂_{t'}
  end for
end if
```

Algorithm 7. Backward transformation.

```
procedure BackwardTransformation(t, V, var y, ŷ);
if sons(s) = ∅ then
  y ← y + V_t ŷ_t
else
  for t' ∈ sons(t) do
    ŷ_{t'} ← ŷ_{t'} + E_{t'} ŷ_t;
    BackwardTransformation(t', V, y, ŷ)
  end for
end if
```

Lemma 3.41 (Forward and backward transformation). *Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a nested cluster basis with transfer matrices $(E_t)_{t\in\mathcal{T}_\mathcal{I}}$ and rank distribution $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$. Let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ be defined as in (3.16) . Then the Algorithms 6 and 7 require not more than*

$$2\sum_{t\in\mathcal{T}_\mathcal{I}} k_t \# K_t \leq 2\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^2$$

arithmetic operations.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. If t is a leaf cluster, both algorithms multiply the matrix $V_t \in \mathbb{R}^{\hat{t}\times K_t}$ or its adjoint by a vector. This requires not more than $2(\#\hat{t})(\#K_t) \leq 2k_t\#K_t$ arithmetic operations.

If t is not a leaf cluster, both algorithms multiply the matrices $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$ or their adjoints by vectors for all sons $t' \in \operatorname{sons}(t)$. For one matrix this takes not more than $2(\#K_{t'})(\#K_t)$ operations, for all sons we get a bound of

$$\sum_{t'\in\operatorname{sons}(t)} 2(\#K_{t'})(\#K_t) = 2k_t\#K_t.$$

Summing over all $t \in \mathcal{T}_\mathcal{I}$ yields the desired bound. □

A closer look at the algorithms reveals that for each coefficient stored in the nested representation of V the algorithms perform one multiplication and at most one addition, so we could also have based the proof on the estimate of Lemma 3.35.

Algorithm 8. Matrix-vector multiplication.

procedure MVM(X, x, **var** y);
ForwardTransformation(root($\mathcal{T}_{\mathcal{J}}$), W, x, $\hat{x}$);
for $t \in \mathcal{T}_{\mathcal{I}}$ **do**
  $\hat{y}_t \leftarrow 0$;
  for $s \in \operatorname{row}^+(t)$ **do**
    $\hat{y}_t \leftarrow \hat{y}_t + S_{t,s}\hat{x}_s$
  end for
end for;
BackwardTransformation(root($\mathcal{T}_{\mathcal{I}}$), V, y, $\hat{y}$);
for $b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ **do**
  $y \leftarrow y + \chi_t X \chi_s x$
end for

Theorem 3.42 (Matrix-vector multiplication). *Let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse and admissible, let K and L be the rank distributions of V and W, and let $(k_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(l_s)_{s\in\mathcal{T}_{\mathcal{J}}}$ be defined as in (3.16) and (3.18). The Algorithm 8 requires not more than*

$$(C_{\mathrm{sp}} + 2)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2\Big)$$

operations.

Proof. Both the forward and backward transformation are covered by Lemma 3.41.

The multiplication step is carried out for each admissible block $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and requires $2(\#K_t)(\#L_s) \leq 2k_t l_s$ operations for multiplying by the matrix S_b and adding the result to $\hat{y}_t$.

The nearfield step is carried out for each inadmissible leaf $b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ and requires $2(\#\hat{t})(\#\hat{s})$ operations for multiplying by the matrix $\chi_t X \chi_s$ and adding the result to y. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, both t and s are leaf clusters, so we have $\#\hat{t} \leq k_t$ and $\#\hat{s} \leq l_s$ and can bound the number of operations in this case by $2k_t l_s$.

Summing over all leaf blocks yields a bound of

$$\begin{aligned}\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} 2k_t l_s &\leq \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} k_t^2 + l_s^2 \\ &= \sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\operatorname{row}(t)} k_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \sum_{t\in\operatorname{col}(s)} l_s^2 \\ &\leq C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + C_{\mathrm{sp}} \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2,\end{aligned}$$

and adding the estimates of Lemma 3.41 completes the proof. □

In the model case, we have $C_{\mathrm{sp}} = 6$, $k_t, l_s = 2^{p_0} \leq 4m$ for leaf clusters and $k_t, l_s \leq 2m$ for non-leaf clusters, and Theorem 3.42 yields a bound of

$$16\Big(\sum_{t\in\mathcal{L}_I} (\#\hat{t})^2 + (2^p - 1)(2m)^2\Big) \leq 16(4mn + 2mn) = 96mn.$$

Our detailed analysis of the $\mathcal{H}^2$-matrix representation in the model case shows that not more than $17mn$ units of storage are required, and since we have seen that at most two operations are performed for each coefficient, we can derive the improved estimate of $34mn$ operations. The difference between this result and the bound provided by the general Theorem 3.42 is again due to the fact that we have simplified the proof by treating admissible and inadmissible blocks similarly.

3.8 Complexity estimates for bounded rank distributions

Our bounds for the storage requirements and the number of operations involved in the matrix-vector multiplication are sums of powers of the quantities k_t and l_s describing the work corresponding to individual clusters. In the model problem, k_t is uniformly bounded by $4m$, and this leads to relatively simple estimates for the complexity.

A closer look reveals that we can get useful bounds for the number of operations and for the storage requirements even for unbounded k_t as long as large values occur only for a small number of clusters. This property is very important for variable-order techniques that can significantly improve the efficiency of $\mathcal{H}^2$-matrix approximations (cf. Sections 4.7 and 6.8).

The storage requirements of the $\mathcal{H}^2$-matrices used for the model problem grow linearly with the number of degrees of freedom. This is a very desirable property, and we would like to preserve it even if the ranks of the cluster basis are unbounded. In order to do so, we have to ensure that there are only a few clusters that require a larger amount of work. We have seen that the amount of work for a cluster t depends on

$$k_t := \begin{cases} \max\{\#K_t, \#\hat{t}\} & \text{if } \operatorname{sons}(t) = \emptyset, \\ \max\left\{\#K_t, \sum_{t'\in\operatorname{sons}(t)} \#K_{t'}\right\} & \text{otherwise,} \end{cases}$$

and in order to bound this quantity, we have to bound $\#K_t$, and we also have to bound $\#\hat{t}$ if t is a leaf and $\#\operatorname{sons}(t)$ if it is not. In order to keep the definition as general as possible, we use an indirect approach: we require that the number of clusters decreases exponentially as the amount of work increases algebraically. This guarantees that the sum of the amounts of work for all clusters converges.

Definition 3.43 (Bounded cluster tree). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree, let $C_{\mathrm{bc}} \in \mathbb{R}_{\geq 1}$, $\alpha \in \mathbb{R}_{>0}$, $\beta \in \mathbb{R}_{\geq 0}$, $r \in \mathbb{R}_{\geq 1}$, $\xi \in \mathbb{R}_{>1}$. The tree $\mathcal{T}_\mathcal{I}$ is $(C_{\mathrm{bc}}, \alpha, \beta, r, \xi)$-*bounded* if

$$\#\{t \in \mathcal{L}_\mathcal{I} : \#\hat{t} > (\alpha + \beta(\ell - 1))^r\} \leq C_{\mathrm{bc}} \xi^{-\ell} \#\mathcal{T}_\mathcal{I} \quad \text{holds for all } \ell \in \mathbb{N}$$

and

$$\#\operatorname{sons}(t) \leq C_{\mathrm{bc}} \quad \text{holds for all } t \in \mathcal{T}_\mathcal{I}.$$

Let $k \in \mathbb{N}$. The tree $\mathcal{T}_\mathcal{I}$ is (C_{bc}, k)-*bounded* if

$$\begin{aligned} \#\hat{t} &\leq k && \text{holds for all } t \in \mathcal{L}_\mathcal{I}, \\ \#\operatorname{sons}(t) &\leq C_{\mathrm{bc}} && \text{holds for all } t \in \mathcal{T}_\mathcal{I}. \end{aligned}$$

Definition 3.44 (Bounded rank distribution). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree, let $K = (K_t)_{t \in \mathcal{T}_\mathcal{I}}$ be a rank distribution for $\mathcal{T}_\mathcal{I}$.

Let $C_{\mathrm{bn}} \in \mathbb{R}_{\geq 1}$, $\alpha \in \mathbb{R}_{>0}$, $\beta \in \mathbb{R}_{\geq 0}$, $r \in \mathbb{R}_{\geq 1}$ and $\xi \in \mathbb{R}_{>1}$. The rank distribution K is $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-*bounded*, if

$$\#\{t \in \mathcal{T}_\mathcal{I} : \#K_t > (\alpha + \beta(\ell - 1))^r\} \leq C_{\mathrm{bn}} \xi^{-\ell} \#\mathcal{T}_\mathcal{I} \quad \text{holds for all } \ell \in \mathbb{N}. \tag{3.19}$$

Let $k \in \mathbb{N}$. The rank distribution K is k-*bounded* if

$$\#K_t \leq k \quad \text{holds for all } t \in \mathcal{T}_\mathcal{I}.$$

If a rank distribution K is k-bounded, it is also $(1, k, 0, 1, \xi)$-bounded for arbitrary values of $\xi \in \mathbb{R}_{>1}$. If, on the other hand, K is $(C_{\mathrm{bn}}, \alpha, 0, r, \xi)$-bounded, taking the limit $\ell \to \infty$ of (3.19) implies $\#K_t \leq \alpha^r$ for all $t \in \mathcal{T}_\mathcal{I}$, i.e., K is α^r-bounded, and both definitions are equivalent for $\beta = 0$.

In the model case, the rank distribution is $4m$-bounded, since we have $k_t = 2^{p_0} < 4m$ for all leaf clusters $t \in \mathcal{L}_\mathcal{I}$ and even $k_t = 2m$ for the remaining clusters $t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}$.

For variable-order approximations (cf. Section 4.7), a simple approach is to allow the order to increase as the level decreases, i.e., to use

$$\#K_t = \alpha + \beta(p_\mathcal{I} - \operatorname{level}(t)),$$

where $p_\mathcal{I}$ is the maximal level and α and β are parameters that can be used to control the approximation error. The resulting rank distribution is illustrated in Figure 3.11 for $\alpha = \beta = 1$. We can see that

$$\alpha + \beta(p_\mathcal{I} - \operatorname{level}(t)) = \#K_t > \alpha + \beta(\ell - 1)$$

implies $\operatorname{level}(t) < p_\mathcal{I} + 1 - \ell$, so there are exactly $2^{p_\mathcal{I}+1-\ell} - 1$ clusters $t \in \mathcal{T}_\mathcal{I}$ with $\#K_t > \alpha + \beta(\ell - 1)$. The total number of clusters satisfies

$$\begin{aligned} \#\mathcal{T}_\mathcal{I} &= 2^{p_\mathcal{I}+1} - 1 = 2^\ell (2^{p_\mathcal{I}+1-\ell} - 2^{-\ell}) \\ &> 2^\ell (2^{p_\mathcal{I}+1-\ell} - 1) = 2^\ell \#\{t \in \mathcal{T}_\mathcal{I} : \#K_t > \alpha + \beta(\ell - 1)\}, \end{aligned}$$

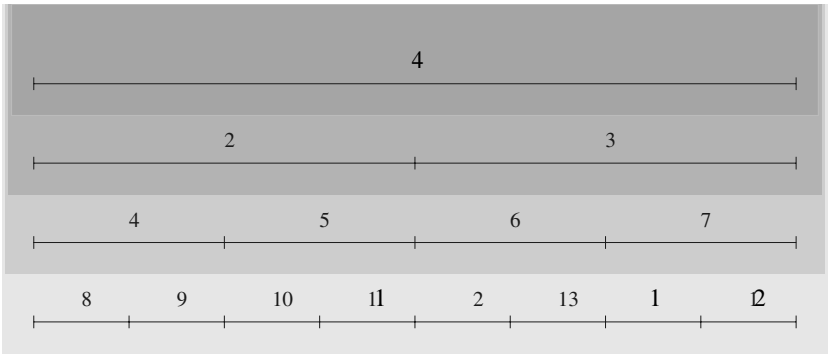


Figure 3.11. Rank distribution for variable-order approximation based on the level, $(1, 1, 1, 1, 2)$-bounded.

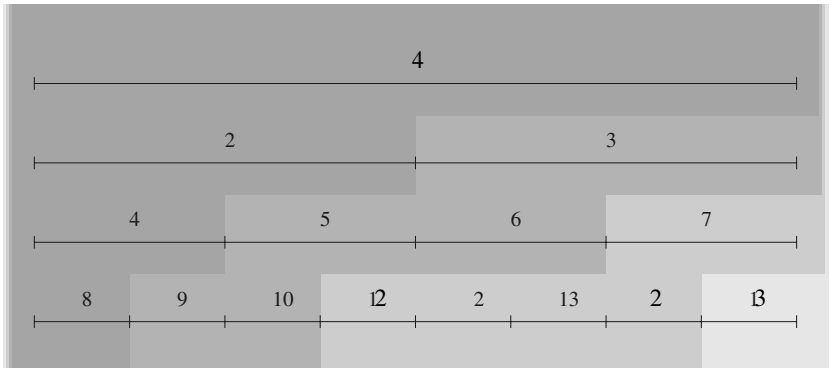


Figure 3.12. Rank distribution for variable-order approximation with an additional singularity at the left endpoint, $(C_{\mathrm{bn}}, 1, 1, 1, \xi)$-bounded.

therefore our rank distribution is $(1, \alpha, \beta, 1, 2)$-bounded.

We conclude the discussion of Definition 3.44 by a non-trivial example. Sometimes, e.g., when approximating certain integral operators on manifolds [80], [81], additional singularities appear at endpoints of intervals, and we have to increase the order close to these points in order to ensure that the error is sufficiently small. Let us consider the situation given in Figure 3.12: we use

$$\#K_t = \alpha + \beta \max\{p_{\mathcal{I}} - \mathrm{level}(t),\, p_{\mathcal{I}} - \lfloor \log_2(\mathrm{dist}(t, 0)/\,\mathrm{diam}(t) + 1) \rfloor\},$$

i.e., clusters close to a singularity assumed at the left endpoint of the interval $[0, 1]$ are assigned a higher order. For this distribution, we can see that

$$\#K_t > \alpha + \beta(\ell - 1)$$

implies

$$\operatorname{level}(t) < p_{\mathcal{I}} + 1 - \ell \quad \text{or} \quad \lfloor \log_2(\operatorname{dist}(t,0)/\operatorname{diam}(t) + 1) \rfloor < p_{\mathcal{I}} + 1 - \ell.$$

We have already seen that the first condition holds for $2^{p_{\mathcal{I}}+1-\ell} - 1$ clusters. The second condition is equivalent to

$$\operatorname{dist}(t,0)/\operatorname{diam}(t) + 1 < 2^{p_{\mathcal{I}}+1-\ell}, \quad \operatorname{dist}(t,0) < \operatorname{diam}(t)(2^{p_{\mathcal{I}}+1-\ell} - 1),$$

so on each level there are at most $2^{p_{\mathcal{I}}+1-\ell} - 1$ clusters matching the second condition. We conclude

$$\begin{aligned}
&\#\{t \in \mathcal{T}_{\mathcal{I}} : \#K_t > \alpha + \beta(\ell - 1)\} \\
&\quad = \#\{t \in \mathcal{T}_{\mathcal{I}} : \operatorname{level}(t) < p_{\mathcal{I}} + 1 - \ell\} \\
&\qquad + \#\{t \in \mathcal{T}_{\mathcal{I}} : \operatorname{level}(t) \geq p_{\mathcal{I}} + 1 - \ell,\ \operatorname{dist}(t,0) < \operatorname{diam}(t)(2^{p_{\mathcal{I}}+1-\ell} - 1)\} \\
&\quad \leq (2^{p_{\mathcal{I}}+1-\ell} - 1) + \ell(2^{p_{\mathcal{I}}+1-\ell} - 1) \\
&\quad = (\ell + 1)(2^{p_{\mathcal{I}}+1-\ell} - 1).
\end{aligned}$$

In order to apply Definition 3.44, we have to prove that this quantity decays exponentially if ℓ grows. Due to the factor $\ell + 1$, we cannot get a convergence rate of $1/2$, but we can get arbitrarily close: let $\xi \in (1,2)$, and let $q := \xi/2 < 1$. Due to Lemma 3.50, there is a constant $C \in \mathbb{R}_{\geq 1}$ with

$$(\ell + 1)q^\ell \leq \sum_{\ell=0}^{\infty} (\ell + 1)q^\ell \leq C \quad \text{for all } \ell \in \mathbb{N}_0,$$

and we have

$$\begin{aligned}
&\#\{t \in \mathcal{T}_{\mathcal{I}} : \#K_t > \alpha + \beta(\ell - 1)\} \\
&\qquad \leq (\ell + 1)(2^{p_{\mathcal{I}}+1-\ell} - 1) \\
&\qquad < (\ell + 1)2^{-\ell}(2^{p_{\mathcal{I}}+1} - 1) = (\ell + 1)q^\ell \xi^{-\ell} \#\mathcal{T}_{\mathcal{I}} \leq C\xi^{-\ell} \#\mathcal{T}_{\mathcal{I}}
\end{aligned}$$

for all $\ell \in \mathbb{N}$. This means that our rank distribution is $(C, \alpha, \beta, 1, \xi)$-bounded.

The main advantage of Definition 3.44 is that it allows us to simplify complexity estimates of the form encountered in Lemma 3.38 or Theorem 3.42:

Lemma 3.45 (Complexity bounds). *Let $\mathcal{T}_{\mathcal{I}}$ be $(C_{\mathrm{bc}}, \alpha, \beta, r, \xi)$-bounded, and let $K = (K_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ be a $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded rank distribution. Let $(k_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ be defined as in (3.16), and let $m \in \mathbb{N}$. Then there is a constant $C_{\mathrm{cb}} \in \mathbb{R}_{\geq 1}$ depending only on C_{bc}, C_{bn}, r, ξ and m satisfying*

$$\sum_{t \in \mathcal{T}_{\mathcal{I}}} k_t^m \leq C_{\mathrm{cb}}(\alpha + \beta)^{rm} c_{\mathcal{I}}$$

with $c_{\mathcal{I}} = \#\mathcal{T}_{\mathcal{I}}$ denoting the number of clusters in $\mathcal{T}_{\mathcal{I}}$.

Proof. We split the clusters into a disjoint partition according to the corresponding complexity: we let

$$\mathcal{C}_\ell := \begin{cases} \{t \in \mathcal{T}_\mathcal{I} \,:\, k_t \leq C_{\mathrm{bc}}\alpha^r\} & \text{if } \ell = 0, \\ \{t \in \mathcal{T}_\mathcal{I} \,:\, C_{\mathrm{bc}}(\alpha + \beta(\ell-1))^r < k_t \leq C_{\mathrm{bc}}(\alpha + \beta\ell)^r\} & \text{otherwise} \end{cases}$$

for all $\ell \in \mathbb{N}_0$. We have

$$\mathcal{T}_\mathcal{I} = \dot{\bigcup}_{\ell \in \mathbb{N}_0} \mathcal{C}_\ell,$$

i.e., the *complexity levels* $\mathcal{C}_\ell$ define a disjoint partition of the cluster tree, and

$$k_t \leq C_{\mathrm{bc}}(\alpha + \beta\ell)^r \tag{3.20}$$

holds for all $t \in \mathcal{C}_\ell$ and all $\ell \in \mathbb{N}_0$ by definition.

Let $\ell \in \mathbb{N}$, and let $t \in \mathcal{C}_\ell$. This means

$$C_{\mathrm{bc}}(\alpha + \beta(\ell-1))^r < k_t = \begin{cases} \max\{\#K_t, \#\hat{t}\} & \text{if } \operatorname{sons}(t) = \emptyset, \\ \max\left\{\#K_t, \sum_{t' \in \operatorname{sons}(t)} \#K_{t'}\right\} & \text{otherwise.} \end{cases}$$

If t is a leaf, this implies

$$\#K_t > C_{\mathrm{bc}}(\alpha + \beta(\ell-1))^r \geq (\alpha + \beta(\ell-1))^r$$

or

$$\#\hat{t} > C_{\mathrm{bc}}(\alpha + \beta(\ell-1))^r \geq (\alpha + \beta(\ell-1))^r.$$

If t is not a leaf, we cannot have more than C_{bc} son clusters and can conclude

$$\#K_t > (\alpha + \beta(\ell-1))^r$$

or

$$\#K_{t'} > (\alpha + \beta(\ell-1))^r \qquad \text{for at least one } t' \in \operatorname{sons}(t).$$

This means that t has to be an element of at least one of the three sets

$$\begin{aligned} \mathcal{R}_\ell &:= \{t \in \mathcal{T}_\mathcal{I} \,:\, \#K_t > (\alpha + \beta(\ell-1))^r\}, \\ \mathcal{L}_\ell &:= \{t \in \mathcal{L}_\mathcal{I} \,:\, \#\hat{t} > (\alpha + \beta(\ell-1))^r\}, \\ \mathcal{S}_\ell &:= \{t \in \mathcal{T}_\mathcal{I} \,:\, \#K_{t'} > (\alpha + \beta(\ell-1)^r \text{ for at least one } t' \in \operatorname{sons}(t)\}, \end{aligned}$$

and we have proven the inclusion

$$\mathcal{C}_\ell \subseteq \mathcal{R}_\ell \cup \mathcal{L}_\ell \cup \mathcal{S}_\ell.$$

Due to our assumptions we have

$$\#\mathcal{R}_\ell \leq C_{\mathrm{bn}}\xi^{-\ell}c_\mathcal{I}, \quad \#\mathcal{L}_\ell \leq C_{\mathrm{bc}}\xi^{-\ell}c_\mathcal{I},$$

and since each cluster can have at most one father, we also find

$$\mathcal{S}_\ell = \{\operatorname{father}(t) \,:\, t \in \mathcal{T}_\mathcal{I},\ \#K_t > (\alpha + \beta(\ell-1))^r\} = \{\operatorname{father}(t) \,:\, t \in \mathcal{R}_\ell\},$$
$$\#\mathcal{S}_\ell \leq \#\mathcal{R}_\ell \leq C_{\mathrm{bn}}\xi^{-\ell} c_\mathcal{I}$$

and obtain the bound

$$\#\mathcal{C}_\ell \leq (2C_{\mathrm{bn}} + C_{\mathrm{bc}})\xi^{-\ell} c_\mathcal{I} \quad \text{for all } \ell \in \mathbb{N}_0.$$

Combining this estimate with (3.20) yields

$$\begin{aligned}\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^m &= \sum_{\ell\in\mathbb{N}_0} \sum_{t\in\mathcal{C}_\ell} k_t^m \\ &\leq \sum_{\ell\in\mathbb{N}_0} (\alpha + \beta\ell)^{rm} \#\mathcal{C}_\ell \\ &\leq C_{\mathrm{bc}}(2C_{\mathrm{bn}} + C_{\mathrm{bc}})\Big(\sum_{\ell\in\mathbb{N}_0} (\alpha + \beta\ell)^{rm}\xi^{-\ell}\Big) c_\mathcal{I}.\end{aligned}$$

Due to Lemma 3.50, there is an $a_{rm} \in \mathbb{R}_{\geq 0}$ depending only on ξ, r and m with

$$\sum_{\ell\in\mathbb{N}_0} (\alpha + \beta\ell)^{rm}\xi^{-\ell} \leq (a_{rm} + 1)(\alpha + \beta)^{rm},$$

and we conclude

$$\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^m \leq C_{\mathrm{bc}}(2C_{\mathrm{bn}} + C_{\mathrm{bc}})(a_{rm} + 1)(\alpha + \beta)^{rm} c_\mathcal{I}.$$

We set $C_{\mathrm{cb}} := C_{\mathrm{bc}}(2C_{\mathrm{bn}} + C_{\mathrm{bc}})(a_{rm} + 1)$ to complete the proof. □

Using this result we can translate complexity estimates into a more manageable form:

Corollary 3.46 ($\mathcal{H}^2$-matrix complexity). *Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be $(C_{\mathrm{bc}}, \alpha, \beta, r, \xi)$-bounded cluster trees. Let V and W be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ with $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded rank distributions K and L. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible C_{sp}-sparse block cluster tree, and let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. The storage requirements for the $\mathcal{H}^2$-matrix representation of X are in $\mathcal{O}((\alpha+\beta)^{2r}(c_\mathcal{I} + c_\mathcal{J}))$, and the matrix-vector multiplication can be performed in $\mathcal{O}((\alpha+\beta)^{2r}(c_\mathcal{I}+c_\mathcal{J}))$ operations, with $c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}$ and $c_\mathcal{J} := \#\mathcal{T}_\mathcal{J}$ denoting the numbers of clusters in $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$.*

Proof. Combine Lemma 3.38 and Theorem 3.42 with Lemma 3.45. □

Typically, we are not interested in complexity estimates based on the number of clusters, but on estimates based on the number of degrees of freedom. In order to relate both numbers, we require lower bounds for the size of leaf clusters and for the number of sons of non-leaf clusters:

Definition 3.47 (Regular cluster tree). Let $\mathcal{T}_\mathcal{I}$ be a cluster tree. Let $C_{\mathrm{rc}} \in \mathbb{R}_{\geq 1}$, $\alpha \in \mathbb{R}_{>0}$, $\beta \in \mathbb{R}_{\geq 0}$, $r \in \mathbb{R}_{\geq 1}$ and $\xi \in \mathbb{R}_{>1}$. The tree $\mathcal{T}_\mathcal{I}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-*regular* if it is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-bounded and satisfies

$$\begin{aligned} C_{\mathrm{rc}} \#\hat{t} &\geq (\alpha + \beta)^r && \text{for all leaf clusters } t \in \mathcal{L}_\mathcal{I}, \\ \#\operatorname{sons}(t) &\geq 2 && \text{for all non-leaf clusters } t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}. \end{aligned}$$

The tree $\mathcal{T}_\mathcal{I}$ is (C_{rc}, k)-*regular* if it is (C_{rc}, k)-bounded and satisfies

$$\begin{aligned} C_{\mathrm{rc}} \#\hat{t} &\geq k && \text{for all } t \in \mathcal{L}_\mathcal{I}, \\ \#\operatorname{sons}(t) &\geq 2 && \text{for all non-leaf clusters } t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}. \end{aligned}$$

In the model case, the cluster tree is constructed by splitting clusters into two equal-sized sons until their size drops below $4m$. Due to the regular structure, each leaf cluster still contains at least $2m$ indices, so the cluster tree is $(2, 4m, 0, 1)$-regular.

Lemma 3.48 (Number of clusters). *Let $\mathcal{T}_\mathcal{I}$ be a $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular cluster tree. Then we have*

$$c_\mathcal{I} = \#\mathcal{T}_\mathcal{I} \leq 2C_{\mathrm{rc}} \frac{n_\mathcal{I}}{(\alpha + \beta)^r},$$

i.e., we can bound the number of clusters $c_\mathcal{I}$ by the number of indices.

Proof. According to Definition 3.47, each leaf of the cluster tree contains at least $C_{\mathrm{rc}}^{-1}(\alpha + \beta)^r$ indices. Due to Corollary 3.9, each index $i \in \mathcal{I}$ appears in exactly one leaf, and this means

$$\#\mathcal{L}_\mathcal{I} \leq \frac{n_\mathcal{I}}{C_{\mathrm{rc}}^{-1}(\alpha + \beta)^r} = C_{\mathrm{rc}} \frac{n_\mathcal{I}}{(\alpha + \beta)^r}.$$

Since $\#\operatorname{sons}(t) \geq 2$ holds for all non-leaf clusters $t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}$, we can use Lemma 3.52 to get

$$c_\mathcal{I} = \#\mathcal{T}_\mathcal{I} \leq 2\#\mathcal{L}_\mathcal{I} \leq 2C_{\mathrm{rc}} \frac{n_\mathcal{I}}{(\alpha + \beta)^r}.$$ □

Regular cluster trees allow us to bound the number of clusters by the number of indices and thus get complexity estimates similar to the ones derived for the model case:

Corollary 3.49 (Linear complexity). *Let V and W be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ with $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded rank distributions K and L. Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible C_{sp}-sparse block cluster tree, and let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. The storage requirements for the $\mathcal{H}^2$-matrix representation of X are in $\mathcal{O}((\alpha+\beta)^r(n_\mathcal{I}+n_\mathcal{J}))$, and the matrix-vector multiplication can be performed in $\mathcal{O}((\alpha + \beta)^r(n_\mathcal{I} + n_\mathcal{J}))$ operations.*

Proof. Combine Corollary 3.46 with Lemma 3.48. □

3.9 Technical lemmas

In order to derive bounds for the storage complexity for nested cluster bases, we have to investigate the relationship between the polynomial growth of $\#K_t$ and the exponential decay of the number of clusters. Our theory is based on the following estimate:

Lemma 3.50 (Exponential decay and polynomial growth). *Let $q \in [0,1)$. We define $(a_k)_{k\in\mathbb{N}_0}$ by the recurrence relation*

$$a_0 = \frac{1}{1-q}, \quad a_{k+1} = \frac{q}{1-q}\sum_{j=0}^{k}\binom{k+1}{j}a_j \tag{3.21}$$

and find

$$\sum_{\ell=0}^{\infty}(\alpha+\beta\ell)^k q^\ell \leq (a_k+1)(\alpha+\beta)^k \tag{3.22}$$

for all $\alpha,\beta \in \mathbb{R}_{\geq 0}$ and all $k \in \mathbb{N}_0$.

Proof. For all $n \in \mathbb{N}_0$ and $k \in \mathbb{N}_0$, we introduce the partial sums

$$a_k^n := \sum_{\ell=0}^{n}\ell^k q^\ell.$$

We prove $\lim_{n\to\infty} a_k^n = a_k$ by induction on $k \in \mathbb{N}_0$.

In the case $k = 0$, the geometric summation formula implies

$$a_0^n = \sum_{\ell=0}^{n} q^\ell = \frac{1-q^{\ell+1}}{1-q},$$

and this converges to $a_0 = 1/(1-q)$.

Let now $k \in \mathbb{N}_0$ be such that $\lim_{n\to\infty} a_j^n = a_j$ holds for all $j \in \{0,\ldots,k\}$. The definition of a_k^n implies

$$\begin{aligned} a_{k+1}^{n+1} &= \sum_{\ell=0}^{n+1}\ell^{k+1}q^\ell = \sum_{\ell=1}^{n+1}\ell^{k+1}q^\ell = \sum_{\ell=0}^{n}(\ell+1)^{k+1}q^{\ell+1} \\ &= q\sum_{\ell=0}^{n}\sum_{j=0}^{k+1}\binom{k+1}{j}\ell^j q^\ell = q\sum_{j=0}^{k+1}\binom{k+1}{j}\sum_{\ell=0}^{n}\ell^j q^\ell = q\sum_{j=0}^{k+1}\binom{k+1}{j}a_j^n \end{aligned}$$

for $n \in \mathbb{N}$, so we find

$$\frac{a_{k+1}^{n+1} - q a_{k+1}^n}{1-q} = \frac{q}{1-q}\sum_{j=0}^{k}\binom{k+1}{j}a_j^n. \tag{3.23}$$

Due to our assumption, the right-hand side of equation (3.23) converges from below to a_{k+1}, and due to

$$a_{k+1}^n = \frac{a_{k+1}^n - q a_{k+1}^n}{1-q} \leq \frac{a_{k+1}^{n+1} - q a_{k+1}^n}{1-q} = \frac{q}{1-q} \sum_{j=0}^{k} \binom{k+1}{j} a_j^n \leq a_{k+1},$$

the increasing sequence $(a_{k+1}^n)_{n \in \mathbb{N}_0}$ also converges to a limit $\alpha \leq a_{k+1}$. Taking equation (3.23) to the limit $n \to \infty$ yields $\alpha = a_{k+1}$ and concludes the induction, thus proving

$$\sum_{m=0}^{\infty} m^k q^m = a_k. \tag{3.24}$$

Due to

$$\sum_{m=0}^{\infty} (\alpha + \beta m)^k q^m \leq \alpha^k + \sum_{m=1}^{\infty} (\alpha m + \beta m)^k q^m \leq (\alpha+\beta)^k + (\alpha+\beta)^k \sum_{m=1}^{\infty} m^k q^m,$$

we can use equation (3.24) to prove (3.22). □

Lemma 3.51 (Bound of a_k). *The quantities $(a_k)_{k \in \mathbb{N}_0}$ defined in Lemma 3.50 satisfy*

$$a_k \leq \frac{1}{1-q} \left(\frac{q}{1-q} + \frac{1}{2} \right)^k k! \tag{3.25}$$

for all $k \in \mathbb{N}_0$.

Proof. We introduce

$$\alpha := \frac{q}{1-q} + \frac{1}{2} \geq \frac{1}{2}$$

and prove

$$a_j \leq \frac{1}{1-q} \alpha^j j! \tag{3.26}$$

for all $j \in \mathbb{N}_0$ by induction. For $j = 0$, this estimate is obvious. Let $k \in \mathbb{N}_0$ be such that (3.26) holds for all $j \in \{0, \dots, k\}$. The recurrence relation (3.21) implies

$$\begin{aligned} a_{k+1} &= \frac{q}{1-q} \sum_{j=0}^{k} \binom{k+1}{j} a_j \leq \frac{1}{1-q} \frac{q}{1-q} \sum_{j=0}^{k} \frac{(k+1)!}{(k+1-j)! j!} \alpha^j j! \\ &= \frac{q(k+1)!}{(1-q)^2} \sum_{j=0}^{k} \frac{\alpha^j}{(k+1-j)!} \leq \frac{q(k+1)!}{(1-q)^2} \sum_{j=0}^{k} \frac{\alpha^j}{2^{k-j}} \\ &= \frac{q(k+1)!}{(1-q)^2} 2^{-k} \sum_{j=0}^{k} (2\alpha)^j \end{aligned}$$

$$
\begin{aligned}
&= \frac{q(k+1)!}{(1-q)^2} 2^{-k} \frac{(2\alpha)^{k+1}-1}{2\alpha-1} \leq \frac{q(k+1)!}{(1-q)^2} 2^{-k} \frac{(2\alpha)^{k+1}}{2\alpha-1} \\
&= \frac{q(k+1)!}{(1-q)^2} 2 \frac{\alpha^{k+1}}{2\alpha-1} \\
&= \left(\frac{2q}{(1-q)^2} \frac{1}{2\alpha-1} \right) \alpha^{k+1} (k+1)! \\
&= \left(\frac{2q}{(1-q)^2} \frac{1-q}{2q} \right) \alpha^{k+1} (k+1)! \\
&= \frac{1}{1-q} \alpha^{k+1} (k+1)!,
\end{aligned}
$$

which proves (3.26) for $j = k+1$ and concludes the induction. □

Lemma 3.52 (Bounding nodes by leaves). *Let $\sigma \in \mathbb{N}_{\geq 2}$. Let $\mathcal{T}_I$ be a cluster tree satisfying*

$$\#\operatorname{sons}(t) \geq \sigma \quad \text{for all non-leaf clusters } t \in \mathcal{T}_I \setminus \mathcal{L}_I.$$

Then we have

$$\#\operatorname{sons}^*(t) \leq \frac{\sigma}{\sigma-1} \#(\mathcal{L}_I \cap \operatorname{sons}^*(t)) - \frac{1}{\sigma-1} \leq 2\#(\mathcal{L}_I \cap \operatorname{sons}^*(t)) \tag{3.27}$$

for all $t \in \mathcal{T}_I$. Applying this estimate to $t = \operatorname{root}(\mathcal{T}_I)$ yields

$$\#\mathcal{T}_I \leq \frac{\sigma}{\sigma-1} \#\mathcal{L}_I - \frac{1}{\sigma-1} \leq 2\#\mathcal{L}_I,$$

i.e., that the number of clusters can be bounded by the number of leaves.

Proof. We prove (3.27) for all $t \in \mathcal{T}_I$ by induction on $\#\operatorname{sons}^*(t) \in \mathbb{N}$.

Let $t \in \mathcal{T}_I$ with $\#\operatorname{sons}^*(t) = 1$. This implies $\operatorname{sons}(t) = \emptyset$, i.e., $t \in \mathcal{L}_I$ and

$$\#\operatorname{sons}^*(t) = 1 = \frac{\sigma-1}{\sigma-1} = \frac{\sigma}{\sigma-1} \#(\mathcal{L}_I \cap \operatorname{sons}^*(t)) - \frac{1}{\sigma-1}.$$

Now let $n \in \mathbb{N}$. We assume that (3.27) holds for all $t \in \mathcal{T}_I$ with $\operatorname{sons}^*(t) \leq n$. Let $t \in \mathcal{T}_I$ with $\#\operatorname{sons}^*(t) = n+1$. Due to $\#\operatorname{sons}^*(t) > 1$, we have

$$\operatorname{sons}^*(t) = \{t\} \cup \bigcup_{t' \in \operatorname{sons}(t)} \operatorname{sons}^*(t')$$

and observe

$$\#\operatorname{sons}^*(t) = 1 + \sum_{t' \in \operatorname{sons}(t)} \#\operatorname{sons}^*(t').$$

Since $\mathrm{sons}^*(t') \subseteq \mathrm{sons}^*(t) \setminus \{t\}$ holds, we have $\#\,\mathrm{sons}^*(t') \leq n$ and can use the assumption in order to conclude

$$\begin{aligned}
\#\,\mathrm{sons}^*(t) &= 1 + \sum_{t' \in \mathrm{sons}(t)} \#\,\mathrm{sons}^*(t') \\
&\leq 1 + \sum_{t' \in \mathrm{sons}(t)} \left(\frac{\sigma}{\sigma - 1} \#(\mathfrak{L}_{\mathcal{I}} \cap \mathrm{sons}^*(t')) - \frac{1}{\sigma - 1} \right) \\
&= \frac{\sigma}{\sigma - 1} \#(\mathfrak{L}_{\mathcal{I}} \cap \mathrm{sons}^*(t)) + 1 - \frac{\#\,\mathrm{sons}(t)}{\sigma - 1} \\
&\leq \frac{\sigma}{\sigma - 1} \#(\mathfrak{L}_{\mathcal{I}} \cap \mathrm{sons}^*(t)) + \frac{\sigma - 1}{\sigma - 1} - \frac{\sigma}{\sigma - 1} \\
&= \frac{\sigma}{\sigma - 1} \#(\mathfrak{L}_{\mathcal{I}} \cap \mathrm{sons}^*(t)) - \frac{1}{\sigma - 1}.
\end{aligned}$$

□

Chapter 4
Application to integral operators

We now consider the application of the general $\mathcal{H}^2$-matrix structures introduced in Chapter 3 to a class of problems in which densely populated matrices occur naturally: the numerical treatment of integral operators.

Our approach is similar to the one applied to the model problem in Chapter 2. We find a separable approximation of the kernel function underlying the integral operator, and this approximation gives rise to an $\mathcal{H}^2$-matrix approximation of the corresponding stiffness matrix. This is also the basic idea of the well-known panel clustering [72] and multipole [88], [58] techniques.

A separable approximation can be constructed in many different ways. The generalization of the truncated Taylor expansion used in the model problem (and in the original paper [72] introducing the panel clustering algorithm) to the multi-dimensional setting is straightforward and yields an efficient technique if the coefficients of the Taylor expansion, i.e., arbitrary derivatives of the kernel function up to a certain order, can be computed efficiently. The latter requirement can be satisfied by using a recursion formula for the derivatives, but since these have to be derived by hand for each individual kernel function, the range of problems that can be covered by Taylor expansions is limited. Typical multipole methods [58], [60] use a more efficient approximation of the kernel function instead of the Taylor expansion, but they share the latter's limitations to specific functions.

A more general approach is based on interpolation: instead of approximating the kernel function by a Taylor expansion, we interpolate it by a polynomial [45], [65]. Using tensor-product interpolation operators in the multi-dimensional setting leads to a method that requires only pointwise evaluations of the kernel function instead of derivatives, so it is far better suited for general kernel functions and its implementation is very simple. The theoretical investigation of the interpolation approach is significantly more complicated than in the Taylor-based case, but it also yields significantly better results: it is possible to prove that the interpolant will always converge as long as the kernel function is locally analytic, and that in situations in which both interpolant and Taylor expansion converge, the interpolant will converge faster.

In practical applications, we sometimes have to handle integral operators based on derivatives of analytic kernel functions. Instead of approximating the derivative directly, we can also use the derivatives of an approximation. The latter approach is usually preferable from the viewpoint of implementation, and it is possible to prove similar convergence rates for both techniques.

Using the error estimates derived for the approximation of the kernel function, it is possible to find bounds for the error of the matrix approximation both in the Frobenius and the spectral norm. In case of the spectral norm, the error in each block is determined

by the product of the kernel approximation error and the ratio of the measure of the support of a cluster and a power of its diameter.

If the singularity of the kernel function is sufficiently weak, the error is dominated by the large clusters and converges to zero as the clusters shrink. This suggests a variable-order scheme [91], [23], [100]: if we increase the order of the approximation in large clusters, we can ensure convergence of the global spectral error without harming the asymptotic complexity. In order to ensure that the cluster bases are nested despite the varying orders, the construction of the approximation scheme has to be modified. It can be proven that only a minimal change in the implementation is required to provide us with a stable and convergent method.

This chapter is organized as follows:

- Section 4.1 introduces the basic problem we want to solve.
- Section 4.2 describes a general symmetric panel-clustering approach and demonstrates that it leads to an $\mathcal{H}^2$-matrix approximation of the relevant stiffness matrix.
- Section 4.3 summarizes the well-known basic properties of Taylor approximations of asymptotically smooth kernel functions.
- Section 4.4 describes the approach based on interpolation of the kernel function and presents the corresponding approximation results (cf. [65], [23]).
- Section 4.5 gives error estimates for the derivatives of interpolants, which are useful for approximating, e.g., the classical double layer potential operator or the hyper-singular operator (cf. Section 4.1 in [17]).
- Section 4.6 uses the results of the previous three sections to establish error estimates for the $\mathcal{H}^2$-matrix in the spectral and Frobenius norm.
- Section 4.7 introduces the variable-order interpolation scheme (cf. [91], [23]).
- Section 4.8 contains a number of auxiliary results required in the other sections. The proofs are only included for the sake of completeness.
- Section 4.9 presents numerical experiments that demonstrate that the theoretical error and complexity estimates are close to optimal.

Assumptions in this chapter: We assume that cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ for the finite index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, are given. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let $n_\mathcal{I} := \#\mathcal{I}$ and $n_\mathcal{J} := \#\mathcal{J}$ denote the number of indices in $\mathcal{I}$ and $\mathcal{J}$, and let $c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}$ and $c_\mathcal{J} := \#\mathcal{T}_\mathcal{J}$ denote the number of clusters in $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$.

4.1 Integral operators

In the model problem described in Chapter 2, we have considered the integral operator

$$\mathcal{G}[u](x) := -\int_0^1 \log|x-y|\, u(y)\, dy.$$

Its discretization by Galerkin's method using a family $(\varphi_i)_{i=1}^n$ of basis functions leads to a matrix $G \in \mathbb{R}^{n\times n}$ given by

$$G_{ij} := -\int_0^1 \varphi_i(x) \int_0^1 \log|x-y|\, \varphi_j(y)\, dy\, dx \quad \text{for all } i,j \in \{1,\ldots,n\},$$

which is dense in general. Handling this matrix directly leads to an algorithmic complexity of $\mathcal{O}(n^2)$, i.e., the method will require too much storage and too much time for large values of n. Using an $\mathcal{H}^2$-matrix approximation, it is possible to reduce the complexity to $\mathcal{O}(nm)$, where $m \in \mathbb{N}$ is a parameter controlling the accuracy of the approximation.

Let us now consider the general integral operator

$$\mathcal{G}[u](x) := \int_\Omega g(x,y)u(y)\, dy, \tag{4.1}$$

defined for a suitable compact subdomain or submanifold of $\mathbb{R}^d$, a suitable *kernel function* g and functions u from a suitable linear space.

Discretizing the operator $\mathcal{G}$ by a Petrov–Galerkin method using finite families $(\varphi_i)_{i\in\mathcal{I}}$ and $(\psi_j)_{j\in\mathcal{J}}$ of basis functions leads to a matrix $G \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ given by

$$G_{ij} := \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\psi_j(y)\, dy\, dx \quad \text{for all } i \in \mathcal{I}, j \in \mathcal{J}. \tag{4.2}$$

In typical applications, the kernel function g has global support, i.e., in general the matrix G is dense. Handling the matrix G directly leads to algorithms with a complexity of $\mathcal{O}(n_{\mathcal{I}} n_{\mathcal{J}})$, which is unacceptable for large-scale computations.

4.2 Low-rank approximation

Using the techniques introduced in Section 3.3, we can construct suitable cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ for the index sets $\mathcal{I}$ and $\mathcal{J}$ and an admissible block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ for these trees.

As in the case of the model problem, the approximation of the matrix G will be based on local separable approximations of the kernel function g.

We define families of supports $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ for $\mathcal{I}$ and $\mathcal{J}$ by

$$\Omega_i := \operatorname{supp}\varphi_i \quad \text{for all } i \in \mathcal{I}, \qquad \Omega_j := \operatorname{supp}\psi_j \quad \text{for all } j \in \mathcal{J}.$$

As in Definition 3.20, we define the supports of clusters by

$$\Omega_t := \bigcup_{i \in \hat{t}} \Omega_i \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$
$$\Omega_s := \bigcup_{j \in \hat{s}} \Omega_j \quad \text{for all } s \in \mathcal{T}_{\mathcal{J}}.$$

Based on these subsets of Ω, we can now introduce "nested cluster bases" in infinite-dimensional spaces of functions:

Definition 4.1 (Expansion system). Let $K = (K_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ be a rank distribution for $\mathcal{T}_{\mathcal{I}}$. A family $(v_{t,\nu})_{t \in \mathcal{T}_{\mathcal{I}}, \nu \in K_t}$ of functions defined on the domain Ω is called an *expansion system* for $\mathcal{T}_{\mathcal{I}}$ with rank distribution K.

An expansion system $(v_{t,\nu})_{t \in \mathcal{T}_{\mathcal{I}}, \nu \in K_t}$ is called *nested* if there is a family $(E_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ of matrices satisfying the following conditions:

- For all $t \in \mathcal{T}_{\mathcal{I}}$ and all $t' \in \operatorname{sons}(t)$, we have $E_{t'} \in \mathbb{R}^{K_{t'} \times K_t}$.
- For all $t \in \mathcal{T}_{\mathcal{I}}$ and all $t' \in \operatorname{sons}(t)$, we have

$$v_{t,\nu}(x) = \sum_{\nu' \in K_{t'}} (E_{t'})_{\nu'\nu} v_{t',\nu'}(x) \quad \text{for all } \nu \in K_t, x \in \Omega_{t'}.$$

The matrices E_t are called *transfer matrices* or *expansion matrices*.

Let $K = (K_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ and $L = (L_s)_{s \in \mathcal{T}_{\mathcal{J}}}$ be rank distributions for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Let $(v_{t,\nu})_{t \in \mathcal{T}_{\mathcal{I}}, \nu \in K_t}$ be a nested expansion system for $\mathcal{T}_{\mathcal{I}}$ with expansion matrices $(E_t)_{t \in \mathcal{T}_{\mathcal{I}}}$. Let $(w_{s,\mu})_{s \in \mathcal{T}_{\mathcal{J}}, \mu \in L_s}$ be a nested expansion system for $\mathcal{T}_{\mathcal{J}}$ with expansion matrices $(F_s)_{s \in \mathcal{T}_{\mathcal{J}}}$. Let $S = (S_b)_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}}$ be a family of matrices satisfying $S_b \in \mathbb{R}^{K_t \times L_s}$ for all admissible blocks $b = (t, s) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}$.

For all admissible blocks $b = (t, s) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}$ in $\mathcal{T}_{\mathcal{I} \times \mathcal{J}}$, we can define the separable approximation

$$\tilde{g}_{t,s}(x, y) := \sum_{\nu \in K_t} \sum_{\mu \in L_s} (S_b)_{\nu\mu} v_{t,\nu}(x) w_{s,\mu}(y) \quad \text{for all } x \in \Omega_t, y \in \Omega_s \tag{4.3}$$

of the function g in the subdomain $\Omega_t \times \Omega_s$ corresponding to the supports of t and s. Replacing g by $\tilde{g}_{t,s}$ in (4.2) gives rise to an approximation

$$\begin{aligned}
(\tilde{G}_{t,s})_{ij} &:= \int_\Omega \varphi_i(x) \int_\Omega \tilde{g}_{t,s}(x, y) \psi_j(y) \, dy \, dx \\
&= \sum_{\nu \in K_t} \sum_{\mu \in L_s} (S_b)_{\nu\mu} \int_\Omega \varphi_i(x) v_{t,\nu}(x) \, dx \int_\Omega \psi_j(y) w_{s,\mu}(y) \, dy \qquad (4.4) \\
&= \sum_{\nu \in K_t} \sum_{\mu \in L_s} (S_b)_{\nu\mu} (V_t)_{i\nu} (W_s)_{j\mu} = (V_t S_b W_s^*)_{ij} \quad \text{for all } i \in \hat{t}, j \in \hat{s},
\end{aligned}$$

where the matrices $V_t \in \mathbb{R}_{\hat{t}}^{\mathcal{I}\times K_t}$ and $W_s \in \mathbb{R}_{\hat{s}}^{\mathcal{J}\times L_s}$ are given by

$$(V_t)_{iv} := \begin{cases} \int_\Omega v_{t,v}(x)\varphi_i(x)\,dx & \text{if } i \in \hat{t}, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, i \in \mathcal{I}, v \in K_t,$$

$$(W_s)_{j\mu} := \begin{cases} \int_\Omega w_{s,\mu}(y)\psi_j(y)\,dy & \text{if } j \in \hat{s}, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J}, j \in \mathcal{J}, \mu \in L_s.$$

For all $t \in \mathcal{T}_\mathcal{I}$, $t' \in \operatorname{sons}(t)$, $i \in \hat{t}'$ and $v \in K_t$, we observe

$$\begin{aligned}(V_t)_{iv} &:= \int_\Omega v_{t,v}(x)\varphi_i(x)\,dx = \int_{\Omega_{t'}} v_{t,v}(x)\varphi_i(x)\,dx \\ &= \sum_{v'\in K_{t'}} \int_{\Omega_{t'}} (E_t)_{v'v} v_{t',v'}(x)\varphi_i(x)\,dx = \sum_{v'\in K_{t'}} (E_t)_{v'v}(V_{t'})_{iv'} = (V_{t'}E_{t'})_{iv},\end{aligned}$$

and this is equivalent to

$$V_t = \sum_{t'\in\operatorname{sons}(t)} V_{t'}E_{t'},$$

i.e., the family $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ is a nested cluster basis for the row cluster tree $\mathcal{T}_\mathcal{I}$ with transfer matrices $(E_t)_{t\in\mathcal{T}_\mathcal{I}}$. By the same arguments, we can prove that $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ is a nested cluster basis for the column cluster tree $\mathcal{T}_\mathcal{J}$ with transfer matrices $(F_s)_{s\in\mathcal{T}_\mathcal{J}}$.

Using the family $S = (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of coupling matrices, we find that we have constructed an $\mathcal{H}^2$-matrix approximation

$$\widetilde{G} = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_b W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t G \chi_s \tag{4.5}$$

of the matrix G.

If $\tilde{g}_{t,s}|_{\Omega_t\times\Omega_s}$ is a sufficiently accurate approximation of $g|_{\Omega_t\times\Omega_s}$, the factorized matrix $\widetilde{G}_{t,s} := V_t S_b W_s^*$ is an approximation of the block $\chi_t G \chi_s$ of the original matrix G.

Remark 4.2 (General discretization schemes). Introducing the families of functionals $(\Phi_i)_{i\in\mathcal{I}}$ and $(\Psi_j)_{j\in\mathcal{J}}$ defined by

$$\Phi_i(f) := \int_\Omega f(x)\varphi_i(x)\,dx, \quad \Psi_j(f) := \int_\Omega f(y)\psi_j(y)\,dy,$$

we can write equation (4.2) in the form

$$G_{ij} = (\Phi_i \otimes \Psi_j)(g) \quad \text{for all } i \in \mathcal{I}, j \in \mathcal{J}.$$

For an admissible block $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, replacing g by $\tilde{g}_{t,s}$ yields

$$\begin{aligned}(\widetilde{G}_{t,s})_{ij} &= (\Phi_i \otimes \Psi_j)(\tilde{g}_{t,s}) = \sum_{\nu\in K_t}\sum_{\mu\in L_s} (S_b)_{\nu\mu}\Phi_i(v_{t,\nu})\Psi_j(w_{s,\mu}) \\ &= (V_t S_b W_s^*)_{ij} \quad \text{for all } i \in \hat{t},\ j \in \hat{s},\end{aligned}$$

where the matrices $V_t \in \mathbb{R}^{\mathcal{I}\times K_t}_{\hat{t}}$ and $W_s \in \mathbb{R}^{\mathcal{J}\times L_s}_{\hat{s}}$ are defined by

$$(V_t)_{i\nu} = \begin{cases}\Phi_i(v_{t,\nu}) & \text{if } i \in \hat{t},\\ 0 & \text{otherwise,}\end{cases}$$

$$(W_s)_{j\mu} = \begin{cases}\Psi_j(w_{s,\mu}) & \text{if } j \in \hat{s},\\ 0 & \text{otherwise}\end{cases}$$

for all $i \in \mathcal{I}$, $j \in \mathcal{J}$, $\nu \in K_t$ and $\mu \in L_s$.

Using this more general form allows us to handle general discretization schemes: a Nyström method corresponds to functionals

$$\Phi_i(f) := f(\xi_i), \quad \Psi_j(f) := f(\zeta_j)$$

for families $(\xi_i)_{i\in\mathcal{I}}$ and $(\zeta_j)_{j\in\mathcal{J}}$ of interpolation points, while a collocation method corresponds to

$$\Phi_i(f) := f(\xi_i), \quad \Psi_j(f) := \int_\Omega f(y)\psi_j(y)\,dy$$

for collocation points $(\xi_i)_{i\in\mathcal{I}}$ and basis functions $(\psi_j)_{j\in\mathcal{J}}$. □

4.3 Approximation by Taylor expansion

We have seen that a separable approximation of the type

$$\tilde{g}_{t,s}(x, y) = \sum_{\nu\in K_t}\sum_{\mu\in L_s} s_{t,s,\nu,\mu} v_{t,\nu}(x) w_{s,\mu}(y)$$

can be used to find factorized representations for admissible blocks $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

In the model problem discussed in Chapter 2, this approximation is derived by using the Taylor expansion of g, which leads to (2.2).

Multi-dimensional Taylor expansion

In order to generalize this approach to the d-dimensional case, we rely on the usual multi-index notation: a tuple $\nu \in \mathbb{N}_0^d$ is called a *multi-index*. The norm and the factorial of multi-indices are defined by

$$|\nu| := \sum_{\iota=1}^{d} \nu_\iota = \nu_1 + \cdots + \nu_d \qquad \text{for all } \nu \in \mathbb{N}_0^d,$$

$$\nu! := \prod_{\iota=1}^{d} \nu_\iota! = (\nu_1!) \dots (\nu_d!) \qquad \text{for all } \nu \in \mathbb{N}_0^d.$$

A partial order on the set of multi-indices in $\mathbb{N}_0^d$ is defined by

$$\nu \leq \mu :\iff \forall \iota \in \{1, \dots, d\} : \nu_\iota \leq \mu_\iota \qquad \text{for all } \nu, \mu \in \mathbb{N}_0^d.$$

We can raise a vector to a multi-index power given by

$$p^\nu := \prod_{\iota=1}^{d} p_\iota^{\nu_\iota} = p_1^{\nu_1} \dots p_d^{\nu_d} \qquad \text{for all } \nu \in \mathbb{N}_0^d, p \in \mathbb{R}^d.$$

As a "special case" of this notation, we introduce the partial differential operator

$$\partial^\nu := \partial_1^{\nu_1} \dots \partial_d^{\nu_d} \qquad \text{for all } \nu \in \mathbb{N}_0^d.$$

Using these notations, we can introduce the truncated Taylor expansion: let $z_0 \in \mathbb{R}^d$, let $\omega \subseteq \mathbb{R}^d$ be a star-shaped domain with center z_0, let $m \in \mathbb{N}$ and let $f \in C^m(\omega)$. The m-th order Taylor approximation of f in z_0 is given by

$$\tilde{f}_{z_0,m} \colon \omega \to \mathbb{R}, \quad z \mapsto \sum_{|\nu|<m} \partial^\nu f(z_0) \frac{(z - z_0)^\nu}{\nu!}.$$

Separable approximation

In the model case discussed in Chapter 2, the kernel function g is shift-invariant, i.e., we have $g(x, y) = g(x - y, 0)$, which allows us to treat g as a function of only one variable and apply a Taylor approximation directly.

In the general case, we cannot assume shift-invariance. Instead, we use the fact that we have multi-dimensional Taylor expansions at our disposal: we handle the kernel function g directly as a $2d$-dimensional function.

We assume that d-dimensional balls $\mathcal{K}_t$ and $\mathcal{K}_s$ satisfying $\Omega_t \subseteq \mathcal{K}_t$ and $\Omega_s \subseteq \mathcal{K}_s$ are given. Since $\mathcal{K}_t$ and $\mathcal{K}_s$ are star-shaped with respect to their centers x_t and y_s,

respectively, the $2d$-dimensional domain $\omega := \mathcal{K}_t \times \mathcal{K}_s$ is star-shaped with respect to $z_0 := (x_t, y_s) \in \mathbb{R}^{2d}$.

We assume that $g|_\omega \in C^m(\omega)$ holds, and let $\tilde{g}_{t,s}$ be the m-th order Taylor expansion of g in z_0. It is given by

$$\begin{aligned}\tilde{g}_{t,s}(x,y) &= \sum_{|\nu+\mu|<m} \frac{\partial^{(\nu,\mu)} g(x_t, y_s)}{(\nu,\mu)!}(x-x_t, y-y_s)^{(\nu,\mu)} \\ &= \sum_{|\nu+\mu|<m} \frac{d^{\nu+\mu} g(x_t, y_s)}{dx^\nu dy^\mu} \frac{(x-x_t)^\nu}{\nu!} \frac{(y-y_s)^\mu}{\mu!}\end{aligned} \tag{4.6}$$

for all $x \in \mathcal{K}_t$ and all $y \in \mathcal{K}_s$. We can see that this function is of the form (4.3) if we let

$$K_t = L_s = \{\nu \in \mathbb{N}_0^d \,:\, |\nu| < m\}.$$

As in the general case, the separable expansion gives rise to an approximation

$$\chi_t G \chi_s \approx V_t S_b W_s^*$$

of the matrix block corresponding to $b = (t,s)$, where the matrices $V_t \in \mathbb{R}^{\mathcal{I} \times K_t}_{\hat{t}}$, $W_s \in \mathbb{R}^{\mathcal{J} \times L_s}$ and $S_b \in \mathbb{R}^{K_t \times L_s}$ are given by

$$(V_t)_{i\nu} := \begin{cases} \int_\Omega \frac{(x-x_t)^\nu}{\nu!} \varphi_i(x)\,dx & \text{if } i \in \hat{t}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{I},\ \nu \in K_t,$$

$$(W_s)_{j\mu} := \begin{cases} \int_\Omega \frac{(y-y_s)^\mu}{\mu!} \psi_j(y)\,dy & \text{if } j \in \hat{s}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } j \in \mathcal{J},\ \mu \in L_s,$$

$$(S_b)_{\nu\mu} := \begin{cases} \frac{d^{\nu+\mu} g}{dx^\nu dy^\mu}(x_t, y_s) & \text{if } |\nu+\mu| < m, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu \in K_t,\ \mu \in L_s.$$

In order to reach the optimal order of complexity, the cluster bases $V := (V_t)_{t \in \mathcal{T}_\mathcal{I}}$ and $W := (W_s)_{s \in \mathcal{T}_\mathcal{J}}$ have to be nested. In the case of Taylor expansions, we find

$$\begin{aligned}\frac{(x-x_t)^\nu}{\nu!} &= \frac{(x-x_{t'}+x_{t'}-x_t)^\nu}{\nu!} = \frac{1}{\nu!} \sum_{\nu' \le \nu} \binom{\nu}{\nu'} (x-x_{t'})^{\nu'} (x_{t'}-x_t)^{\nu-\nu'} \\ &= \sum_{\nu' \le \nu} \frac{(x-x_{t'})^{\nu'}}{\nu'!} \frac{(x_{t'}-x_t)^{\nu-\nu'}}{(\nu-\nu')!}\end{aligned}$$

for all $t, t' \in \mathcal{T}_\mathcal{I}$ and all $\nu \in K_t$. This identity implies that for all $t \in \mathcal{T}_\mathcal{I}$ and all $t' \in \operatorname{sons}(t)$ the matrices $E_{t'} \in \mathbb{R}^{K_{t'} \times K_t}$ defined by

$$(E_{t'})_{\nu'\nu} := \begin{cases} \frac{(x_{t'}-x_t)^{\nu-\nu'}}{(\nu-\nu')!} & \text{if } \nu' \le \nu, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu' \in K_{t'}, \nu \in K_t,$$

satisfy the equation

$$V_t = \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'},$$

that is, the matrices $(E_t)_{t \in \mathcal{T}_\mathcal{I}}$ are transfer matrices for the nested cluster basis $V = (V_t)_{t \in \mathcal{T}_\mathcal{I}}$. A family $(F_s)_{s \in \mathcal{T}_\mathcal{J}}$ of transfer matrices for the nested cluster basis $W = (W_s)_{s \in \mathcal{T}_\mathcal{J}}$ can be constructed by a similar argument.

Remark 4.3 (Implementation). Computing the entries of V_t and W_s is straightforward: if the basis functions φ_i and ψ_j are piecewise polynomials, we can use standard quadrature rules to evaluate the integrals. If $\nu, \mu \in K_t$ satisfy

$$\mu = (\nu_1, \ldots, \nu_{\iota-1}, \nu_\iota + 1, \nu_{\iota+1}, \ldots, \nu_d)$$

for a $\iota \in \{1, \ldots, d\}$, we get

$$\frac{(x - x_t)^\mu}{\mu!} = \frac{(x - x_t)^\nu}{\nu!} \frac{x_\iota - x_{t,\iota}}{\nu_\iota + 1},$$

and this equation allows us to use an efficient recurrence to evaluate the monomials in a given quadrature point rapidly.

The computation of the transfer matrices E_t and F_s is even simpler: we only have to evaluate monomials and factorials at points in $\mathbb{R}^d$, no quadrature is required.

The construction of the matrices S_b is more challenging. In order to find an efficient algorithm, recursive expressions for the derivatives of g have to be derived by hand. □

Lemma 4.4 (Complexity). *Let $m \in \mathbb{N}$. We have*

$$\#K_t = \binom{m-1+d}{d} \leq m^d \quad \text{for all } t \in \mathcal{T}_\mathcal{I},$$

i.e., the rank distribution is m^d-bounded.

Proof. According to Lemma 4.74, we have

$$\#K_t = \#\{\nu \in \mathbb{N}_0^d \,:\, |\nu| < m\} = \binom{m-1+d}{d}.$$

Elementary computations yield

$$\begin{aligned}\binom{m-1+d}{d} = \frac{(m-1+d)!}{(m-1)!d!} &= \prod_{k=1}^{d} \frac{m-1+k}{k} \\ &\leq \prod_{k=1}^{d} \frac{k(m-1)+k}{k} = \prod_{k=1}^{d} m = m^d,\end{aligned}$$

so we conclude $\#K_t \leq m^d$. □

Error analysis

Let us now investigate the error introduced by the truncated Taylor expansion. The multi-dimensional counterpart of the representation (2.5) of this error is given by Lemma 4.75. In order to stress the similarities between the multi-dimensional error term (4.79) and its one-dimensional counterpart (2.5), we introduce the *directional derivative*

$$\partial_p^m f(z) := \sum_{|\nu|=m} \frac{m!}{\nu!} \partial^\nu f(z) p^\nu$$

for $z \in \omega$ and a vector $p \in \mathbb{R}^d$. Using $p := z - z_0$ in Lemma 4.75 yields

$$f(z) - \tilde{f}_{z_0,m}(z) = \frac{1}{(m-1)!} \int_0^1 (1-t)^{m-1} \partial_p^m f(z_0 + tp)\, dt. \tag{4.7}$$

Let us now investigate the approximation properties of the function $\tilde{g}_{t,s}$. In order to prove useful bounds for the approximation error introduced by replacing g with $\tilde{g}_{t,s}$, we have to be able to control the growth of the derivatives of the original kernel function.

Since we frequently have to work with quotients of factorials, we use the notation

$$\begin{bmatrix} n \\ k \end{bmatrix} := \frac{(n+k)!}{k!} = \begin{cases} \prod_{\ell=1}^n (k+\ell) & \text{if } n > 0, \\ 1 & \text{otherwise} \end{cases} \quad \text{for all } n, k \in \mathbb{N}_0. \tag{4.8}$$

Definition 4.5 (Asymptotically smooth kernels). Let $g \in \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Let $C_{\mathrm{as}} \in \mathbb{R}_{>0}$, $c_0 \in \mathbb{R}_{\geq 1}$ and $\sigma \in \mathbb{N}$.

The function g is called $(C_{\mathrm{as}}, \sigma, c_0)$-*asymptotically smooth* (cf. [28], [27]) if

$$|\partial_p^\nu g(x,y)| \leq C_{\mathrm{as}} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \frac{c_0^\nu \|p\|_2^\nu}{\|x-y\|_2^{\sigma+\nu}} \tag{4.9}$$

holds for all $\nu \in \mathbb{N}_0$, $x, y \in \mathbb{R}^d$ with $x \neq y$ and all directions $p \in \mathbb{R}^d \times \mathbb{R}^d$.

For $\sigma = 0$, the function g is called $(C_{\mathrm{as}}, 0, c_0)$-*asymptotically smooth* if

$$|\partial_p^\nu g(x,y)| \leq C_{\mathrm{as}} (\nu-1)! \frac{c_0^\nu \|p\|_2^\nu}{\|x-y\|_2^\nu} \tag{4.10}$$

holds for all $\nu \in \mathbb{N}$, $x, y \in \mathbb{R}^d$ with $x \neq y$ and all directions $p \in \mathbb{R}^d \times \mathbb{R}^d$.

In this context σ is called the *order of the singularity* of g, $\sigma = 0$ corresponds to logarithmic singularities.

Example 4.6. The most important examples of asymptotically smooth kernel functions are

$$g_3 \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}, \quad (x,y) \mapsto \begin{cases} \frac{1}{4\pi} \frac{1}{\|x-y\|_2} & \text{if } x \neq y, \\ 0 & \text{otherwise,} \end{cases}$$

the fundamental solution of Poisson's equation in three-dimensional space, and

$$g_2 \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \quad (x,y) \mapsto \begin{cases} -\frac{1}{2\pi} \log \|x-y\|_2 & \text{if } x \neq y, \\ 0 & \text{otherwise,} \end{cases}$$

its two-dimensional counterpart.

According to [63], Appendix E, both functions are asymptotically smooth: for a given $c_0 > 1$, [63], Satz E.2.1, yields a constant $C_{\mathrm{as}} \in \mathbb{R}_{>0}$ such that g_3 is $(C_{\mathrm{as}}, 1, c_0)$-asymptotically smooth and g_2 is $(C_{\mathrm{as}}, 0, c_0)$-asymptotically smooth.

In order to be able to formulate the approximation error estimate in the familiar terms of diameters and distances, we require the following result (which is obvious if considered geometrically, cf. Figure 4.1):

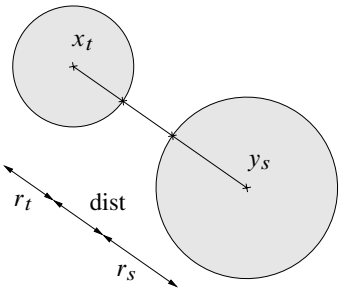


Figure 4.1. Distance of the centers x_t and y_s of two circles expressed by the distance of the circles and their radii.

Lemma 4.7 (Distance of centers). *Let $x_t \in \mathcal{K}_t$ and $y_s \in \mathcal{K}_s$ be the centers of the balls $\mathcal{K}_t$ and $\mathcal{K}_s$, respectively. Let $r_t \in \mathbb{R}_{\geq 0}$ and $r_s \in \mathbb{R}_{\geq 0}$ be the radii of $\mathcal{K}_t$ and $\mathcal{K}_s$, respectively. If*

$$\operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) > 0$$

holds, we have

$$\|x_t - y_s\|_2 \geq \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) + r_t + r_s.$$

Proof. We define the continuous functions

$$\begin{aligned} x \colon [0, r_t] &\to \mathbb{R}^d, & \alpha &\mapsto x_t - \alpha \frac{x_t - y_s}{\|x_t - y_s\|_2}, \\ y \colon [0, r_s] &\to \mathbb{R}^d, & \beta &\mapsto y_s + \beta \frac{x_t - y_s}{\|x_t - y_s\|_2}, \\ h \colon [0, r_t] \times [0, r_s] &\to \mathbb{R}, & (\alpha, \beta) &\mapsto \|x_t - y_s\|_2 - \alpha - \beta, \end{aligned}$$

and observe

$$x(\alpha) \in \mathcal{K}_t, \quad y(\beta) \in \mathcal{K}_s \quad \text{for all } \alpha \in [0, r_t], \beta \in [0, r_s],$$

which implies

$$\begin{aligned} 0 < \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) \leq \|x(\alpha) - y(\beta)\|_2 &= \left\| (\|x_t - y_s\|_2 - \alpha - \beta) \frac{x_t - y_s}{\|x_t - y_s\|_2} \right\|_2 \\ &= |\|x_t - y_s\|_2 - \alpha - \beta| = |h(\alpha, \beta)| \end{aligned}$$

for all $\alpha \in [0, r_t]$ and $\beta \in [0, r_s]$. Since h is continuous with

$$h(0,0) = \|x_t - y_s\|_2 \geq \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) > 0,$$

we conclude $h(\alpha, \beta) > 0$ for all $\alpha \in [0, r_t]$ and all $\beta \in [0, r_s]$, i.e.,

$$\operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) \leq |h(r_t, r_s)| = h(r_t, r_s) = \|x_t - y_s\|_2 - r_t - r_s,$$

and this is the desired estimate. □

Now we can proceed to prove an estimate for the approximation error resulting from Taylor approximation:

Theorem 4.8 (Approximation error). *Let $\eta \in \mathbb{R}_{>0}$. Let $\mathcal{K}_t$ and $\mathcal{K}_s$ be d-dimensional balls satisfying the admissibility condition*

$$\operatorname{diam}(\mathcal{K}_t) + \operatorname{diam}(\mathcal{K}_s) \leq 2\eta \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s). \tag{4.11}$$

Let the kernel function g be $(C_{\mathrm{as}}, \sigma, c_0)$-asymptotically smooth. Let $x_t \in \mathcal{K}_t$ and $y_s \in \mathcal{K}_s$ be the centers of $\mathcal{K}_t$ and $\mathcal{K}_s$, respectively. Then

$$|g(x,y) - \tilde{g}_{t,s}(x,y)| \leq \begin{cases} C_{\mathrm{as}} c_0 \log(\eta + 1) \left(\frac{c_0 \eta}{\eta+1}\right)^{m-1} & \text{if } \sigma = 0, \\ C_{\mathrm{as}} \binom{m-1}{\sigma} \frac{c_0}{\operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s)^\sigma} \left(\frac{c_0 \eta}{\eta+1}\right)^{m-1} & \text{otherwise} \end{cases} \tag{4.12}$$

holds for all $m \in \mathbb{N}$, $x \in \mathcal{K}_t$ and all $y \in \mathcal{K}_s$.

Proof. Combining equation (4.7) with the bound (4.9) yields

$$\begin{aligned} |g(x,y) - \tilde{g}_{t,s}(x,y)| &= |g(x,y) - \tilde{g}_{z_0,m}(x,y)| \\ &= \frac{1}{(m-1)!} \int_0^1 (1-t)^{m-1} |\partial_p^m f(z_0 + pt)| \, dt \\ &\leq \hat{C} c_0^m \|p\|_2^m \int_0^1 \frac{(1-t)^{m-1}}{\|(x_t - y_s) + (x - x_t - (y - y_s))t\|_2^{m+\sigma}} \, dt \end{aligned}$$

for $p = (x - x_t, y - y_s)$, $z_0 = (x_t, y_s)$ and

$$\widehat{C} := \begin{cases} \frac{C_{\mathrm{as}}}{(m-1)!}(m-1)! = C_{\mathrm{as}} & \text{if } \sigma = 0, \\ \frac{C_{\mathrm{as}}}{(m-1)!}\left[\begin{smallmatrix} m \\ \sigma-1 \end{smallmatrix}\right] = C_{\mathrm{as}} \frac{(m-1+\sigma)!}{(m-1)!\,(\sigma-1)!} & \text{otherwise.} \end{cases}$$

In order to derive a useful estimate, we have to find a bound for the integral term.

To this end, we introduce relative coordinates $\zeta := x - y$ and $\zeta_0 := x_t - y_s$ and observe

$$|g(x,y) - \tilde{g}_{t,s}(x,y)| \leq \widehat{C} c_0^m \|p\|_2^m \int_0^1 \frac{(1-t)^{m-1}}{\|\zeta_0 + (\zeta - \zeta_0)t\|_2^{m+\sigma}}\, dt. \tag{4.13}$$

Let $r_t := \operatorname{diam}(\mathcal{K}_t)/2$ and $r_s := \operatorname{diam}(\mathcal{K}_s)/2$ denote the radii of $\mathcal{K}_t$ and $\mathcal{K}_s$, respectively. We let

$$\alpha := r_t + r_s, \quad \beta := \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s)$$

and use Lemma 4.7 in order to prove

$$\begin{aligned} \|\zeta_0\|_2 &= \|x_t - y_s\|_2 \geq \operatorname{dist}(\mathcal{K}_t, \mathcal{K}_s) + r_t + r_s = \alpha + \beta, \\ \|\zeta - \zeta_0\|_2 &= \|x - x_t - (y - y_s)\|_2 \leq \|x - x_t\|_2 + \|y - y_s\|_2 \leq r_t + r_s = \alpha, \\ \|p\|_2 &= (\|x - x_t\|_2^2 + \|y - y_s\|_2^2)^{1/2} \leq \|x - x_t\|_2 + \|y - y_s\|_2 \leq r_t + r_s = \alpha, \end{aligned}$$

which implies

$$\|\zeta_0 + (\zeta - \zeta_0)t\|_2 \geq \|\zeta_0\|_2 - \|\zeta - \zeta_0\|_2 t \geq \alpha + \beta - \alpha t = \alpha(1-t) + \beta$$

for all $t \in [0,1]$. Combining this inequality with (4.13) yields

$$\begin{aligned} |g(x,y) - \tilde{g}_{t,s}(x,y)| &\leq \widehat{C} c_0^m \|p\|_2^m \int_0^1 \frac{(1-t)^{m-1}}{\|\zeta_0 + (\zeta - \zeta_0)t\|_2^{m+\sigma}}\, dt \\ &\leq \widehat{C} c_0^m \int_0^1 \frac{(1-t)^{m-1}\alpha^m}{((1-t)\alpha + \beta)^{m+\sigma}}\, dt \\ &\leq \widehat{C} c_0^m \int_0^1 \left(\frac{(1-t)\alpha}{(1-t)\alpha + \beta}\right)^{m-1} \frac{\alpha}{((1-t)\alpha + \beta)^{\sigma+1}}\, dt. \end{aligned}$$

The assumption (4.11) implies $\alpha \leq \eta\beta$, i.e.,

$$\frac{(1-t)\alpha}{(1-t)\alpha + \beta} \leq \frac{(1-t)\alpha}{(1-t)\alpha + \alpha/\eta} = \frac{1-t}{1-t+1/\eta} \leq \frac{1}{1+1/\eta} = \frac{\eta}{\eta+1}.$$

If $\sigma = 0$ holds, we have

$$\int_0^1 \frac{\alpha}{((1-t)\alpha + \beta)^{\sigma+1}}\, dt = \int_0^1 \frac{\alpha}{s\alpha + \beta}\, ds = \log|\alpha + \beta| - \log|\beta| = \log\left|\frac{\alpha + \beta}{\beta}\right|$$

$$\leq \log\left|\frac{\beta(\eta+1)}{\beta}\right| = \log(\eta+1).$$

For $\sigma > 0$, we find

$$\begin{aligned}\int_0^1 \frac{\alpha}{((1-t)\alpha+\beta)^{\sigma+1}}\,dt &= \int_0^1 \frac{\alpha}{(s\alpha+\beta)^{\sigma+1}}\,ds = -\frac{1}{\sigma}\left(\frac{1}{(\alpha+\beta)^\sigma} - \frac{1}{\beta^\sigma}\right)\\ &\leq \frac{1}{\sigma}\left(\frac{1}{\beta^\sigma} - \frac{1}{(\eta\beta+\beta)^\sigma}\right) = \frac{1}{\sigma}\left(\frac{(\eta+1)^\sigma - 1}{(\eta+1)^\sigma\beta^\sigma}\right)\\ &\leq \frac{1}{\sigma\beta^\sigma}\end{aligned}$$

and can conclude

$$|g(x,y) - \tilde{g}_{t,s}(x,y)| \leq \begin{cases} \widehat{C}\,c_0\log(\eta+1)\left(\frac{c_0\eta}{\eta+1}\right)^{m-1} & \text{if } \sigma = 0,\\ \widehat{C}\,\frac{c_0}{\sigma\,\operatorname{dist}^\sigma(\mathcal{K}_t,\mathcal{K}_s)}\left(\frac{c_0\eta}{\eta+1}\right)^{m-1} & \text{otherwise,}\end{cases}$$

and substituting the value of $\widehat{C}$ completes the proof. □

Theorem 4.8 is a generalization of the result of Lemma 2.2: the former implies the latter if we let $C_{\mathrm{as}} = 1$, $\sigma = 0$ and $c_0 = 1$.

If the distance of the d-dimensional balls $\mathcal{K}_t$ and $\mathcal{K}_s$ is sufficiently large compared to their diameters, i.e., if $q_{\mathrm{opt}} := c_0\eta/(\eta+1) < 1$ holds, Theorem 4.8 yields that the approximant $\tilde{g}_{t,s}|_{\mathcal{K}_t\times\mathcal{K}_s}$ converges to the kernel function $g|_{\mathcal{K}_t\times\mathcal{K}_s}$ at an exponential rate if m grows: for fixed $\mathcal{K}_t$ and $\mathcal{K}_s$, the error is proportional to

$$\frac{(m+\sigma-1)!}{(m-1)!}q_{\mathrm{opt}}^{m-1},$$

and here the exponential decay of the second factor dominates the polynomial growth of the first one: for each $q \in \mathbb{R}_{>q_{\mathrm{opt}}}$, we can find a constant $C_{\mathrm{ta}} \in \mathbb{R}_{>0}$ such that

$$\|g - \tilde{g}_{t,s}\|_{\mathcal{K}_t\times\mathcal{K}_s} \leq \frac{C_{\mathrm{ta}}}{\operatorname{dist}(\mathcal{K}_t,\mathcal{K}_s)^\sigma}q^m \quad \text{holds for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, m \in \mathbb{N}.$$

4.4 Approximation by interpolation

Using Taylor expansions to construct separable approximations of the kernel function has many advantages: significant portions of the resulting transfer and coupling matrices contain only zero entries, i.e., we can save storage by using efficient data formats, the evaluation of the monomials corresponding to the cluster bases is straightforward, and the error analysis is fairly simple.

Unfortunately, the approach via Taylor expansions has also two major disadvantages: the construction of the coupling matrices requires the efficient evaluation of derivatives of the kernel function g, e.g., by recursion formulas that have to be derived by hand, and the error estimates are not robust with respect to the parameter c_0 appearing in the Definition 4.5 of asymptotic smoothness: if c_0 grows, we have to adapt η in order to guarantee exponential convergence. Both properties limit the applicability of Taylor-based approximations in general situations.

We can overcome the disadvantages by using an alternative approximation scheme: instead of constructing a polynomial approximation of the kernel function g by Taylor expansion, we use Lagrangian interpolation.

One-dimensional interpolation

Let us first consider the one-dimensional case. For each interpolation order $m \in \mathbb{N}$, we fix interpolation points $(\xi_{m,\nu})_{\nu=1}^{m}$ in the interval $[-1, 1]$. We require that the points corresponding to one $m \in \mathbb{N}$ are pairwise different, i.e., that

$$\nu \neq \mu \Rightarrow \xi_{m,\nu} \neq \xi_{m,\mu} \quad \text{holds for all } m \in \mathbb{N} \text{ and } \nu, \mu \in \{1, \dots, m\}. \tag{4.14}$$

The one-dimensional interpolation operator of order $m \in \mathbb{N}$ is given by

$$\mathfrak{I}_m \colon C[-1, 1] \to \mathcal{P}_m, \quad f \mapsto \sum_{\nu=1}^{m} f(\xi_{m,\nu}) \mathcal{L}_{m,\nu},$$

where the Lagrange polynomials $\mathcal{L}_{m,\nu} \in \mathcal{P}_m$ are given by

$$\mathcal{L}_{m,\nu}(x) := \prod_{\substack{\mu=1 \\ \mu \neq \nu}}^{m} \frac{x - \xi_{m,\mu}}{\xi_{m,\nu} - \xi_{m,\mu}} \quad \text{for all } x \in \mathbb{R},\ m \in \mathbb{N},\ \nu \in \{1, \dots, m\}.$$

Since $\mathcal{L}_{m,\nu}(\xi_{m,\mu}) = \delta_{\nu\mu}$ holds for all $m \in \mathbb{N}$ and $\nu, \mu \in \{1, \dots, m\}$, we have

$$\mathfrak{I}_m[f](\xi_{m,\nu}) = f(\xi_{m,\nu}) \quad \text{for all } f \in C[-1, 1],\ m \in \mathbb{N} \text{ and } \nu \in \{1, \dots, m\}.$$

Combining (4.14) with this equation and the identity theorem for polynomials yields

$$\mathfrak{I}_m[p] = p \quad \text{for all } m \in \mathbb{N} \text{ and } p \in \mathcal{P}_m, \tag{4.15}$$

i.e., the interpolation $\mathfrak{I}_m$ is a *projection* with range $\mathcal{P}_m$.

In order to define an interpolation operator for general non-empty intervals $[a, b]$, we consider the linear mapping

$$\Phi_{[a,b]} \colon [-1, 1] \to [a, b], \quad t \mapsto \frac{b + a}{2} + \frac{b - a}{2} t,$$

and define the transformed interpolation operator $\mathfrak{I}_m^{[a,b]}\colon C[a,b] \to \mathcal{P}_m$ by

$$\mathfrak{I}_m^{[a,b]}[f] := (\mathfrak{I}_m[f \circ \Phi_{[a,b]}]) \circ \Phi_{[a,b]}^{-1} \quad \text{for all } m \in \mathbb{N},$$

i.e., by mapping f into $C[-1,1]$, applying the original interpolation operator, and mapping the resulting polynomial back to the interval $[a,b]$. Since $\Phi_{[a,b]}^{-1}$ is an affine mapping, the result will still be a polynomial in $\mathcal{P}_m$.

Let $m \in \mathbb{N}$. The definition of $\mathfrak{I}_m$ yields

$$\mathfrak{I}_m^{[a,b]}[f] = \sum_{\nu=1}^{m} f(\Phi_{[a,b]}(\xi_{m,\nu}))\mathcal{L}_{m,\nu} \circ \Phi_{[a,b]}^{-1},$$

and defining the transformed interpolation points $\left(\xi_{m,\nu}^{[a,b]}\right)_{\nu=1}^{m}$ in the interval $[a,b]$ by

$$\xi_{m,\nu}^{[a,b]} := \Phi_{[a,b]}(\xi_{m,\nu}) = \frac{b+a}{2} + \frac{b-a}{2}\xi_{m,\nu}$$

and the corresponding transformed Lagrange polynomials $\left(\mathcal{L}_{m,\nu}^{[a,b]}\right)_{\nu=1}^{m}$ by

$$\mathcal{L}_{m,\nu}^{[a,b]} := \mathcal{L}_{m,\nu} \circ \Phi_{[a,b]}^{-1},$$

we get the more compact notation

$$\mathfrak{I}_m^{[a,b]} = \sum_{\nu=1}^{m} f\left(\xi_{m,\nu}^{[a,b]}\right)\mathcal{L}_{m,\nu}^{[a,b]}.$$

Since the equation

$$\begin{aligned}\mathcal{L}_{m,\nu}^{[a,b]}(\xi_{m,\mu}^{[a,b]}) &= \mathcal{L}_{m,\nu} \circ \Phi_{[a,b]}^{-1}(\Phi_{[a,b]}(\xi_{m,\mu})) = \mathcal{L}_{m,\nu}(\xi_{m,\mu}) = \delta_{\nu\mu} \\ &= \prod_{\substack{\kappa=1 \\ \kappa\neq\nu}}^{m} \frac{\xi_{m,\mu}^{[a,b]} - \xi_{m,\kappa}^{[a,b]}}{\xi_{m,\nu}^{[a,b]} - \xi_{m,\kappa}^{[a,b]}}\end{aligned}$$

holds for all $\nu, \mu \in \{1,\dots,m\}$, the identity theorem for polynomials yields the equation

$$\mathcal{L}_{m,\nu}^{[a,b]}(x) = \prod_{\substack{\mu=1 \\ \mu\neq\nu}}^{m} \frac{x - \xi_{m,\mu}^{[a,b]}}{\xi_{m,\nu}^{[a,b]} - \xi_{m,\mu}^{[a,b]}} \quad \text{for all } x \in \mathbb{R},\ m \in \mathbb{N} \text{ and } \nu \in \{1,\dots,m\},$$

which we can use to evaluate the transformed Lagrange polynomials efficiently.

Separable approximation by multi-dimensional interpolation

Since we intend to apply interpolation to construct a separable approximation of the kernel function g defined in a multi-dimensional domain, we require multi-dimensional interpolation operators.

Let us consider an axis-parallel d-dimensional box

$$\mathcal{Q} = [a_1, b_1] \times \cdots \times [a_d, b_d]$$

with $a_1 < b_1$, ..., $a_d < b_d$. The order m of the one-dimensional interpolation scheme is replaced by an *order vector* $m \in \mathbb{N}^d$, and the corresponding tensor-product interpolation operator $\mathcal{I}_m^{\mathcal{Q}}$ is defined by

$$\mathcal{I}_m^{\mathcal{Q}} := \mathcal{I}_{m_1}^{[a_1,b_1]} \otimes \cdots \otimes \mathcal{I}_{m_d}^{[a_d,b_d]}. \tag{4.16}$$

We can observe that it takes the familiar form

$$\mathcal{I}_m^{\mathcal{Q}}[f] = \sum_{0<\nu\leq m} f(\xi_{m,\nu}^{\mathcal{Q}}) \mathcal{L}_{m,\nu}^{\mathcal{Q}} \quad \text{for all } f \in C(\mathcal{Q}),$$

if the multi-dimensional interpolation points and Lagrange polynomials are given by

$$\begin{aligned} \xi_{m,\nu}^{\mathcal{Q}} &:= (\xi_{m_1,\nu_1}^{[a_1,b_1]}, \ldots, \xi_{m_d,\nu_d}^{[a_d,b_d]}) && \text{for all } \nu \in \mathbb{N}^d \text{ with } \nu \leq m, \\ \mathcal{L}_{m,\nu}^{\mathcal{Q}} &:= \mathcal{L}_{m_1,\nu_1}^{[a_1,b_1]} \otimes \cdots \otimes \mathcal{L}_{m_d,\nu_d}^{[a_d,b_d]} && \text{for all } \nu \in \mathbb{N}^d \text{ with } \nu \leq m. \end{aligned}$$

In practice, the multi-dimensional Lagrange polynomials can be easily evaluated by using the equation

$$\begin{aligned} \mathcal{L}_{m,\nu}^{\mathcal{Q}}(x) &= \prod_{\iota=1}^{d} \mathcal{L}_{m_\iota,\nu_\iota}^{[a_\iota,b_\iota]}(x_\iota) \\ &= \prod_{\iota=1}^{d} \prod_{\substack{\mu=1 \\ \mu\neq\nu_\iota}}^{m_\iota} \frac{x_\iota - \xi_{m_\iota,\mu}^{[a_\iota,b_\iota]}}{\xi_{m_\iota,\nu_\iota}^{[a_\iota,b_\iota]} - \xi_{m_\iota,\mu}^{[a_\iota,b_\iota]}} \quad \text{for all } x \in \mathbb{R}^d \text{ and } \nu \in \mathbb{N}_{\leq m}^d. \end{aligned}$$

In order to construct separable approximations of g, we enclose the supports Ω_t and Ω_s of all clusters $t \in \mathcal{T}_I$ and $s \in \mathcal{T}_J$ in axis-parallel bounding boxes (cf. 3.10), i.e., we fix axis-parallel boxes $(\mathcal{Q}_t)_{t\in\mathcal{T}_I}$ and $(\mathcal{Q}_s)_{s\in\mathcal{T}_J}$ satisfying

$$\Omega_t \subseteq \mathcal{Q}_t, \quad \Omega_s \subseteq \mathcal{Q}_s \quad \text{for all } t \in \mathcal{T}_I,\ s \in \mathcal{T}_J.$$

We also fix order vectors $(m_t)_{t\in\mathcal{T}_I}$ and $(m_s)_{s\in\mathcal{T}_J}$ for all clusters in $\mathcal{T}_I$ and $\mathcal{T}_J$, respectively.

Using these bounding boxes and order vectors, we can define interpolation operators $(\mathcal{I}_t)_{t\in\mathcal{T}_I}$ and $(\mathcal{I}_s)_{s\in\mathcal{T}_J}$ for all clusters in $\mathcal{T}_I$ and $\mathcal{T}_J$, respectively, by

$$\mathcal{I}_t := \mathcal{I}_{m_t}^{\mathcal{Q}_t}, \quad \mathcal{I}_s := \mathcal{I}_{m_s}^{\mathcal{Q}_s} \quad \text{for all } t \in \mathcal{T}_I,\ s \in \mathcal{T}_J. \tag{4.17}$$

If we let

$$K_t := \{\nu \in \mathbb{N}^d \ : \ 1 \leq \nu_\iota \leq m_{t,\iota} \text{ for all } \iota \in \{1,\ldots,d\}\} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},$$
$$L_s := \{\mu \in \mathbb{N}^d \ : \ 1 \leq \mu_\iota \leq m_{s,\iota} \text{ for all } \iota \in \{1,\ldots,d\}\} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J}$$

and define the interpolation points and Lagrange polynomials for all clusters by

$$\xi_{t,\nu} := \xi^{\mathcal{Q}_t}_{m_t,\nu}, \quad \mathcal{L}_{t,\nu} := \mathcal{L}^{\mathcal{Q}_t}_{m_t,\nu} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ \nu \in K_t,$$
$$\xi_{s,\mu} := \xi^{\mathcal{Q}_s}_{m_s,\mu}, \quad \mathcal{L}_{s,\mu} := \mathcal{L}^{\mathcal{Q}_s}_{m_s,\mu} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J},\ \mu \in L_s,$$

we can see that the interpolation operators take the form

$$\begin{aligned} \mathcal{I}_t[f] &= \sum_{\nu\in K_t} f(\xi_{t,\nu})\mathcal{L}_{t,\nu} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ f \in C(\mathcal{Q}_t), \\ \mathcal{I}_s[f] &= \sum_{\mu\in L_s} f(\xi_{s,\mu})\mathcal{L}_{s,\mu} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J},\ f \in C(\mathcal{Q}_s). \end{aligned} \tag{4.18}$$

Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ be an admissible pair of clusters. By construction, $\mathcal{Q} := \mathcal{Q}_t \times \mathcal{Q}_s$ is an axis-parallel box in $\mathbb{R}^{2d}$ and $m := (m_t, m_s)$ is a $2d$-dimensional order vector. By definition, we have

$$\mathcal{I}^{\mathcal{Q}}_m = \mathcal{I}^{\mathcal{Q}_t}_{m_t} \otimes \mathcal{I}^{\mathcal{Q}_s}_{m_s} = \mathcal{I}_t \otimes \mathcal{I}_s$$

and the $2d$-dimensional interpolant of g is given by

$$\tilde{g}_{t,s}(x,y) := \mathcal{I}^{\mathcal{Q}}_m[g](x,y) = \sum_{\nu\in K_t}\sum_{\mu\in L_s} g(\xi^t_\nu, \xi^s_\mu)\mathcal{L}^t_\nu(x)\mathcal{L}^s_\mu(y) \qquad \text{for all } x,y \in \mathbb{R}^d, \tag{4.19}$$

i.e., the $2d$-dimensional interpolation leads to a separable approximation of the desired form (4.3).

We can use this approximation of g to define an approximation

$$\chi_t G \chi_s \approx V_t S_b W_s^*$$

of the matrix block corresponding to $b = (t,s)$, where the matrices $V_t \in \mathbb{R}^{\mathcal{I}\times K_t}_{\hat{t}}$, $W_s \in \mathbb{R}^{\mathcal{J}\times L_s}_{\hat{s}}$ and $S_b \in \mathbb{R}^{K_t\times L_s}$ are given by

$$(V_t)_{i\nu} := \begin{cases} \int_\Omega \mathcal{L}_{t,\nu}(x)\varphi_i(x)\,dx & \text{if } i \in \hat{t}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{I},\ \nu \in K_t, \tag{4.20a}$$

$$(W_s)_{j\mu} := \begin{cases} \int_\Omega \mathcal{L}_{s,\mu}(y)\psi_j(y)\,dy & \text{if } j \in \hat{s}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } j \in \mathcal{J},\ \mu \in L_s, \tag{4.20b}$$

$$(S_b)_{\nu\mu} := g(\xi_{t,\nu}, \xi_{s,\mu}) \quad \text{for all } \nu \in K_t,\ \mu \in L_s. \tag{4.20c}$$

We can only obtain an algorithm with optimal order of complexity if the resulting cluster bases $V := (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W := (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ are nested.

We assume $m_t \leq m_{t'}$ for all $t \in \mathcal{T}_\mathcal{I}$ and all $t' \in \operatorname{sons}(t)$ and observe that (4.15) implies

$$\mathfrak{I}_{t'}[\mathfrak{L}_{t,\nu}] = \mathfrak{L}_{t,\nu} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},\ t' \in \operatorname{sons}(t) \text{ and all } \nu \in K_t.$$

Due to (4.18), this means

$$\sum_{\nu'\in K_{t'}} \mathfrak{L}_{t,\nu}(\xi_{t',\nu'})\mathfrak{L}_{t',\nu'} = \mathfrak{L}_{t,\nu},$$

and we find that for all $t \in \mathcal{T}_\mathcal{I}$ and all $t' \in \operatorname{sons}(t)$ the matrices $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$ defined by

$$(E_{t'})_{\nu'\nu} := \mathfrak{L}_{t,\nu}(\xi_{t',\nu'}) \quad \text{for all } \nu' \in K_{t'},\ \nu \in K_t,$$

satisfy the equation

$$V_t = \sum_{t'\in\operatorname{sons}(t)} V_{t'}E_{t'},$$

i.e., the matrices $(E_t)_{t\in\mathcal{T}_\mathcal{I}}$ are transfer matrices for the nested cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$. A family $(F_s)_{s\in\mathcal{T}_\mathcal{J}}$ of transfer matrices for the nested cluster basis $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ can be constructed by a similar argument.

Remark 4.9 (Implementation). We can see that the construction of the coupling matrices S_b for this approximation scheme is simpler than in the case of Taylor approximations: instead of having to evaluate arbitrarily high derivatives of the kernel function g, interpolation only requires the evaluation of the function g itself. This means that we can apply interpolation in quite general situations as a "black box" strategy.

Finding the correct interpolation points is straightforward: we have seen in Section 3.3 that optimal bounding boxes $\mathcal{Q}_t$ and $\mathcal{Q}_s$ can be constructed by a simple and efficient algorithm. Once these boxes and one-dimensional interpolation points $(\xi_\nu)_{\nu=1}^m$ are given, the construction of tensor interpolation points is trivial.

Based on these interpolation points, the Lagrange polynomials $\mathfrak{L}_t$ can be evaluated efficiently, i.e., the computation of the entries of the transfer matrices and the values of the integrands appearing in the definition of V_t and W_s is easily accomplished. A simple quadrature rule can be used to take care of the integration.

The transfer matrices $E_{t'}$ are of a special structure: we have

$$(E_{t'})_{\nu'\nu} = \mathfrak{L}_{t,\nu}(\xi_{t',\nu'}) = \prod_{\iota=1}^{d} \underbrace{\mathfrak{L}_{\nu_\iota}^{[a_\iota,b_\iota]}(\xi_{\nu'_\iota}^{[a'_\iota,b'_\iota]})}_{=:(E_{t',\iota})_{\nu'_\iota\nu_\iota}}$$

for $\mathcal{Q}_t = [a_1, \ldots, a_d]$ and $\mathcal{Q}_{t'} = [a'_1, \ldots, a'_d]$. We can prepare the $d(m+1)^2$ coefficients of the d auxiliary matrices $E_{t',\iota}$ using only $4d(m+1)^3$ operations, and the computation of one entry of $E_{t'}$ then requires only $d-1$ multiplications. □

Lemma 4.10 (Complexity). *If $\mu \in \mathbb{N}$ satisfies*

$$m_{t,\iota} \leq \mu \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ \iota \in \{1, \ldots, d\},$$

we have

$$\#K_t = \prod_{\iota=1}^{d} m_{t,\iota} \leq \mu^d \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

i.e., the rank distribution is μ^d-bounded.

Proof. Similar to the proof of Lemma 4.4. □

Error analysis in the one-dimensional case

Let $m \in \mathbb{N}$. The interpolation operator $\mathfrak{I}_m$ is continuous since we have

$$\begin{aligned} \|\mathfrak{I}_m[f]\|_{\infty,[-1,1]} &= \max\left\{\left|\textstyle\sum_{\nu=1}^{m} f(\xi_{m,\nu})\mathcal{L}_{m,\nu}(x)\right| \,:\, x \in [-1,1]\right\} \\ &\leq \max\left\{\textstyle\sum_{\nu=1}^{m} \|f\|_{\infty,[-1,1]}|\mathcal{L}_{m,\nu}(x)| \,:\, x \in [-1,1]\right\} \\ &= \Lambda_m \|f\|_{\infty,[-1,1]} \end{aligned} \tag{4.21}$$

for all $f \in C[-1,1]$ with the *Lebesgue constant*

$$\Lambda_m := \max\left\{\textstyle\sum_{\nu=1}^{m} |\mathcal{L}_{m,\nu}(x)| \,:\, x \in [-1,1]\right\}.$$

Combining the stability estimate (4.21) with the projection property (4.15) yields the lower bound $\Lambda_m \geq 1$ for all $m \in \mathbb{N}$.

Definition 4.11 (Stable interpolation scheme). A family $(\mathfrak{I}_m)_{m=1}^{\infty}$ of interpolation operators is called an *interpolation scheme*.

If there are constants $\Lambda, \lambda \in \mathbb{R}_{\geq 1}$ satisfying

$$\Lambda_m \leq \Lambda(m+1)^{\lambda} \quad \text{for all } m \in \mathbb{N}, \tag{4.22}$$

the interpolation scheme $(\mathfrak{I}_m)_{m=0}^{\infty}$ is called (Λ, λ)-*stable*.

Example 4.12 (Chebyshev interpolation). The *Chebyshev interpolation points* are given by

$$\xi_{m,\nu} := \cos\left(\frac{2\nu - 1}{2m}\right) \quad \text{for all } m \in \mathbb{N},\ \nu \in \{1, \ldots, m\}.$$

The corresponding interpolation scheme has the advantage that its Lebesgue constants satisfy

$$\Lambda_m \leq \frac{2}{\pi} \ln(m+1) + 1 \leq m+1 \quad \text{for all } m \in \mathbb{N}, \tag{4.23}$$

i.e., the interpolation scheme is $(1,1)$-stable [87].

Let $m \in \mathbb{N}$, and let $[a,b]$ be a non-trivial interval. Since we have

$$\begin{aligned} \|\mathfrak{I}_m^{[a,b]}[f]\|_{\infty,[a,b]} &= \|(\mathfrak{I}_m[f \circ \Phi_{[a,b]}]) \circ \Phi_{[a,b]}^{-1}\|_{\infty,[a,b]} \\ &= \|\mathfrak{I}_m[f \circ \Phi_{[a,b]}]\|_{\infty,[-1,1]} \\ &\leq \Lambda_m \|f \circ \Phi_{[a,b]}\|_{\infty,[-1,1]} = \Lambda_m \|f\|_{\infty,[a,b]} \end{aligned} \tag{4.24}$$

for all $f \in C[a,b]$, we can conclude that the transformed interpolation operator $\mathfrak{I}_m^{[a,b]}$ has the same Lebesgue constant as $\mathfrak{I}_m$.

Using a bound for the Lebesgue constant of an interpolation scheme, we can demonstrate that the interpolant is close to the best possible polynomial approximation:

Lemma 4.13 (Best approximation). *Let $m \in \mathbb{N}$ and $f \in C[-1,1]$. We have*

$$\|f - \mathfrak{I}_m[f]\|_{\infty,[-1,1]} \leq (\Lambda_m + 1)\|f - p\|_{\infty,[-1,1]} \quad \text{for all } p \in \mathcal{P}_m. \tag{4.25}$$

Proof. Let $p \in \mathcal{P}_m$. Due to (4.15), we have $\mathfrak{I}_m[p] = p$ and find

$$\begin{aligned} \|f - \mathfrak{I}_m[f]\|_{\infty,[-1,1]} &= \|f - p + \mathfrak{I}_m[p] - \mathfrak{I}_m[f]\|_{\infty,[-1,1]} \\ &\leq \|f - p\|_{\infty,[-1,1]} + \|\mathfrak{I}_m[p-f]\|_{\infty,[-1,1]} \\ &\leq \|f - p\|_{\infty,[-1,1]} + \Lambda_m \|f - p\|_{\infty,[-1,1]} \\ &= (\Lambda_m + 1)\|f - p\|_{\infty,[-1,1]}. \end{aligned}$$

□

Due to this best-approximation property, we can bound the interpolation error by finding an approximating polynomial. We follow the approach described in [23], Lemma 3.13: we construct a holomorphic extension of f into an elliptic subdomain of the complex plane $\mathbb{C}$ and use the following result to find the desired polynomial:

Lemma 4.14 (Approximation of holomorphic functions). *Let $\varrho \in \mathbb{R}_{>1}$ and*

$$\mathcal{E}_\varrho := \left\{ x + iy \ : \ x, y \in \mathbb{R}, \left(\frac{2x}{\varrho + 1/\varrho}\right)^2 + \left(\frac{2y}{\varrho - 1/\varrho}\right)^2 \leq 1 \right\}. \tag{4.26}$$

Let $f \in C^\infty(\mathcal{E}_\varrho)$ be a holomorphic function. We have

$$\min_{p \in \mathcal{P}_m} \|f - p\|_{\infty,[-1,1]} \leq \frac{2\varrho}{\varrho - 1} \varrho^{-m} \|f\|_{\infty,\mathcal{E}_\varrho} \quad \text{for all } m \in \mathbb{N}.$$

Proof. See, e.g., [42], Chapter 7, Section 8, equation (8.7). □

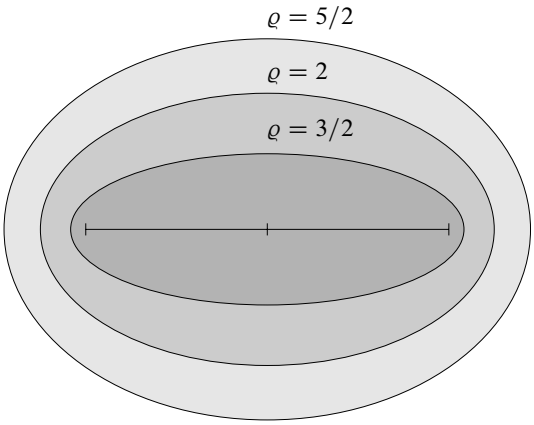


Figure 4.2. Analyticity domains $\mathcal{E}_\varrho$ for different values of ϱ.

In order to apply this lemma, we have to be able to find holomorphic extensions of functions $f \in C^\infty[-1,1]$ to a regularity ellipse $\mathcal{E}_\varrho$. If the derivatives of f are bounded, the extension exists and we obtain the following existence result for polynomial approximations:

Lemma 4.15 (Polynomial approximation). *Let $f \in C^\infty[-1,1]$, $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that*

$$|f^{(\nu)}(x)| \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{holds for all } x \in [-1,1],\ \nu \in \mathbb{N}_0. \tag{4.27}$$

For all $m \in \mathbb{N}$, we can then find a polynomial $p \in \mathcal{P}_m$ with

$$\|f - p\|_{\infty,[-1,1]} \leq 2C_f e(m+1)^\sigma \frac{\gamma_f + 2}{\gamma_f} \left(\gamma_f + \sqrt{1+\gamma_f^2}\right)^{-m},$$

Proof. Let $m \in \mathbb{N}$. Let $r \in]0, \gamma_f[$. Due to Lemma 4.76, we can find a holomorphic extension $\tilde{f}$ of f to the domain $\mathcal{R}_r$ which satisfies

$$\|\tilde{f}\|_{\infty,\mathcal{R}_r} \leq C_f \left(\frac{\gamma_f}{\gamma_f - r}\right)^\sigma.$$

Let $\varrho := r + \sqrt{1+r^2}$. We can apply Lemma 4.77 in order to prove $\mathcal{E}_\varrho \subseteq \mathcal{R}_r$, i.e.,

$$\|\tilde{f}\|_{\infty,\mathcal{E}_\varrho} \leq C_f \left(\frac{\gamma_f}{\gamma_f - r}\right)^\sigma.$$

Due to Lemma 4.14, we can find a polynomial $p \in \mathcal{P}_m$ with

$$\begin{aligned}\|f - p\|_{\infty,[-1,1]} = \|\tilde{f} - p\|_{\infty,[-1,1]} &\leq \frac{2\varrho}{\varrho - 1}\varrho^{-m}\|\tilde{f}\|_{\infty,\mathcal{E}_\varrho} \\ &\leq C_f \frac{2\varrho}{\varrho - 1}\varrho^{-m}\left(\frac{\gamma_f}{\gamma_f - r}\right)^\sigma.\end{aligned}$$

Using the definition of ϱ yields

$$\begin{aligned}\frac{\varrho}{\varrho - 1} &= \frac{r + \sqrt{1 + r^2}}{r - 1 + \sqrt{1 + r^2}} = \frac{(r + \sqrt{1 + r^2})(r - 1 - \sqrt{1 + r^2})}{(r - 1 + \sqrt{1 + r^2})(r - 1 - \sqrt{1 + r^2})} \\ &= \frac{r^2 - r - r\sqrt{1 + r^2} + r\sqrt{1 + r^2} - \sqrt{1 + r^2} - 1 - r^2}{r^2 - 2r + 1 - 1 - r^2} \\ &= \frac{-r - 1 - \sqrt{1 + r^2}}{-2r} = \frac{r + 1 + \sqrt{1 + r^2}}{2r} \leq \frac{r + 1 + \sqrt{1 + 2r + r^2}}{2r} \\ &= \frac{2r + 2}{2r} = \frac{r + 1}{r}.\end{aligned}$$

Applying these estimates to $r := \alpha\gamma_f$ with $\alpha \in]0, 1[$ yields

$$\begin{aligned}\|f - p\|_{\infty,[-1,1]} &\leq C_f \frac{2\varrho}{\varrho - 1}\varrho^{-m}\left(\frac{\gamma_f}{\gamma_f - r}\right)^\sigma \\ &\leq 2C_f \frac{r + 1}{r}\left(r + \sqrt{1 + r^2}\right)^{-m}\left(\frac{\gamma_f}{\gamma_f - r}\right)^\sigma \\ &= 2C_f \frac{\alpha\gamma_f + 1}{\alpha\gamma_f}\left(\alpha\gamma_f + \sqrt{1 + \alpha^2\gamma_f^2}\right)^{-m}\left(\frac{\gamma_f}{\gamma_f - \alpha\gamma_f}\right)^\sigma \\ &\leq 2C_f \frac{\alpha\gamma_f + 1}{\alpha\gamma_f}\left(\alpha\gamma_f + \alpha\sqrt{1 + \gamma_f^2}\right)^{-m}\frac{1}{(1 - \alpha)^\sigma} \\ &= 2C_f \frac{\alpha\gamma_f + 1}{\alpha\gamma_f}\frac{1}{\alpha^m(1 - \alpha)^\sigma}\left(\gamma_f + \sqrt{1 + \gamma_f^2}\right)^{-m}.\end{aligned}$$

We choose $\alpha := m/(m + 1)$ and get

$$\begin{aligned}\frac{\alpha\gamma_f + 1}{\alpha\gamma_f} &= \frac{\gamma_f + 1/\alpha}{\gamma_f} = \frac{\gamma_f + (m + 1)/m}{\gamma_f} \leq \frac{\gamma_f + 2}{\gamma_f}, \\ \frac{1}{\alpha^m(1 - \alpha)^\sigma} &= \left(\frac{m + 1}{m}\right)^m\left(1 - \frac{m}{m + 1}\right)^{-\sigma} = \left(1 + \frac{1}{m}\right)^m(m + 1)^\sigma < e(m + 1)^\sigma\end{aligned}$$

and conclude

$$\|f - p\|_{\infty,[-1,1]} < 2C_f e(m + 1)^\sigma \frac{\gamma_f + 2}{\gamma_f}\left(\gamma_f + \sqrt{1 + \gamma_f^2}\right)^{-m}. \qquad \square$$

Using this general approximation result for analytic functions in the interval $[-1, 1]$, we can now bound the interpolation error using the best-approximation result Lemma 4.13.

Theorem 4.16 (Interpolation error). *Let $[a, b] \subseteq \mathbb{R}$ be a non-trivial interval. Let $m \in \mathbb{N}$ and $n \in \{1, \dots, m\}$. Let $f \in C^\infty[a, b]$, $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that*

$$|f^{(\nu)}(x)| \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{holds for all } x \in [a, b],\ \nu \in \mathbb{N}_0.$$

For the function

$$\varrho \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 1}, \quad r \mapsto r + \sqrt{1 + r^2}, \tag{4.28}$$

we have

$$\|f - \mathfrak{I}_m^{[a,b]}[f]\|_{\infty,[a,b]} \leq 2eC_f(\Lambda_m + 1)(n + 1)^\sigma \left(1 + \frac{b - a}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{b - a}\right)^{-n}. \tag{4.29}$$

Proof. We let $\hat{f} := f \circ \Phi_{[a,b]}$ and $\hat{\gamma}_f := 2\gamma_f/(b - a)$ and observe

$$\begin{aligned} |\hat{f}^{(\nu)}(x)| &= \left|\left(\frac{b - a}{2}\right)^\nu f^{(\nu)}(\Phi_{[a,b]}(x))\right| \\ &\leq \frac{C_f}{\hat{\gamma}_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{for all } x \in [-1, 1],\ \nu \in \mathbb{N}_0, \end{aligned}$$

so we can apply Lemma 4.15 to construct a polynomial $\hat{p} \in \mathcal{P}_n$ satisfying

$$\|\hat{f} - \hat{p}\|_{\infty,[-1,1]} \leq 2C_f e(n + 1)^\sigma \left(1 + \frac{2}{\hat{\gamma}_f}\right) \left(\hat{\gamma}_f + \sqrt{1 + \hat{\gamma}_f^2}\right)^{-n}.$$

Due to Lemma 4.13, this implies

$$\begin{aligned} \|f - \mathfrak{I}_m^{[a,b]}[f]\|_{\infty,[a,b]} &= \|f \circ \Phi_{[a,b]} - \mathfrak{I}_m^{[a,b]}[f] \circ \Phi_{[a,b]}\|_{\infty,[-1,1]} \\ &= \|\hat{f} - \mathfrak{I}_m[\hat{f}]\|_{\infty,[-1,1]} \\ &\leq 2eC_f(\Lambda_m + 1)(n + 1)^\sigma \left(1 + \frac{2}{\hat{\gamma}_f}\right) \left(\hat{\gamma}_f + \sqrt{1 + \hat{\gamma}_f^2}\right)^{-n}, \end{aligned}$$

which is the required estimate. □

Remark 4.17 (Rate of convergence). The function ϱ determining the rate of convergence introduced in (4.28) is monotonic increasing, since it is the sum of two monotonic increasing functions. It can be bounded from below by

$$\varrho(r) = r + \sqrt{1 + r^2} > r + 1 \quad \text{and} \quad \varrho(r) = r + \sqrt{1 + r^2} > r + r = 2r$$

for all $r \in \mathbb{R}_{>0}$. The first estimate guarantees convergence even if a holomorphic extension exists only in a small neighbourhood of $[-1, 1]$. The second estimates shows that for $r \geq 1$ we get the improved convergence rate of standard Chebyshev interpolation. □

Error analysis in the multi-dimensional case

We now consider the d-dimensional interpolation operator $\mathfrak{I}_m^{\mathcal{Q}}$ defined by (4.16) for the axis-parallel box

$$\mathcal{Q} = [a_1, b_1] \times \cdots \times [a_d, b_d].$$

For all $\iota \in \{1, \ldots, d\}$, we let

$$\mathfrak{I}_{m,\iota}^{\mathcal{Q}} := \underbrace{I \otimes \cdots \otimes I}_{\iota-1 \text{ times}} \otimes \mathfrak{I}_{m_\iota}^{[a_\iota, b_\iota]} \otimes \underbrace{I \otimes \cdots \otimes I}_{d-\iota \text{ times}}. \tag{4.30}$$

For an index $\iota \in \{1, \ldots, d\}$ and a function $f \in C(\mathcal{Q})$, the interpolant $\mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]$ is polynomial in the ι-th coordinate: we have

$$\mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](x) = \sum_{\nu_\iota=1}^{m_\iota} f(x_1, \ldots, x_{\iota-1}, \xi_{m_\iota,\nu_\iota}^{[a_\iota,b_\iota]}, x_{\iota+1}, \ldots, x_d) \mathcal{L}_{m_\iota,\nu_\iota}^{[a_\iota,b_\iota]}(x_\iota) \tag{4.31}$$

for all $x \in \mathcal{Q}$. The definition of $\mathfrak{I}_m^{\mathcal{Q}}$ implies

$$\mathfrak{I}_m^{\mathcal{Q}} = \prod_{\iota=1}^{d} \mathfrak{I}_{m,\iota}^{\mathcal{Q}},$$

therefore we can analyze the tensor interpolation operator by analyzing interpolation in each coordinate direction separately.

Lemma 4.18 (Stability of directional interpolation). *Let $\iota \in \{1, \ldots, d\}$ be a coordinate direction. We have*

$$\|\mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq \Lambda_{m_\iota} \|f\|_{\infty,\mathcal{Q}} \quad \text{for all } f \in C(\mathcal{Q}).$$

Proof. Let $f \in C(\mathcal{Q})$. We fix $y_\kappa \in [a_\kappa, b_\kappa]$ for all $\kappa \in \{1, \ldots, \iota-1, \iota+1, \ldots, d\}$. We define the function

$$f_\iota \colon [a_\iota, b_\iota] \to \mathbb{R}, \quad x \mapsto f(y_1, \ldots, y_{\iota-1}, x, y_{\iota+1}, \ldots, y_d).$$

We have

$$\begin{aligned} f(y_1, \ldots, y_{\iota-1}, x, y_{\iota+1}, \ldots, y_d) &= f_\iota(x) && \text{for all } x \in [a_\iota, b_\iota], \\ \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](y_1, \ldots, y_{\iota-1}, x, y_{\iota+1}, \ldots, y_d) &= \mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota](x) && \text{for all } x \in [a_\iota, b_\iota], \end{aligned}$$

therefore the estimate (4.24) yields

$$\begin{aligned}\left|\mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d)\right| &= |\mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota](x)| \\ &\le \Lambda_m \|f_\iota\|_{\infty,[a_\iota,b_\iota]} \le \Lambda_m \|f\|_{\infty,\mathcal{Q}}\end{aligned}$$

for all $x \in [a_\iota, b_\iota]$. Since this estimate holds for all $y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d$, it implies the desired bound. □

Lemma 4.19 (Directional interpolation error). *Let $\iota \in \{1,\dots,d\}$. Let ϱ be defined by (4.28). Let $f \in C^\infty(\mathcal{Q})$, $C_f \in \mathbb{R}_{\ge 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that*

$$\|\partial_\iota^\nu f\|_{\infty,\mathcal{Q}} \le \frac{C_f}{\gamma_f^\nu}\begin{bmatrix}\nu\\ \sigma-1\end{bmatrix} \quad \text{holds for all } \nu \in \mathbb{N}_0. \tag{4.32}$$

Let $m \in \mathbb{N}^d$ be an order vector, and let $n \in \{1,\dots,m_\iota\}$. We have

$$\|f - \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \le 2eC_f(\Lambda_{m_\iota}+1)(n+1)^\sigma\left(1+\frac{b_\iota-a_\iota}{\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n}.$$

Proof. We fix $y_\kappa \in [a_\kappa, b_\kappa]$ for all $\kappa \in \{1,\dots,\iota-1,\iota+1,\dots,d\}$. We define the function

$$f_\iota\colon [a_\iota,b_\iota] \to \mathbb{R}, \quad x \mapsto f(y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d),$$

and observe that

$$\begin{aligned}&|f(y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d) \\ &\quad - \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d)| = |f_\iota(x) - \mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota](x)|\end{aligned} \tag{4.33}$$

holds for all $x \in [a_\iota, b_\iota]$. Our assumption (4.32) implies

$$|f_\iota^{(i)}(x)| = |\partial_\iota^i f(y_1,\dots,y_{\iota-1},x,y_{\iota+1},\dots,y_d)| \le \frac{C_f}{\gamma_f^\nu}\begin{bmatrix}\nu\\ \sigma-1\end{bmatrix}$$

for all $x \in [a_\iota, b_\iota]$ and $\nu \in \mathbb{N}_0$. Due to Theorem 4.16, this means

$$\|f_\iota - \mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}\|_{\infty,[a_\iota,b_\iota]} \le 2eC_f(\Lambda_{m_\iota}+1)(n+1)^\sigma\left(1+\frac{b_\iota-a_\iota}{\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n}.$$

Applying (4.33) concludes the proof. □

Combining the stability estimate of Lemma 4.18 with the interpolation error estimate of Lemma 4.19 yields the following result:

Theorem 4.20 (Multi-dimensional interpolation error). *Let ϱ be defined as in (4.28). Let $f \in C^\infty(\mathcal{Q})$, $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that*

$$\|\partial_\iota^\nu f\|_{\infty,\mathcal{Q}} \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{holds for all } \iota \in \{1,\dots,d\},\ \nu \in \mathbb{N}_0. \tag{4.34}$$

Let $m, n \in \mathbb{N}^d$ be order vectors with $n \leq m$. We let

$$\Lambda_m := \prod_{\iota=1}^{d} (\Lambda_{m_\iota} + 1), \quad \operatorname{diam}_\infty(\mathcal{Q}) := \max\{b_\iota - a_\iota \,:\, \iota \in \{1,\dots,d\}\}.$$

Then the following error estimate holds:

$$\|f - \mathfrak{I}_m^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq 2eC_f \Lambda_m \left(1 + \frac{\operatorname{diam}_\infty(\mathcal{Q})}{\gamma_f}\right) \sum_{\iota=1}^{d} (n_\iota + 1)^\sigma \varrho\left(\frac{2\gamma_f}{b_\iota - a_\iota}\right)^{-n_\iota}. \tag{4.35}$$

Proof. Let $\delta := \operatorname{diam}_\infty(\mathcal{Q})$. For all $\iota \in \{0,\dots,d\}$, we define the operator

$$P_\iota := \prod_{\kappa=1}^{\iota} \mathfrak{I}_{m,\kappa}^{\mathcal{Q}}.$$

We can see that

$$P_\iota = P_{\iota-1} \mathfrak{I}_{m,\iota}^{\mathcal{Q}} \quad \text{holds for all } \iota \in \{1,\dots,d\},$$

and due to $P_0 = I$ and $P_d = \mathfrak{I}_m^{\mathcal{Q}}$, we have

$$\begin{aligned} \|f - \mathfrak{I}_m^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} &= \|P_0[f] - P_d[f]\|_{\infty,\mathcal{Q}} \leq \sum_{\iota=1}^{d} \|P_{\iota-1}[f] - P_\iota[f]\|_{\infty,\mathcal{Q}} \\ &= \sum_{\iota=1}^{d} \left\| P_{\iota-1}\left[f - \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\right] \right\|_{\infty,\mathcal{Q}}. \end{aligned}$$

According to Lemma 4.18, we find

$$\|P_{\iota-1}[g]\|_{\infty,\mathcal{Q}} \leq \Big(\prod_{\kappa=1}^{\iota-1} \Lambda_{m_\kappa} \Big) \|g\|_{\infty,\mathcal{Q}} \quad \text{for all } g \in C(\mathcal{Q}),$$

and applying this to $g = f - \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]$ for $\iota \in \{1,\dots,m\}$ leads to

$$\|f - \mathfrak{I}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq \sum_{\iota=1}^{d} \Big(\prod_{\kappa=1}^{\iota-1} \Lambda_{m_\kappa} \Big) \|f - \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}}.$$

Now we can use Lemma 4.19 to conclude

$$\begin{aligned}
&\|f - \mathfrak{I}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \\
&\quad \leq 2eC_f \left(1 + \frac{\delta}{\gamma_f}\right) \sum_{\iota=1}^{d} \Big(\prod_{\kappa=1}^{\iota-1} \Lambda_{m_\kappa} \Big) (\Lambda_{m_\iota} + 1)(n_\iota + 1)^\sigma \varrho \left(\frac{2\gamma_f}{b_\iota - a_\iota} \right)^{-n_\iota} \\
&\quad \leq 2eC_f \left(1 + \frac{\delta}{\gamma_f}\right) \sum_{\iota=1}^{d} \Big(\prod_{\kappa=1}^{\iota} (\Lambda_{m_\kappa} + 1) \Big) (n_\iota + 1)^\sigma \varrho \left(\frac{2\gamma_f}{b_\iota - a_\iota} \right)^{-n_\iota} \\
&\quad \leq 2eC_f \Lambda_m \left(1 + \frac{\delta}{\gamma_f}\right) \sum_{\iota=1}^{d} (n_\iota + 1)^\sigma \varrho \left(\frac{2\gamma_f}{b_\iota - a_\iota} \right)^{-n_\iota}.
\end{aligned}$$

□

Theorem 4.20 provides us with a fairly general interpolation error estimate for the case of anisotropic boxes $\mathcal{Q}$: if the extent of the box in one direction is significantly smaller than in the other ones, estimate (4.35) allows us to use a lower interpolation order in this direction, i.e., to reach a higher efficiency.

In most applications, the clustering strategy is chosen in such a way that anisotropic boxes are avoided. In this situation, we can afford to choose approximately identical interpolation orders in all coordinate directions and base the error estimate only on the minimal order:

Corollary 4.21 (Isotropic interpolation). *Let $f \in C^\infty(\mathcal{Q})$, $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that (4.34) holds. Let $m \in \mathbb{N}^d$ be an order vector, and let*

$$n := \min\{m_\iota \,:\, \iota \in \{1, \ldots, d\}\}, \quad \Lambda_m := \prod_{\iota=1}^{d} \Lambda_{m_\iota},$$
$$\operatorname{diam}_\infty(\mathcal{Q}) := \max\{b_\iota - a_\iota \,:\, \iota \in \{1, \ldots, d\}\}.$$

Then we have

$$\|f - \mathfrak{I}_m^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq 2edC_f \Lambda_m (n+1)^\sigma \left(1 + \frac{\operatorname{diam}_\infty(\mathcal{Q})}{\gamma_f}\right) \varrho \left(\frac{2\gamma_f}{\operatorname{diam}_\infty(\mathcal{Q})} \right)^{-n}. \tag{4.36}$$

Proof. Since ϱ is monotonic and

$$\frac{2\gamma_f}{b_\iota - a_\iota} \geq \frac{2\gamma_f}{\operatorname{diam}_\infty(\mathcal{Q})} \quad \text{holds for all } \iota \in \{1, \ldots, d\},$$

this is a simple consequence of Theorem 4.20. □

Application to the kernel function

Let us now apply the general error estimates of Theorem 4.20 and Corollary 4.21 to the problem of approximating the kernel function g by $\tilde{g}_{t,s}$.

Theorem 4.22 (Approximation error). *Let $\eta \in \mathbb{R}_{>0}$. We consider d-dimensional axis-parallel bounding boxes*

$$\mathcal{Q}_t = [a_1, b_1] \times \cdots \times [a_d, b_d], \quad \mathcal{Q}_s = [a_{d+1}, b_{d+1}] \times \cdots \times [a_{2d}, b_{2d}],$$

satisfying the admissibility condition

$$\max\{\operatorname{diam}_\infty(\mathcal{Q}_t), \operatorname{diam}_\infty(\mathcal{Q}_s)\} = \operatorname{diam}_\infty(\mathcal{Q}_t \times \mathcal{Q}_s) \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s). \tag{4.37}$$

Let the kernel function g be $(C_{\mathrm{as}}, \sigma, c_0)$-asymptotically smooth with $\sigma > 0$. Let $m := (m_t, m_s)$ and

$$n := \min\{m_\iota \,:\, \iota \in \{1, \ldots, 2d\}\}, \quad \Lambda_m := \prod_{\iota=1}^{2d} (\Lambda_{m_\iota} + 1).$$

Then the separable approximation $\tilde{g}_{t,s}$ defined by (4.19) satisfies

$$|g(x,y) - \tilde{g}_{t,s}(x,y)| \leq \frac{4edC_{\mathrm{as}}\Lambda_m(n+1)^\sigma}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma}(1 + 2c_0\eta)\varrho\left(\frac{1}{c_0\eta}\right)^{-n}$$

for all $x \in \mathcal{Q}_t$ and $y \in \mathcal{Q}_s$.

Proof. Since g is $(C_{\mathrm{as}}, \sigma, c_0)$-asymptotically smooth, we have

$$\|\partial_\iota^\nu g\|_{\infty, \mathcal{Q}_t \times \mathcal{Q}_s} \leq \frac{C_{\mathrm{as}}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \frac{c_0^\nu}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} = \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix}$$

for all $\nu \in \mathbb{N}_0$, $\iota \in \{1, \ldots, 2d\}$, where we let

$$C_f := \frac{C_{\mathrm{as}}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma}, \quad \gamma_f := \frac{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)}{c_0}.$$

Applying Corollary 4.21 and observing

$$\frac{\operatorname{diam}_\infty(\mathcal{Q}_t \times \mathcal{Q}_s)}{\gamma_f} \leq 2c_0\eta, \quad \frac{2\gamma_f}{\operatorname{diam}_\infty(\mathcal{Q}_t \times \mathcal{Q}_s)} \geq \frac{1}{c_0\eta}$$

concludes the proof. □

The convergence rate of the interpolant is determined by $\varrho(1/(c_0\eta))^{-n}$. Since working with this quantity is slightly inconvenient, we replace it by simpler expressions: due to Remark 4.17, we can bound $\varrho(1/(c_0\eta))^{-1}$ from above by $(c_0\eta)/(c_0\eta + 1) < 1$ and see that the interpolant will always converge as long as $\eta > 0$ holds. On the other hand, we can bound $\varrho(1/(c_0\eta))^{-1}$ from above by $(c_0\eta)/2$ and thus reproduce the convergence rate estimates predicted by classical results. It is even possible to combine both estimates and simplify the estimate even further:

Remark 4.23. Let us assume that the interpolation scheme $(\mathfrak{I}_m)_{m=1}^{\infty}$ is (Λ, λ)-stable and that we use interpolation of constant order n, i.e., that the order vectors $(m_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(m_s)_{s\in\mathcal{T}_\mathcal{J}}$ satisfy $m_t \equiv n \equiv m_s$ for all $t \in \mathcal{T}_\mathcal{I}$, $s \in \mathcal{T}_\mathcal{J}$. Under these assumptions, the estimate of Theorem 4.22 can be simplified: we let

$$q := \min\left\{\frac{c_0\eta}{c_0\eta+1}, \frac{c_0\eta}{2}\right\} > \frac{c_0\eta}{1+\sqrt{c_0^2\eta^2+1}} = \frac{1}{\frac{1}{c_0\eta}+\sqrt{1+\frac{1}{c_0^2\eta^2}}} = \frac{1}{\varrho(1/(c_0\eta))},$$

and combining the estimate $\zeta := q\varrho(1/(c_0\eta)) > 1$ with Lemma 3.50 yields that there is a constant $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ satisfying

$$4edC_{\mathrm{as}}\Lambda^{2d}(n+2)^{2d\lambda}(n+1)^{\sigma}(1+2c_0\eta)\zeta^{-n} \leq C_{\mathrm{in}} \quad \text{for all } n \in \mathbb{N}.$$

Due to the stability assumption and $m_\iota = n$, we have

$$\Lambda_m = \prod_{\iota=1}^{2d}(\Lambda_{m_\iota}+1) \leq \prod_{\iota=1}^{2d}(\Lambda(m_\iota+1)^{\lambda}+1) \leq \prod_{\iota=1}^{2d}(\Lambda(n+2)^{\lambda}) = \Lambda^{2d}(n+2)^{2d\lambda}$$

and Theorem 4.22 yields

$$\|g-\tilde{g}_{t,s}\|_{\mathcal{Q}_t\times\mathcal{Q}_s} \leq \frac{C_{\mathrm{in}}}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^{\sigma}}q^n \quad \text{for all } b=(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, n \in \mathbb{N},$$

i.e., the interpolant will converge exponentially if the order n is increased. □

Remark 4.24 (Taylor expansion and interpolation). Although both Taylor expansion and interpolation provide polynomial approximations of the kernel function g, the latter approach has several significant advantages: the convergence rate of the Taylor expansion is bounded by $(c_0\eta)/(\eta+1)$, and we can expect no convergence if $c_0\eta$ is too large. The convergence rate of interpolation, on the other hand, is bounded by $(c_0\eta)/(c_0\eta+1) < 1$, so exponential convergence is always guaranteed, even though the rate of convergence may deteriorate for large values of the product $c_0\eta$. If η is small enough, the interpolant will converge with a rate of $c_0\eta/2$.

In addition, the parameter η in the admissibility condition (4.11) for the Taylor expansion has to be large enough to bound the *sum* of the diameters of $\mathcal{K}_t$ and $\mathcal{K}_s$, while in the condition (4.37) for interpolation only the maximum of the diameters has to be bounded. If the clusters are of comparable size, this means that the parameter η for the Taylor expansion has to be chosen twice as large as for interpolation, i.e., interpolation will converge even faster for similar block cluster trees. □

4.5 Approximation of derivatives

The theory presented so far requires the kernel function g to be asymptotically smooth on the entire domain $\{(x,y) \in \mathbb{R}^d \times \mathbb{R}^d : x \neq y\}$. In some applications, e.g., when

dealing with boundary integral equations, the function g will not be globally defined, i.e., our definition of asymptotically smooth functions will not apply. Fortunately, g can usually be expressed as a product of a derivative of a globally-defined asymptotically smooth *generator function*

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

and a separable function.

An example is the classical double layer potential operator

$$\mathcal{G}_{\mathrm{DLP}}[u](x) = \frac{1}{2\pi} \int_\Omega \frac{\langle x - y, n(y)\rangle}{\|x - y\|_2^2} u(y)\, dy$$

given on a curve $\Omega \subset \mathbb{R}^2$ in two spatial dimensions, where for each $y \in \Omega$, $n(y) \in \mathbb{R}^2$ is the outward-pointing normal vector. The corresponding kernel function

$$g(x, y) = \frac{1}{2\pi} \frac{\langle x - y, n(y)\rangle}{\|x - y\|_2^2}$$

is only defined on Ω and will, in general, not be smooth, since n does not have to be a smooth function.

Construction of a separable approximation

We can see that

$$g(x, y) = \langle \operatorname{grad}_y \gamma(x, y), n(y)\rangle \quad \text{holds for all } x, y \in \Omega \text{ with } x \neq y, \tag{4.38}$$

where the generator function is given by

$$\gamma\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \quad (x, y) \mapsto \begin{cases} -\frac{1}{2\pi} \log \|x - y\|_2 & \text{if } x \neq y, \\ 0 & \text{otherwise.} \end{cases}$$

The generator function is asymptotically smooth (cf. Appendix E of [63]). The equation (4.38) suggests two different approaches to approximating g: since γ is asymptotically smooth, the same will hold for the components of $\operatorname{grad}_y \gamma$, so we can use Taylor expansions or interpolation to find separable approximations of these components and combine them to form a separable approximation of the gradient. This approach will lead to optimal convergence, but it will also increase the rank of the approximation by a factor of d, since each component of $\operatorname{grad}_y \gamma$ has to be approximated individually.

We can avoid the higher rank and the corresponding higher complexity of the implementation by constructing a local approximation $\tilde{\gamma}_{t,s}$ of the generator function γ and using $\operatorname{grad}_y \tilde{\gamma}_{t,s}$ as an approximation of $\operatorname{grad}_y \gamma$.

In short: the first approach uses an approximation of the derivative, while the second uses the derivative of an approximation. The first approach is covered by the theory in the previous section, so we focus on the second approach.

For each admissible block $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we can construct a local separable approximation

$$\tilde{\gamma}_{t,s}(x, y) = \sum_{\nu\in K_t} \sum_{\mu\in L_s} (S_b)_{\nu\mu} v_{t,\nu}(x) w_{s,\mu}(y) \quad \text{for all } x \in \Omega_t,\ y \in \Omega_s,$$

by Taylor expansion or interpolation, where $S_b \in \mathbb{R}^{K_t\times L_s}$ is the corresponding coupling matrix and $(v_{t,\nu})_{\nu\in K_t}$ and $(w_{s,\mu})_{\mu\in L_s}$ are suitable expansion functions.

If we assume that the functions $(w_{s,\mu})_{\mu\in L_s}$ are differentiable, we can replace γ in (4.38) by $\tilde{\gamma}_{t,s}$ and obtain

$$\begin{aligned}\tilde{g}_{t,s}(x, y) &:= \langle \operatorname{grad}_y \tilde{\gamma}_{t,s}(x, y), n(y)\rangle \\ &= \sum_{\nu\in K_t} \sum_{\mu\in L_s} (S_b)_{\nu\mu} v_{t,\nu}(x) \langle \operatorname{grad}_y w_{s,\mu}(y), n(y)\rangle \\ &= \sum_{\nu\in K_t} \sum_{\mu\in L_s} (S_b)_{\nu\mu} v_{t,\nu}(x) \tilde{w}_{s,\mu}(y) \quad \text{for all } x \in \Omega_t,\ y \in \Omega_s,\end{aligned}$$

where we let

$$\tilde{w}_{s,\mu}(y) := \langle \operatorname{grad}_y w_{s,\mu}(y), n(y)\rangle \quad \text{for all } y \in \Omega,\ \mu \in L_s,$$

i.e., we have found a separable approximation $\tilde{g}_{t,s}$ of g.

In order to bound the error introduced by replacing g with $\tilde{g}_{t,s}$, we have to find estimates for

$$\begin{aligned}|g(x, y) - \tilde{g}_{t,s}(x, y)| &= |\langle \operatorname{grad}_y(\gamma - \tilde{\gamma}_{t,s})(x, y), n(y)\rangle| \\ &\leq \|\operatorname{grad}_y(\gamma - \tilde{\gamma}_{t,s})(x, y)\|_2 \|n(y)\|_2\end{aligned}$$

in all points $x, y \in \Omega_t \times \Omega_s$. Since $\|n(y)\|_2 = 1$ holds for all $y \in \Omega$, we face the task of finding a bound for $\|\operatorname{grad}_y(\gamma - \tilde{\gamma}_{t,s})(x, y)\|_2$ in all points $x \in \Omega_t$, $y \in \Omega_s$, i.e., we require error bounds for the derivatives of Taylor expansions or interpolants.

Error analysis in the one-dimensional case

Due to its theoretical and practical advantages (cf. Remarks 4.9 and 4.24), we focus on approximation by interpolation, i.e., we will only consider the approximation

$$\tilde{\gamma}_{t,s}(x, y) := \mathcal{I}^{\mathcal{Q}_t\times\mathcal{Q}_s}_{(m_t,m_s)}[\gamma](x, y) = \sum_{\nu\in K_t} \sum_{\mu\in L_s} \gamma(\xi_{t,\nu}, \xi_{s,\mu}) \mathcal{L}_{t,\nu}(x) \mathcal{L}_{s,\mu}(y) \tag{4.39}$$

for $x \in \mathcal{Q}_t$, $y \in \mathcal{Q}_s$, where K_t, L_s, $(\xi_{t,\nu})_{\nu\in K_t}$, $(\xi_{s,\mu})_{\mu\in L_s}$, $(\mathcal{L}_{t,\nu})_{\nu\in K_t}$ and $(\mathcal{L}_{s,\mu})_{\mu\in L_s}$ are defined as in Section 4.4.

We base our analysis on one-dimensional results and derive multi-dimensional estimates by tensor arguments.

Let us start by considering the approximation of f' for a function $f \in C^1[-1,1]$. We have to prove that the interpolation error $\|f' - (\mathfrak{I}_m[f])'\|_{\infty,[-1,1]}$ is small.

As in the previous section, the proof is split into two parts: we start by proving a stability estimate for derivatives of polynomials, since this allows us to bound the error of the interpolation by $\|f' - p'\|_{\infty,[-1,1]}$ for an arbitrary polynomial $p \in \mathcal{P}_m$.

Now we have to construct a suitable polynomial $p \in \mathcal{P}_m$. We do this by an indirect approach: we find a polynomial $p_0 \in \mathcal{P}_{m-1}$ approximating the derivative f' of f and let p be an antiderivative of p_0, i.e., we have $p' = p_0$ and $\|f' - p'\|_{\infty,[-1,1]} = \|f' - p_0\|_{\infty,[-1,1]}$. This allows us to re-use the approximation results of the previous section.

The stability estimate for derivatives of polynomials is based on an inverse estimate for polynomials:

Lemma 4.25 (Markov's inequality). *Let $m \in \mathbb{N}$. We have*

$$\|u'\|_{\infty,[-1,1]} \leq m^2 \|u\|_{\infty,[-1,1]} \quad \text{for all } u \in \mathcal{P}_m.$$

Proof. See, e.g., Theorem 4.1.4 in [42]. □

In order to be able to handle higher derivatives, we use the following straightforward generalization of Markov's inequality:

Lemma 4.26 (Markov's inequality iterated). *Let $m \in \mathbb{N}$ and $\ell \in \mathbb{N}_0$. We have*

$$\|u^{(\ell)}\|_{\infty,[-1,1]} \leq \begin{cases} \left(\frac{m!}{(m-\ell)!}\right)^2 \|u\|_{\infty,[-1,1]} & \text{if } \ell \leq m, \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } u \in \mathcal{P}_m. \tag{4.40}$$

Proof. By induction on ℓ. For $\ell = 0$, the inequality is trivial.

Let now $\ell \in \mathbb{N}_0$ be such that (4.40) holds. We have to prove the estimate for $\ell + 1$. If $\ell \geq m$ holds, we have $\ell + 1 > m$, i.e., $u^{(\ell+1)} = 0$ and (4.40) is trivial.

Otherwise, we have $u^{(\ell)} \in \mathcal{P}_{m-\ell}$ and can apply Lemma 4.25 in order to get

$$\|u^{(\ell+1)}\|_{\infty,[-1,1]} = \|(u^{(\ell)})'\|_{\infty,[-1,1]} \leq (m-\ell)^2 \|u^{(\ell)}\|_{\infty,[-1,1]}.$$

Using the induction assumption, we can conclude

$$\begin{aligned} \|u^{(\ell+1)}\|_{\infty,[-1,1]} &\leq (m-\ell)^2 \|u^{(\ell)}\|_{\infty,[-1,1]} \leq (m-\ell)^2 \left(\frac{m!}{(m-\ell)!}\right)^2 \|u\|_{\infty,[-1,1]} \\ &= \left(\frac{m!}{(m-(\ell+1))!}\right)^2 \|u\|_{\infty,[-1,1]}, \end{aligned}$$

and this is the estimate (4.40) for $\ell + 1$. □

Combining Markov's inequality with a simple Taylor approximation yields a stability estimate for derivatives of interpolants:

Lemma 4.27 (Stability of derivatives). *Let $[a,b] \subseteq \mathbb{R}$ be a non-trivial interval. Let $\ell \in \mathbb{N}_0$ and let $m \in \mathbb{N}_{\geq \ell}$. We have*

$$\|(\mathfrak{I}_m^{[a,b]}[f])^{(\ell)}\|_{\infty,[a,b]} \leq \Lambda_m^{(\ell)} \|f^{(\ell)}\|_{\infty,[a,b]} \quad \text{for all } f \in C^\ell[a,b],$$

where the stability constant is given by

$$\Lambda_m^{(\ell)} := \frac{\Lambda_m}{\ell!} \left(\frac{m!}{(m-\ell)!} \right)^2.$$

Proof. Let $f \in C^\ell[a,b]$. Let $\hat{f} := f \circ \Phi_{[a,b]} \in C^\ell[-1,1]$. Since we are only interested in the ℓ-th derivative of $\hat{f}$, we can subtract any polynomial of degree less than ℓ without changing the derivative. We choose the truncated Taylor expansion

$$\tilde{f} \colon [-1,1] \to \mathbb{R}, \quad x \mapsto \sum_{\nu=0}^{\ell-1} \hat{f}^{(\nu)}(0) \frac{x^\nu}{\nu!},$$

observe $\tilde{f}^{(\ell)} = 0$ and $\mathfrak{I}_m[\tilde{f}] = \tilde{f}$ due to $\ell \leq m$, and conclude

$$(\mathfrak{I}_m[\hat{f}])^{(\ell)} = (\mathfrak{I}_m[\hat{f}])^{(\ell)} - \tilde{f}^{(\ell)} = (\mathfrak{I}_m[\hat{f}] - \mathfrak{I}_m[\tilde{f}])^{(\ell)} = (\mathfrak{I}_m[\hat{f} - \tilde{f}])^{(\ell)}. \tag{4.41}$$

We apply Lemma 4.26 and the stability estimate (4.21) to $u := \mathfrak{I}_m[\hat{f} - \tilde{f}] \in \mathcal{P}_m$ and get

$$\begin{aligned} \|(\mathfrak{I}_m[\hat{f} - \tilde{f}])^{(\ell)}\|_{\infty,[-1,1]} &\leq \left(\frac{m!}{(m-\ell)!} \right)^2 \|\mathfrak{I}_m[\hat{f} - \tilde{f}]\|_{\infty,[-1,1]} \\ &\leq \Lambda_m \left(\frac{m!}{(m-\ell)!} \right)^2 \|\hat{f} - \tilde{f}\|_{\infty,[-1,1]}. \end{aligned} \tag{4.42}$$

Since $\tilde{f}$ is the Taylor expansion of $\hat{f}$, we can apply Lemma 4.75 in order to get

$$\begin{aligned} |\hat{f}(z) - \tilde{f}(z)| &\leq \int_0^1 (1-t)^{\ell-1} |\hat{f}^{(\ell)}(tz)| \frac{|z|^\ell}{(\ell-1)!} \, dt \\ &\leq \frac{\|\hat{f}^{(\ell)}\|_{\infty,[-1,1]}}{(\ell-1)!} \int_0^1 (1-t)^{\ell-1} \, dt \\ &= \frac{\|\hat{f}^{(\ell)}\|_{\infty,[-1,1]}}{\ell!} \end{aligned} \tag{4.43}$$

for all $z \in [-1,1]$. Combining (4.41), (4.42) and (4.43) yields

$$\|(\mathfrak{I}_m[\hat{f}])^{(\ell)}\|_{\infty,[-1,1]} \leq \frac{\Lambda_m}{\ell!} \left(\frac{m!}{(m-\ell)!} \right) \|\hat{f}^{(\ell)}\|_{\infty,[-1,1]} = \Lambda_m^{(\ell)} \|\hat{f}^{(\ell)}\|_{\infty,[-1,1]}.$$

Due to the definition of $\mathfrak{I}_m^{[a,b]}$, we have

$$\begin{aligned}
\|(\mathfrak{I}_m^{[a,b]}[f])^{(\ell)}\|_{\infty,[a,b]} &= \|(\mathfrak{I}_m[\hat{f}] \circ \Phi_{[a,b]}^{-1})^{(\ell)}\|_{\infty,[a,b]} \\
&= \left(\frac{2}{b-a}\right)^{\ell} \|(\mathfrak{I}_m[\hat{f}])^{(\ell)} \circ \Phi_{[a,b]}^{-1}\|_{\infty,[a,b]} = \left(\frac{2}{b-a}\right)^{\ell} \|(\mathfrak{I}_m[\hat{f}])^{(\ell)}\|_{\infty,[-1,1]} \\
&\leq \Lambda_m^{(\ell)} \left(\frac{2}{b-a}\right)^{\ell} \|\hat{f}^{(\ell)}\|_{\infty,[-1,1]} = \Lambda_m^{(\ell)} \left(\frac{2}{b-a}\right)^{\ell} \|\hat{f}^{(\ell)} \circ \Phi_{[a,b]}^{-1}\|_{\infty,[a,b]} \\
&= \Lambda_m^{(\ell)} \|(\hat{f} \circ \Phi_{[a,b]}^{-1})^{(\ell)}\|_{\infty,[a,b]},
\end{aligned}$$

and this concludes the proof. □

We can combine this stability result with Lemma 4.15 to prove that the derivative of $\mathfrak{I}_m^{[a,b]}[f]$ is a good approximation of the derivative of f. Since we have to apply the lemma to the derivative, we have to modify the analyticity condition (4.27) accordingly.

Theorem 4.28 (Derived interpolation). *Let $[a,b] \in \mathbb{R}$ be a non-trivial interval. Let $\ell \in \mathbb{N}_0$. Let $f \in C^\infty[a,b]$, $C_f^{(\ell)} \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma^{(\ell)} \in \mathbb{N}$ be such that*

$$|f^{(\ell+\nu)}(x)| \leq \frac{C_f^{(\ell)}}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma^{(\ell)} - 1 \end{bmatrix} \quad \text{holds for all } x \in [a,b],\ \nu \in \mathbb{N}_0.$$

Let ϱ be defined as in (4.28). We have

$$\begin{aligned}
&\|f^{(\ell)} - (\mathfrak{I}_m^{[a,b]}[f])^{(\ell)}\|_{\infty,[a,b]} \\
&\qquad \leq 2eC_f^{(\ell)}(\Lambda_m^{(\ell)} + 1)(n+1)^{\sigma^{(\ell)}} \left(1 + \frac{b-a}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{b-a}\right)^{-n}
\end{aligned}$$

for all $m \in \mathbb{N}_{\geq \ell}$ and all $n \in \{1, \dots, m-\ell\}$.

Proof. Let $m \in \mathbb{N}_{\geq \ell}$ and $n \in \{0, \dots, m-\ell\}$. We let $\hat{f} := f \circ \Phi_{[a,b]} \in C^\infty[-1,1]$ and find

$$\begin{aligned}
|\hat{f}^{(\ell+\nu)}(x)| &= \left| \left(\frac{b-a}{2}\right)^{\ell+\nu} f^{(\ell+\nu)}(\Phi_{[a,b]}(x)) \right| \leq \frac{C_f^{(\ell)}}{\gamma_f^\nu} \left(\frac{b-a}{2}\right)^{\ell+\nu} \begin{bmatrix} \nu \\ \sigma^{(\ell)} - 1 \end{bmatrix} \\
&= \frac{\hat{C}_f^{(\ell)}}{\hat{\gamma}_f^\nu} \begin{bmatrix} \nu \\ \sigma^{(\ell)} - 1 \end{bmatrix} \quad \text{for all } x \in [-1,1],\ \nu \in \mathbb{N}_0
\end{aligned}$$

with the constants

$$\hat{C}_f^{(\ell)} := C_f^{(\ell)} \left(\frac{b-a}{2}\right)^{\ell}, \quad \hat{\gamma}_f := \gamma_f \frac{2}{b-a},$$

i.e., the derivative $\hat{f}^{(\ell)}$ satisfies the conditions of Lemma 4.15. This result allows us to find a polynomial $\hat{p}_0 \in \mathcal{P}_n$ satisfying

$$\|\hat{f}^{(\ell)} - \hat{p}_0\|_{\infty,[-1,1]} \leq 2e\widehat{C}_f^{(\ell)}(n+1)^{\sigma^{(\ell)}} \left(1 + \frac{2}{\hat{\gamma}_f}\right) \varrho(\hat{\gamma}_f)^{-n}.$$

For all $i \in \{1, \ldots, \ell\}$, we define polynomials $\hat{p}_i \in \mathcal{P}_{n+i}$ by

$$\hat{p}_i(x) = \int_{-1}^{x} \hat{p}_{i-1}(y)\, dy \quad \text{for all } x \in [-1, 1].$$

Due to this definition, we have $\hat{p}_i' = \hat{p}_{i-1}$ for all $i \in \{1, \ldots, \ell\}$. We let $\hat{p} := \hat{p}_\ell \in \mathcal{P}_{n+\ell} \subseteq \mathcal{P}_m$ and observe $\hat{p}^{(\ell)} = \hat{p}_0$. This means

$$\|\hat{f}^{(\ell)} - \hat{p}^{(\ell)}\|_{\infty,[-1,1]} = \|\hat{f}^{(\ell)} - \hat{p}_0\|_{\infty,[-1,1]} \leq 2e\widehat{C}_f^{(\ell)}(n+1)^{\sigma^{(\ell)}} \left(1 + \frac{2}{\hat{\gamma}_f}\right) \varrho(\hat{\gamma}_f)^{-n}.$$

We define $p := \hat{p} \circ \Phi_{[a,b]}^{-1}$ and find

$$\begin{aligned}
\|f^{(\ell)} - p^{(\ell)}\|_{\infty,[a,b]} &= \left(\frac{2}{b-a}\right)^{\ell} \|\hat{f}^{(\ell)} \circ \Phi_{[a,b]}^{-1} - \hat{p}^{(\ell)} \circ \Phi_{[a,b]}^{-1}\|_{\infty,[a,b]} \\
&= \left(\frac{2}{b-a}\right)^{\ell} \|\hat{f}^{(\ell)} - \hat{p}^{(\ell)}\|_{\infty,[-1,1]} \\
&\leq 2e \left(\frac{2}{b-a}\right)^{\ell} \widehat{C}_f^{(\ell)}(n+1)^{\sigma^{(\ell)}} \left(1 + \frac{2}{\hat{\gamma}_f}\right) \varrho(\hat{\gamma}_f)^{-n} \\
&\leq 2eC_f^{(\ell)}(n+1)^{\sigma^{(\ell)}} \left(1 + \frac{2}{\hat{\gamma}_f}\right) \varrho(\hat{\gamma}_f)^{-n}.
\end{aligned}$$

Since $\hat{p} \in \mathcal{P}_{n+\ell} \subseteq \mathcal{P}_m$ holds and since $\Phi_{[a,b]}$ is an affine map, we have $p \in \mathcal{P}_m$ and can use the stability result of Lemma 4.27 in order to conclude

$$\begin{aligned}
\|f^{(\ell)} - (\mathfrak{I}_m^{[a,b]}[f])^{(\ell)}\|_{\infty,[a,b]} &\leq \|f^{(\ell)} - p^{(\ell)}\|_{\infty,[a,b]} + \|(\mathfrak{I}_m^{[a,b]}[f-p])^{(\ell)}\|_{\infty,[a,b]} \\
&\leq (1 + \Lambda_m^{(\ell)})\|f^{(\ell)} - p^{(\ell)}\|_{\infty,[a,b]}.
\end{aligned}$$

Applying the bound for the right-hand term completes the proof. □

Error analysis in the multi-dimensional case

As in Section 4.4, we now extend the one-dimensional approximation result to the multi-dimensional setting by investigating the properties of the directional interpolation operators $\mathfrak{I}_{m,t}^{\mathfrak{Q}}$. We first establish the counterpart of Lemma 4.27:

Lemma 4.29 (Stability of directional interpolation). *Let $\mu \in \mathbb{N}_0^d$ with $\mu \leq m$. Let $\iota \in \{1, \dots, d\}$. We have*

$$\|\partial^\mu \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq \Lambda_{m_\iota}^{(\mu_\iota)} \|\partial^\mu f\|_{\infty,\mathcal{Q}} \quad \text{for all } f \in C^\infty(\mathcal{Q}).$$

Proof. Let $f \in C^\infty(\mathcal{Q})$. Due to (4.31), we have

$$\partial_\kappa^{\mu_\kappa} \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f] = \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[\partial_\kappa^{\mu_\kappa} f] \quad \text{for all } \kappa \in \{1, \dots, \iota-1, \iota+1, \dots, d\}. \tag{4.44}$$

In order to handle the differential operator $\partial_\iota^{\mu_\iota}$, we proceed as in the proof of Lemma 4.18: we fix $y_\kappa \in [a_\kappa, b_\kappa]$ for all $\kappa \in \{1, \dots, \iota-1, \iota+1, \dots, d\}$ and define the function

$$f_\iota \colon [a_\iota, b_\iota] \to \mathbb{R}, \quad x \mapsto f(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d).$$

We observe

$$\begin{aligned}
\partial_\iota^{\mu_\iota} f(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) &= f_\iota^{(\mu_\iota)}(x) && \text{for all } x \in [a_\iota, b_\iota], \\
\partial_\iota^{\mu_\iota} \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) &= (\mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota])^{(\mu_\iota)}(x) && \text{for all } x \in [a_\iota, b_\iota],
\end{aligned}$$

and applying Lemma 4.27 yields

$$\begin{aligned}
\left|\partial_\iota^{\mu_\iota} \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f](y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d)\right| &= \left|(\mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota])^{(\mu_\iota)}(x)\right| \\
&\leq \Lambda_{m_\iota}^{(\mu_\iota)} \|f_\iota^{(\mu_\iota)}\|_{\infty,[a_\iota,b_\iota]} \\
&= \Lambda_{m_\iota}^{(\mu_\iota)} \|f\|_{\infty,\mathcal{Q}}.
\end{aligned} \tag{4.45}$$

By definition, we have

$$\partial^\mu = \partial_1^{\mu_1} \dots \partial_d^{\mu_d},$$

so combining (4.44) and (4.45) concludes the proof. □

Next, we require a multi-dimensional counterpart of the one-dimensional interpolation error estimate provided by Theorem 4.28:

Lemma 4.30 (Directional interpolation error). *Let $\mu \in \mathbb{N}_0^d$ with $\mu \leq m$. Let $\iota \in \{1, \dots, d\}$. Let ϱ be defined by (4.28). Let $f \in C^\infty(\mathcal{Q})$, $C_f^{(\mu)} \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma^{(\mu)} \in \mathbb{N}$ be such that*

$$\|\partial_\iota^\nu \partial^\mu f\|_{\infty,\mathcal{Q}} \leq \frac{C_f^{(\mu)}}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma^{(\mu)} - 1 \end{bmatrix} \quad \text{holds for all } \nu \in \mathbb{N}_0.$$

Let $n \in \{1, \dots, m_\iota - \mu_\iota\}$. We have

$$\begin{aligned}
&\|\partial^\mu f - \partial^\mu \mathfrak{I}_{m,\iota}^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \\
&\quad \leq 2eC_f^{(\mu)}(\Lambda_{m_\iota}^{(\mu_\iota)} + 1)(n+1)^{\sigma^{(\mu)}} \left(1 + \frac{b_\iota - a_\iota}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{b_\iota - a_\iota}\right)^{-n}.
\end{aligned}$$

Proof. Let $\mu' := (\mu_1, \dots, \mu_{\iota-1}, 0, \mu_{\iota+1}, \dots, \mu_d) \in \mathbb{N}_0^d$. Let $y_\kappa \in [a_\kappa, b_\kappa]$ for all $\kappa \in \{1, \dots, \iota-1, \iota+1, \dots, d\}$. We define the function

$$f_\iota \colon [a_\iota, b_\iota] \to \mathbb{R}, \quad x \mapsto \partial^{\mu'} f(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d),$$

and since (4.44) implies $\partial^{\mu'} \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f] = \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[\partial^{\mu'} f]$, we observe

$$\begin{aligned}\partial^{\mu} f(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) &= \partial_\iota^{\mu_\iota} (\partial^{\mu'} f)(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) \\ &= f_\iota^{(\mu_\iota)}(x) \quad \text{for all } x \in [a_\iota, b_\iota],\end{aligned}$$

and

$$\begin{aligned}&\partial^{\mu} \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f](y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) \\ &\quad = \partial_\iota^{\mu_\iota} \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[\partial^{\mu'} f](y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d) \\ &\quad = (\mathfrak{I}_{m_\iota}^{[a_\iota,b_\iota]}[f_\iota])^{(\mu_\iota)}(x) \quad \text{for all } x \in [a_\iota, b_\iota].\end{aligned}$$

Our assumption implies

$$\begin{aligned}|f_\iota^{(\nu+\mu_\iota)}(x)| &= |\partial_\iota^{\nu} \partial^{\mu} f(y_1, \dots, y_{\iota-1}, x, y_{\iota+1}, \dots, y_d)| \\ &\le \frac{C_f^{(\mu)}}{\gamma_f^{\nu}} \begin{bmatrix} \nu \\ \sigma^{(\mu)} - 1 \end{bmatrix} \quad \text{for all } x \in [a_\iota, b_\iota],\ \nu \in \mathbb{N}_0,\end{aligned}$$

and we can apply Theorem 4.28 to obtain

$$\begin{aligned}&\| f_\iota^{(\mu_\iota)} - (\mathfrak{I}_m^{[a_\iota,b_\iota]}[f_\iota])^{(\mu_\iota)} \|_{\infty,[a_\iota,b_\iota]} \\ &\quad \le 2e C_f^{(\mu)} (\Lambda_{m_\iota}^{(\mu_\iota)} + 1)(n+1)^{\sigma^{(\mu)}} \left(1 + \frac{b_\iota - a_\iota}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{b_\iota - a_\iota}\right)^{-n}.\end{aligned}$$

Since the right-hand side of this estimate does not depend on the choice of the variables $y_1, \dots, y_{\iota-1}, y_{\iota+1}, \dots, y_d$, a look at the definition of f_ι reveals that the proof is already complete. □

Combining the Lemmas 4.29 and 4.30 allows us to prove the necessary estimate for the derivatives of the multi-dimensional tensor product interpolation operator $\mathfrak{I}_m^{\mathfrak{Q}}$:

Theorem 4.31 (Multi-dimensional interpolation error). *Let ϱ be defined as in (4.28). Let $\mu \in \mathbb{N}_0^d$. Let $f \in C^\infty(\mathfrak{Q})$, $C_f^{(\mu)} \in \mathbb{R}_{\ge 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma^{(\mu)} \in \mathbb{N}$ be such that*

$$\|\partial_\iota^{\nu} \partial^{\mu} f\|_{\infty,\mathfrak{Q}} \le \frac{C_f^{(\mu)}}{\gamma_f^{\nu}} \begin{bmatrix} \nu \\ \sigma^{(\mu)} - 1 \end{bmatrix} \quad \text{holds for all } \iota \in \{1, \dots, d\},\ \nu \in \mathbb{N}_0. \tag{4.46}$$

Let $m, n \in \mathbb{N}^d$ be order vectors with $m \ge \mu$ and $n \le m - \mu$. We let

$$\Lambda_m^{(\mu)} := \prod_{\iota=1}^{d} (\Lambda_{m_\iota}^{(\mu_\iota)} + 1), \quad \operatorname{diam}_\infty(\mathfrak{Q}) := \max\{b_\iota - a_\iota \ : \ \iota \in \{1, \dots, d\}\}.$$

Then the following error estimate holds:

$$\begin{aligned}&\|\partial^\mu f - \partial^\mu \mathfrak{I}_m^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}}\\ &\quad\le 2eC_f^{(\mu)}\Lambda_m^{(\mu)}\left(1+\frac{\operatorname{diam}_\infty(\mathfrak{Q})}{\gamma_f}\right)\sum_{\iota=1}^{d}(n_\iota+1)^{\sigma^{(\mu)}}\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n_\iota}.\end{aligned}$$

Proof. Let $\delta := \operatorname{diam}_\infty(\mathfrak{Q})$. As in the proof of Theorem 4.20, we define the operator

$$P_\iota := \prod_{\kappa=1}^{\iota}\mathfrak{I}_{m,\kappa}^{\mathfrak{Q}} \quad \text{for all } \iota\in\{0,\dots,d\},$$

and apply the triangle inequality to obtain

$$\|\partial^\mu f - \partial^\mu \mathfrak{I}_m^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}} \le \sum_{\iota=1}^{d}\left\|\partial^\mu P_{\iota-1}\left[f-\mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f]\right]\right\|_{\infty,\mathfrak{Q}}.$$

We can use Lemma 4.29 in order to prove

$$\|\partial^\mu f - \partial^\mu \mathfrak{I}_m^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}} \le \sum_{\iota=1}^{d}\Big(\prod_{\kappa=1}^{\iota-1}\Lambda_{m_\kappa}^{(\mu_\kappa)}\Big)\left\|\partial^\mu f - \partial^\mu \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f]\right\|_{\infty,\mathfrak{Q}}.$$

Since $n_\iota \le m_\iota - \mu_\iota$ holds for all $\iota\in\{1,\dots,d\}$, we can apply Lemma 4.30 to find

$$\begin{aligned}&\|\partial^\mu f - \partial^\mu \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}}\\ &\quad\le 2eC_f^{(\mu)}(\Lambda_{m_\iota}^{(\mu_\iota)}+1)(n_\iota+1)^{\sigma^{(\mu)}}\left(1+\frac{b_\iota-a_\iota}{\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n_\iota}\\ &\quad\le 2eC_f^{(\mu)}(\Lambda_{m_\iota}^{(\mu_\iota)}+1)(n_\iota+1)^{\sigma^{(\mu)}}\left(1+\frac{\delta}{\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n_\iota}\end{aligned}$$

and conclude

$$\begin{aligned}&\|\partial^\mu f - \partial^\mu \mathfrak{I}_m^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}}\\ &\quad\le \sum_{\iota=1}^{d}\Big(\prod_{\kappa=1}^{\iota-1}\Lambda_{m_\kappa}^{(\mu_\kappa)}\Big)\|\partial^\mu f - \partial^\mu \mathfrak{I}_{m,\iota}^{\mathfrak{Q}}[f]\|_{\infty,\mathfrak{Q}}\\ &\quad\le 2eC_f^{(\mu)}\left(1+\frac{\delta}{\gamma_f}\right)\sum_{\iota=1}^{d}\Big(\prod_{\kappa=1}^{\iota-1}\Lambda_{m_\kappa}^{(\mu_\kappa)}\Big)(\Lambda_{m_\iota}^{(\mu_\iota)}+1)(n_\iota+1)^{\sigma^{(\mu)}}\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n_\iota}\\ &\quad\le 2eC_f^{(\mu)}\Lambda_m^{(\mu)}\left(1+\frac{\delta}{\gamma_f}\right)\sum_{\iota=1}^{d}(n_\iota+1)^{\sigma^{(\mu)}}\varrho\left(\frac{2\gamma_f}{b_\iota-a_\iota}\right)^{-n_\iota}.\end{aligned}$$

□

In an isotropic situation, we can simplify the error estimate provided by Theorem 4.31:

Corollary 4.32 (Isotropic interpolation). *Let ϱ be defined as in (4.28). Let $\mu \in \mathbb{N}_0^d$. Let $f \in C^\infty(\mathcal{Q})$, $C_f^{(\mu)} \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma^{(\mu)} \in \mathbb{N}$ be such that (4.46) holds. Let $m \in \mathbb{N}^d$ be an order vector with $m \geq \mu$, and let*

$$n := \min\{m_\iota - \mu_\iota \,:\, \iota \in \{1,\dots,d\}\}, \quad \Lambda_m^{(\mu)} := \prod_{\iota=1}^{d} (\Lambda_{m_\iota}^{(\mu_\iota)} + 1),$$
$$\operatorname{diam}_\infty(\mathcal{Q}) := \max\{b_\iota - a_\iota \,:\, \iota \in \{1,\dots,d\}\}.$$

Then the following error estimate holds:

$$\|\partial^\mu f - \partial^\mu \mathcal{I}_m^{\mathcal{Q}}[f]\|_{\infty,\mathcal{Q}} \leq 2de C_f^{(\mu)} \Lambda_m^{(\mu)} (n+1)^{\sigma^{(\mu)}} \left(1 + \frac{\operatorname{diam}_\infty(\mathcal{Q})}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{\operatorname{diam}_\infty(\mathcal{Q})}\right)^{-n}.$$

Proof. This is a simple consequence of Theorem 4.31. □

Application to the kernel function

Now we can apply the general result of Theorem 4.31 to the kernel function $g = \partial^\mu \gamma$. In order to keep the expressions from becoming too complicated, we only consider the isotropic case:

Theorem 4.33 (Approximation error). *Let $\eta \in \mathbb{R}_{>0}$. We consider d-dimensional axis-parallel bounding boxes*

$$\mathcal{Q}_t = [a_1, b_1] \times \cdots \times [a_d, b_d], \quad \mathcal{Q}_s = [a_{d+1}, b_{d+1}] \times \cdots \times [a_{2d}, b_{2d}],$$

satisfying the admissibility condition

$$\max\{\operatorname{diam}_\infty(\mathcal{Q}_t), \operatorname{diam}_\infty(\mathcal{Q}_s)\} = \operatorname{diam}_\infty(\mathcal{Q}_t \times \mathcal{Q}_s) \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s).$$

Let $\mu \in \mathbb{N}_0^{2d}$ and let $m \in \mathbb{N}^{2d}$ be an order vector with $m \geq \mu$. Let $\partial^\mu \gamma$ be $(C_{\mathrm{as}}^{(\mu)}, \sigma^{(\mu)}, c_0)$-asymptotically smooth with $\sigma^{(\mu)} > 0$. We let

$$n := \min\{m_\iota - \mu_\iota \,:\, \iota \in \{1,\dots,2d\}\}, \quad \Lambda_m^{(\mu)} := \prod_{\iota=1}^{2d} (\Lambda_{m_\iota}^{(\mu_\iota)} + 1).$$

Then the separable approximation $\tilde{\gamma}_{t,s}$ defined by (4.39) satisfies

$$|\partial^\mu \gamma(x,y) - \partial^\mu \tilde{\gamma}_{t,s}(x,y)| \leq \frac{4ed C_{\mathrm{as}}^{(\mu)} \Lambda_m^{(\mu)} (n+1)^{\sigma^{(\mu)}}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma^{(\mu)}}} (1 + 2c_0\eta) \varrho\left(\frac{1}{c_0\eta}\right)^{-n}$$

for all $x \in \mathcal{Q}_t$ and $y \in \mathcal{Q}_s$.

Proof. We let $\mathcal{Q} := \mathcal{Q}_t \times \mathcal{Q}_s$, define

$$\delta := \operatorname{diam}_\infty(\mathcal{Q}), \quad C_f^{(\mu)} := \frac{C_{\mathrm{as}}^{(\mu)}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma(\mu)}}, \quad \gamma_f := \frac{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)}{c_0},$$

and observe that the asymptotic smoothness of the derivative $\partial^\mu \gamma$ of γ implies

$$\|\partial_t^\nu \partial^\mu \gamma\|_{\infty,\mathcal{Q}} \leq C_{\mathrm{as}}^{(\mu)} \begin{bmatrix} \nu \\ \sigma(\mu) - 1 \end{bmatrix} \frac{c_0^\nu}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma(\mu)+\nu}} = \frac{C_f^{(\mu)}}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma(\mu) - 1 \end{bmatrix}.$$

This means that we can apply Corollary 4.32 to $\mathcal{Q}$ in order to get

$$\begin{aligned} \|\partial^\mu \gamma - \partial^\mu \mathfrak{I}_m^{\mathcal{Q}}[\gamma]\|_{\infty,\mathcal{Q}} &\leq 4de C_f^{(\mu)} \Lambda_m^{(\mu)} (n+1)^{\sigma(\mu)} \left(1 + \frac{\delta}{\gamma_f}\right) \varrho\left(\frac{2\gamma_f}{\delta}\right)^{-n} \\ &= 4de \frac{C_{\mathrm{as}}^{(\mu)}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma(\mu)}} \Lambda_m^{(\mu)} (n+1)^{\sigma(\mu)} (1 + 2c_0\eta) \varrho\left(\frac{1}{c_0\eta}\right)^{-n} \end{aligned}$$

by using the admissibility condition $\delta \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)$. □

Remark 4.34. If σ is the order of the singularity of the generator function γ, its μ-th derivative will typically have a singularity of order $\sigma^{(\mu)} := \sigma + |\mu|$. If we disregard the polynomial factors, the results from Theorem 4.22 and Theorem 4.33 are quite similar: in the isotropic case, the first corresponds to the direct approximation of $g = \partial^\mu \gamma$ by $\tilde{g}$ and yields an estimate of the type

$$\|g - \tilde{g}\|_{\infty,\mathcal{Q}_t \times \mathcal{Q}_s} \leq \frac{C(m)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma+|\mu|}} \varrho\left(\frac{1}{c_0\eta}\right)^{-m}$$

for an interpolation order m, the second correspond to the approximation of g by $\partial^\mu \tilde{\gamma}$ and yields an estimate of the type

$$\|g - \partial^\mu \tilde{\gamma}\|_{\infty,\mathcal{Q}_t \times \mathcal{Q}_s} \leq \frac{C(m)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^{\sigma+|\mu|}} \varrho\left(\frac{1}{c_0\eta}\right)^{-m+|\mu|_\infty},$$

where we define $|\mu|_\infty := \max_\iota \mu_\iota$. Roughly speaking, the second approach yields the same approximation quality as the first for the reduced order $m - |\mu|_\infty$ of interpolation. This is not surprising, since polynomials of low order will not effect the derivative of $\tilde{\gamma}$ and therefore contribute nothing to the approximation. □

4.6 Matrix approximation

Due to the Theorems 4.8, 4.22 and 4.33, we can find bounds $(\epsilon_b)_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+}$ such that

$$|g(x,y) - \tilde{g}_{t,s}(x,y)| \leq \epsilon_b \quad \text{holds for all } x \in \Omega_t, y \in \Omega_s \tag{4.47}$$

and all admissible blocks $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Based on these bounds, we now investigate the error of the approximation of the matrix G by

$$\widetilde{G} = \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t G \chi_s + \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_b W_s^*$$

constructed in (4.5).

In order to simplify the presentation, we only consider the case that the basis functions $(\varphi_i)_{i\in\mathcal{I}}$ and $(\psi_j)_{j\in\mathcal{J}}$ are elements of $L^2(\Omega)$. More general vector spaces can be handled by the approach described in Remark 4.2.

We measure the error introduced by the $\mathcal{H}^2$-matrix approximation in the Frobenius and in the spectral norm.

Frobenius norm estimates

Let us first consider the Frobenius norm.

Definition 4.35 (Frobenius norm). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. The *Frobenius norm* of X is given by

$$\|X\|_F := \Big(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} X_{ij}^2\Big)^{1/2}.$$

In the model problem, we can base the analysis of the error on the element-wise estimate provided by (2.10), since the supports of the basis functions are disjoint and we can find bounds for their L^2-norms.

In general, we cannot assume that the supports are disjoint, since most popular finite element spaces do not satisfy this requirement. Instead, we assume that the overlap of the supports can be bounded by a constant, i.e., that in each point $x \in \Omega$ only a bounded number of basis functions can differ from zero.

Definition 4.36 (Overlapping supports). Let $(\Omega_i)_{i\in\mathcal{I}}$ be a family of supports in Ω (cf. Definition 3.20). Let $C_{\mathrm{ov}} \in \mathbb{N}$. If

$$\#\{i \in \mathcal{I} \,:\, x \in \Omega_i\} \leq C_{\mathrm{ov}} \quad \text{holds for almost all } x \in \Omega,$$

we call the family $(\Omega_i)_{i\in\mathcal{I}}$ C_{ov}-*overlapping*.

In the model case, i.e., for non-overlapping supports, the measure of Ω is the sum of the measures of the supports Ω_i. If a family of supports if C_{ov}-overlapping, we can still bound the sum of the measures of all supports Ω_i by the measure of Ω multiplied by the constant C_{ov}.

Lemma 4.37 (Bounded overlap). *Let $(\Omega_i)_{i\in\mathcal{I}}$ be a C_{ov}-overlapping family of supports in Ω. Then we have*

$$\sum_{i\in\hat{t}} |\Omega_i| \leq C_{\mathrm{ov}}|\Omega_t| \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$
$$\sum_{i\in\mathcal{I}} |\Omega_i| \leq C_{\mathrm{ov}}|\Omega|.$$

Proof. For all $i \in \mathcal{I}$, we introduce

$$\chi_i : \Omega \to \mathbb{R}, \quad x \mapsto \begin{cases} 1 & \text{if } x \in \Omega_i, \\ 0 & \text{otherwise.} \end{cases}$$

Let $t \in \mathcal{T}_{\mathcal{I}}$. Since $(\Omega_i)_{i\in\mathcal{I}}$ is C_{ov}-overlapping, we have

$$\begin{aligned}\sum_{i\in\hat{t}} |\Omega_i| &= \sum_{i\in\hat{t}} \int_\Omega \chi_i(x)\,dx = \int_\Omega \sum_{i\in\hat{t}} \chi_i(x)\,dx = \int_{\Omega_t} \sum_{i\in\hat{t}} \chi_i(x)\,dx \\ &= \int_{\Omega_t} \#\{i \in \mathcal{I} : x \in \Omega_i\}\,dx \leq \int_{\Omega_t} C_{\mathrm{ov}}\,dx = C_{\mathrm{ov}}|\Omega_t|.\end{aligned}$$

Applying this result to the root of $\mathcal{T}_{\mathcal{I}}$ yields the second estimate of this lemma. □

Using this estimate, we can derive the following estimate for the Frobenius norm of the blockwise error:

Lemma 4.38 (Blockwise error). *Let $(\epsilon_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be a family satisfying (4.47). Let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ be C_{ov}-overlapping. Let*

$$C_{\mathcal{I}} := \max\{\|\varphi_i\|_{L^2} : i \in \hat{t}\}, \quad C_{\mathcal{J}} := \max\{\|\psi_j\|_{L^2} : j \in \hat{s}\}. \tag{4.48}$$

We have

$$\|\chi_t G \chi_s - V_t S_b W_s^*\|_F \leq C_{\mathcal{I}} C_{\mathcal{J}} \epsilon_b \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)^{1/2} \Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2}$$

for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Proof. Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. According to Definition 4.35 and (4.4), we have

$$\begin{aligned}\|\chi_t G \chi_s - V_t S_b W_s^*\|_F^2 &= \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} \left(\int_\Omega \varphi_i(x) \int_\Omega (g(x,y) - \tilde{g}_{t,s}(x,y))\psi_j(y)\,dy\,dx\right)^2 \\ &\leq \epsilon_b^2 \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} \left(\int_\Omega |\varphi_i(x)|\,dx\right)^2 \left(\int_\Omega |\psi_j(y)|\,dy\right)^2.\end{aligned}$$

Let us consider the sum over $i \in \mathcal{I}$. We use the Cauchy–Schwarz inequality in order to get

$$\begin{aligned}\sum_{i\in\hat{t}} \left(\int_\Omega |\varphi_i(x)| \, dx \right)^2 &\leq \sum_{i\in\hat{t}} |\Omega_i| \int_\Omega \varphi_i(x)^2 \, dx \\ &\leq \left(\max\{ \|\varphi_i\|_{L^2}^2 \, : \, i \in \hat{t} \} \right) \sum_{i\in\hat{t}} |\Omega_i| \\ &\leq C_{\mathcal{I}} \sum_{i\in\hat{t}} |\Omega_i|.\end{aligned}$$

Applying the same reasoning to the sum over $j \in \mathcal{J}$ yields the desired estimate. □

Now we can combine the blockwise error estimates in order to derive a global estimate. Due to the structure of the Frobenius norm, combining blockwise estimates is straightforward:

Lemma 4.39 (Global Frobenius norm). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|X\|_F = \Big(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|\chi_t X \chi_s\|_F^2 \Big)^{1/2}.$$

Proof. We once more use the equation

$$X = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s.$$

The matrices χ_t and χ_s are orthogonal projections for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$, and the definition of the norm implies

$$\begin{aligned}\|X\|_F^2 = \langle X, X \rangle_F &= \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle \chi_t X \chi_s, X \rangle_F \\ &= \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle \chi_t X \chi_s, \chi_t X \chi_s \rangle_F \\ &= \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|\chi_t X \chi_s\|_F^2.\end{aligned}$$

□

Using Lemma 4.39 in combination with the blockwise error estimate of Lemma 4.38 yields the following error bound:

Lemma 4.40 (Frobenius error bound). *Let $(\epsilon_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be a family satisfying (4.47). Let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ be C_{ov}-overlapping. Let $C_{\mathcal{I}}, C_{\mathcal{J}} \in \mathbb{R}_{\geq 0}$ be defined as in (4.48). We have*

$$\|G - \tilde{G}\|_F \leq C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} |\Omega| \max\{\epsilon_b \, : \, b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}.$$

Proof. We apply Lemma 4.39 to the matrix $X := G - \widetilde{G}$ and use Lemma 4.38 in order to obtain

$$\begin{aligned}\|G-\widetilde{G}\|_F^2 &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \|\chi_t G\chi_s - V_t S_b W_s^*\|_F^2\\ &\leq C_{\mathcal{I}}^2 C_{\mathcal{J}}^2 \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \epsilon_b^2 \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)\\ &\leq C_{\mathcal{I}}^2 C_{\mathcal{J}}^2 \max\{\epsilon_b^2 \,:\, b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\} \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big).\end{aligned}$$

Combining Lemma 3.14 with Corollary 3.9 yields that $\{\hat{t}\times\hat{s} \,:\, b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}\}$ is a disjoint partition of $\mathcal{I}\times\mathcal{J}$, and we find

$$\begin{aligned}\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big) &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} |\Omega_i|\,|\Omega_j|\\ &= \Big(\sum_{i\in\mathcal{I}} |\Omega_i|\Big)\Big(\sum_{j\in\mathcal{J}} |\Omega_j|\Big) \leq C_{\mathrm{ov}}^2 |\Omega|^2\end{aligned}$$

by using Lemma 4.37 in the last step. Now we can conclude

$$\|G-\widetilde{G}\|_F^2 \leq C_{\mathrm{ov}}^2 C_{\mathcal{I}}^2 C_{\mathcal{J}}^2 \max\{\epsilon_b^2 \,:\, b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}|\Omega|^2. \qquad \square$$

This result is a generalization of the corresponding estimate for the model problem: if we let $C_{\mathrm{ov}} = 1$ (i.e., no overlap between the supports of the basis functions), $C_{\mathcal{I}} = C_{\mathcal{J}} = n^{-1/2}$ (corresponding to piecewise constant basis functions) and use the bound $\epsilon_b = \log(\eta+1)(\eta/(\eta+1))^{m-1}$ established in Corollary 2.3, we recover the estimate (2.19) derived for the model problem in Chapter 2.

Let us now consider the application of this norm estimate to the kernel approximations constructed by Taylor expansion and interpolation. We consider only the case $\sigma > 0$. Comparing the Theorems 4.8, 4.22 and 4.33 yields the error bounds

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{K}_t\times\mathcal{K}_s} \leq \frac{C_g q^n}{\operatorname{dist}(\mathcal{K}_t,\mathcal{K}_s)^\sigma} \qquad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

if Taylor expansions are used,

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq \frac{C_g q^n}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^\sigma} \qquad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

if interpolation is applied and

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq \frac{C_g q^n}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^{\sigma+|\mu|}} \qquad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

if the μ-th partial derivative of an interpolant is considered. Here $C_g \in \mathbb{R}_{>0}$ is a constant which does not depend on n and b, and $q \in [0,1[$ is the rate of convergence (recall that the admissibility parameter η has to be sufficiently small in the case of the Taylor expansion).

In the case of the Taylor approximation, the corresponding admissibility condition (4.11) implies

$$\begin{aligned}\operatorname{diam}(\mathcal{K}_t)^{1/2}\operatorname{diam}(\mathcal{K}_s)^{1/2} &\leq \frac{1}{2}(\operatorname{diam}(\mathcal{K}_t)+\operatorname{diam}(\mathcal{K}_s))\\ &\leq \eta\operatorname{dist}(\mathcal{K}_t,\mathcal{K}_s) \quad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},\end{aligned}$$

and the error estimate takes the form

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{K}_t\times\mathcal{K}_s} \leq \frac{C_g\eta^\sigma q^n}{\operatorname{diam}(\mathcal{K}_t)^{\sigma/2}\operatorname{diam}(\mathcal{K}_s)^{\sigma/2}}.$$

In order to derive a similar result for approximations constructed by interpolation, we have to replace the admissibility condition (4.37) by the slightly stronger condition

$$\max\{\operatorname{diam}(\mathcal{Q}_t),\operatorname{diam}(\mathcal{Q}_s)\} \leq 2\eta\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s) \tag{4.49}$$

and observe that it yields

$$\begin{aligned}\operatorname{diam}(\mathcal{Q}_t)^{1/2}\operatorname{diam}(\mathcal{Q}_s)^{1/2} &\leq \max\{\operatorname{diam}(\mathcal{Q}_t),\operatorname{diam}(\mathcal{Q}_s)\}\\ &\leq 2\eta\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s) \quad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\end{aligned} \tag{4.50}$$

if the block cluster tree is constructed based on the new admissibility condition. Then the error estimate implies

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq \frac{C_g(2\eta)^\sigma q^n}{\operatorname{diam}(\mathcal{Q}_t)^{\sigma/2}\operatorname{diam}(\mathcal{Q}_s)^{\sigma/2}}$$

in the case of the interpolation and

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq \frac{C_g(2\eta)^{\sigma+|\mu|}q^n}{\operatorname{diam}(\mathcal{Q}_t)^{(\sigma+|\mu|)/2}\operatorname{diam}(\mathcal{Q}_s)^{(\sigma+|\mu|)/2}}$$

if μ-th partial derivatives of interpolants are used.

Since all three cases can be handled in a similar fashion, we restrict our attention to approximations constructed by interpolation, i.e., we use

$$\|g-\tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq \epsilon_b := \frac{C_{g,\eta}q^n}{\operatorname{diam}(\mathcal{Q}_t)^{\sigma/2}\operatorname{diam}(\mathcal{Q}_s)^{\sigma/2}} \tag{4.51}$$

for all $b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, where we let $C_{g,\eta} := C_g(2\eta)^\sigma$. Combining this estimate with Lemma 4.40 yields the following global error bound for the Frobenius norm:

Theorem 4.41 (Frobenius approximation error). *Let $C_{\mathcal{I}}, C_{\mathcal{J}} \in \mathbb{R}_{\geq 0}$ be defined as in (4.48), and let $q \in [0,1[$ be the rate of convergence introduced in (4.51). Let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ be C_{ov}-overlapping. For all admissible leaves $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we assume that the $\tilde{g}_{t,s}$ satisfies (4.51). Then we have*

$$\|G - \widetilde{G}\|_F \leq \frac{C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} C_{g,\eta} |\Omega|}{\min\{\operatorname{diam}(\mathcal{Q}_t)\operatorname{diam}(\mathcal{Q}_s) \,:\, b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}^{\sigma/2}} q^n. \tag{4.52}$$

Proof. Combine Lemma 4.40 with (4.51). □

Remark 4.42 (Asymptotic behaviour of the error). Let us assume that Ω is a d_Ω-dimensional subset or submanifold of $\mathbb{R}^d$, and that the discretization is based on a quasi-uniform hierarchy of grids with a decreasing sequence $h_0, h_1, \ldots$ of mesh parameters.

The minimum in estimate (4.52) is attained for leaf clusters, and we can assume that the diameters of their bounding boxes are approximately proportional to the grid parameter. If we neglect the constants $C_{\mathcal{I}}$ and $C_{\mathcal{J}}$ corresponding to the scaling of the basis functions, the Frobenius approximation error on mesh level ℓ can be expected to behave like $h_\ell^{-\sigma} q^n$. □

Spectral norm estimates

Now let us focus on the spectral norm of the error $G - \widetilde{G}$ of the $\mathcal{H}^2$-matrix approximation.

Definition 4.43 (Spectral norm). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. The *spectral norm* (or *operator norm*) of X is given by

$$\|X\|_2 := \sup\left\{\frac{\|Xu\|_2}{\|u\|_2} \,:\, u \in \mathbb{R}^{\mathcal{J}} \setminus \{0\}\right\}.$$

Under the same conditions as in the case of the Frobenius norm we can also derive a blockwise error estimate for the spectral norm:

Lemma 4.44 (Blockwise error). *Let $(\epsilon_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be a family satisfying (4.47). Let $C_{\mathcal{I}}, C_{\mathcal{J}} \in \mathbb{R}_{>0}$ be defined as in (4.48). We have*

$$\|\chi_t G \chi_s - V_t S_b W_s^*\|_2 \leq C_{\mathcal{I}} C_{\mathcal{J}} \epsilon_b \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)^{1/2} \Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2}$$

for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Proof. Let $E := \chi_t G \chi_s - V_t S_b W_s^*$. Let $u \in \mathbb{R}^{\mathcal{J}}$. Due to

$$\|Eu\|_2^2 = \sum_{i\in\mathcal{I}} (Eu)_i^2 = \sum_{i\in\mathcal{I}} \Big(\sum_{j\in\mathcal{J}} E_{ij} u_j\Big)^2 \leq \sum_{i\in\mathcal{I}} \Big(\sum_{j\in\mathcal{J}} E_{ij}^2\Big)\Big(\sum_{j\in\mathcal{J}} u_j^2\Big)$$

$$= \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} E_{ij}^2 \|u\|_2^2 = \|E\|_F^2 \|u\|_2^2,$$

we can conclude by using the Frobenius norm estimate of Lemma 4.38. □

In order to find a global error bound, we have to combine these blockwise error bounds using the operator-norm counterpart of Lemma 4.39:

Lemma 4.45 (Global spectral norm). *Let $X \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$. We have*

$$\|X\|_2 \leq \Big(\sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t X \chi_s\|_2^2 \Big)^{1/2}.$$

Proof. Let $v \in \mathbb{R}^{\mathcal{I}}$ and $u \in \mathbb{R}^{\mathcal{J}}$. We have

$$\begin{aligned} |\langle v, Xu \rangle_2| &= \Big| \sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \langle v, \chi_t X \chi_s u \rangle_2 \Big| \leq \sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} |\langle \chi_t v, \chi_t X \chi_s \chi_s u \rangle_2| \\ &\leq \sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t X \chi_s\|_2 \|\chi_t v\|_2 \|\chi_s u\|_2 \\ &\leq \Big(\sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t X \chi_s\|_2^2 \Big)^{1/2} \Big(\sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t v\|_2^2 \|\chi_s u\|_2^2 \Big)^{1/2}. \end{aligned}$$

Combining Lemma 3.14 and Corollary 3.9 yields that $\mathcal{L}_{\mathcal{I} \times \mathcal{J}}$ corresponds to a disjoint partition of $\mathcal{I} \times \mathcal{J}$, i.e., we have

$$\sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t v\|_2^2 \|\chi_s u\|_2^2 = \|v\|_2^2 \|u\|_2^2$$

and conclude

$$|\langle v, Xu \rangle_2| \leq \Big(\sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}}} \|\chi_t X \chi_s\|_2^2 \Big)^{1/2} \|u\|_2 \|v\|_2.$$

Setting $v := Xu$ and proceeding as in the proof of Lemma 4.44 proves our claim. □

We can combine the Lemmas 4.44 and 4.45 in order to prove the following bound for the $\mathcal{H}^2$-matrix approximation error:

Lemma 4.46 (Spectral error bound). *Let $(\epsilon_b)_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}}$ be a family of positive real numbers satisfying (4.47). Let the families $(\Omega_i)_{i \in \mathcal{I}}$ and $(\Omega_j)_{j \in \mathcal{J}}$ be C_{ov}-overlapping, and let $C_{\mathcal{I}}, C_{\mathcal{J}} \in \mathbb{R}_{>0}$ be defined as in (4.48). We have*

$$\|G - \widetilde{G}\|_2 \leq C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} |\Omega| \max\{\epsilon_b \ : \ b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}\}.$$

Proof. We apply Lemma 4.45 to the matrix $X := G - \widetilde{G}$ and use Lemma 4.44 in order to obtain

$$\begin{aligned}\|G-\widetilde{G}\|_2^2 &\leq \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \|\chi_t G \chi_s - V_t S_b W_s^*\|_2^2 \\ &\leq C_{\mathcal{I}}^2 C_{\mathcal{J}}^2 \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \epsilon_b^2 \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big) \\ &\leq C_{\mathcal{I}}^2 C_{\mathcal{J}}^2 \max\{\epsilon_b^2 \,:\, b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\} \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big).\end{aligned}$$

As in the proof of Lemma 4.40, we combine Lemma 3.14 with Corollary 3.9 and find

$$\begin{aligned}\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big) &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} |\Omega_i|\,|\Omega_j| \\ &= \Big(\sum_{i\in\mathcal{I}} |\Omega_i|\Big)\Big(\sum_{j\in\mathcal{J}} |\Omega_j|\Big) \leq C_{\mathrm{ov}}^2 |\Omega|^2\end{aligned}$$

by using Lemma 4.37 in the last step. □

The estimate of Lemma 4.45 is quite straightforward, general and completely sufficient for many standard applications, but it is, in general, far from optimal: the Lemmas 4.40 and 4.46 even yield exactly the same upper bounds for Frobenius and spectral norm, although the spectral norm will indeed be significantly smaller than the Frobenius norm in practical applications.

To illustrate this, let us consider the example of the identity matrix

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

It can be considered as an $n \times n$ block matrix, and Lemma 4.45 yields the estimate $\|I\|_2 \leq \sqrt{n}$, which is obviously far from the optimal bound $\|I\|_2 = 1$.

Therefore we now consider an improved estimate (a generalization of [49], Satz 6.2) which takes the block structure of a matrix into account:

Theorem 4.47 (Global spectral norm). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, and let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse. Let $p_{\mathcal{I}}$ and $p_{\mathcal{J}}$ be the depths of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Let $(\epsilon_{\mathcal{I},\ell})_{\ell=0}^{p_{\mathcal{I}}}$ and $(\epsilon_{\mathcal{J},\ell})_{\ell=0}^{p_{\mathcal{J}}}$ be families in $\mathbb{R}_{\geq 0}$ satisfying*

$$\|\chi_t X \chi_s\|_2 \leq \epsilon_{\mathcal{I},\mathrm{level}(t)}^{1/2} \epsilon_{\mathcal{J},\mathrm{level}(s)}^{1/2} \quad \text{for all } b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}. \tag{4.53}$$

Then we have

$$\|X\|_2 \leq C_{\mathrm{sp}} \Big(\sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell} \Big)^{1/2} \Big(\sum_{\ell=0}^{p_{\mathcal{J}}} \epsilon_{\mathcal{J},\ell} \Big)^{1/2}.$$

Proof. Let $v \in \mathbb{R}^{\mathcal{I}}$ and $u \in \mathbb{R}^{\mathcal{J}}$. Due to the triangle inequality, estimate (4.53), and the Cauchy–Schwarz inequality, we have

$$\begin{aligned}
|\langle v, Xu \rangle_2| &\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\langle \chi_t v, \chi_t X \chi_s \chi_s u \rangle_2| \\
&\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|\chi_t X \chi_s\|_2 \|\chi_t v\|_2 \|\chi_s u\|_2 \\
&\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{I},\mathrm{level}(t)}^{1/2} \|\chi_t v\|_2 \, \epsilon_{\mathcal{J},\mathrm{level}(s)}^{1/2} \|\chi_s u\|_2 \\
&\leq \Big(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{I},\mathrm{level}(t)} \|\chi_t v\|_2^2 \Big)^{1/2} \Big(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{J},\mathrm{level}(s)} \|\chi_s u\|_2^2 \Big)^{1/2}. \qquad (4.54)
\end{aligned}$$

Let us take a look at the first factor. Using sparsity and Corollary 3.10, we find

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{I},\mathrm{level}(t)} \|\chi_t v\|_2^2 &\leq C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} \epsilon_{\mathcal{I},\mathrm{level}(t)} \|\chi_t v\|_2^2 \\
&= C_{\mathrm{sp}} \sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell} \sum_{t\in\mathcal{T}_{\mathcal{I}}^{\ell}} \|\chi_t v\|_2^2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell} \|v\|_2^2 \\
&= C_{\mathrm{sp}} \Big(\sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell} \Big) \|v\|_2^2.
\end{aligned}$$

By the same reasoning, the second factor in (4.54) can be bounded by

$$\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{J},\mathrm{level}(s)} \|\chi_s u\|_2^2 \leq C_{\mathrm{sp}} \Big(\sum_{\ell=0}^{p_{\mathcal{J}}} \epsilon_{\mathcal{J},\ell} \Big) \|u\|_2^2,$$

combining both bounds yields

$$|\langle v, Xu \rangle_2| \leq C_{\mathrm{sp}} \Big(\sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell} \Big)^{1/2} \Big(\sum_{\ell=0}^{p_{\mathcal{J}}} \epsilon_{\mathcal{J},\ell} \Big)^{1/2} \|v\|_2 \|u\|_2,$$

and we can complete the proof by setting $v = Xu$. □

Let us now consider the application of these estimates to the kernel function. Combining Lemma 4.44 with (4.51) yields the following result:

Lemma 4.48 (Factorized error estimate). *Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfy the strong admissibility condition (4.49). Let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ of supports be C_{ov}-overlapping. Let the approximation $\tilde{g}_{t,s}$ satisfy the error estimate (4.51). Then we have*

$$\|\chi_t G \chi_s - V_t S_b W_s^*\|_2 \leq C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} C_{g,\eta} q^n \left(\frac{|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2} \left(\frac{|\Omega_s|}{\operatorname{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}. \tag{4.55}$$

Proof. Combining Lemma 4.44 with (4.51) yields

$$\|\chi_t G \chi_s - V_t S_b W_s^*\|_2 \leq \frac{C_{\mathcal{I}} C_{\mathcal{J}} C_{g,\eta} q^n}{\operatorname{diam}(\mathcal{Q}_t)^{\sigma/2} \operatorname{diam}(\mathcal{Q}_s)^{\sigma/2}} \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)^{1/2} \Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2}.$$

Due to Lemma 4.37, we have

$$\sum_{i\in\hat{t}} |\Omega_i| \leq C_{\mathrm{ov}} |\Omega_t|, \quad \sum_{j\in\hat{s}} |\Omega_j| \leq C_{\mathrm{ov}} |\Omega_s|,$$

so combining both estimates concludes the proof. □

This lemma provides us with error bounds matching the requirements of Theorem 4.47 perfectly.

Theorem 4.49 (Spectral approximation error). *Let $C_{\mathcal{I}}, C_{\mathcal{J}} \in \mathbb{R}_{>0}$ be given as in (4.48). Let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ of supports be C_{ov}-overlapping. Let the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse and let all of its admissible leaves satisfy the strong admissibility condition (4.49). Let $p_{\mathcal{I}}$ and $p_{\mathcal{J}}$ be the depths of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, respectively. Let*

$$\epsilon_{\mathcal{I},\ell} := \max\left\{\frac{|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} \;:\; t \in \mathcal{T}_{\mathcal{I}}^{(\ell)}\right\} \quad \text{for all } \ell \in \{0,\dots,p_{\mathcal{I}}\},$$
$$\epsilon_{\mathcal{J},\ell} := \max\left\{\frac{|\Omega_s|}{\operatorname{diam}(\mathcal{Q}_s)^\sigma} \;:\; s \in \mathcal{T}_{\mathcal{J}}^{(\ell)}\right\} \quad \text{for all } \ell \in \{0,\dots,p_{\mathcal{J}}\}.$$

Then we have

$$\|G - \widetilde{G}\|_2 \leq C_{\mathrm{sp}} C_{g,\eta} C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} q^n \Big(\sum_{\ell=0}^{p_{\mathcal{I}}} \epsilon_{\mathcal{I},\ell}\Big)^{1/2} \Big(\sum_{\ell=0}^{p_{\mathcal{J}}} \epsilon_{\mathcal{J},\ell}\Big)^{1/2}. \tag{4.56}$$

Proof. We combine Lemma 4.48 with Theorem 4.47. □

Remark 4.50 (Asymptotic behaviour of the error). As in Remark 4.42, let us assume that Ω is a d_Ω-dimensional subset or submanifold and that the discretization is based on a quasi-uniform grid hierarchy with a decreasing sequence $h_0, h_1, \dots$ of mesh

parameters. We again neglect the scaling of the basis functions $(\varphi_i)_{i\in\mathcal{I}}$ and $(\psi_j)_{j\in\mathcal{J}}$ captured by the constants $C_{\mathcal{I}}$ and $C_{\mathcal{J}}$.

For piecewise smooth geometries, we can expect the diameters $\operatorname{diam}(\Omega_t)$ of cluster supports and $\operatorname{diam}(Q_t)$ of the corresponding bounding boxes to be approximately proportional, and we can expect that $|\Omega_t|$ behaves like $\operatorname{diam}(\Omega_t)^{d_\Omega}$.

Under these assumptions, we find

$$\epsilon_{\mathcal{I},\ell} \sim \max\{\operatorname{diam}(\Omega_t)^{d_\Omega-\sigma} : t \in \mathcal{T}_{\mathcal{I}}^{(\ell)}\} \quad \text{for all } \ell \in \{0,\ldots,p_{\mathcal{I}}\},$$
$$\epsilon_{\mathcal{J},\ell} \sim \max\{\operatorname{diam}(\Omega_s)^{d_\Omega-\sigma} : s \in \mathcal{T}_{\mathcal{J}}^{(\ell)}\} \quad \text{for all } \ell \in \{0,\ldots,p_{\mathcal{J}}\}$$

and have to distinguish between three different cases:

- If $d_\Omega > \sigma$ holds, i.e., if the singularity of the kernel function is weak, the sums appearing in the estimate (4.56) will be dominated by the large clusters. Since the large clusters will remain essentially unchanged when refining the grid, the sums can be bounded by a constant, and we can conclude that the spectral error will behave like q^n on all levels of the grid.

- If $d_\Omega = \sigma$ holds, all terms in the sums appearing in (4.56) can be individually bounded by a constant. Therefore we can expect that the error is approximately proportional to the depth of the cluster tree, i.e., that it will behave like $|\log h_\ell| q^n$ and grow very slowly when the grid is refined.

- If $d_\Omega < \sigma$ holds, i.e., if the kernel function is strongly singular, the sums appearing in (4.56) will be dominated by the small clusters, and therefore the spectral error will behave like $h_\ell^{d_\Omega-\sigma} q^n$.

In all three cases, the spectral error estimate is better than the Frobenius error estimate given in Remark 4.42 as $h_\ell^{-\sigma} q^n$. □

4.7 Variable-order approximation

Until now, we have assumed that the order of the Taylor expansion or the interpolation scheme is constant for all clusters. We have seen in Lemma 4.48 that the size of the support of a cluster plays a major role in determining the spectral approximation error, and we can now take advantage of this observation in order to construct approximation schemes that lead to better error estimates than the ones provided by Theorem 4.49.

The fundamental idea is to use different approximation orders for different cluster sizes. It was introduced in [70] and analyzed for Taylor expansions and weakly singular kernel functions in [90], [91]. A refined analysis of this approach was presented in [101], [100]. By using interpolation instead of Taylor expansions, the convergence results can be significantly improved [23].

The restriction to weakly singular kernel functions can sometimes be overcome by using suitable globally-defined antiderivatives of the kernel function [25], but since this approach does not fit the concept of $\mathcal{H}^2$-matrices introduced here and since its error analysis requires fairly advanced tools which cannot be introduced in the context of this book, we do not discuss it further.

Motivation

We assume that the kernel function is only weakly singular, i.e., that the order of the singularity σ is smaller than the dimension d_Ω of the subset or submanifold Ω. For this case, Remark 4.50 states that the spectral error will be dominated by the error arising in the large clusters, while the blockwise error in leaf clusters will behave like $h_\ell^{d_\Omega-\sigma}$, i.e., it will decrease as the meshwidth h_ℓ decreases.

We would like to ensure the same favourable convergence behaviour for all clusters, and thus for the entire matrix $\widetilde{G}$. In order to do this, we reconsider Lemma 4.48: let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ be an admissible block. If we can ensure

$$\|g - \tilde{g}_{t,s}\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \lesssim \left(\frac{q^{n_t}}{\operatorname{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2}\left(\frac{q^{n_s}}{\operatorname{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}$$

instead of (4.51), where $(n_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(n_s)_{s\in\mathcal{T}_\mathcal{J}}$ are suitably-chosen families of parameters, the estimate (4.55) takes the form

$$\|\chi_t G \chi_s - V_t S_b W_s^*\|_2 \lesssim \left(\frac{q^{n_t}|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2}\left(\frac{q^{n_s}|\Omega_s|}{\operatorname{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}, \tag{4.57}$$

and proceeding as in Theorem 4.49 yields

$$\|G - \widetilde{G}\|_2 \lesssim \Big(\sum_{\ell=0}^{p_\mathcal{I}} \epsilon_{\mathcal{I},\ell}\Big)^{1/2}\Big(\sum_{\ell=0}^{p_\mathcal{J}} \epsilon_{\mathcal{J},\ell}\Big)^{1/2}$$

for the families $(\epsilon_{\mathcal{I},\ell})_{\ell=0}^{p_\mathcal{I}}$ and $(\epsilon_{\mathcal{J},\ell})_{\ell=0}^{p_\mathcal{J}}$ given by

$$\epsilon_{\mathcal{I},\ell} := \max\left\{\frac{q^{n_t}|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} \;:\; t \in \mathcal{T}_\mathcal{I}^{(\ell)}\right\} \quad \text{for all } \ell \in \{0,\dots,p_\mathcal{I}\},$$

$$\epsilon_{\mathcal{J},\ell} := \max\left\{\frac{q^{n_s}|\Omega_s|}{\operatorname{diam}(\mathcal{Q}_s)^\sigma} \;:\; s \in \mathcal{T}_\mathcal{J}^{(\ell)}\right\} \quad \text{for all } \ell \in \{0,\dots,p_\mathcal{J}\}.$$

This error estimate differs significantly from the one provided by Theorem 4.49: instead of working with the same interpolation order for all clusters, we can use a different order for each individual cluster. This variable-order approach [91], [90] allows us to compensate the growth of the cluster supports $|\Omega_t|$ by increasing the order of interpolation.

We assume that the measure of the support of a cluster can be bounded by the diameter of the corresponding bounding box, i.e., that there is a constant $C_{\mathrm{cu}} \in \mathbb{R}_{>0}$ satisfying

$$|\Omega_t| \leq C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}, \tag{4.58a}$$

$$|\Omega_s| \leq C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega} \quad \text{for all } s \in \mathcal{T}_{\mathcal{J}}. \tag{4.58b}$$

The size of the constant C_{cu} is determined by the geometry: if Ω is a subset of $\mathbb{R}^d$, we always have $C_{\mathrm{cu}} \leq 1$. If Ω is a submanifold, the size of C_{cu} depends on how "tightly folded" Ω is, e.g., on the curvature of the manifold (cf. Figure 4.3).

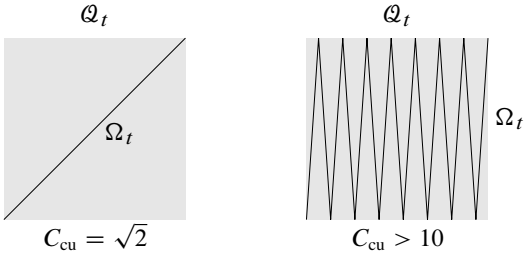


Figure 4.3. Influence of the geometry of Ω on the constant C_{cu}.

We also have to assume that the diameters of bounding boxes of leaf clusters are approximately proportional to the mesh parameter h and that they do not grow too rapidly as we proceed from the leaves of the cluster trees towards the root, i.e., that there are constants $C_{\mathrm{gr}} \in \mathbb{R}_{>0}$ and $\zeta \in \mathbb{R}_{\geq 1}$ satisfying

$$\operatorname{diam}(\mathcal{Q}_t) \leq C_{\mathrm{gr}} h \zeta^{p_{\mathcal{I}} - \operatorname{level}(t)} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}, \tag{4.59a}$$

$$\operatorname{diam}(\mathcal{Q}_s) \leq C_{\mathrm{gr}} h \zeta^{p_{\mathcal{J}} - \operatorname{level}(s)} \quad \text{for all } s \in \mathcal{T}_{\mathcal{J}}. \tag{4.59b}$$

Based on the assumptions (4.58) and (4.59), we find

$$\begin{aligned}
\epsilon_{\mathcal{I},\ell} &= \max\left\{ \frac{q^{n_t} |\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} \ : \ t \in \mathcal{T}_{\mathcal{I}}^{(\ell)} \right\} \\
&\leq C_{\mathrm{cu}} \max\{ q^{n_t} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega - \sigma} \ : \ t \in \mathcal{T}_{\mathcal{I}}^{(\ell)} \} \\
&\leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} h^{d_\Omega - \sigma} \max\{ q^{n_t} (\zeta^{d_\Omega - \sigma})^{p_{\mathcal{I}} - \ell} \ : \ t \in \mathcal{T}_{\mathcal{I}}^{(\ell)} \}.
\end{aligned}$$

For arbitrary parameters $\alpha, \beta \in \mathbb{N}_0$, we can choose the interpolation orders high enough to ensure $n_t \geq \alpha + \beta(p_{\mathcal{I}} - \operatorname{level}(t))$ for all $t \in \mathcal{T}_{\mathcal{I}}$. Then the error estimates takes the form

$$\epsilon_{\mathcal{I},\ell} \leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} h^{d_\Omega - \sigma} q^\alpha (q^\beta \zeta^{d_\Omega - \sigma})^{p_{\mathcal{I}} - \ell}.$$

For any given $\xi \in]0,1[$, we can let

$$\beta \geq \frac{\log \xi - (d_\Omega - \sigma)\log \zeta}{\log q}$$

and observe

$$q^\beta \zeta^{d_\Omega - \sigma} \leq \xi,$$

which implies the bound

$$\epsilon_{\mathcal{I},\ell} \leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega-\sigma} h^{d_\Omega-\sigma} q^\alpha \xi^{p_\mathcal{I}-\ell}.$$

This estimate allows us to bound the sum over all levels ℓ by a geometric sum, and bounding the sum by its limit yields

$$\begin{aligned}\sum_{\ell=0}^{p_\mathcal{I}} \epsilon_{\mathcal{I},\ell} &\leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega-\sigma} h^{d_\Omega-\sigma} q^\alpha \sum_{\ell=0}^{p_\mathcal{I}} \xi^{p_\mathcal{I}-\ell} = C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega-\sigma} h^{d_\Omega-\sigma} q^\alpha \sum_{\ell=0}^{p_\mathcal{I}} \xi^\ell \\ &\leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega-\sigma} h^{d_\Omega-\sigma} q^\alpha \sum_{\ell=0}^{\infty} \xi^\ell = C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega-\sigma} h^{d_\Omega-\sigma} q^\alpha \frac{1}{1-\xi}.\end{aligned}$$

Applying the same reasoning to the family $(\epsilon_{\mathcal{J},\ell})_{\ell=0}^{p_\mathcal{J}}$ yields

$$\|G - \widetilde{G}\|_2 \lesssim h^{d_\Omega-\sigma} \frac{q^\alpha}{1-\xi}. \tag{4.60}$$

Since we have assumed $\sigma < d_\Omega$, this estimate implies that the approximation error will decrease like $h^{d_\Omega-\sigma}$ if the grid is refined. The additional parameter α can be used to ensure that the approximation error is sufficiently small.

The major advantage of this *variable-order approximation scheme* is that the order of the interpolation is bounded in the leaf clusters and grows only slowly if we proceed towards the root of the cluster tree. This means that the leaf clusters, and these dominate in the complexity estimates, are handled very efficiently. A detailed complexity analysis shows that the resulting rank distribution is bounded, i.e., the computational and storage complexity is optimal.

Re-interpolation scheme

We have seen that we have to ensure

$$n_t \geq \alpha + \beta(p_\mathcal{I} - \mathrm{level}(t)) \quad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

if we want to reach the desirable error estimate (4.60). If we restrict our analysis to the case of isotropic interpolation, this can be achieved by using the order vectors

$$m_{t,\iota} := \alpha + \beta(p_\mathcal{I} - \mathrm{level}(t)) \quad \text{for all } t \in \mathcal{T}_\mathcal{I}, \iota \in \{1,\ldots,d\}. \tag{4.61}$$

Since larger clusters now use a higher approximation order than smaller clusters, we have to expect that

$$\mathfrak{I}_{t'}[\mathfrak{L}_{t,v}] \neq \mathfrak{L}_{t,v}$$

will hold for certain clusters $t \in \mathcal{T}_I$ with $t' \in \operatorname{sons}(t)$ and certain $v \in K_t$: if the order in the cluster t' is lower than the order used in t, not all Lagrange polynomials used for t can be represented in the basis used for t'. According to (4.20a), this can lead to

$$V_t \neq \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'},$$

i.e., the cluster basis $V = (V_t)_{t \in \mathcal{T}_I}$ will in general not be nested.

Losing the nested structure of the cluster basis is not acceptable, since it would lead to a less efficient representation of the matrix approximation.

As in [23], [22], we fix this by constructing a nested cluster basis based on the – no longer nested – cluster basis V: we define $\widetilde{V} = (\widetilde{V}_t)_{t \in \mathcal{T}_I}$ by

$$\widetilde{V}_t := \begin{cases} V_t & \text{if } \operatorname{sons}(t) = \emptyset, \\ \sum_{t' \in \operatorname{sons}(t)} \widetilde{V}_{t'} E_{t'} & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_I.$$

This cluster basis is obviously nested and uses the same rank distribution $K = (K_t)_{t \in \mathcal{T}_I}$ as V. The same reasoning applies to the cluster basis $W = (W_s)_{s \in \mathcal{T}_\mathcal{J}}$, and we introduce a similarly modified nested clusterbasis $\widetilde{W} = (\widetilde{W}_s)_{s \in \mathcal{T}_\mathcal{J}}$ by

$$\widetilde{W}_s := \begin{cases} W_s & \text{if } \operatorname{sons}(s) = \emptyset, \\ \sum_{s' \in \operatorname{sons}(s)} \widetilde{W}_{s'} F_{s'} & \text{otherwise} \end{cases} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J}.$$

The rank distribution for $\widetilde{W}$ is the same as for W. Instead of approximating an admissible block $b = (t, s) \in \mathcal{L}^+_{I \times \mathcal{J}}$ by $V_t S_b W_s^*$, we now approximate it by $\widetilde{V}_t S_b \widetilde{W}_s^*$.

Remark 4.51 (Implementation). The algorithms for constructing $\widetilde{V}$ and $\widetilde{W}$ require no modifications compared to the standard case, we only have to ensure that the correct interpolation orders and Lagrange polynomials are used. □

Lemma 4.52 (Complexity). *According to Lemma 4.10, we have*

$$\#K_t = (\alpha + \beta(p_I - \operatorname{level}(t)))^d \quad \text{for all } t \in \mathcal{T}_I.$$

Let $\mathcal{T}_I$ be quasi-balanced, i.e., let there be constants $C_{\mathrm{ba}} \in \mathbb{N}$ and $\xi \in \mathbb{R}_{>1}$ satisfying

$$\#\{t \in \mathcal{T}_I : \operatorname{level}(t) = \ell\} \leq C_{\mathrm{ba}} \xi^{\ell - p_I} c_I \quad \text{for all } \ell \in \mathbb{N}_0. \tag{4.62}$$

Then the rank distribution $K = (K_t)_{t \in \mathcal{T}_I}$ is $(C_{\mathrm{bn}}, \alpha, \beta, d, \xi)$-bounded with

$$C_{\mathrm{bn}} := C_{\mathrm{ba}} \frac{\xi}{\xi - 1}.$$

Proof. Let $\ell \in \mathbb{N}$ and

$$\mathcal{R}_\ell := \{t \in \mathcal{T}_\mathcal{I} : \#K_t > (\alpha + \beta(\ell-1))^d\}.$$

For a cluster $t \in \mathcal{R}_\ell$, we have

$$\begin{aligned}(\alpha + \beta(\ell-1))^d &< \#K_t = (\alpha + \beta(p_\mathcal{I} - \operatorname{level}(t)))^d,\\ \alpha + \beta(\ell-1) &< \alpha + \beta(p_\mathcal{I} - \operatorname{level}(t)),\\ \ell - 1 &< p_\mathcal{I} - \operatorname{level}(t),\\ \operatorname{level}(t) &< p_\mathcal{I} - \ell + 1,\\ \operatorname{level}(t) &\leq p_\mathcal{I} - \ell.\end{aligned}$$

Since $\mathcal{T}_\mathcal{I}$ is quasi-balanced, the estimate (4.62) yields

$$\begin{aligned}\mathcal{R}_\ell &\subseteq \bigcup_{n=0}^{p_\mathcal{I}-\ell} \{t \in \mathcal{T}_\mathcal{I} : \operatorname{level}(t) = n\},\\ \#\mathcal{R}_\ell &\leq \sum_{n=0}^{p_\mathcal{I}-\ell} \#\{t \in \mathcal{T}_\mathcal{I} : \operatorname{level}(t) = n\} \leq \sum_{n=0}^{p_\mathcal{I}-\ell} C_{\mathrm{ba}} \xi^{n-p_\mathcal{I}} c_\mathcal{I}\\ &= C_{\mathrm{ba}} c_\mathcal{I} \sum_{n=\ell}^{p_\mathcal{I}} \xi^{-n} = C_{\mathrm{ba}} c_\mathcal{I} \xi^{-\ell} \sum_{n=0}^{p_\mathcal{I}-\ell} \xi^{-n} < C_{\mathrm{ba}} c_\mathcal{I} \xi^{-\ell} \frac{1}{1-1/\xi} = C_{\mathrm{bn}} \xi^{-\ell} c_\mathcal{I}\end{aligned}$$

with $C_{\mathrm{bn}} = C_{\mathrm{ba}} \xi/(\xi-1)$. □

In order to analyze the approximation error corresponding to the new cluster bases $\widetilde{V}$ and $\widetilde{W}$, we have to relate them to suitable approximation schemes for the kernel function.

Let $t \in \mathcal{T}_\mathcal{I}$ and $i \in \hat{t}$. If $\operatorname{sons}(t) = \emptyset$ holds, we have

$$(V_t)_{i\nu} = \int_\Omega \mathcal{L}_{t,\nu}(x)\varphi_i(x)\,dx \quad \text{for all } \nu \in K_t$$

by definition. Otherwise, we let $t' \in \operatorname{sons}(t)$ with $i \in \hat{t}'$. If $\operatorname{sons}(t') = \emptyset$ holds, we have

$$(V_{t'})_{i\mu} = \int_\Omega \mathcal{L}_{t',\mu}(x)\varphi_i(x)\,dx \quad \text{for all } \mu \in K_{t'},$$

and since the transfer matrix $E_{t'}$ is given by

$$(E_{t'})_{\mu\nu} = \mathcal{L}_{t,\nu}(x_{t',\mu}) \quad \text{for all } \nu \in K_t, \mu \in K_{t'},$$

the definition of $\widetilde{V}_t$ implies

$$(\widetilde{V}_t)_{i\nu} = \widetilde{V}_{t'} E_{t'} = V_{t'} E_{t'} = \sum_{\mu \in K_{t'}} \mathcal{L}_{t,\nu}(x_{t',\mu}) \int_\Omega \mathcal{L}_{t',\mu}(x)\varphi_i(x)\,dx$$

$$= \int_\Omega \mathfrak{I}_{t'}[\mathfrak{L}_{t,\nu}](x)\varphi_i(x)\,dx \quad \text{for all } \nu \in K_t,$$

i.e., the Lagrange polynomial $\mathfrak{L}_{t,\nu}$ corresponding to t is replaced by its lower-order interpolant $\mathfrak{I}_{t'}[\mathfrak{L}_{t,\nu}]$.

If t' is not a leaf, we proceed by recursively applying interpolation operators until we reach a leaf cluster containing i. The resulting nested interpolation operator is defined as follows:

Definition 4.53 (Re-interpolation). For all $t \in \mathcal{T}_\mathcal{I}$ and all $r \in \operatorname{sons}^*(t)$, we define the *re-interpolation operator* $\mathfrak{I}_{r,t}$ by

$$\mathfrak{I}_{r,t} := \begin{cases} \mathfrak{I}_{r,t'}\mathfrak{I}_t & \text{if there is a } t' \in \operatorname{sons}(t) \text{ with } r \in \operatorname{sons}^*(t'), \\ \mathfrak{I}_t & \text{otherwise, i.e., if } t = r. \end{cases}$$

Due to $\widetilde{V}_t = V_t$ for $t \in \mathcal{L}_\mathcal{I}$, we can derive interpolation operators corresponding to the cluster basis $\widetilde{V}$ inductively starting from the leaves of $\mathcal{T}_\mathcal{I}$. We collect all leaf clusters influencing a cluster in the families $(\mathcal{L}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(\mathcal{L}_s)_{s\in\mathcal{T}_\mathcal{J}}$ of sets given by

$$\mathcal{L}_t := \{r \in \mathcal{L}_\mathcal{I} : r \in \operatorname{sons}^*(t)\} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},$$
$$\mathcal{L}_s := \{r \in \mathcal{L}_\mathcal{J} : r \in \operatorname{sons}^*(s)\} \quad \text{for all } s \in \mathcal{T}_\mathcal{J}.$$

and can express the connection between re-interpolation operators and the new cluster basis $\widetilde{V}$ by a simple equation:

Lemma 4.54 (Re-interpolation). *Let $t \in \mathcal{T}_\mathcal{I}$, and let $r \in \mathcal{L}_t$. Then we have*

$$(\widetilde{V}_t)_{i\nu} = \int_\Omega \mathfrak{I}_{r,t}[\mathfrak{L}_{t,\nu}](x)\varphi_i(x)\,dx \quad \text{for all } i \in \hat{r},\ \nu \in K_t. \tag{4.63}$$

Proof. By induction on $\operatorname{level}(r) - \operatorname{level}(t)$. Let $t \in \mathcal{T}_\mathcal{I}$ and $r \in \mathcal{L}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = 0$. We have $t = r \in \mathcal{L}_\mathcal{I}$, i.e., $\widetilde{V}_t = V_t$ by definition. According to

$$\mathfrak{I}_t[\mathfrak{L}_{t,\nu}] = \sum_{\mu\in K_t} \mathfrak{L}_{t,\nu}(x_{t,\mu})\mathfrak{L}_{t,\mu} = \sum_{\mu\in K_t} \delta_{\nu\mu}\mathfrak{L}_{t,\mu} = \mathfrak{L}_{t,\nu},$$

we find

$$\begin{aligned}(\widetilde{V}_t)_{i\nu} = (V_t)_{i\nu} &= \int_\Omega \mathfrak{L}_{t,\nu}(x)\varphi_i(x)\,dx \\ &= \int_\Omega \mathfrak{I}_t[\mathfrak{L}_{t,\nu}](x)\varphi_i(x)\,dx \quad \text{for all } i \in \hat{t},\ \nu \in K_t.\end{aligned}$$

Let now $n \in \mathbb{N}_0$ be such that (4.63) holds for all $t \in \mathcal{T}_\mathcal{I}$ and $r \in \mathcal{L}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = n$. Let $t \in \mathcal{T}_\mathcal{I}$ and $r \in \mathcal{L}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = n + 1$. Due to the

definition of sons$^*(t)$, we can find $t' \in \operatorname{sons}(t)$ with $r \in \operatorname{sons}^*(t')$. The definition of $\widetilde{V}_t$ yields

$$\begin{aligned}(\widetilde{V}_t)_{iv} = (\widetilde{V}_{t'}E_{t'})_{iv} &= \sum_{\mu\in K_{t'}} (\widetilde{V}_{t'})_{i\mu}(E_{t'})_{\mu v} \\ &= \sum_{\mu\in K_{t'}} (\widetilde{V}_{t'})_{i\mu}\mathcal{L}_{t,v}(x_{t',\mu}) \quad \text{for all } i \in \hat{r},\ v \in K_t.\end{aligned}$$

Since $\operatorname{level}(r) - \operatorname{level}(t') = \operatorname{level}(r) - \operatorname{level}(t) - 1 = n$ holds, we can apply the induction assumption and get

$$\begin{aligned}(\widetilde{V}_t)_{iv} &= \sum_{\mu\in K_{t'}} \mathcal{L}_{t,v}(x_{t',\mu}) \int_\Omega \mathfrak{I}_{r,t'}[\mathcal{L}_{t',\mu}](x)\varphi_i(x)\,dx \\ &= \int_\Omega \mathfrak{I}_{r,t'}\Big[\sum_{\mu\in K_{t'}} \mathcal{L}_{t,v}(x_{t',\mu})\mathcal{L}_{t',\mu}\Big](x)\varphi_i(x)\,dx \\ &= \int_\Omega \mathfrak{I}_{r,t'}[\mathfrak{I}_{t'}[\mathcal{L}_{t,v}]](x)\varphi_i(x)\,dx \\ &= \int_\Omega \mathfrak{I}_{r,t}[\mathcal{L}_{t,v}](x)\varphi_i(x)\,dx \quad \text{for all } i \in \hat{r},\ v \in K_t,\end{aligned}$$

which concludes the induction. □

Using this lemma, we can find a representation for subblocks of the matrix $\widetilde{V}_t S_b \widetilde{W}_s^*$ used to approximate a matrix block:

Lemma 4.55 (Re-interpolated kernel). *Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Let $t^* \in \mathcal{L}_t$ and $s^* \in \mathcal{L}_s$. We have*

$$(\widetilde{V}_t S_b \widetilde{W}_s^*)_{ij} = \int_\Omega \varphi_i(x) \int_\Omega (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g](x,y)\varphi_j(y)\,dy\,dx$$

for all $i \in \hat{t}^*$, $j \in \hat{s}^*$.

Proof. Let $i \in \hat{t}^*$ and $j \in \hat{s}^*$. Lemma 4.54 and the definition of S_b (cf. 4.20c) yield

$$\begin{aligned}(\widetilde{V}_t S_b \widetilde{W}_s^*)_{ij} &= \sum_{v\in K_t}\sum_{\mu\in L_s} (\widetilde{V}_t)_{iv}(S_b)_{v\mu}(\widetilde{W}_s)_{j\mu} \\ &= \sum_{v\in K_t}\sum_{\mu\in L_s} \int_\Omega \mathfrak{I}_{t^*,t}[\mathcal{L}_{t,v}](x)\varphi_i(x) \int_\Omega \mathfrak{I}_{s^*,s}[\mathcal{L}_{s,\mu}](y)\psi_j(y)g(x_{t,v},x_{s,\mu})\,dy\,dx \\ &= \int_\Omega \varphi_i(x) \int_\Omega (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s}) \\ &\qquad \Big[\sum_{v\in K_t}\sum_{\mu\in L_s} g(x_{t,v},x_{s,\mu})\mathcal{L}_{t,v} \otimes \mathcal{L}_{s,\mu}\Big](x,y)\psi_j(y)\,dy\,dx\end{aligned}$$

$$= \int_\Omega \varphi_i(x) \int_\Omega (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[\mathfrak{I}_t \otimes \mathfrak{I}_s[g]](x,y)\psi_j(y)\,dy\,dx$$
$$= \int_\Omega \varphi_i(x) \int_\Omega (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\psi_j(y)\,dy\,dx.$$

In the last step, we have used the fact that $\mathfrak{I}_t$ and $\mathfrak{I}_s$ are projections. □

Using this representation of the matrix approximation, we can generalize the error estimate of Lemma 4.38 to the case of variable-order approximation:

Lemma 4.56 (Blockwise error). *Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, and let $\epsilon_b \in \mathbb{R}_{>0}$ satisfy*

$$\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathfrak{Q}_{t^*}\times\mathfrak{Q}_{s^*}} \leq \epsilon_b \quad \text{for all } t^* \in \mathcal{L}_t, s^* \in \mathcal{L}_s.$$

Let $C_\mathcal{I}$ and $C_\mathcal{J}$ satisfy (4.48). Then we have

$$\|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_2 \leq \|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_F \leq C_\mathcal{I} C_\mathcal{J} \epsilon_b \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)^{1/2} \Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2}.$$

Proof. Let $X := \chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*$. For all $u \in \mathbb{R}^\mathcal{J}$, we have

$$\|Xu\|_2^2 = \sum_{i\in\mathcal{I}} \Big(\sum_{j\in\mathcal{J}} X_{ij} u_j\Big)^2 \leq \sum_{i\in\mathcal{I}} \Big(\sum_{j\in\mathcal{J}} X_{ij}^2\Big)\Big(\sum_{j\in\mathcal{J}} u_j^2\Big) = \|X\|_F^2 \|u\|_2^2$$

and can conclude $\|X\|_2 \leq \|X\|_F$.

Corollary 3.9 and Lemma 3.8 yield that $\{\hat{t}^* : t^* \in \mathcal{L}_t\}$ and $\{\hat{s}^* : s^* \in \mathcal{L}_s\}$ are disjoint partitions of $\hat{t}$ and $\hat{s}$, respectively, and we find

$$\|X\|_F^2 = \sum_{t^*\in\mathcal{L}_t} \sum_{s^*\in\mathcal{L}_s} \|\chi_{t^*} X \chi_{s^*}\|_F^2 = \sum_{t^*\in\mathcal{L}_t} \sum_{s^*\in\mathcal{L}_s} \sum_{i\in\hat{t}^*} \sum_{j\in\hat{s}^*} X_{ij}^2,$$

and we can apply Lemma 4.55 to each $t^* \in \operatorname{sons}^*(t) \cap \mathcal{L}_\mathcal{I}$ and $s^* \in \operatorname{sons}^*(s) \cap \mathcal{L}_\mathcal{J}$ in order to find

$$\begin{aligned}
|X_{ij}| &= |G_{ij} - (\widetilde{V}_t S_b \widetilde{W}_s^*)_{ij}| \\
&= \left| \int_\Omega \varphi_i(x) \int_\Omega (g(x,y) - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g](x,y))\psi_j(y)\,dy\,dx \right| \\
&\leq \epsilon_b \int_\Omega |\varphi_i(x)|\,dx \int_\Omega |\psi_j(y)|\,dy \leq \epsilon_b |\Omega_i|^{1/2} \|\varphi_i\|_{L^2} |\Omega_j|^{1/2} \|\psi_j\|_{L^2} \\
&\leq \epsilon_b C_\mathcal{I} C_\mathcal{J} |\Omega_i|^{1/2} |\Omega_j|^{1/2}.
\end{aligned}$$

Due to this estimate, we can conclude

$$\|X\|_F^2 \leq \epsilon_b^2 C_\mathcal{I}^2 C_\mathcal{J}^2 \sum_{t^*\in\mathcal{L}_t} \sum_{s^*\in\mathcal{L}_s} \sum_{i\in\hat{t}^*} \sum_{j\in\hat{s}^*} |\Omega_i|\,|\Omega_j| = \epsilon_b^2 C_\mathcal{I}^2 C_\mathcal{J}^2 \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} |\Omega_i|\,|\Omega_j|,$$

which is the desired result. □

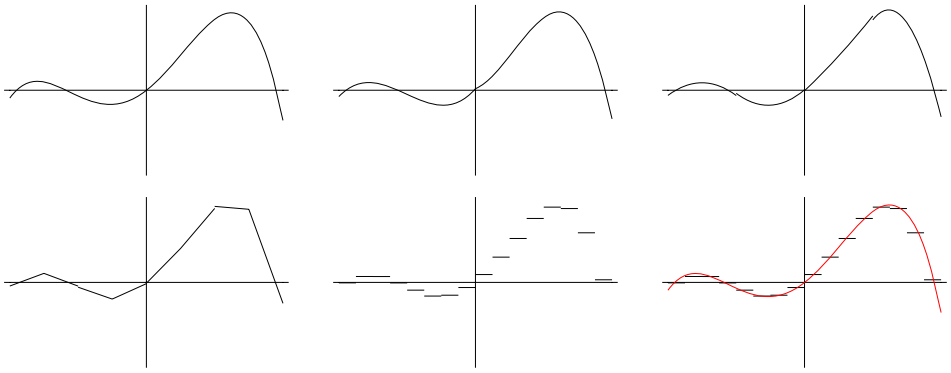


Figure 4.4. Re-interpolation of a Lagrange polynomial.

Assuming that the overlap of the supports is bounded, we can simplify this result:

Corollary 4.57 (Blockwise spectral error). *Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, and let $\epsilon_b \in \mathbb{R}_{>0}$ satisfy*

$$\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \leq \epsilon_b \quad \text{for all } t^* \in \mathcal{L}_t, s^* \in \mathcal{L}_s.$$

Let $C_{\mathcal{I}}$ and $C_{\mathcal{J}}$ satisfy (4.48), and let the families $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ be C_{ov}-overlapping. Then we have

$$\|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_2 \leq C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} \epsilon_b |\Omega_t|^{1/2} |\Omega_s|^{1/2}.$$

Proof. Combine Lemma 4.56 with Lemma 4.37. □

Error analysis in the one-dimensional case

Before we can analyze the properties of the re-interpolation operators $\mathfrak{I}_{t^*,t}$ in d-dimensional space, we have to investigate the one-dimensional case.

Let $\alpha \in \mathbb{N}$ and $\beta \in \mathbb{N}_0$. Let $p \in \mathbb{N}$, and let $(J_\ell)_{\ell=0}^p$ be a family of non-trivial intervals $J_\ell = [a_\ell, b_\ell]$ satisfying

$$J_{\ell+1} \subseteq J_\ell \quad \text{for all } \ell \in \{0, \ldots, p-1\}.$$

For each $\ell \in \{0, \ldots, p\}$, we introduce the interpolation operator

$$\mathfrak{I}_\ell^{J,\alpha,\beta} := \mathfrak{I}^{J_\ell}_{\alpha+\beta(p-\ell)}.$$

The one-dimensional re-interpolation operator $\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell}$ is given recursively by

$$\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell} := \begin{cases} \mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell+1} \mathfrak{I}^{J,\alpha,\beta}_\ell & \text{if } \ell < \ell^*, \\ \mathfrak{I}^{J,\alpha,\beta}_\ell & \text{otherwise} \end{cases} \quad \text{for all } \ell, \ell^* \in \{0, \ldots, p\} \text{ with } \ell^* \geq \ell. \tag{4.64}$$

We can see that this definition implies

$$\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell} = \mathfrak{I}^{J,\alpha,\beta}_{\ell^*} \mathfrak{I}^{J,\alpha,\beta}_{\ell^*-1} \dots \mathfrak{I}^{J,\alpha,\beta}_{\ell+1} \mathfrak{I}^{J,\alpha,\beta}_{\ell} \quad \text{for all } \ell, \ell^* \in \{0, \dots, p\} \text{ with } \ell^* \geq \ell. \tag{4.65}$$

In order to bound the interpolation error by a best-approximation estimate, we have to establish the stability of $\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell}$. The simple approach

$$\begin{aligned}
\|\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell}[f]\|_{\infty,[a_{\ell^*},b_{\ell^*}]} &\leq \Lambda_{\alpha+\beta(p-\ell^*)} \|\mathfrak{I}^{J,\alpha,\beta}_{\ell^*-1,\ell}[f]\|_{\infty,[a_{\ell^*},b_{\ell^*}]} \\
&\leq \Lambda_{\alpha+\beta(p-\ell^*)} \|\mathfrak{I}^{J,\alpha,\beta}_{\ell^*-1,\ell}[f]\|_{\infty,[a_{\ell^*-1},b_{\ell^*-1}]} \\
&\leq \dots \\
&\leq \Lambda_{\alpha+\beta(p-\ell^*)} \Lambda_{\alpha+\beta(p-\ell^*+1)} \dots \Lambda_{\alpha+\beta(p-\ell)} \|f\|_{\infty,[a_\ell,b_\ell]}
\end{aligned}$$

suggested by (4.65) will not yield a sufficiently good bound, since the resulting "stability constant" will grow too rapidly if $\ell^* - \ell$ grows.

An improved estimate can be derived if we require that the intervals shrink sufficiently fast:

Definition 4.58 (ξ-regular intervals). Let $\xi \in]0, 1[$. If

$$\frac{|J_{\ell+1}|}{|J_\ell|} \leq \xi \quad \text{holds for all } \ell \in \{0, \dots, p-1\},$$

we call the family $J = (J_\ell))_{\ell=0}^p$ *ξ-regular*.

In the ξ-regular case, we can exploit the fact that all but the rightmost interpolation operators in (4.65) are applied to polynomials and not to general functions: the growth of polynomials can be controlled, and applying Lemma 4.14 yields the following improved stability estimate:

Theorem 4.59 (Stability of re-interpolation). *Let the family $J = (J_\ell)_{\ell=0}^p$ be ξ-regular. Let the interpolation scheme $(\mathfrak{I}_m)_{m=1}^\infty$ be (Λ, λ)-stable. Then there is a constant $C_{\mathrm{re}} \in \mathbb{R}_{\geq 1}$ depending only on ξ, Λ, λ and β which satisfies*

$$\|\mathfrak{I}^{J,\alpha,\beta}_{\ell^*,\ell}[f]\|_{\infty,J_{\ell^*}} \leq C_{\mathrm{re}} \Lambda_{\alpha+\beta(p-\ell)} \|f\|_{\infty,J_\ell}$$

for all $\ell, \ell^* \in \{0, \dots, p\}$ with $\ell \leq \ell^*$ and all $f \in C(J_\ell)$.

Proof. Cf. Theorem 3.11 in [23]. □

Using this inequality, we can derive an error estimate for the re-interpolation operator:

Lemma 4.60 (Re-interpolation error). *Let the family $J = (J_\ell)_{\ell=0}^p$ be ξ-regular. Let the interpolation scheme be (Λ, λ)-stable. Let $C_{\mathrm{re}} \in \mathbb{R}_{\geq 1}$ be the constant introduced in Theorem 4.59. Let $\ell, \ell^* \in \{0, \ldots, p\}$ with $\ell \leq \ell^*$. We have*

$$\|f - \mathfrak{I}_{\ell^*,\ell}^{J,\alpha,\beta}[f]\|_{\infty,J_{\ell^*}} \leq C_{\mathrm{re}}\Lambda \sum_{n=\ell}^{\ell^*} (\alpha + \beta(p-n) + 1)^\lambda \|f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\|_{\infty,J_n} \quad (4.66)$$

for all $f \in C(J_\ell)$.

Proof. Let $f \in C(J_\ell)$. We define

$$P_n := \begin{cases} I & \text{if } n = \ell^* + 1, \\ \mathfrak{I}_{\ell^*,n}^{\alpha,\beta} & \text{otherwise} \end{cases} \qquad \text{for all } n \in \{\ell, \ldots, \ell^* + 1\}.$$

According to (4.64), we have $P_n = P_{n+1}\mathfrak{I}_n^{J,\alpha,\beta}$, and Theorem 4.59 yields

$$\|P_n[g]\|_{\infty,J_{\ell^*}} \leq C_{\mathrm{re}}\Lambda_{\alpha+\beta(p-n)}\|g\|_{\infty,J_n} \qquad \text{for all } g \in C(J_n),\ n \in \{\ell, \ldots, \ell^*\}.$$

We can use the (Λ, λ)-stability of the interpolation scheme in order to prove

$$\begin{aligned} \|P_{n+1}[g]\|_{\infty,J_{\ell^*}} &\leq \|g\|_{\infty,J_n} \begin{cases} C_{\mathrm{re}}\Lambda(\alpha + \beta(p-(n+1)) + 1)^\lambda & \text{if } n < \ell^*, \\ 1 & \text{otherwise} \end{cases} \\ &\leq C_{\mathrm{re}}\Lambda(\alpha + \beta(p-n) + 1)^\lambda \|g\|_{\infty,J_n} \end{aligned}$$

for all $g \in C(J_n)$ and all $n \in \{\ell, \ldots, \ell^*\}$. Using a telescoping sum, we get

$$\begin{aligned} \|f - \mathfrak{I}_{\ell^*,\ell}^{J,\alpha,\beta}[f]\|_{\infty,J_{\ell^*}} &= \Big\| \sum_{n=\ell}^{\ell^*} P_{n+1}[f] - P_n[f] \Big\|_{\infty,J_{\ell^*}} \\ &= \Big\| \sum_{n=\ell}^{\ell^*} P_{n+1}\big[f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\big] \Big\|_{\infty,J_{\ell^*}} \\ &\leq \sum_{n=\ell}^{\ell^*} \Big\| P_{n+1}\big[f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\big] \Big\|_{\infty,J_{\ell^*}} \\ &\leq C_{\mathrm{re}}\Lambda \sum_{n=\ell}^{\ell^*} (\alpha + \beta(p-n) + 1)^\lambda \|f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\|_{\infty,J_n}, \end{aligned}$$

and this is the required estimate. □

We can use Theorem 4.16 in order to bound the terms in the sum (4.66): if we again assume

$$|f^{(\nu)}(x)| \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \qquad \text{for all } x \in [a_\ell, b_\ell],\ \nu \in \mathbb{N}_0,$$

we observe

$$\|f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\|_{\infty,J_n} \leq 2eC_f(\Lambda_{\alpha+\beta(p-n)}+1)(\alpha+\beta(p-n)+1)^{\sigma} \left(1+\frac{|J_n|}{\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{|J_n|}\right)^{-\alpha-\beta(p-n)}$$

for all $n \in \{\ell,\dots,\ell^*\}$. Due to ξ-regularity, we have

$$|J_n| \leq \xi^{n-\ell}|J_\ell| \quad \text{for all } n \in \{\ell,\dots,\ell^*\},$$

and Lemma 4.78 yields that there is a $\hat{\xi} \in]0,1[$ satisfying

$$\varrho\left(\frac{2\gamma_f}{|J_n|}\right) \geq \varrho\left(\frac{2\gamma_f}{\xi^{\iota-\ell}|J_\ell|}\right) \geq \left(\frac{1}{\hat{\xi}}\right)^{\iota-\ell}\varrho\left(\frac{2\gamma_f}{|J_\ell|}\right).$$

We can use the additional factor $\hat{\xi}^{\iota-\ell}$ in order to bound the sum (4.66):

Theorem 4.61 (Re-interpolation error). *Let the family $(J_\ell)_{\ell=0}^p$ be ξ-regular. Let the interpolation scheme be (Λ,λ)-stable. Let a rank parameter $\beta \in \mathbb{N}_0$ and a lower bound $r_0 \in \mathbb{R}_{>0}$ for the convergence radius be given.*

There are constants $C_{\mathrm{ri}} \in \mathbb{R}_{\geq 1}$, $\alpha_0 \in \mathbb{N}$ such that

$$\|f - \mathfrak{I}_{\ell^*,\ell}^{J,\alpha,\beta}[f]\|_{\infty,J_{\ell^*}} \leq C_f C_{\mathrm{ri}}(\alpha+\beta(p-\ell)+1)^{2\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\varrho(r_0)^{-\alpha-\beta(p-\ell)}$$

holds for all $\ell,\ell^* \in \{0,\dots,p\}$ with $\ell^* \geq \ell$, all $\alpha \geq \alpha_0$ and all $f \in C^\infty(J_\ell)$ satisfying

$$|f^{(\nu)}(x)| \leq \frac{C_f}{\gamma_f^\nu}\begin{bmatrix}\nu\\ \sigma-1\end{bmatrix} \quad \text{for all } x \in J_\ell,\ \nu \in \mathbb{N}_0 \tag{4.67}$$

for constants $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ with $2\gamma_f/|J_\ell| \geq r_0$.

Proof. Let $C_{\mathrm{re}} \in \mathbb{R}_{\geq 1}$ be the constant introduced in Theorem 4.59. According to Lemma 4.78, there is a constant $\hat{\xi} \in]0,1[$ satisfying

$$\varrho\left(\frac{r}{\xi}\right) \geq \frac{1}{\hat{\xi}}\varrho(r) \quad \text{for all } r \in \mathbb{R}_{\geq r_0}.$$

Let $w \in]0,1[$. Due to $\hat{\xi} < 1$, we can find $\alpha_0 \in \mathbb{N}$ such that

$$\hat{\xi}^{\alpha_0} \leq w\varrho(r_0)^{-\beta}$$

holds. We define $C_{\mathrm{ri}} := 2e\Lambda^2 C_{\mathrm{re}}/(1-w)$.

Let now $\ell,\ell^* \in \{0,\dots,p\}$ with $\ell^* \geq \ell$, $\alpha \in \mathbb{N}_{\geq\alpha_0}$, and let $f \in C^\infty(J_\ell)$ be a function satisfying (4.67) for constants $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ with $2\gamma_f/|J_\ell| \geq r_0$. According to Theorem 4.16, we have

$$
\begin{aligned}
&\|f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\|_{\infty,J_n} \\
&\leq 2eC_f(\Lambda_{\alpha+\beta(p-n)}+1)(\alpha+\beta(p-n)+1)^{\sigma}\left(1+\frac{|J_n|}{2\gamma_f}\right)\varrho\left(\frac{2\gamma_f}{|J_n|}\right)^{-\alpha-\beta(p-n)} \\
&\leq 2e\Lambda C_f(\alpha+\beta(p-n)+1)^{\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\varrho\left(\frac{2\gamma_f}{\xi^{n-\ell}|J_\ell|}\right)^{-\alpha-\beta(p-n)} \\
&\leq 2e\Lambda C_f(\alpha+\beta(p-n)+1)^{\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\hat{\xi}^{(n-\ell)(\alpha+\beta(p-n))}\varrho\left(\frac{2\gamma_f}{|J_\ell|}\right)^{-\alpha-\beta(p-n)} \\
&\leq 2e\Lambda C_f(\alpha+\beta(p-\ell)+1)^{\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\hat{\xi}^{\alpha_0(n-\ell)}\varrho(r_0)^{-\alpha-\beta(p-n)} \\
&\leq 2e\Lambda C_f(\alpha+\beta(p-\ell)+1)^{\lambda+\sigma}\left(1+\frac{1}{r_0}\right)w^{n-\ell}\varrho(r_0)^{-\beta(n-\ell)}\varrho(r_0)^{-\alpha-\beta(p-n)} \\
&= 2e\Lambda C_f(\alpha+\beta(p-\ell)+1)^{\lambda+\sigma}\left(1+\frac{1}{r_0}\right)w^{n-\ell}\varrho(r_0)^{-\alpha-\beta(p-\ell)}
\end{aligned}
$$

for all $n \in \{\ell,\dots,\ell^*\}$. We can combine this estimate with Lemma 4.60 in order to find

$$
\begin{aligned}
&\|f - \mathfrak{I}_{\ell^*,\ell}^{J,\alpha,\beta}[f]\|_{\infty,J_{\ell^*}} \\
&\quad\leq C_{\mathrm{re}}\Lambda\sum_{n=\ell}^{\ell^*}(\alpha+\beta(p-n))^{\lambda}\|f - \mathfrak{I}_n^{J,\alpha,\beta}[f]\|_{\infty,J_n} \\
&\quad\leq C_{\mathrm{re}}2e\Lambda^2 C_f(\alpha+\beta(p-\ell)+1)^{2\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\varrho(r_0)^{-\alpha-\beta(p-\ell)}\sum_{n=\ell}^{\ell^*}w^{n-\ell} \\
&\quad\leq C_f\frac{C_{\mathrm{re}}2e\Lambda^2}{1-w}(\alpha+\beta(p-\ell)+1)^{2\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\varrho(r_0)^{-\alpha-\beta(p-\ell)} \\
&\quad= C_f C_{\mathrm{ri}}(\alpha+\beta(p-\ell)+1)^{2\lambda+\sigma}\left(1+\frac{1}{r_0}\right)\varrho(r_0)^{-\alpha-\beta(p-\ell)},
\end{aligned}
$$

which is the desired result. □

Error analysis in the multi-dimensional case

Now we can turn our attention to the multi-dimensional case. Let $p \in \mathbb{N}_0$. Let $(\mathcal{Q}_\ell)_{\ell=0}^p$ be a family of non-trivial axis-parallel boxes

$$\mathcal{Q}_\ell = [a_{\ell,1}, b_{\ell,1}] \times \cdots \times [a_{\ell,d}, b_{\ell,d}]$$

satisfying

$$\mathcal{Q}_{\ell+1} \subseteq \mathcal{Q}_\ell \quad \text{for all } \ell \in \{0,\dots,p-1\}.$$

We let $m_\ell := \alpha + \beta(p-\ell)$ for all $\ell \in \{0,\dots,p\}$ and introduce a tensor-product interpolation operator

$$\mathfrak{I}_\ell^{\mathcal{Q},\alpha,\beta} := \mathfrak{I}_{m_\ell}^{\mathcal{Q}_\ell} \quad \text{for all } \ell \in \{0,\dots,p\}. \tag{4.68}$$

The multi-dimensional re-interpolation operators are given by

$$\mathfrak{I}_{\ell^*,\ell}^{\mathcal{Q},\alpha,\beta} := \begin{cases} \mathfrak{I}_{\ell^*,\ell+1}^{\mathcal{Q},\alpha,\beta}\mathfrak{I}_\ell^{\mathcal{Q},\alpha,\beta} & \text{if } \ell < \ell^*, \\ \mathfrak{I}_\ell^{\mathcal{Q},\alpha,\beta} & \text{otherwise} \end{cases} \quad \text{for all } \ell,\ell^* \in \{0,\dots,p\} \text{ with } \ell^* \geq \ell. \tag{4.69}$$

For each $\iota \in \{1,\dots,d\}$, the family $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ gives rise to a family $J_\iota := (J_{\iota,\ell})_{\ell=0}^p$ of non-trivial intervals $J_{\iota,\ell} := [a_{\ell,\iota}, b_{\ell,\iota}]$ satisfying

$$J_{\iota,\ell+1} \subseteq J_{\iota,\ell} \quad \text{for all } \iota \in \{1,\dots,d\}, \ell \in \{0,\dots,p-1\},$$

and the equation

$$\mathcal{Q}_\ell = J_{1,\ell} \times \cdots \times J_{d,\ell}$$

implies that

$$\mathfrak{I}_\ell^{\mathcal{Q},\alpha,\beta} = \mathfrak{I}_\ell^{J_1,\alpha,\beta} \otimes \cdots \otimes \mathfrak{I}_\ell^{J_d,\alpha,\beta} \quad \text{holds for all } \ell \in \{0,\dots,p\}.$$

Using the definition of the re-interpolation operators yields

$$\mathfrak{I}_{\ell^*,\ell}^{\mathcal{Q},\alpha,\beta} = \mathfrak{I}_{\ell^*,\ell}^{J_1,\alpha,\beta} \otimes \cdots \otimes \mathfrak{I}_{\ell^*,\ell}^{J_d,\alpha,\beta} \quad \text{for all } \ell,\ell^* \in \{0,\dots,p\} \text{ with } \ell \leq \ell^*,$$

i.e., we have found a tensor-product representation for the multi-dimensional re-interpolation operator. We proceed as in Section 4.4: the operators $\mathfrak{I}_{\ell^*,\ell}^{\mathcal{Q},\alpha,\beta}$ are characterized by directional re-interpolation operators, which can be analyzed by using one-dimensional results.

For all $\iota \in \{1,\dots,d\}$ and all $\ell,\ell^* \in \{0,\dots,p\}$ with $\ell \leq \ell^*$, we let

$$\mathfrak{I}_{\ell^*,\ell,\iota}^{\mathcal{Q},\alpha,\beta} := \underbrace{I \otimes \cdots \otimes I}_{\iota-1 \text{ times}} \otimes \mathfrak{I}_{\ell^*,\ell}^{J_\iota,\alpha,\beta} \otimes \underbrace{I \otimes \cdots \otimes I}_{d-\iota \text{ times}}$$

and observe

$$\mathfrak{I}_{\ell^*,\ell}^{\mathcal{Q},\alpha,\beta} = \prod_{\iota=1}^d \mathfrak{I}_{\ell^*,\ell,\iota}^{\mathcal{Q},\alpha,\beta} \quad \text{for all } \ell,\ell^* \in \{0,\dots,p\} \text{ with } \ell \leq \ell^*.$$

The directional operators can again be expressed as polynomials in one variable: for $\iota \in \{1,\dots,d\}$, $\ell,\ell^* \in \{0,\dots,p\}$ with $\ell \leq \ell^*$, a function $f \in C(\mathcal{Q}_\ell)$ and all points $x \in \mathcal{Q}_{\ell^*}$, we have

$$\mathfrak{I}_{\ell^*,\ell,\iota}^{\mathcal{Q},\alpha,\beta}[f](x) = \sum_{\nu_\iota=1}^{m_\ell} f(x_1,\dots,x_{\iota-1},\xi_{m_\ell,\nu_\iota}^{J_{\iota,\ell}},x_{\iota+1},\dots,x_d)\mathfrak{I}_{\ell^*,\ell}^{J_\iota,\alpha,\beta}[\mathfrak{L}_{m_\ell,\nu_\iota}^{J_{\iota,\ell}}](x_\iota).$$

Now we can turn our attention to the problem of establishing stability and approximation error estimates for the directional operators.

For the one-dimensional re-interpolation operators, the ξ-regularity of the sequence of intervals is of central importance, therefore it is straightforward to introduce its multi-dimensional counterpart:

Definition 4.62 (ξ-regular boxes). Let $\xi \in]0,1[$. The family $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ of boxes is called *ξ-regular* if we have

$$\frac{|J_{\iota,\ell+1}|}{|J_{\iota,\ell}|} \leq \xi \quad \text{for all } \ell \in \{0,\dots,p-1\}, \iota \in \{1,\dots,d\},$$

i.e., if all the families J_ι are ξ-regular, where

$$\mathcal{Q}_\ell = J_{1,\ell} \times \cdots \times J_{d,\ell} \quad \text{for all } \ell \in \{0,\dots,p\}.$$

If the bounding boxes are ξ-regular, we can generalize the Lemmas 4.18 and 4.19:

Lemma 4.63 (Stability of directional re-interpolation). *Let $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ be ξ-regular. Let the interpolation scheme be (Λ,λ)-stable. Then there is a constant $C_{\mathrm{re}} \in \mathbb{R}_{\geq 1}$ depending only on ξ, Λ, λ and β which satisfies*

$$\|\mathfrak{I}^{\mathcal{Q},\alpha,\beta}_{\ell^*,\ell,\iota}[f]\|_{\infty,\mathcal{Q}_{\ell^*}} \leq C_{\mathrm{re}} \Lambda_{\alpha+\beta(p-\ell)} \|f\|_{\infty,\mathcal{Q}_\ell}$$

for all $\iota \in \{1,\dots,d\}$, $f \in C(\mathcal{Q}_\ell)$ and $\ell, \ell^* \in \{0,\dots,p\}$ with $\ell \leq \ell^*$.

Proof. Combine the technique used in the proof of Lemma 4.18 with the result of Theorem 4.59. □

Lemma 4.64 (Directional re-interpolation error). *Let $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ be a family of ξ-regular boxes. Let the interpolation scheme be (Λ,λ)-stable. Let a rank parameter $\beta \in \mathbb{N}_0$ and a lower bound $r_0 \in \mathbb{R}_{>0}$ for the convergence radius be given. There are constants $C_{\mathrm{ri}} \in \mathbb{R}_{\geq 1}$, $\alpha_0 \in \mathbb{N}$ such that*

$$\|f - \mathfrak{I}^{\mathcal{Q},\alpha,\beta}_{\ell^*,\ell,\iota}[f]\|_{\infty,\mathcal{Q}_{\ell^*}} \leq C_f C_{\mathrm{ri}} (\alpha+\beta(p-\ell)+1)^{2\lambda+\sigma} \left(1+\frac{1}{r_0}\right) \varrho(r_0)^{-\alpha-\beta(p-\ell)}$$

holds for all $\iota \in \{1,\dots,d\}$, all $\ell, \ell^* \in \{0,\dots,p\}$ with $\ell \leq \ell^*$, all $\alpha \in \mathbb{N}_{\geq \alpha_0}$ and all $f \in C^\infty(\mathcal{Q}_\ell)$ satisfying

$$\|\partial_\iota^\nu f\|_{\infty,\mathcal{Q}_\ell} \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma-1 \end{bmatrix} \quad \text{for all } \nu \in \mathbb{N}_0$$

for constants $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ with $2\gamma_f/|J_{\iota,\ell}| \geq r_0$.

Proof. Proceed as in the proof of Lemma 4.19 and use Theorem 4.61. □

Combining the Lemmas 4.63 and 4.64 yields the following multi-dimensional error estimate for the re-interpolation operator:

Theorem 4.65 (Multi-dimensional re-interpolation error). *Let $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ be ξ-regular. Let the interpolation scheme be (Λ, λ)-stable. Let a rank parameter $\beta \in \mathbb{N}_0$ and a lower bound $r_0 \in \mathbb{R}_{>0}$ for the convergence radius be given. There are constants $C_{\mathrm{mr}} \in \mathbb{R}_{\geq 1}$, $\alpha_0 \in \mathbb{N}$ such that*

$$\begin{aligned}\|f - \mathfrak{I}_{\ell^*,\ell}^{\mathcal{Q},\alpha,\beta}[f]\|_{\infty,\mathcal{Q}_{\ell^*}} \\ \leq C_f C_{\mathrm{mr}}(\alpha + \beta(p-\ell) + 1)^{(d+1)\lambda+\sigma} \left(1 + \frac{1}{r_0}\right) \varrho(r_0)^{-\alpha-\beta(p-\ell)}\end{aligned}$$

holds for all $\ell, \ell^* \in \{0, \ldots, p\}$ with $\ell \leq \ell^*$, all $\alpha \in \mathbb{N}_{\geq \alpha_0}$ and all $f \in C^\infty(\mathcal{Q}_\ell)$ satisfying

$$\|\partial_\iota^\nu f\|_{\infty,\mathcal{Q}_\ell} \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{for all } \nu \in \mathbb{N}_0,\ \iota \in \{1, \ldots, d\}$$

and $2\gamma_f / |J_{\iota,\ell}| \geq r_0$ for all $\iota \in \{1, \ldots, d\}$ for constants $C_f \in \mathbb{R}_{\geq 0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$.

Proof. As in the proof of Theorem 4.20, replacing the Lemmas 4.18 and 4.19 by the Lemmas 4.63 and 4.64, and using $C_{\mathrm{mr}} := C_{\mathrm{ri}} C_{\mathrm{re}}^{d-1}$. □

Application to the kernel function

Before we can apply Theorem 4.65, we have to ensure that all relevant sequences of bounding boxes are ξ-regular.

Definition 4.66 (ξ-regular bounding boxes). Let $\xi \in]0, 1[$. Let $(\mathcal{Q}_t)_{t \in \mathcal{T}_\mathcal{I}}$ be a family of bounding boxes for a cluster tree $\mathcal{T}_\mathcal{I}$. It is called *ξ-regular* if all boxes have to form

$$\mathcal{Q}_t = J_{t,1} \times \cdots \times J_{t,d}$$

for suitable non-trivial intervals $J_{t,1}, \ldots, J_{t,d}$ and if

$$J_{t',\iota} \subseteq J_{t,\iota}, \quad \frac{|J_{t',\iota}|}{|J_{t,\iota}|} \leq \xi \quad \text{holds for all } t \in \mathcal{T}_\mathcal{I}, t' \in \operatorname{sons}(t), \iota \in \{1, \ldots, d\}.$$

Let $(\mathcal{Q}_t)_{t \in \mathcal{T}_\mathcal{I}}$ and $(\mathcal{Q}_s)_{s \in \mathcal{T}_\mathcal{J}}$ be families of bounding boxes for the cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. In order to be able to apply Theorem 4.65, we have to find a sequence of bounding boxes for each pair $t \in \mathcal{T}_\mathcal{I}$, $t^* \in \mathcal{L}_t$ of clusters.

Lemma 4.67 (Sequence of ancestors). *For each $t \in \mathcal{T}_\mathcal{I}$ and each $t^* \in \mathcal{L}_t$, there is a sequence $(t_\ell)_{\ell=0}^p$ with $p = \operatorname{level}(t^*)$ satisfying*

$$t_0 = \operatorname{root}(\mathcal{T}_\mathcal{I}), \quad t_{\operatorname{level}(t)} = t, \quad t_p = t^*$$

and

$$t_{\ell+1} \in \operatorname{sons}(t_\ell) \quad \text{for all } \ell \in \{0, \dots, p-1\}.$$

Proof. By induction on $\operatorname{level}(t^*) \in \mathbb{N}_0$. If $\operatorname{level}(t^*) = 0$ holds, $t_0 = t = \operatorname{root}(\mathcal{T}_\mathcal{I})$ is the solution.

Now let $n \in \mathbb{N}_0$ be such that we can find the desired sequence for all $t \in \mathcal{T}_\mathcal{I}$, $t^* \in \mathcal{L}_t$ with $\operatorname{level}(t^*) = n$. Let $t \in \mathcal{T}_\mathcal{I}$ and $t^* \in \mathcal{L}_t$ with $\operatorname{level}(t^*) = n+1$. Due to Definition 3.6, there is a cluster $t^+ \in \mathcal{T}_\mathcal{I}$ with $t^* \in \operatorname{sons}(t^+)$ and $\operatorname{level}(t^+) = n$. We can apply the induction assumption to get a sequence $t_0, \dots, t_n$ of clusters with $t_0 = \operatorname{root}(\mathcal{T}_\mathcal{I})$, $t_n = t^+$ and $t_{\ell+1} \in \operatorname{sons}(t_\ell)$ for all $\ell \in \{0, \dots, n-1\}$. We let $p := n+1$ and $t_p := t^*$. Due to definition, we have $t^* \in \operatorname{sons}^*(t_{\operatorname{level}(t)})$ and $t^* \in \operatorname{sons}^*(t)$, i.e., $\emptyset \neq \hat{t}^* \subseteq \hat{t}_{\operatorname{level}(t)} \cap \hat{t}$, and since the levels of $t_{\operatorname{level}(t)}$ and t are identical, Lemma 3.8 yields $t = t_{\operatorname{level}(t)}$, thus concluding the induction. □

We can use this result to construct sequences of bounding boxes corresponding to the re-interpolation operators $\mathfrak{I}_{t^*,t}$ and $\mathfrak{I}_{s^*,s}$ and get the following error estimate:

Theorem 4.68 (Variable-order interpolation). *Let $(\mathcal{Q}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(\mathcal{Q}_s)_{s\in\mathcal{T}_\mathcal{J}}$ be ξ-regular families of bounding boxes. Let the kernel function g be $(C_{\mathrm{as}}, \sigma, c_0)$-asymptotically smooth with $\sigma > 0$. Let the interpolation scheme be (Λ, λ)-stable. Let $\eta \in \mathbb{R}_{>0}$. Let $\beta \in \mathbb{N}_0$. There are constants $C_{\mathrm{vo}} \in \mathbb{R}_{\geq 1}$ and $\alpha_0 \in \mathbb{N}$ such that*

$$\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \leq \frac{C_{\mathrm{vo}}(\alpha + \beta\ell^+ + 1)^{(2d+1)\lambda+\sigma}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta\ell^-}$$

with

$$\ell^+ := \max\{p_\mathcal{I} - \operatorname{level}(t), p_\mathcal{J} - \operatorname{level}(s)\},$$
$$\ell^- := \min\{p_\mathcal{I} - \operatorname{level}(t), p_\mathcal{J} - \operatorname{level}(s)\}$$

holds for all $\alpha \in \mathbb{N}_{\geq\alpha_0}$, all blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfying the standard admissibility condition

$$\max\{|J_{t,\iota}|, |J_{s,\iota}| \,:\, \iota \in \{1, \dots, d\}\} \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s) \tag{4.70}$$

and all $t^* \in \mathcal{L}_t$ and $s^* \in \mathcal{L}_s$.

Proof. Let $\alpha \in \mathbb{N}_{\geq\alpha_0}$, let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfying (4.70), and let $t^* \in \mathcal{L}_t$ and $s^* \in \mathcal{L}_s$. In order to be able to apply Theorem 4.65, we have to find families of boxes $\mathcal{Q} = (\mathcal{Q}_\ell)_{\ell=0}^p$ such that the corresponding re-interpolation operators $\mathfrak{I}^{\mathcal{Q},\alpha,\beta}_{\ell^*,\ell}$ coincide with $\mathfrak{I}_{t^*,t}$ or $\mathfrak{I}_{s^*,s}$.

Without loss of generality, we only consider $\mathfrak{I}_{t^*,t}$. Let $\ell_t := \operatorname{level}(t)$ and $\ell_t^* := \operatorname{level}(t^*)$. According to Lemma 4.67, there is a sequence $(t_\ell)_{\ell=0}^{\ell_t^*}$ satisfying $t_\ell = t$, $t_{\ell^*} = t^*$ and

$$t_{\ell+1} \in \operatorname{sons}(t_\ell) \quad \text{for all } \ell \in \{0, \ldots, \ell_t^* - 1\}.$$

We define the family $\mathcal{Q} := (\mathcal{Q}_\ell)_{\ell=0}^{\ell_t^*}$ by $\mathcal{Q}_\ell := \mathcal{Q}_{t_\ell}$. This family is ξ-regular due to our assumptions, and comparing Definition 4.53 and (4.68), (4.69) yields

$$\mathfrak{I}_{\ell_t^*,\ell_t}^{\mathcal{Q},\alpha_t,\beta} = \mathfrak{I}_{t^*,t} \quad \text{with } \alpha_t := \alpha + \beta(p_{\mathcal{I}} - \ell_t^*) \geq \alpha.$$

The asymptotic smoothness of g implies that we can apply the re-interpolation error estimate provided by Theorem 4.65 in order to get

$$\begin{aligned}
&\|g - (\mathfrak{I}_{t^*,t} \otimes I)[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_s} \\
&\quad \leq \frac{C_f C_{\mathrm{mr}}(\alpha_t + \beta(\ell_t^* - \ell_t))^{(d+1)\lambda+\sigma}(1 + c_0\eta)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho \left(\frac{1}{c_0\eta}\right)^{-\alpha_t - \beta(\ell_t^* - \ell_t)} \\
&\quad \leq \frac{C_f C_{\mathrm{mr}}(\alpha + \beta(p_{\mathcal{I}} - \ell_t))^{(d+1)\lambda+\sigma}(1 + c_0\eta)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho \left(\frac{1}{c_0\eta}\right)^{-\alpha - \beta(p_{\mathcal{I}} - \ell_t)} \\
&\quad \leq \frac{C_f C_{\mathrm{mr}}(\alpha + \beta\ell^+)^{(d+1)\lambda+\sigma}(1 + c_0\eta)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho \left(\frac{1}{c_0\eta}\right)^{-\alpha - \beta\ell^-}.
\end{aligned}$$

Applying the same reasoning to $\mathfrak{I}_{s^*,s}$ yields

$$\begin{aligned}
&\|g - (I \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_{s^*}} \\
&\quad \leq \frac{C_f C_{\mathrm{mr}}(\alpha + \beta\ell^+)^{(d+1)\lambda+\sigma}(1 + c_0\eta)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho \left(\frac{1}{c_0\eta}\right)^{-\alpha - \beta\ell^-}.
\end{aligned}$$

Due to the stability estimate of Theorem 4.59, we have

$$\|(\mathfrak{I}_{t^*,t} \otimes I)[f]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \leq C_{\mathrm{re}}^d \Lambda^d (\alpha + \beta\ell^+)^d \|f\|_{\mathcal{Q}_t\times\mathcal{Q}_{s^*}}$$

for all $f \in C(\mathcal{Q}_t \times \mathcal{Q}_s)$, and we can conclude

$$\begin{aligned}
&\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\
&\quad \leq \|g - (\mathfrak{I}_{t^*,t} \otimes I)[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\
&\qquad + \|(\mathfrak{I}_{t^*,t} \otimes I)[g - (I \otimes \mathfrak{I}_{s^*,s})[g]]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\
&\quad \leq \|g - (\mathfrak{I}_{t^*,t} \otimes I)[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_s} \\
&\qquad + C_{\mathrm{re}}^d \Lambda^d (\alpha + \beta\ell^+)^d \|g - (I \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_{s^*}} \\
&\quad \leq \frac{C_f C_{\mathrm{mr}} C_{\mathrm{re}}^d \Lambda^d (\alpha + \beta\ell^+ + 1)^{(2d+1)\lambda+\sigma}(1 + c_0\eta)}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \varrho \left(\frac{1}{c_0\eta}\right)^{-\alpha - \beta\ell^-}.
\end{aligned}$$

Defining the constant C_{vo} as

$$C_{\mathrm{vo}} := C_f C_{\mathrm{mr}} C_{\mathrm{re}}^d \Lambda^d (1 + c_0 \eta)$$

yields the desired estimate. □

In order to be able to apply Theorem 4.47, we have to separate the clusters t and s appearing in the estimate of Theorem 4.68. We can do this by assuming that the quantities $p_{\mathcal{I}} - \mathrm{level}(t)$ and $p_{\mathcal{J}} - \mathrm{level}(s)$ describing the number of re-interpolation steps are close to each other:

Definition 4.69 (C_{cn}-consistency). Let $C_{\mathrm{cn}} \in \mathbb{N}_0$, and let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree. $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called C_{cn}-*consistent* if

$$|(p_{\mathcal{I}} - \mathrm{level}(t)) - (p_{\mathcal{J}} - \mathrm{level}(s))| \leq C_{\mathrm{cn}} \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

For standard grid hierarchies, we can find a uniform bound for the difference between $p_{\mathcal{I}}$ and $p_{\mathcal{J}}$ for all meshes in the hierarchy, i.e., the left-hand term in Definition 4.69 can be bounded uniformly for all meshes. This implies that the corresponding block cluster trees will be C_{cn}-consistent for a constant C_{cn} which does not depend on the mesh, but only on the underlying geometry and the discretization scheme.

Before we can apply Theorem 4.47 to bound the spectral error of the variable-order approximation, we have to establish the factorized error estimates of the form (4.53).

Corollary 4.70 (Factorized estimate). *Let $(\mathcal{Q}_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(\mathcal{Q}_s)_{s\in\mathcal{T}_{\mathcal{J}}}$ be ξ-regular families of bounding boxes. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{cn}-consistent. Let the kernel function g be $(C_{\mathrm{as}}, \sigma, c_0)$-asymptotically smooth with $\sigma > 0$. Let the interpolation scheme be (Λ, λ)-stable. Let $\eta \in \mathbb{R}_{>0}$. Let $\beta \in \mathbb{N}_0$. There are constants $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ and $\alpha_0 \in \mathbb{N}$ such that*

$$\begin{aligned}\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\ \leq \left(\frac{C_{\mathrm{in}} q^{\alpha+\beta(p_{\mathcal{I}}-\mathrm{level}(t))}}{\mathrm{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2} \left(\frac{C_{\mathrm{in}} q^{\alpha+\beta(p_{\mathcal{J}}-\mathrm{level}(s))}}{\mathrm{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}\end{aligned} \tag{4.71}$$

holds with

$$q := \min\left\{\frac{c_0\eta}{c_0\eta+1}, \frac{c_0\eta}{2}\right\}$$

for all $\alpha \in \mathbb{N}_{\geq\alpha_0}$, all blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfying the admissibility condition

$$\max\{\mathrm{diam}(\mathcal{Q}_t), \mathrm{diam}(\mathcal{Q}_s)\} \leq 2\eta\, \mathrm{dist}(\mathcal{Q}_t, \mathcal{Q}_s) \tag{4.72}$$

and all $t^* \in \mathcal{L}_t$ and $s^* \in \mathcal{L}_s$.

Proof. Let $\alpha \in \mathbb{N}_{\geq \alpha_0}$ and let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfy the admissibility condition (4.72).

We have to find bounds for the three important factors appearing in the error estimate of Theorem 4.68. The admissibility condition (4.72) yields

$$\begin{aligned}\frac{1}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^\sigma} &\leq \frac{(2\eta)^\sigma}{\max\{\operatorname{diam}(\mathcal{Q}_t)^\sigma, \operatorname{diam}(\mathcal{Q}_s)^\sigma\}}\\ &\leq (2\eta)^\sigma \frac{1}{\operatorname{diam}(\mathcal{Q}_t)^{\sigma/2}} \frac{1}{\operatorname{diam}(\mathcal{Q}_s)^{\sigma/2}}.\end{aligned} \tag{4.73}$$

Using the consistency of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we find

$$\begin{aligned}\alpha + \beta\ell^+ + 1 &\leq \alpha + \beta(p_{\mathcal{I}} - \operatorname{level}(t)) + 1 + C_{\mathrm{cn}}\beta\\ &\leq (C_{\mathrm{cn}}\beta + 1)(\alpha + \beta(p_{\mathcal{I}} - \operatorname{level}(t)) + 1),\\ \alpha + \beta\ell^+ + 1 &\leq \alpha + \beta(p_{\mathcal{J}} - \operatorname{level}(s)) + 1 + C_{\mathrm{cn}}\beta\\ &\leq (C_{\mathrm{cn}}\beta + 1)(\alpha + \beta(p_{\mathcal{J}} - \operatorname{level}(s)) + 1),\end{aligned}$$

and can bound the stability term by

$$\begin{aligned}(\alpha + \beta\ell^+ + 1)^{(2d+1)\lambda+\sigma} &\leq (C_{\mathrm{cn}}\beta + 1)^{1/2}(\alpha + \beta(p_{\mathcal{I}} - \operatorname{level}(t)) + 1)^{1/2}\\ &\quad (\alpha + \beta(p_{\mathcal{J}} - \operatorname{level}(s)) + 1)^{1/2}.\end{aligned} \tag{4.74}$$

The convergence term can also be bounded by using the consistency. We get

$$\begin{aligned}\varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta\ell^-} &\leq \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta(p_{\mathcal{I}}-\operatorname{level}(t))+C_{\mathrm{cn}}\beta}\\ &= \varrho\left(\frac{1}{c_0\eta}\right)^{C_{\mathrm{cn}}\beta} \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta(p_{\mathcal{I}}-\operatorname{level}(t))},\\ \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta\ell^-} &\leq \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta(p_{\mathcal{J}}-\operatorname{level}(s))+C_{\mathrm{cn}}\beta}\\ &= \varrho\left(\frac{1}{c_0\eta}\right)^{C_{\mathrm{cn}}\beta} \varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta(p_{\mathcal{J}}-\operatorname{level}(s))}\end{aligned},$$

and conclude

$$\begin{aligned}\varrho\left(\frac{1}{c_0\eta}\right)^{-\alpha-\beta\ell^-} \leq \varrho\left(\frac{1}{c_0\eta}\right)^{C_{\mathrm{cn}}\beta/2} &\varrho\left(\frac{1}{c_0\eta}\right)^{-(\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t)))/2}\\ &\varrho\left(\frac{1}{c_0\eta}\right)^{-(\alpha+\beta(p_{\mathcal{J}}-\operatorname{level}(s)))/2}.\end{aligned} \tag{4.75}$$

Combining Theorem 4.68 with (4.73), (4.74) and (4.75) yields the factorized error

estimate

$$\begin{aligned}\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\ \leq \left(\frac{C'(\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))+1)^{(2d+1)\lambda+\sigma}}{\operatorname{diam}(\mathcal{Q}_t)^{\sigma}\varrho(1/(c_0\eta))^{\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))}}\right)^{1/2} \\ \left(\frac{C'(\alpha+\beta(p_{\mathcal{J}}-\operatorname{level}(s))+1)^{(2d+1)\lambda+\sigma}}{\operatorname{diam}(\mathcal{Q}_s)^{\sigma}\varrho(1/(c_0\eta))^{\alpha+\beta(p_{\mathcal{J}}-\operatorname{level}(s))}}\right)^{1/2},\end{aligned} \tag{4.76}$$

where we abbreviate

$$C' := C_{\mathrm{vo}}(2\eta)^{\sigma}(C_{\mathrm{cn}}\beta+1)^{1/2}\varrho\left(\frac{1}{c_0\eta}\right)^{C_{\mathrm{cn}}\beta/2}.$$

As in Remark 4.23, we can see that

$$q > \frac{1}{\varrho(1/(c_0\eta))}$$

holds, i.e., we have $\zeta := q\varrho(1/(c_0\eta)) > 1$.

We consider only the first factor on the right-hand side of estimate (4.76): introducing the polynomial

$$S\colon \mathbb{R}\to\mathbb{R}, \quad x \mapsto C'(x+1)^{(2d+1)\lambda+\sigma}$$

yields

$$\begin{aligned}\frac{C'(\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))+1)^{(2d+1)\lambda+\sigma}}{\varrho(1/(c_0\eta))^{\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))}} &= \frac{S(\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t)))}{\varrho(1/(c_0\eta))^{\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))}} \\ &= \frac{S(n)}{(\zeta/q)^n} = q^n\frac{S(n)}{\zeta^n}\end{aligned}$$

for $n := \alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))$. Since S is a polynomial and due to $\zeta > 1$, we can find a constant $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ such that

$$\frac{S(x)}{\zeta^x} \leq C_{\mathrm{in}} \quad \text{holds for all } x \in \mathbb{R}_{\geq 1},$$

and we conclude

$$\frac{C'(\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))+1)^{(2d+1)\lambda+\sigma}}{\varrho(1/(c_0\eta))^{\alpha+\beta(p_{\mathcal{I}}-\operatorname{level}(t))}} \leq C_{\mathrm{in}}q^n. \qquad \square$$

Global spectral error estimate

Now we can assemble the spectral error estimate provided by Theorem 4.47, the blockwise estimate of Lemma 4.45 and the factorized error estimate of Corollary 4.70.

We assume that the families $(\mathcal{Q}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(\mathcal{Q}_s)_{s\in\mathcal{T}_\mathcal{J}}$ of bounding boxes are ξ-regular, that the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is C_{cn}-consistent and C_{sp}-sparse, that the kernel function is $(C_{\mathrm{as}},\sigma,c_0)$-asymptotically smooth with $\sigma > 0$, that the interpolation scheme is (Λ,λ)-stable, that the supports $(\Omega_i)_{i\in\mathcal{I}}$ and $(\Omega_j)_{j\in\mathcal{J}}$ of the basis functions are C_{ov}-overlapping and that the L^2-norms of the basis functions $(\varphi_i)_{i\in\mathcal{I}}$ and $(\psi_j)_{j\in\mathcal{J}}$ are bounded by constants $C_\mathcal{I}, C_\mathcal{J} \in \mathbb{R}_{>0}$ defined as in (4.48).

We fix an admissibility parameter $\eta \in \mathbb{R}_{>0}$ and assume that (4.58) holds, i.e., that the domain Ω is not folded too tightly, and that (4.59) holds, i.e., that the bounding boxes do not grow too rapidly.

Lemma 4.71 (Factorized error estimate). *Let $\beta \in \mathbb{N}_0$. With the constants $C_{\mathrm{in}} \in \mathbb{R}_{>0}$, $q \in]0,1[$ and $\alpha_0 \in \mathbb{N}$ from Corollary 4.70, we have*

$$\begin{aligned}\|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_2 \\ \leq \left(C_{\mathrm{in}} C_\mathcal{I}^2 C_{\mathrm{ov}} |\Omega_t| \frac{q^{\alpha+\beta(p_\mathcal{I}-\mathrm{level}(t))}}{\operatorname{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2} \left(C_{\mathrm{in}} C_\mathcal{J}^2 C_{\mathrm{ov}} |\Omega_s| \frac{q^{\alpha+\beta(p_\mathcal{J}-\mathrm{level}(s))}}{\operatorname{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}\end{aligned}$$

for all $\alpha \in \mathbb{N}_{\geq\alpha_0}$ and all blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ satisfying the admissibility condition (4.72).

Proof. We can use Corollary 4.70 to bound the interpolation error by

$$\begin{aligned}\|g - (\mathfrak{I}_{t^*,t} \otimes \mathfrak{I}_{s^*,s})[g]\|_{\infty,\mathcal{Q}_{t^*}\times\mathcal{Q}_{s^*}} \\ \leq \epsilon_b := \left(\frac{C_{\mathrm{in}} q^{\alpha+\beta(p_\mathcal{I}-\mathrm{level}(t))}}{\operatorname{diam}(\mathcal{Q}_t)^\sigma}\right)^{1/2} \left(\frac{C_{\mathrm{in}} q^{\alpha+\beta(p_\mathcal{J}-\mathrm{level}(s))}}{\operatorname{diam}(\mathcal{Q}_s)^\sigma}\right)^{1/2}\end{aligned}$$

and since ϵ_b does not depend on t^* or s^*, we can use Lemma 4.56 to conclude

$$\|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_2 \leq C_\mathcal{I} C_\mathcal{J} \epsilon_b \Big(\sum_{i\in\hat{t}} |\Omega_i|\Big)^{1/2} \Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2}.$$

Since the supports are C_{ov}-overlapping, this estimate can be simplified:

$$\|\chi_t G \chi_s - \widetilde{V}_t S_b \widetilde{W}_s^*\|_2 \leq C_\mathcal{I} C_\mathcal{J} \epsilon_b |\Omega_t|^{1/2} |\Omega_s|^{1/2}.$$

Combining this estimate with the definition of ϵ_b yields the desired bound for the blockwise error. □

Combining Theorem 4.47 with this estimate yields the following error bound for the global spectral error:

Theorem 4.72 (Variable-order spectral error). *Let $\beta \in \mathbb{N}_0$, and let $d_\Omega \geq \sigma$. With the constants $C_{\mathrm{in}} \in \mathbb{R}_{>0}$, $q \in]0,1[$ and $\alpha_0 \in \mathbb{N}$ from Corollary 4.70 and $C_{\mathrm{cu}} \in \mathbb{R}_{>0}$ and*

$d_\Omega \in \mathbb{N}$ from (4.58), we have

$$\|G - \widetilde{G}\|_2 \leq C_{\mathrm{sp}} C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} C_{\mathrm{in}} C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma} q^\alpha \Big(\sum_{\ell=0}^{p_{\mathcal{I}}} (q^\beta \zeta^{d_\Omega - \sigma})^{p_{\mathcal{I}} - \ell} \Big)^{1/2} \Big(\sum_{\ell=0}^{p_{\mathcal{J}}} (q^\beta \zeta^{d_\Omega - \sigma})^{p_{\mathcal{J}} - \ell} \Big)^{1/2}$$

for all $\alpha \in \mathbb{R}_{\geq \alpha_0}$.

Proof. We combine Theorem 4.47 with Lemma 4.71. In the resulting estimate, we use (4.58) and (4.59) in order to find

$$\begin{aligned} \frac{|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} &\leq \frac{C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega}}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} = C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega - \sigma} \\ &\leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} h^{d_\Omega - \sigma} \zeta^{(d_\Omega - \sigma)(p_{\mathcal{I}} - \operatorname{level}(t))} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}, \\ \frac{|\Omega_s|}{\operatorname{diam}(\mathcal{Q}_s)^\sigma} &\leq \frac{C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega}}{\operatorname{diam}(\mathcal{Q}_s)^\sigma} = C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega - \sigma} \\ &\leq C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} h^{d_\Omega - \sigma} \zeta^{(d_\Omega - \sigma)(p_{\mathcal{J}} - \operatorname{level}(s))} \qquad \text{for all } s \in \mathcal{T}_{\mathcal{J}} \end{aligned}$$

and notice that combining these factors with the ones from Lemma 4.71 yields the desired result. □

We can use the rank parameter β to compensate the growth of the clusters and the rank parameter α to reduce the global error to any prescribed accuracy:

Corollary 4.73 (Variable-order approximation). *Let $d_\Omega \geq \sigma$. Let $C \in \mathbb{R}_{>0}$. We can ensure*

$$\|G - \widetilde{G}\|_2 \leq C C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma}$$

by choosing the rank parameters $\alpha \in \mathbb{N}$ and $\beta \in \mathbb{N}_0$ appropriately.

Proof. Let $\beta \in \mathbb{N}_0$ with

$$q^\beta \zeta^{d_\Omega - \sigma} < 1/2.$$

According to Theorem 4.72, we can choose $\alpha \in \mathbb{N}_{\geq \alpha_0}$ with

$$2 C_{\mathrm{sp}} C_{\mathrm{cu}} C_{\mathrm{gr}}^{d_\Omega - \sigma} C_{\mathrm{in}} C_{\mathrm{ov}} q^\alpha \leq C$$

and observe

$$\begin{aligned} \sum_{\ell=0}^{p_{\mathcal{I}}} (q^\beta \zeta^{d_\Omega - \sigma})^{p_{\mathcal{I}} - \ell} &\leq \sum_{\ell=0}^{p_{\mathcal{I}}} \left(\frac{1}{2} \right)^{p_{\mathcal{I}} - \ell} \leq 2, \\ \sum_{\ell=0}^{p_{\mathcal{J}}} (q^\beta \zeta^{d_\Omega - \sigma})^{p_{\mathcal{J}} - \ell} &\leq \sum_{\ell=0}^{p_{\mathcal{J}}} \left(\frac{1}{2} \right)^{p_{\mathcal{J}} - \ell} \leq 2, \end{aligned}$$

and applying Theorem 4.72 yields the desired bound. □

4.8 Technical lemmas

Lemma 4.74 (Cardinality of multi-index sets). *We have*

$$\#\{\nu \in \mathbb{N}_0^d \,:\, |\nu| \le n\} = \binom{n+d}{d} \tag{4.77}$$

for all $n \in \mathbb{N}_0$ *and all* $d \in \mathbb{N}$.

Proof. We first prove

$$\sum_{j=0}^{n} \binom{j+d}{d} = \binom{n+d+1}{d+1} \tag{4.78}$$

for all $d \in \mathbb{N}$ and all $n \in \mathbb{N}_0$ by induction on n.

Let $d \in \mathbb{N}$. For $n = 0$, the equation is trivial. Assume now that (4.78) holds for $n \in \mathbb{N}_0$. We conclude the induction by observing

$$\begin{aligned}
\sum_{j=0}^{n+1} \binom{j+d}{d} &= \sum_{j=0}^{n} \binom{j+d}{d} + \binom{n+d+1}{d} = \binom{n+d+1}{d+1} + \binom{n+d+1}{d} \\
&= \frac{(n+d+1)!}{n!(d+1)!} + \frac{(n+d+1)!}{(n+1)!d!} \\
&= \frac{(n+d+1)!(n+1)}{(n+1)!(d+1)!} + \frac{(n+d+1)!(d+1)}{(n+1)!(d+1)!} \\
&= \frac{(n+d+1)!(n+d+2)}{(n+1)!(d+1)!} = \binom{n+d+2}{d+1}.
\end{aligned}$$

Now we prove (4.77) by induction on $d \in \mathbb{N}$. For $d = 1$, we have

$$\{\nu \in \mathbb{N}_0^d \,:\, |\nu| \le n\} = \{0, \dots, n\},$$

i.e.,

$$\#\{\nu \in \mathbb{N}_0^d \,:\, |\nu| \le n\} = n+1 = \frac{(n+1)!}{1!} n! = \binom{n+d}{d}.$$

Let us now assume that (4.77) holds for a given $d \in \mathbb{N}$ and all $n \in \mathbb{N}_0$. Let $n \in \mathbb{N}_0$. We have

$$\{\nu \in \mathbb{N}_0^{d+1} \,:\, |\nu| \le n\} = \bigcup_{i=0}^{n} \{(\nu, i) \,:\, \nu \in \mathbb{N}_0^d, |\nu| \le n-i\},$$

which implies

$$\begin{aligned}
\#\{\nu \in \mathbb{N}_0^{d+1} \,:\, |\nu| \le n\} &= \sum_{i=0}^{n} \#\{(\nu, i) \,:\, \nu \in \mathbb{N}_0^d, |\nu| \le n-i\} \\
&= \sum_{i=0}^{n} \{\nu \in \mathbb{N}_0^d \,:\, |\nu| \le n-i\}
\end{aligned}$$

$$= \sum_{i=0}^{n} \binom{n-i+d}{d} = \sum_{j=0}^{n} \binom{j+d}{d} = \binom{n+d+1}{d+1},$$

where we have used (4.78) in the last step. □

Lemma 4.75 (Taylor approximation error). *Let $\omega \subseteq \mathbb{R}^d$ be a star-shaped domain with center $z_0 \in \mathbb{R}^d$. Let $m \in \mathbb{N}$, and let $f \in C^m(\omega)$. For all $z \in \omega$, the approximation error of the Taylor approximation*

$$\tilde{f}_{z_0,m}(z) = \sum_{|\nu|<m} \partial^\nu f(z_0) \frac{(z-z_0)^\nu}{\nu!}$$

is given by

$$f(z) - \tilde{f}_{z_0,m}(z) = m \sum_{|\nu|=m} \int_0^1 (1-t)^{m-1} \partial^\nu f(z_0 + (z-z_0)t)\, dt \frac{(z-z_0)^\nu}{\nu!}. \quad (4.79)$$

Proof. Let $z \in \omega$ and $p := z - z_0$. We define the function

$$F\colon [0,1] \to \mathbb{R}, \quad t \mapsto f(z_0 + pt),$$

which satisfies $F(1) = f(z)$, and we define its m-th order Taylor approximation

$$\tilde{F}_m\colon [0,1] \to \mathbb{R}, \quad t \mapsto \sum_{i=0}^{m-1} \frac{t^i}{i!} F^{(i)}(0),$$

for all $m \in \mathbb{N}$. A simple induction yields

$$F^{(i)}(t) = \sum_{|\nu|=i} \frac{i!}{\nu!} \partial^\nu f(z_0 + pt) p^\nu, \quad \tilde{f}_{z_0,m}(z) = \tilde{F}_m(1) \quad \text{for all } i \in \{0,\dots,m\},$$

so we can apply Lemma 2.1 to get

$$\begin{aligned} F(1) - \tilde{F}_m(1) &= \frac{1}{(m-1)!} \int_0^1 (1-t)^{m-1} F^{(m)}(t)\, dt \\ &= \frac{1}{(m-1)!} \sum_{|\nu|=m} \frac{m!}{\nu!} \int_0^1 (1-t)^{m-1} \partial^\nu f(z_0 + pt) p^\nu\, dt \\ &= m \sum_{|\nu|=m} \int_0^1 (1-t)^{m-1} \partial^\nu f(z_0 + pt)\, dt \frac{(z-z_0)^\nu}{\nu!}, \end{aligned}$$

and this is the desired result. □

Lemma 4.76 (Holomorphic extension). *Let $f \in C^\infty[-1,1]$, $C_f \in \mathbb{R}_{>0}$, $\gamma_f \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ be such that*

$$|f^{(\nu)}(x)| \leq \frac{C_f}{\gamma_f^\nu} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \quad \text{holds for all } \nu \in \mathbb{N}_0, x \in [-1,1]. \tag{4.80}$$

Let $r \in [0, \gamma_f[$, and let

$$\mathcal{R}_r := \{z \in \mathbb{C} \ : \ \text{there is a } z_0 \in [-1,1] \text{ with } |z - z_0| \leq r\}.$$

We have $[-1,1] \subseteq \mathcal{R}_r$, and there is a holomorphic extension $\tilde{f} \in C^\infty(\mathcal{R}_r)$ of f satisfying

$$|\tilde{f}(z)| \leq C_f \left(\frac{\gamma_f}{\gamma_f - r} \right)^\sigma \quad \text{for all } z \in \mathcal{R}_r. \tag{4.81}$$

Proof. For all $\tau \in \mathbb{N}$, we define the function

$$s_\tau \colon [0,1[\to \mathbb{R}, \quad \zeta \mapsto \frac{1}{(1-\zeta)^\tau},$$

and observe that

$$s_\tau^{(\ell)} = \frac{(\tau - 1 + \ell)!}{(\tau - 1)!} s_{\tau+\ell} = \begin{bmatrix} \ell \\ \tau - 1 \end{bmatrix} s_{\tau+\ell} \quad \text{holds for all } \ell \in \mathbb{N}_0, \ \tau \in \mathbb{N}.$$

Computing the Taylor expansion of s_τ at zero yields

$$s_\tau(\zeta) = \sum_{\nu=0}^{\infty} \begin{bmatrix} \nu \\ \tau - 1 \end{bmatrix} \zeta^\nu \quad \text{for all } \zeta \in [0,1[. \tag{4.82}$$

Let us now turn our attention to the function f. For all $z_0 \in [-1,1]$, we define the disc

$$\mathcal{D}_{z_0} := \{z \in \mathbb{C} \ : \ |z - z_0| \leq r\}$$

centered at z_0 with radius r.

Let $z_0 \in [-1,1]$. For all $z \in \mathcal{D}_{z_0}$, we have $|z - z_0| \leq r < \gamma_f$, and the upper bound (4.80) implies

$$\begin{aligned} \sum_{\nu=0}^{\infty} \left| f^{(\nu)}(z_0) \frac{(z-z_0)^\nu}{\nu!} \right| &\leq C_f \sum_{\nu=0}^{\infty} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \left(\frac{|z - z_0|}{\gamma_f} \right)^\nu \\ &\leq C_f \sum_{\nu=0}^{\infty} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \left(\frac{r}{\gamma_f} \right)^\nu = C_f \sum_{\nu=0}^{\infty} \begin{bmatrix} \nu \\ \sigma - 1 \end{bmatrix} \zeta^\nu \end{aligned}$$

for $\zeta := r/\gamma_f \in [0,1[$. Comparing this equation with (4.82) yields

$$\sum_{\nu=0}^{\infty} \left| f^{(\nu)}(z_0) \frac{(z-z_0)^\nu}{\nu!} \right| \leq C_f s_\sigma(\zeta) < \infty, \tag{4.83}$$

i.e., the Taylor series in the point z_0 converges on $\mathcal{D}_{z_0}$, therefore the function

$$\tilde{f}_{z_0} \colon \mathcal{D}_{z_0} \to \mathbb{C}, \quad z \mapsto \sum_{i=0}^{\infty} f^{(i)}(z_0) \frac{(z-z_0)^i}{i!},$$

is holomorphic.

Let $z \in \mathcal{D}_{z_0} \cap [-1,1] \subseteq \mathbb{R}$. Combining (4.27) and the error equation (2.5) of the truncated Taylor expansion, we find

$$\begin{aligned}
&\left| f(z) - \sum_{i=0}^{m-1} f^{(i)}(z_0) \frac{(z-z_0)^i}{i!} \right| \\
&\quad = \left| \int_0^1 (1-t)^{m-1} f^{(m)}(z_0 + t(z-z_0)) \frac{(z-z_0)^m}{(m-1)!} \, dt \right| \\
&\quad \leq \int_0^1 (1-t)^{m-1} |f^{(m)}(z_0 + t(z-z_0))| \frac{|z-z_0|^m}{(m-1)!} \, dt \\
&\quad \leq C_f \left(\frac{|z-z_0|}{\gamma_f} \right)^m (\sigma - 1 + m)! \int_0^1 \frac{(1-t)^{m-1}}{(m-1)!} \, dt \\
&\quad \leq C_f \frac{(\sigma - 1 + m)!}{m!} \left(\frac{r}{\gamma_f} \right)^m
\end{aligned}$$

for all $m \in \mathbb{N}$. Due to $r/\gamma_f < 1$, sending m to infinity yields $f(z) = \tilde{f}_{z_0}(z)$ for all $z \in \mathcal{D}_{z_0} \cap [-1,1]$, i.e., $\tilde{f}_{z_0}$ is an extension of f into $\mathcal{D}_{z_0}$.

Now we can define the extension $\tilde{f}$ of f into $\mathcal{R}_r$. For all $z \in \mathcal{R}_r$, we pick a point $z_0 \in [-1,1]$ with $|z - z_0| \leq r$ and define $\tilde{f}(z) := \tilde{f}_{z_0}(z)$. Since all holomorphic functions $\tilde{f}_{z_0}$ are extensions of f, the identity theorem for holomorphic functions yields that $\tilde{f}$ is a well-defined holomorphic extension of f in $\mathcal{R}_r$. The inequality (4.83) implies (4.81). □

Lemma 4.77 (Covering of $\mathcal{E}_\varrho$). *Let $r \in \mathbb{R}_{>0}$ and $\varrho := r + \sqrt{1+r^2}$. Let $\mathcal{R}_r \subseteq \mathbb{C}$ be defined in Lemma 4.76 and let $\mathcal{E}_\varrho \subseteq \mathbb{C}$ be the regularity ellipse defined in Lemma 4.14. Then we have $\mathcal{E}_\varrho \subseteq \mathcal{R}_r$.*

Proof. Let $z \in \mathcal{E}_\varrho$, and let $x, y \in \mathbb{R}$ with $z = x + iy$. We have to find $z_0 \in [-1,1]$ with $|z - z_0| \leq r$. The definition (4.26) implies

$$\left(\frac{2x}{\varrho + 1/\varrho} \right)^2 + \left(\frac{2y}{\varrho - 1/\varrho} \right)^2 \leq 1,$$

and we can conclude

$$x^2 \leq \frac{(\varrho + 1/\varrho)^2}{4}, \quad y^2 \leq \frac{(\varrho - 1/\varrho)^2}{4}.$$

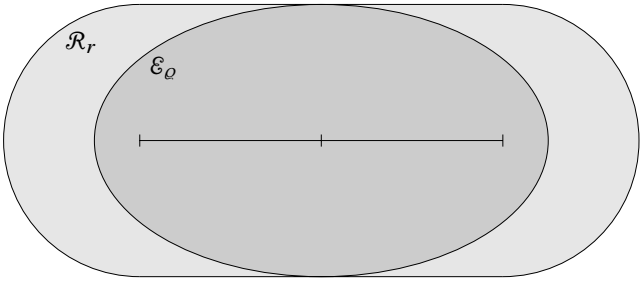


Figure 4.5. Covering $\mathcal{E}_\varrho$ by a family of discs.

If $x \in [-1, 1]$ holds, we let $z_0 := x$ and observe

$$\begin{aligned} |z - z_0| = |y| \le \frac{\varrho - 1/\varrho}{2} &= \frac{\varrho^2 - 1}{2\varrho} = \frac{(r + \sqrt{1 + r^2})^2 - 1}{2\varrho} \\ &= \frac{r^2 + 2r\sqrt{1 + r^2} + 1 + r^2 - 1}{2\varrho} = \frac{2r^2 + 2r\sqrt{1 + r^2}}{2\varrho} \\ &= \frac{2r(r + \sqrt{1 + r^2})}{2\varrho} = r, \end{aligned}$$

i.e., $z \in \mathcal{R}_r$.

Otherwise we have $|x| > 1$. Due to symmetry, we can restrict our attention to the case $x > 1$. We let $z_0 := 1$. We have

$$1 < x \le \frac{\varrho^2 + 1}{2\varrho},$$

which implies

$$4x^2\varrho^2 \le 4\varrho^2 \frac{(\varrho^2 + 1)^2}{4\varrho^2} = (\varrho^2 + 1)^2,$$

and we can conclude

$$\begin{aligned} x^2(\varrho^2 - 1)^2 &= x^2(\varrho^4 - 2\varrho^2 + 1) = x^2(\varrho^4 + 2\varrho^2 + 1) - 4x^2\varrho^2 \\ &= x^2(\varrho^2 + 1)^2 - 4x^2\varrho^2 \ge x^2(\varrho^2 + 1)^2 - (\varrho^2 + 1)^2 \\ &= (x^2 - 1)(\varrho^2 + 1)^2 > (x^2 - 2x + 1)(\varrho^2 + 1)^2 = (x - 1)^2(\varrho^2 + 1)^2. \end{aligned}$$

This estimate is equivalent to

$$\frac{4x^2}{(\varrho^2 + 1)^2} > \frac{4(x - 1)^2}{(\varrho^2 - 1)^2},$$

so we get

$$1 \geq \left(\frac{2x}{\varrho + 1/\varrho}\right)^2 + \left(\frac{2y}{\varrho - 1/\varrho}\right)^2 = \frac{4x^2\varrho^2}{(\varrho^2+1)^2} + \frac{4y^2\varrho^2}{(\varrho^2-1)^2}$$
$$> \frac{4(x-1)^2\varrho^2}{(\varrho^2-1)^2} + \frac{4y^2\varrho^2}{(\varrho^2-1)^2} = \frac{4(x-1)^2}{(\varrho-1/\varrho)^2} + \frac{4y^2}{(\varrho-1/\varrho)^2} = \frac{(x-1)^2}{r^2} + \frac{y^2}{r^2},$$

i.e., $(x-1)^2 + y^2 < r^2$. This is equivalent to $|z - z_0| < r$ and implies $z \in \mathcal{R}_r$. □

Lemma 4.78 (Scaling of ϱ). *Let*

$$\varrho\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 1}, \quad r \mapsto r + \sqrt{1+r^2}.$$

Let $r_0 \in \mathbb{R}_{>0}$. For each $\xi \in (0,1)$, there is a $\hat{\xi} \in (0,1)$ such that

$$\varrho\left(\frac{r}{\xi}\right) \geq \frac{1}{\hat{\xi}}\varrho(r) \quad \textit{holds for all } r \in \mathbb{R}_{\geq r_0}.$$

Proof. Let $x \in \mathbb{R}_{>0}$. We start by considering the function

$$f\colon \mathbb{R}_{>1} \to \mathbb{R}_{\geq 0}, \quad \alpha \mapsto \sqrt{1 + x^2\alpha^2}.$$

Elementary computations yield

$$f'(\alpha) = \frac{\alpha x^2}{\sqrt{1+x^2\alpha^2}}, \quad f''(\alpha) = \frac{x^2}{(1+x^2\alpha^2)\sqrt{1+x^2\alpha^2}} \geq 0, \quad \text{for all } \alpha \in \mathbb{R}_{>0}.$$

Applying the error equation (2.5) to the second-order Taylor expansion of f in the point 1 yields

$$f(\alpha) - f(1) - f'(1)(\alpha - 1) \geq 0 \quad \text{for all } \alpha \geq 1,$$

and we conclude

$$\sqrt{1+x^2\alpha^2} = f(\alpha) \geq f(1) + f'(1)(\alpha-1) = \sqrt{1+x^2} + \frac{x^2}{\sqrt{1+x^2}}(\alpha-1). \quad (4.84)$$

Now let $r \in \mathbb{R}_{\geq r_0}$. Applying (4.84) to $x := r$ and $\alpha := 1/\xi > 1$ yields

$$\varrho(r/\xi) = \varrho(r\alpha) = r\alpha + \sqrt{1+r^2\alpha^2} \geq r\alpha + \sqrt{1+r^2} + \frac{r^2}{\sqrt{1+r^2}}(\alpha-1)$$
$$= r + \sqrt{1+r^2} + \frac{r\sqrt{1+r^2}+r^2}{\sqrt{1+r^2}}(\alpha-1)$$
$$= (r + \sqrt{1+r^2})\left(1 + \frac{r}{\sqrt{1+r^2}}(\alpha-1)\right)$$

$$= \varrho(r)\left(1 + \frac{r}{\sqrt{1+r^2}}(\alpha - 1)\right).$$

Due to $r \geq r_0 \geq 0$, we have $r^2 \geq r_0^2$ and $r^2(1+r_0^2) = r^2 + r^2 r_0^2 \geq r_0^2 + r^2 r_0^2 = r_0^2(1+r^2)$. This means

$$\frac{r^2}{1+r^2} \geq \frac{r_0^2}{1+r_0^2}, \quad \text{i.e.,} \quad \frac{r}{\sqrt{1+r^2}} \geq \frac{r_0}{\sqrt{1+r_0^2}},$$

so we can conclude

$$\varrho(r/\xi) \geq \varrho(r)\left(1 + \frac{r_0}{\sqrt{1+r_0^2}}\frac{1-\xi}{\xi}\right). \tag{4.85}$$

We let

$$\hat{\xi} := \frac{\xi\sqrt{1+r_0^2}}{\xi\sqrt{1+r_0^2} + (1-\xi)r_0} < 1$$

and observe

$$\frac{1}{\hat{\xi}} = \frac{\xi\sqrt{1+r_0^2} + (1-\xi)r_0}{\xi\sqrt{1+r_0^2}} = 1 + \frac{1-\xi}{\xi}\frac{r_0}{\sqrt{1+r_0^2}},$$

which combined with (4.85) is the desired result. □

4.9 Numerical experiments

The purpose of this section is to demonstrate that the theoretical properties of the techniques introduced in this chapter can be observed in the implementation, i.e., that the estimates for the complexity and the error are close to the real behaviour of the algorithms.

In order to allow us to observe the relevant effects, we choose a relatively simple setting: we consider the stiffness matrices $V \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and $K \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ corresponding to the three-dimensional single and double layer potential operator, given by

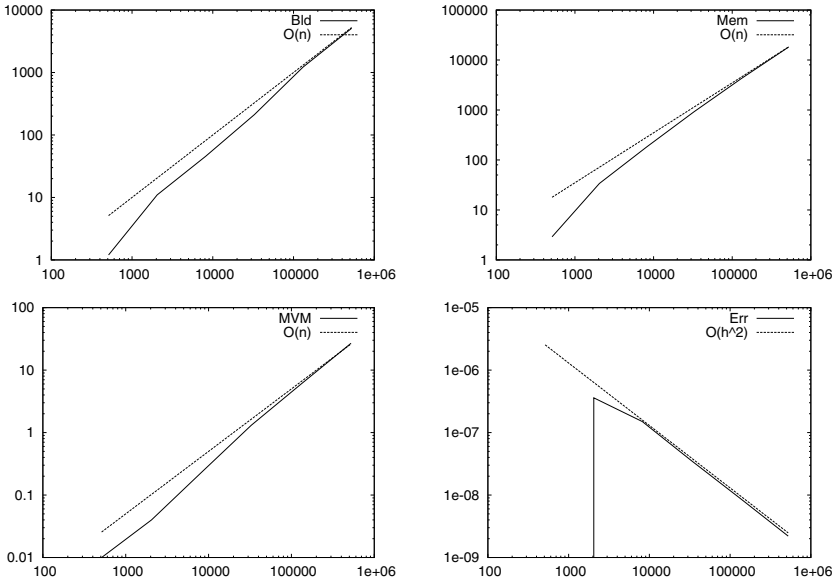
$$V_{ij} := \int_\Gamma \varphi_i(x) \int_\Omega \varphi_j(y) \frac{1}{4\pi\|x-y\|_2}\, dy\, dx \qquad \text{for all } i,j \in \mathcal{I},$$

$$K_{ij} := \int_\Gamma \varphi_i(x) \int_\Omega \varphi_j(y) \frac{\langle x-y, n(y)\rangle}{4\pi\|x-y\|_2^3}\, dy\, dx \qquad \text{for all } i,j \in \mathcal{I},$$

discretized by piecewise constant basis functions $(\varphi_i)_{i\in\mathcal{I}}$ on a piecewise triangular approximation Γ of the unit sphere $\Gamma_S := \{x \in \mathbb{R}^3 : \|x\|_2 = 1\}$.

Table 4.1. Single layer potential operator, approximated by interpolation with $m = 4$ and $\eta = 2$ for different matrix dimensions n.

n	Bld	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	1.2	2.9	5.8	< 0.01	0
2048	11.0	33.9	17.0	0.04	3.6_{-7}
8192	45.7	183.2	22.9	0.23	1.5_{-7}
32768	211.8	923.3	28.9	1.31	3.6_{-8}
131072	1219.6	4274.5	33.4	5.99	9.0_{-9}
524288	5107.4	18330.0	35.8	26.92	2.2_{-9}



The integrals are computed by the black-box quadrature techniques introduced in [89], [43]. Since we are only interested in the approximation properties of the matrix, not in those of the underlying operators, we use a constant order for the quadrature.

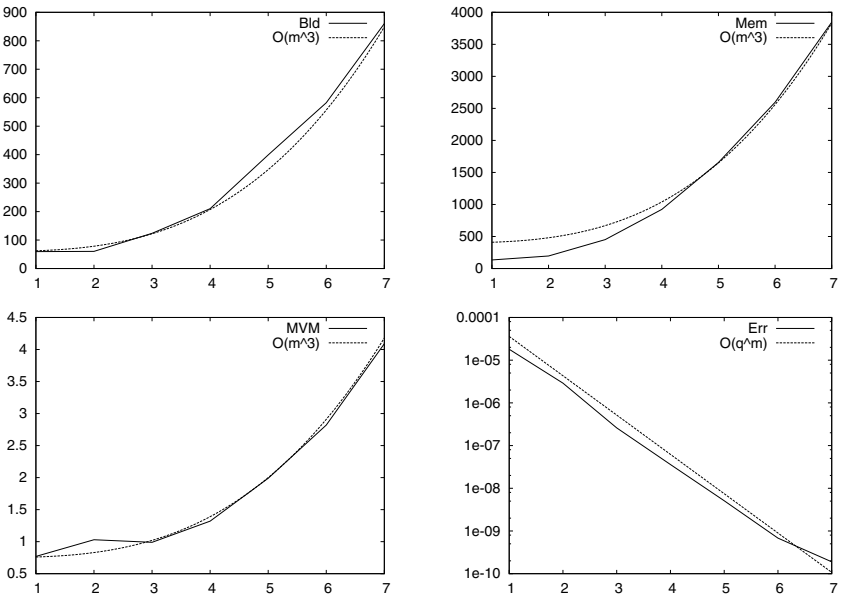
Let us now consider the approximation of the matrix V corresponding to the single layer potential operator by constant-order interpolation (cf. Section 4.4). The manifold Γ_S is approximated by a sequence of polygons Γ with $n \in \{512, 2048, \ldots, 524288\}$ triangles, and one piecewise constant basis function is used in each triangle.

Table 4.1 lists the results for $\mathcal{H}^2$-matrices constructed by using tensor-product interpolation of order $m = 4$ for a block cluster tree satisfying the admissibility condition (4.49) with $\eta = 2$. The table gives the dimension of the matrix ("n"), the time in seconds[1] for the construction of the $\mathcal{H}^2$-matrix approximation ("Bld"), the stor-

[1] On a single 900 MHz UltraSPARC IIIcu processor of a SunFire 6800 computer.

Table 4.2. Single layer potential operator, approximated by interpolation with different orders m and $\eta = 2$ for the matrix dimension $n = 32768$.

m	Bld	Mem	Mem/n	MVM	$\|X - \widetilde{X}\|_2$
1	59.4	134.2	4.2	0.77	1.8_{-5}
2	60.3	195.7	6.1	1.03	2.9_{-6}
3	123.6	451.3	14.1	0.99	2.6_{-7}
4	210.2	923.3	28.9	1.32	3.6_{-8}
5	399.0	1655.8	51.7	1.99	5.1_{-9}
6	582.3	2595.6	81.1	2.82	6.8_{-10}
7	860.3	3845.2	120.2	4.09	1.9_{-10}



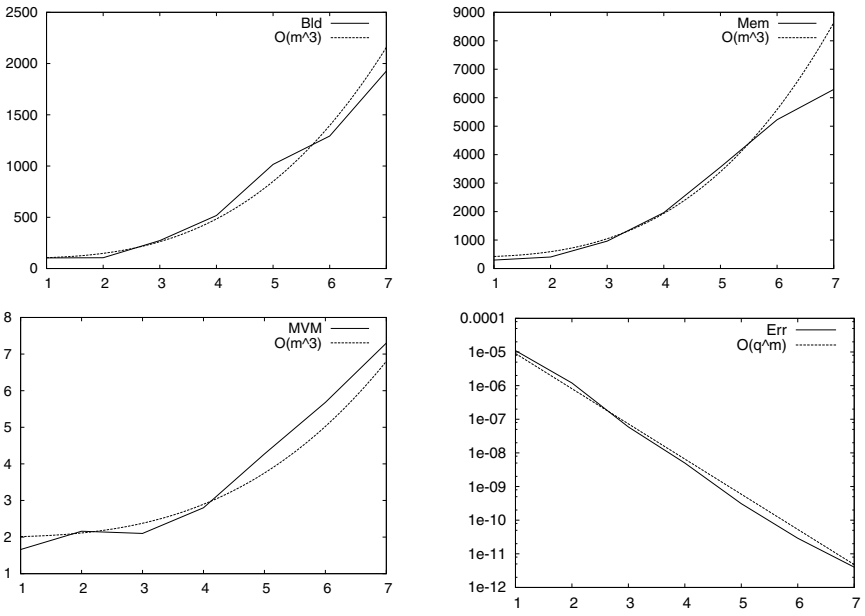
age requirements in MBytes[2] ("Mem"), the storage required per degree of freedom in KBytes[3] ("Mem/n"), the time in seconds for the matrix-vector multiplication and an estimate for the spectral norm of the error ("$\|X - \widetilde{X}\|_2$").

The complexity estimate of Lemma 3.38 leads us to expect that the time for constructing the $\mathcal{H}^2$-matrix and its storage requirements will behave like $\mathcal{O}(n)$, and this is clearly visible in the table. Theorem 3.42 suggests that $\mathcal{O}(n)$ operations are required to perform a matrix-vector multiplication, and this estimate also coincides with the numerical results. Remark 4.50 implies that the error of the kernel approximation will

[2] 1 MByte = 2^{20} Bytes
[3] 1 KByte = 2^{10} Bytes

Table 4.3. Single layer potential operator, approximated by interpolation with different orders m and $\eta = 1$ for the matrix dimension $n = 32768$.

m	Bld	Mem	Mem/n	MVM	$\|X - \widetilde{X}\|_2$
1	103.6	297.0	9.3	1.66	1.1_{-5}
2	105.6	403.7	12.6	2.16	1.2_{-6}
3	272.6	961.8	30.1	2.10	5.9_{-8}
4	517.1	1963.8	61.4	2.80	5.0_{-9}
5	1015.9	3562.9	111.3	4.27	3.1_{-10}
6	1293.0	5226.7	163.3	5.68	2.9_{-11}
7	1926.3	6291.1	216.3	7.30	4.0_{-12}

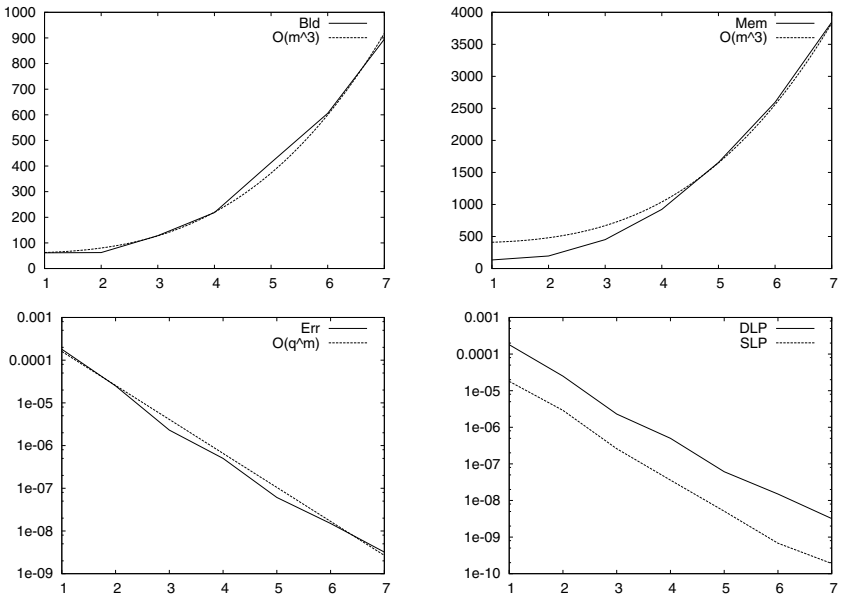


be constant on all grid levels, and the scaling of the basis functions ($C_{\mathcal{I}}$ and $C_{\mathcal{J}}$ are in $\mathcal{O}(h)$) implies that the error should behave like $\mathcal{O}(h^2)$, which is indeed visible. We can conclude that the theoretical predictions concerning the dependence on n match the practical observations.

Now let us investigate the relationship between the order m of the interpolation and the complexity of the $\mathcal{H}^2$-matrix approximation. Table 4.2 lists the storage and time requirements and the accuracy of the $\mathcal{H}^2$-matrix approximation for different choices of the order $m \in \{1, \ldots, 7\}$ and a fixed grid with $n = 32768 = 2^{15}$ triangles. Lemma 4.10 yields that the rank distribution is m^3-bounded. According to Lemma 3.38, we expect the build time and storage requirements to behave like $\mathcal{O}(m^3)$, and according to The-

Table 4.4. Double layer potential operator, approximated by interpolation with different orders m and $\eta = 2$ for the matrix dimension $n = 32768$.

m	Bld	Mem	Mem/n	MVM	$\|X - \widetilde{X}\|_2$
1	61.4	134.2	4.2	0.77	1.8_{-4}
2	62.3	195.7	6.1	1.03	2.5_{-5}
3	128.3	451.3	14.1	0.99	2.3_{-6}
4	218.3	923.3	28.9	1.32	5.0_{-7}
5	413.4	1655.8	51.7	1.99	6.1_{-8}
6	605.8	2595.6	81.1	2.83	1.5_{-8}
7	895.1	3845.2	120.2	3.98	3.2_{-9}

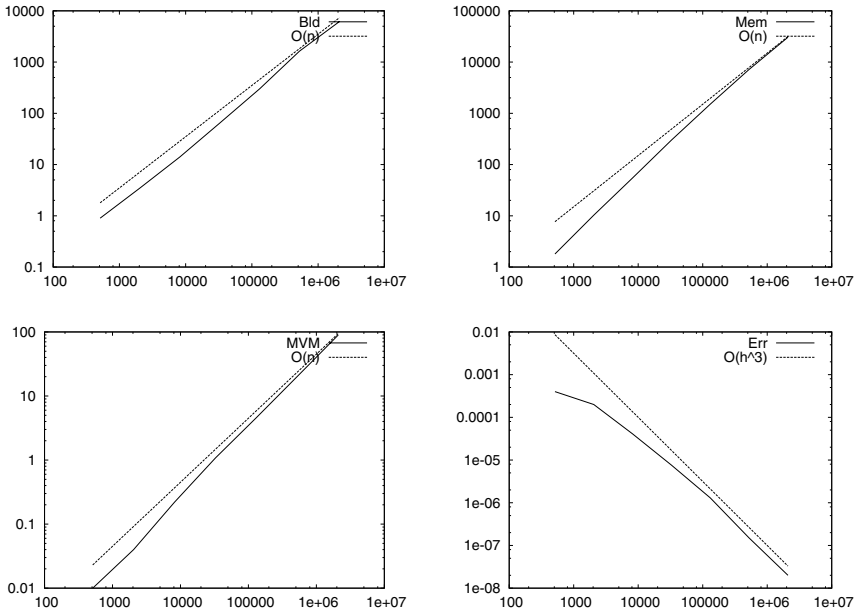


orem 3.42, the matrix-vector multiplication should also require $\mathcal{O}(m^3)$ operations. These predictions coincide with the results given in Table 4.2. Due to Theorem 4.22, we expect the approximation error to decrease exponentially with a rate of $q \approx 2/3$, but we measure a rate of approximately 0.12. Whether this is due to suboptimal estimates or to a pre-asymptotic behaviour of the interpolation is not clear.

Repeating the same experiment for the admissibility parameter $\eta = 1$ gives the results listed in Table 4.3. We can see that the storage and time requirements are approximately doubled, while the approximation error is reduced by a factor between 2 and 50. Theorem 4.22 suggests that we have to expect a convergence rate of $q \approx 1/2$, but we again measure a better rate of approximately 0.09. We can conclude that using a lower value of η will improve the rate of convergence, but that the increase in storage

Table 4.5. Single layer potential operator, approximated by variable-order interpolation with $\alpha = 1$, $\beta = 1$ and $\eta = 2$ for different matrix dimensions n.

n	Bld	Mem	Mem/n	MVM	$\|X - \widetilde{X}\|_2$
512	0.9	1.8	3.6	< 0.01	4.0_{-4}
2048	3.5	10.3	5.2	0.04	2.0_{-4}
8192	14.1	54.8	6.8	0.22	4.1_{-5}
32768	64.6	300.8	9.4	1.08	7.6_{-6}
131072	300.6	1508.3	11.8	4.60	1.3_{-6}
524288	1633.0	7181.7	14.0	20.26	1.5_{-7}
2097152	6177.5	30584.1	14.9	89.50	2.0_{-8}



and algorithmic complexity renders this approach unattractive in practice: we can get a similar improvement of the error by increasing the order instead of decreasing η.

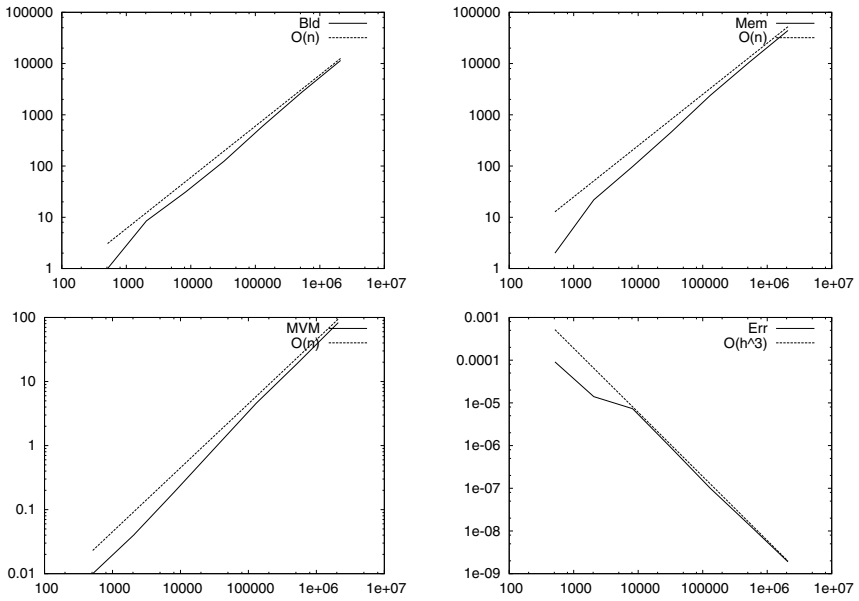
We approximate the matrix K corresponding to the double layer potential operator by the approach described in Section 4.5: we construct an approximation of the kernel function

$$\frac{\langle x - y, n(y)\rangle}{4\pi \|x - y\|_2^3} = \frac{\partial}{\partial n(y)} \frac{1}{4\pi \|x - y\|_2}$$

by taking the normal derivative of an interpolant of the generator function $\gamma(x, y) := 1/(4\pi\|x - y\|_2)$. As before, we test the approximation scheme by using different orders $m \in \{1, \ldots, 7\}$ for a fixed grid with $n = 32768 = 2^{15}$ triangles. The results

Table 4.6. Single layer potential operator, approximated by variable-order interpolation with $\alpha = 2$, $\beta = 1$ and $\eta = 2$ for different matrix dimensions n.

n	Bld	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	1.0	2.0	4.1	< 0.01	9.1_{-5}
2048	8.5	21.9	11.0	0.04	1.4_{-5}
8192	30.7	97.6	12.2	0.19	7.3_{-6}
32768	124.6	463.3	14.5	0.94	8.7_{-7}
131072	607.0	2393.5	18.7	4.61	1.0_{-7}
524288	2743.0	10539.6	20.6	19.19	1.4_{-8}
2097152	11494.6	44237.5	21.6	83.06	1.9_{-9}



are listed in Table 4.4: we can see that, the time and storage requirements behave like $\mathcal{O}(m^3)$ and that the error decreases exponentially. Once again the measured rate of convergence, approximately 0.16, is far better than the predicted rate of $q \approx 2/3$.

Now let us turn our attention to the variable-order interpolation scheme described in Section 4.7. Instead of the isotropic scheme described in this section, we use the more practical anisotropic rank distribution algorithm described in [23], [80]. The results for the rank parameters $\alpha = 1$ and $\beta = 1$ and the dimensions $n \in \{2^9, 2^{11}, \ldots, 2^{21}\}$ are given in Table 4.5. We can see that the time for the construction of the matrix, its storage requirements and the time required for a matrix-vector multiplication converge to $\mathcal{O}(n)$ and that the error decays like $\mathcal{O}(h^3)$, which is the rate predicted by Corollary 4.73 for $d_\Omega = 2$ and $\sigma = 1$.

Table 4.6 is concerned with the case where the "slope" of the order distribution is still $\beta = 1$, but the lower bound is now $\alpha = 2$. In our theory, α is used to compensate certain pollution effects caused by the growth of cluster supports and bounding boxes, therefore a higher value of α can be expected to lead to a more regular convergence of the error. We can observe this behaviour in this experiment: except for a short pre-asymptotic phase, the error converges exactly like $\mathcal{O}(h^3)$. The price for this improved convergence are an increase in storage requirements (by approximately 33%) and a pronounced increase in setup time (which is almost doubled).

Chapter 5
Orthogonal cluster bases and matrix projections

We have seen that we can construct $\mathcal{H}^2$-matrix approximations for certain types of integral operators by using polynomial expansions. These approximations have a reasonable accuracy and can be constructed efficiently, but they require a large amount of storage. On modern computers, which have typically very fast processors and a comparatively small amount of memory, the storage requirements are the limiting factor for most practical applications.

Since the storage requirements of an $\mathcal{H}^2$-matrix are governed by the rank distributions of the row and column cluster bases, we have to reduce these ranks without sacrificing accuracy. Then we have to convert the given matrix into an element of the $\mathcal{H}^2$-matrix space defined by the improved cluster bases.

An elegant solution for the second task is available if the cluster basis matrices are *orthogonal*, i.e., if their columns are pairwise orthogonal vectors. In this case, the best approximation of an arbitrary matrix in the $\mathcal{H}^2$-matrix space can be constructed by a simple orthogonal projection.

The concept of orthogonal cluster basis matrices also provides a good approach to the first task: in an orthogonal cluster basis matrix, no column is redundant, since it cannot be expressed in terms of the other columns.

This chapter introduces algorithms for turning a cluster basis into an orthogonal cluster basis with the same or almost the same approximation properties, for converting an arbitrary matrix into an $\mathcal{H}^2$-matrix using orthogonal cluster bases, and for estimating or even explicitly computing the error introduced by this conversion. It is organized as follows:

- Section 5.1 introduces the concept of orthogonal cluster bases and provides us with a simple algorithm for checking whether a given cluster basis is orthogonal.
- Section 5.2 contains algorithms for converting dense and $\mathcal{H}$-matrices into $\mathcal{H}^2$-matrices using orthogonal cluster bases.
- Section 5.3 defines cluster operators, which are used to describe transformations between different cluster bases and allow us to switch efficiently between different $\mathcal{H}^2$-matrix spaces.
- Section 5.4 is devoted to an algorithm that turns an arbitrary nested cluster basis into an orthogonal nested cluster basis in optimal complexity.
- Section 5.5 introduces a variant of the former algorithm that only produces an approximation of the original cluster basis. A careful error analysis allows us to control the resulting approximation error (cf. [10], [20]).

- Section 5.6 describes algorithms for computing the error introduced by a projection into an $\mathcal{H}^2$-matrix space.

- Section 5.7 contains numerical experiments that demonstrate that the bounds for the complexity of the new algorithms are optimal and that the adaptive error control works as predicted by theory.

Assumptions in this chapter: We assume that cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ for the finite index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, are given. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let $n_\mathcal{I} := \#\mathcal{I}$ and $n_\mathcal{J} := \#\mathcal{J}$ denote the number of indices in $\mathcal{I}$ and $\mathcal{J}$, and let $c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}$ and $c_\mathcal{J} := \#\mathcal{T}_\mathcal{J}$ denote the number of clusters in $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let $p_\mathcal{I}$, $p_\mathcal{J}$ and $p_{\mathcal{I}\times\mathcal{J}}$ be the depths of $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ and $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

5.1 Orthogonal cluster bases

While the construction of cluster bases using polynomials is quite straightforward, it typically includes a certain degree of redundancy: the rank of cluster basis matrices $V_t \in \mathbb{R}^{\hat{t}\times K_t}$ for leaf clusters $t \in \mathcal{L}_\mathcal{I}$ is bounded by $\#\hat{t}$ and $\#K_t$, i.e., if $\#K_t > \#\hat{t}$ holds, the matrix V_t has more columns than necessary, which leads to suboptimal efficiency.

We can avoid redundant columns in cluster basis matrices V_t by ensuring that all columns of V_t are pairwise orthogonal, since this implies that we cannot eliminate any of the columns without changing the range of the matrix.

Definition 5.1 (Orthogonal cluster basis). Let V be a cluster basis for $\mathcal{T}_\mathcal{I}$. V is called *orthogonal* if

$$V_t^* V_t = I \tag{5.1}$$

holds for each $t \in \mathcal{T}_\mathcal{I}$, i.e., if all cluster basis matrices are orthogonal.

For nested cluster bases, we can find a criterion for the orthogonality which depends only on the cluster basis matrices V_t in leaf clusters $t \in \mathcal{L}_\mathcal{I}$ and the transfer matrices E_t, i.e., on quantities that are available in a standard $\mathcal{H}^2$-matrix representation.

We require the following simple auxiliary result:

Lemma 5.2. *Let V_1 and V_2 be cluster bases for $\mathcal{T}_\mathcal{I}$. Let $t \in \mathcal{T}_\mathcal{I}$ with* $\operatorname{sons}(t) \neq \emptyset$. *Let $t_1, t_2 \in \operatorname{sons}(t)$. We have*

$$V_{1,t_1}^* V_{2,t_2} = \begin{cases} V_{1,t_1}^* V_{2,t_1} & \text{if } t_1 = t_2, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Proof. The case $t_1 = t_2$ is obvious, so we only consider $t_1 \neq t_2$. Due to Definition 3.33, we have $V_{1,t_1} = \chi_{t_1} V_{1,t_1}$ and $V_{2,t_2} = \chi_{t_2} V_{2,t_2}$. Definition 3.4 implies $\hat{t}_1 \cap \hat{t}_2 = \emptyset$, and this means $\chi_{t_1} \chi_{t_2} = 0$ due to Definition 3.24. Combining these equations yields

$$V_{1,t_1}^* V_{2,t_2} = (\chi_{t_1} V_{1,t_1})^* \chi_{t_2} V_{2,t_2} = V_{1,t_1}^* \chi_{t_1} \chi_{t_2} V_{2,t_2} = 0$$

and completes the proof. □

Now we can proceed to prove an orthogonality criterion based only on matrices available in the standard representation:

Lemma 5.3 (Orthogonality criterion). *Let $V = (V_t)_{t \in \mathcal{T}_\mathcal{I}}$ be a nested cluster basis for $\mathcal{T}_\mathcal{I}$ with transfer matrices $E = (E_t)_{t \in \mathcal{T}_\mathcal{I}}$. V is orthogonal if and only if*

$$V_t^* V_t = I \quad \text{holds for all leaf clusters } t \in \mathcal{L}_\mathcal{I}$$

and

$$\sum_{t' \in \operatorname{sons}(t)} E_{t'}^* E_{t'} = I \quad \text{holds for all non-leaf clusters } t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}.$$

Proof. Let us first assume that V is orthogonal. Let $t \in \mathcal{T}_\mathcal{I}$. The case $\operatorname{sons}(t) = \emptyset$ is trivial. If $\operatorname{sons}(t) \neq \emptyset$, we have

$$\begin{aligned}\sum_{t' \in \operatorname{sons}(t)} E_{t'}^* E_{t'} &\overset{(5.1)}{=} \sum_{t' \in \operatorname{sons}(t)} E_{t'}^* V_{t'}^* V_{t'} E_{t'} \overset{(5.2)}{=} \sum_{t_1 \in \operatorname{sons}(t)} \sum_{t_2 \in \operatorname{sons}(t)} E_{t_1}^* V_{t_1}^* V_{t_2} E_{t_2} \\ &= \Big(\sum_{t_1 \in \operatorname{sons}(t)} V_{t_1} E_{t_1} \Big)^* \Big(\sum_{t_2 \in \operatorname{sons}(t)} V_{t_2} E_{t_2} \Big) \overset{(3.15)}{=} V_t^* V_t = I,\end{aligned}$$

which yields the desired result.

Let us now assume that $V_t^* V_t = I$ holds for all leaf clusters $t \in \mathcal{L}_\mathcal{I}$ and that $\sum_{t' \in \operatorname{sons}(t)} E_{t'}^* E_{t'} = I$ holds for all remaining clusters $t \in \mathcal{T}_\mathcal{I} \setminus \mathcal{L}_\mathcal{I}$. We prove $V_t^* V_t = I$ by induction on $\#\operatorname{sons}^*(t)$. For $\#\operatorname{sons}^*(t) = 1$, we have $\operatorname{sons}(t) = \emptyset$ and $V_t^* V_t = I$ holds by assumption.

Let now $n \in \mathbb{N}$ be such that $V_t^* V_t = I$ holds for all $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) \leq n$. Let $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) = n + 1$. For all $t' \in \operatorname{sons}(t)$, we have $\#\operatorname{sons}^*(t') \leq \#\operatorname{sons}^*(t) - 1 = n$, and the induction assumption implies $V_{t'}^* V_{t'} = I$. We find

$$\begin{aligned}V_t^* V_t &\overset{(3.15)}{=} \Big(\sum_{t_1 \in \operatorname{sons}(t)} V_{t_1} E_{t_1} \Big)^* \Big(\sum_{t_2 \in \operatorname{sons}(t)} V_{t_2} E_{t_2} \Big)^* \\ &\overset{(5.2)}{=} \sum_{t' \in \operatorname{sons}(t)} E_{t'}^* V_{t'}^* V_{t'} E_{t'} = \sum_{t' \in \operatorname{sons}(t)} E_{t'}^* E_{t'} = I,\end{aligned}$$

which concludes the induction step. □

It is important to note that this criterion can only be used to determine whether the *entire* cluster basis is orthogonal. If there is a cluster $t \in \mathcal{T}_I$ with $V_t^* V_t \neq I$, the criterion cannot be applied to its predecessors.

We can use the criterion of Lemma 5.3 in a recursion to determine whether a given nested cluster basis is orthogonal. This approach is summarized in Algorithm 9. Since it is a special case of Algorithm 13, the complexity analysis of Lemma 5.13 applies.

Algorithm 9. Check a nested cluster basis for orthogonality.

```
function IsOrthogonal(t, V) : boolean;
if sons(t) = ∅ then
  ortho ← (V_t^* V_t = I)
else
  ortho ← true;
  P ← 0;
  for t' ∈ sons(t) do
    ortho ← ortho and IsOrthogonal(t', V);
    P ← P + E_{t'}^* E_{t'}
  end for;
  ortho ← ortho and (P = I)
end if;
return ortho
```

5.2 Projections into $\mathcal{H}^2$-matrix spaces

Remark 3.37 states that $\mathcal{H}^2(\mathcal{T}_{I\times J}, V, W)$ is a subspace of $\mathbb{R}^{I\times J}$. Equipped with the *Frobenius inner product* given by

$$\langle X, Y\rangle_F := \sum_{i\in I}\sum_{j\in J} X_{ij} Y_{ij} = \sum_{i\in I} (XY^*)_{ii} = \sum_{j\in J} (X^*Y)_{jj}$$

for all $X, Y \in \mathbb{R}^{I\times J}$ and the corresponding *Frobenius norm*

$$\|X\|_F := \langle X, X\rangle_F^{1/2} = \Big(\sum_{i\in I}\sum_{j\in J} X_{ij}^2\Big)^{1/2}$$

for all $X \in \mathbb{R}^{I\times J}$, the matrix space $\mathbb{R}^{I\times J}$ is a Hilbert space.

This implies that the best approximation of an arbitrary matrix $X \in \mathbb{R}^{I\times J}$ in the subspace $\mathcal{H}^2(\mathcal{T}_{I\times J}, V, W)$ with respect to the Frobenius norm is given by an orthogonal projection.

Before we can investigate its properties, we need the following auxiliary results:

Lemma 5.4 (Block restriction). *Let $(X_b)_{b\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}}$ be a family of matrices $X_b \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. For all $b' = (t', s') \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, we have*

$$\chi_{t'}\Big(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \chi_t X_b \chi_s\Big)\chi_{s'} = \chi_{t'} X_{b'} \chi_{s'}.$$

Proof. Let $b' = (t', s') \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.

Let $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ with $\chi_{t'}\chi_t X_b \chi_s \chi_{s'} \neq 0$. Due to Definition 3.24, this implies that there are $i \in \hat{t} \cap \hat{t}'$ and $j \in \hat{s} \cap \hat{s}'$, i.e., that $\hat{b} \cap \hat{b}' = (\hat{t} \times \hat{s}) \cap (\hat{t}' \times \hat{s}') \neq \emptyset$.

Due to Lemma 3.14, $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ is the set of leaves of a cluster tree for $\mathcal{I} \times \mathcal{J}$, and due to Corollary 3.9, this means that the corresponding index sets form a disjoint partition of $\mathcal{I} \times \mathcal{J}$, and therefore $\hat{b} \cap \hat{b}' \neq \emptyset$ implies $b = b'$. □

We can use this lemma to pick single blocks from the equations (3.12) and (3.17), which helps us to find an explicit representation of the orthogonal projection from $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ into $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$.

Lemma 5.5 ($\mathcal{H}^2$-matrix projection). *Let V and W be orthogonal cluster bases. The operator*

$$\begin{aligned}\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} \colon \mathbb{R}^{\mathcal{I}\times\mathcal{J}} &\to \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W),\\ X &\mapsto \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t V_t^* X W_s W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s,\end{aligned}$$

is the orthogonal projection into $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. This implies

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} X\|_F = \inf_{\widetilde{X}\in\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W)} \|X - \widetilde{X}\|_F \tag{5.3}$$

for all matrices $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

Proof. Let $\Pi := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}$. In a first step we establish that Π is a projection into $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$, and let $(S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be the corresponding coupling matrices. Due to Lemma 5.4, we have

$$\begin{aligned}\chi_{t'} X \chi_{s'} &= \chi_{t'}\Big(\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t S_b W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s\Big)\chi_{s'}\\ &= \chi_{t'}\Big(\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \chi_t V_t S_b W_s^* \chi_s + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s\Big)\chi_{s'} = V_{t'} S_{b'} W_{s'}^*\end{aligned}$$

and therefore

$$\begin{aligned}V_{t'} V_{t'}^* X W_{s'} W_{s'}^* &= V_{t'} V_{t'}^* (\chi_{t'} X \chi_{s'}) W_{s'} W_{s'}^* = V_{t'} V_{t'}^* (V_{t'} S_{b'} W_{s'}^*) W_{s'} W_{s'}^*\\ &\overset{(5.1)}{=} V_{t'} S_{b'} W_{s'}^* = \chi_{t'} X \chi_{s'}\end{aligned}$$

for all $b' = (t', s') \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. We can apply a similar argument to $b' = (t', s') \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ and add the contributions of all blocks to get $\Pi X = X$, and observing $\operatorname{range}(\Pi) \subseteq \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ concludes this part of the proof.

In order to prove that Π is an *orthogonal* projection, we have to demonstrate $\Pi^* = \Pi$. For $X, Y \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, we find

$$\langle \Pi X, Y\rangle_F = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle \Pi X, \chi_t Y \chi_s\rangle_F = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle \chi_t(\Pi X)\chi_s, Y\rangle_F.$$

Lemma 5.4 implies

$$\begin{aligned}\langle \chi_t(\Pi X)\chi_s, Y\rangle_F &= \langle V_t V_t^* X W_s W_s^*, Y\rangle_F \\ &= \langle X, V_t V_t^* Y W_s W_s^*\rangle_F = \langle X, \chi_t(\Pi Y)\chi_s\rangle_F\end{aligned}$$

for $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and

$$\langle \chi_t(\Pi X)\chi_s, Y\rangle_F = \langle \chi_t X \chi_s, Y\rangle_F = \langle X, \chi_t Y \chi_s\rangle_F = \langle X, \chi_t(\Pi Y)\chi_s\rangle_F$$

for $b = (t, s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$, so we can conclude

$$\begin{aligned}\langle \Pi X, Y\rangle_F &= \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle \chi_t(\Pi X)\chi_s, Y\rangle_F \\ &= \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \langle X, \chi_t(\Pi Y)\chi_s\rangle_F = \langle X, \Pi Y\rangle_F,\end{aligned}$$

which implies $\Pi^* = \Pi$.

Let now $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ and $\widetilde{X} \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. We observe

$$\begin{aligned}\|X - \widetilde{X}\|_F^2 &= \|X - \Pi X + \Pi X - \widetilde{X}\|_F^2 \\ &= \|X - \Pi X\|_F^2 + 2\langle X - \Pi X, \Pi X - \widetilde{X}\rangle_F^2 + \|\Pi X - \widetilde{X}\|_F^2 \\ &= \|X - \Pi X\|_F^2 + 2\langle X - \Pi X, \Pi(X - \widetilde{X})\rangle_F^2 + \|\Pi X - \widetilde{X}\|_F^2 \\ &= \|X - \Pi X\|_F^2 + 2\langle \Pi X - \Pi^2 X, X - \widetilde{X}\rangle_F^2 + \|\Pi X - \widetilde{X}\|_F^2 \\ &= \|X - \Pi X\|_F^2 + 2\langle \Pi X - \Pi X, X - \widetilde{X}\rangle_F^2 + \|\Pi X - \widetilde{X}\|_F^2 \\ &= \|X - \Pi X\|_F^2 + \|\Pi X - \widetilde{X}\|_F^2 \geq \|X - \Pi X\|_F^2,\end{aligned}$$

and this result implies (5.3). □

Let us now turn our attention to the computation of the best approximation $\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} X$ for a given matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. According to Lemma 5.5, we only have to compute

$$S_b := V_t^* X W_s \quad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

since the nearfield blocks can be copied directly. This computation can be split into two parts: first we compute the intermediate matrix $\widehat{X}_{t,s} := W_s^* X^* \chi_t \in \mathbb{R}^{L_s \times \#\hat{t}}$, then we compute the result $S_b = V_t^* X W_s = V_t^* \chi_t X W_s = V_t^* (W_s^* X^* \chi_t^*)^* = V_t^* \widehat{X}_{t,s}^*$. Both steps of this procedure require us to multiply a matrix with the transposed of a cluster basis matrix from the left. The forward transformation Algorithm 6 introduced in Section 3.7 solves this problem by an elegant recursion, and it is straightforward to apply the same recursion to the task at hand.

For a given matrix X and a cluster $t \in \mathcal{T}_\mathcal{I}$, the *block forward transformation* Algorithm 10 computes the matrices $\widehat{X}_r := V_r^* X$ for all descendants $r \in \operatorname{sons}^*(t)$ of t.

Algorithm 10. Block forward transformation.

```
procedure BlockForwardTransformation(t, V, X, var X̂);
if sons(t) = ∅ then
  X̂_t ← V_t^* X
else
  X̂_t ← 0;
  for t' ∈ sons(t) do
    BlockForwardTransformation(t', V, X, X̂);
    X̂_t ← X̂_t + E_{t'}^* X̂_{t'}
  end for
end if
```

Lemma 5.6. *Let $t \in \mathcal{T}_\mathcal{I}$ and let $X \in \mathbb{R}^{\mathcal{I} \times P}$ for a finite index set P. Let $k = (k_t)_{t \in \mathcal{T}_\mathcal{I}}$ be defined as in (3.16). The computation of $\widehat{X}_r := V_r^* X$ for all $r \in \operatorname{sons}^*(t)$ by Algorithm 10 requires not more than*

$$2(\#P) \sum_{r \in \operatorname{sons}^*(t)} k_r (\#K_r) \leq 2(\#P) \sum_{r \in \operatorname{sons}^*(t)} k_r^2$$

operations.

Proof. By induction on $\#\operatorname{sons}^*(t) \in \mathbb{N}$. For $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) = 1$ we have $\operatorname{sons}(t) = \emptyset$, and Algorithm 10 computes the product of the adjoint of $V_t \in \mathbb{R}^{\mathcal{I} \times K_t}_{\hat{t}}$ and $X \in \mathbb{R}^{\mathcal{I} \times P}$ in not more than $2(\#\hat{t})(\#K_t)(\#P) \leq 2k_t(\#K_t)(\#P)$ operations.

Let now $n \in \mathbb{N}$ be such that the bound holds for all $\#\operatorname{sons}^*(t) \leq n$. Let $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}(t) = n + 1$. Then we have $\operatorname{sons}(t) \neq \emptyset$ and $\#\operatorname{sons}(t') \leq n$ for all sons $t' \in \operatorname{sons}(t)$, so the computation of $\widehat{X}_{t'}$ for all sons requires not more than

$$2(\#P) \sum_{r \in \operatorname{sons}^*(t')} k_r (\#K_r)$$

operations due to the induction assumption. Algorithm 10 multiplies each $\widehat{X}_{t'} \in \mathbb{R}^{K_t \times P}$ by the adjoint of the transfer matrix $E_{t'} \in \mathbb{R}^{K_{t'} \times K_t}$, and this takes not more than $2(\#K_{t'})(\#K_t)(\#P)$ operations per son $t' \in \operatorname{sons}(t)$ and

$$\sum_{t' \in \operatorname{sons}(t)} 2(\#K_{t'})(\#K_t)(\#P) \leq 2k_t(\#K_t)(\#P)$$

operations for all sons. Adding the bound for the subtrees corresponding to the sons yields a total of not more than

$$2k_t(\#K_t)(\#P) + 2(\#P) \sum_{r \in \operatorname{sons}^*(t')} k_r(\#K_r) = 2(\#P) \sum_{r \in \operatorname{sons}^*(t)} k_r(\#K_r)$$

operations. □

Algorithm 11 makes use of the block forward transformation to convert an arbitrary dense matrix into an $\mathcal{H}^2$-matrix by computing the coupling matrices for all farfield blocks: the block forward transformation applied to $(\chi_t X \chi_s)^*$ and the cluster basis W yields the matrices $\widehat{X}_r = W_r^* X \chi_t \in \mathbb{R}^{L_r \times \mathcal{I}}$ for all $r \in \operatorname{sons}^*(s)$, and applying the block forward transformation to the matrix $\widehat{X}_s$ and the cluster basis V then yields the matrix $\widehat{Y}_t = V_t^* \widehat{X}_s^* = V_t^* X W_s$, i.e., the desired coupling matrix (cf. Figure 5.1).

Algorithm 11. Convert a dense matrix into an $\mathcal{H}^2$-matrix.

```
procedure ConvertDense(M, V, W, var S);
for b = (t, s) ∈ L^+_{I×J} do
  BlockForwardTransformation(s, W, (χ_t X χ_s)^*, X̂);
  BlockForwardTransformation(t, V, X̂_s^*, Ŷ);
  S_b ← Ŷ_t
end for
```

Let us now consider the complexity of Algorithm 11. Since the block forward transformation of $\widehat{X}^*$ is performed by a sequence of operations for all $r \in \operatorname{sons}^*(t)$, each of which treats $\#L_s \leq \#\hat{s}$ units of data, we need a method for bounding sums of the type

$$\sum_{b=(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \sum_{r \in \operatorname{sons}^*(t)} \#\hat{s} = \sum_{t \in \mathcal{T}_\mathcal{I}} \sum_{s \in \operatorname{row}^+(t)} \sum_{r \in \operatorname{sons}^*(t)} \#\hat{s} = \sum_{r \in \mathcal{T}_\mathcal{I}} \sum_{t \in \operatorname{pred}(r)} \sum_{s \in \operatorname{row}^+(t)} \#\hat{s}.$$

Fortunately, we can find a simple upper bound for the two rightmost sums in this expression by using the following result:

Lemma 5.7 (Extended block rows). *For all $t \in \mathcal{T}_\mathcal{I}$, let*

$$\begin{aligned} \operatorname{row}^*(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t) &:= \{s \in \mathcal{T}_\mathcal{J} \ : \ \textit{there is a } t^+ \in \operatorname{pred}(t) \textit{ with } (t^+, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\} \\ &= \bigcup_{t^+ \in \operatorname{pred}(t)} \operatorname{row}^+(t^+). \end{aligned} \tag{5.4}$$

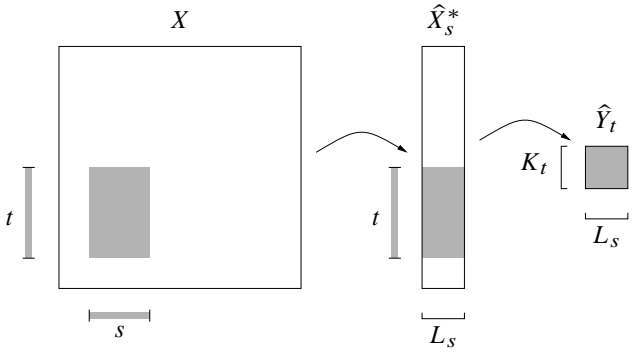


Figure 5.1. Conversion of a dense matrix by block forward transformations applied to columns and rows.

If this does not lead to ambiguity, we use $\operatorname{row}^*(t)$ *instead of* $\operatorname{row}^*(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$.

For $t \in \mathcal{T}_{\mathcal{I}}$ *and* $s_1, s_2 \in \operatorname{row}^*(t)$ *with* $s_1 \neq s_2$, *we have* $\hat{s}_1 \cap \hat{s}_2 = \emptyset$, *i.e., the index sets corresponding to the clusters in* $\operatorname{row}^*(t)$ *are pairwise disjoint.*

For $t \in \mathcal{T}_{\mathcal{I}}$ *and* $t^+ \in \operatorname{pred}(t)$, *we have* $\operatorname{row}^*(t^+) \subseteq \operatorname{row}^*(t)$.

Proof. We prove the first statement by contraposition. Let $t \in \mathcal{T}_{\mathcal{I}}$ and $s_1, s_2 \in \operatorname{row}^*(t)$ with $\hat{s}_1 \cap \hat{s}_2 \neq \emptyset$. By definition, there are $t_1^+, t_2^+ \in \operatorname{pred}(t)$ such that $s_1 \in \operatorname{row}(t_1^+)$ and $s_2 \in \operatorname{row}(t_2^+)$ hold. Without loss of generality, we assume $\operatorname{level}(t_1^+) \leq \operatorname{level}(t_2^+)$, which implies $t_2^+ \in \operatorname{sons}^*(t_1^+)$ due to Lemma 3.8 and therefore $\hat{t}_2^+ \subseteq \hat{t}_1^+$.

Let $j \in \hat{s}_1 \cap \hat{s}_2$ and $i \in \hat{t}_2^+ \subseteq \hat{t}_1^+$. Then we have $(i, j) \in (\hat{t}_1^+ \times \hat{s}_1) \cap (\hat{t}_2^+ \times \hat{s}_2)$, and since (t_1^+, s_1) and (t_2^+, s_2) are leaves of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, Corollary 3.9 implies $(t_1^+, s_1) = (t_2^+, s_2)$, and therefore $s_1 = s_2$ and $t_1^+ = t_s^+$.

The second statement is a simple consequence of the fact that $\operatorname{pred}(t) \subseteq \operatorname{pred}(t')$ holds for all $t' \in \operatorname{sons}(t)$. □

Due to this lemma, we find

$$\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+} \sum_{r\in\operatorname{sons}^*(t)} \#\hat{s} = \sum_{r\in\mathcal{T}_{\mathcal{I}}} \sum_{t\in\operatorname{pred}(r)} \sum_{s\in\operatorname{row}^+(t)} \#\hat{s} = \sum_{r\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\operatorname{row}^*(t)} \#\hat{s} \leq \sum_{r\in\mathcal{T}_{\mathcal{I}}} n_{\mathcal{J}}$$

and can now proceed to prove an upper bound for the complexity of Algorithm 11:

Lemma 5.8 (Conversion of dense matrices). *Let* V *and* W *be orthogonal nested cluster bases with rank distributions* K *and* L, *and let* $(k_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ *and* $(l_s)_{s\in\mathcal{T}_{\mathcal{J}}}$ *be defined as in* (3.16) *and* (3.18). *Let* $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. *The computation of* $\tilde{X} := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} X$ *by Algorithm 11 requires not more than*

$$2n_{\mathcal{J}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} k_t^2 + 2n_{\mathcal{I}} \sum_{s\in\mathcal{T}_{\mathcal{J}}} l_s^2$$

operations. If K, L are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}(n_{\mathcal{I}} n_{\mathcal{J}} (\alpha + \beta)^r)$.

Proof. Let $k = (k_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ and $l = (l_s)_{s \in \mathcal{T}_{\mathcal{J}}}$ be defined as in (3.16) and (3.18) for K and L, respectively. Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}$. According to Lemma 5.6, the computation of $\widehat{X}_s$ requires not more than

$$2(\#\hat{t}) \sum_{r \in \operatorname{sons}^*(s)} l_r^2$$

operations, and the computation of $\widehat{Y}_t$ requires not more than

$$2(\#L_s) \sum_{r \in \operatorname{sons}^*(t)} k_r^2 \leq 2(\#\hat{s}) \sum_{r \in \operatorname{sons}^*(t)} k_r^2$$

operations. Applying Lemma 5.7 to the first sum yields a bound of

$$\begin{aligned}
\sum_{b=(t,s) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}} 2(\#\hat{t}) \sum_{r \in \operatorname{sons}^*(s)} l_r^2 &= 2 \sum_{s \in \mathcal{T}_{\mathcal{J}}} \sum_{t \in \operatorname{col}^+(s)} \sum_{r \in \operatorname{sons}^*(s)} (\#\hat{t}) l_r^2 \\
&= 2 \sum_{r \in \mathcal{T}_{\mathcal{J}}} \sum_{s \in \operatorname{pred}(r)} \sum_{t \in \operatorname{col}^+(s)} (\#\hat{t}) l_r^2 \\
&= 2 \sum_{r \in \mathcal{T}_{\mathcal{J}}} l_r^2 \sum_{t \in \operatorname{col}^*(r)} \#\hat{t} \\
&\leq 2 \sum_{r \in \mathcal{T}_{\mathcal{J}}} l_r^2 n_{\mathcal{I}},
\end{aligned}$$

and by a similar argument we get the bound

$$\sum_{b=(t,s) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{J}}} 2(\#\hat{s}) \sum_{r \in \operatorname{sons}^*(t)} k_r^2 \leq 2 \sum_{r \in \mathcal{T}_{\mathcal{I}}} k_r^2 n_{\mathcal{J}}$$

for the second sum. This proves the estimate for the number of operations.

Using Lemma 3.45 and Lemma 3.48 completes the proof. □

We can conclude that the number of operations required to convert an arbitrary matrix into an $\mathcal{H}^2$-matrix grows quadratically with the matrix dimension. This is not surprising, since in this case the matrix is represented by $n_{\mathcal{I}} n_{\mathcal{J}}$ coefficients, and all of these coefficients have to be taken into account to compute the best approximation.

In order to reach a better than quadratic complexity, we have to assume that the input matrix is given in a more efficient representation. If the matrix $X \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ is given in the hierarchical matrix representation (3.12), we can take advantage of the factorized structure: Lemma 5.4 implies

$$V_t^* X W_s = V_t^* A_b B_b^* W_s = (V_t^* A_b)(W_s^* B_b)^*$$

for all $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. As in the case of the conversion of a dense matrix, we can use the block forward transformation Algorithm 10 to compute

$$\widehat{A}_b := V_t^* A_b, \quad \widehat{B}_b := W_s^* B_b$$

efficiently and then let $S_b := \widehat{A}_b \widehat{B}_b^*$. The result is Algorithm 12.

Algorithm 12. Convert an $\mathcal{H}$-matrix into an $\mathcal{H}^2$-matrix.

```
procedure ConvertH(A, B, V, W, var S);
for b = (t, s) ∈ L^+_{I×J} do
    BlockForwardTransformation(t, V, A_b, Â_b);
    BlockForwardTransformation(s, W, B_b, B̂_b);
    S_b ← X̂_t Ŷ_s^*
end for
```

Lemma 5.9 (Conversion of $\mathcal{H}$-matrices). *Let V and W be orthogonal nested cluster bases with rank distributions K and L, let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be a hierarchical matrix with local rank $k_\mathcal{H}$ given in the form (3.12). Let $p_{\mathcal{I}\times\mathcal{J}}$ be the depth of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. The computation of $\widetilde{X} := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} X$ by Algorithm 12 requires not more than*

$$3C_{\mathrm{sp}} k_\mathcal{H} (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^2 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_s^2 \Big)$$

operations. If K, L are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}(k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_\mathcal{I} + n_\mathcal{J})(\alpha + \beta)^r)$.

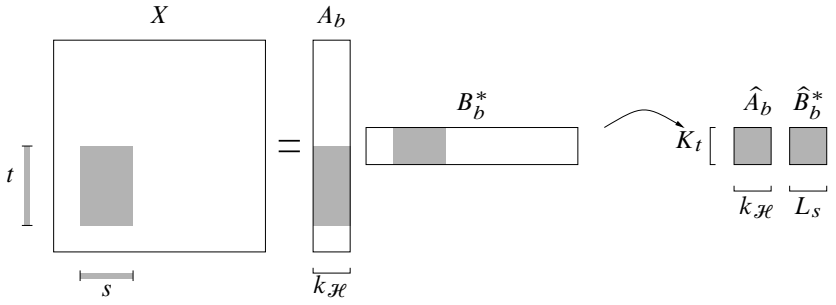


Figure 5.2. Conversion of a hierarchical matrix by block forward transformations applied to the factors A_b and B_b.

Proof. Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. According to Lemma 5.6, the computation of $\widehat{A}_b$ requires not more than

$$2k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(t)} k_r^2$$

operations, and the computation of $\widehat{B}_b$ can be accomplished in not more than

$$2k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(s)} l_r^2$$

operations. Using these auxiliary matrices, S_b can be constructed in not more than $2(\#K_t)(\#L_s)k_{\mathcal{H}} \leq 2k_t l_s k_{\mathcal{H}} \leq k_{\mathcal{H}}(k_t^2 + l_s^2)$ operations, and the handling of the block b requires not more than a total of

$$3k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(t)} k_r^2 + 3k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(s)} l_r^2$$

operations. Definition 3.12 implies

$$\begin{aligned}
&\operatorname{level}(t) \leq \operatorname{level}(b) \leq p_{\mathcal{I}\times\mathcal{J}},\\
&\operatorname{level}(s) \leq \operatorname{level}(b) \leq p_{\mathcal{I}\times\mathcal{J}} \qquad \text{for all } b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},
\end{aligned}$$

and summing over all blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ yields

$$\begin{aligned}
&\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} 3k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(t)} k_r^2\\
&\quad = 3k_{\mathcal{H}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{s\in\operatorname{row}^+(t)} \sum_{r\in\operatorname{sons}^*(t)} k_r^2\\
&\quad \leq 3C_{\mathrm{sp}}k_{\mathcal{H}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{r\in\operatorname{sons}^*(t)} k_r^2\\
&\quad = 3C_{\mathrm{sp}}k_{\mathcal{H}} \sum_{r\in\mathcal{T}_{\mathcal{I}}} \sum_{\substack{t\in\operatorname{pred}(r)\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} k_r^2\\
&\quad = 3C_{\mathrm{sp}}k_{\mathcal{H}} \sum_{r\in\mathcal{T}_{\mathcal{I}}} (\min\{\operatorname{level}(r), p_{\mathcal{I}\times\mathcal{J}}\} + 1)k_r^2\\
&\quad \leq 3C_{\mathrm{sp}}k_{\mathcal{H}}(p_{\mathcal{I}\times\mathcal{J}} + 1) \sum_{r\in\mathcal{T}_{\mathcal{I}}} k_r^2
\end{aligned}$$

for the first term and

$$\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} 3k_{\mathcal{H}} \sum_{r\in\operatorname{sons}^*(t)} l_r^2 \leq 3C_{\mathrm{sp}}k_{\mathcal{H}}(p_{\mathcal{I}\times\mathcal{J}} + 1) \sum_{r\in\mathcal{T}_{\mathcal{J}}} l_r^2$$

for the second. Using Lemma 3.45 and Lemma 3.48 completes the proof. □

As the storage complexity for a hierarchical matrix is $\mathcal{O}(k_{\mathcal{H}}(p_{\mathcal{I}\times\mathcal{J}}+1)(n_{\mathcal{I}}+n_{\mathcal{J}}))$, the number of operations required for the conversion into an $\mathcal{H}^2$-matrix is approximately proportional to the amount of storage required to represent the input data.

5.3 Cluster operators

We have seen that we can convert dense and hierarchical matrices into the $\mathcal{H}^2$-matrix representation efficiently once we have found suitable orthogonal cluster bases.

The efficient conversion of an $\mathcal{H}^2$-matrix X into an $\mathcal{H}^2$-matrix with different cluster bases is a slightly more challenging task: let V_X and W_X be nested cluster bases, let K_X and L_X be the corresponding rank distributions, and let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_X, W_X)$ be given in $\mathcal{H}^2$-matrix representation

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_{X,t} S_{X,b} W_{X,s}^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s$$

(cf. (3.17)) with coupling matrices $(S_{X,b})_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$. We are looking for an approximation of X in the space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Applying the same arguments as in the previous section yields that

$$S_b = V_t^* V_{X,t} S_{X,b} W_{X,s}^* W_s \tag{5.5}$$

is the best coefficient matrix for an admissible block $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. By introducing the auxiliary matrices

$$P_{V,t} := V_t^* V_{X,t} \in \mathbb{R}^{K_t\times K_{X,t}} \quad \text{and} \quad P_{W,s} := W_s^* W_{X,s} \in \mathbb{R}^{L_s\times L_{X,s}}, \tag{5.6}$$

we can write equation (5.5) in the form

$$S_b = P_{V,t} S_{X,b} P_{W,s}^*. \tag{5.7}$$

This expression is well-suited for efficient computations, since it only involves products of matrices with dimensions $\#K_t$,$\#K_{X,t}$,$\#L_s$ and $\#L_{X,s}$, which we can assume to be small.

Definition 5.10 (Cluster operator). Let $K_1 = (K_{1,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ and $K_2 = (K_{2,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ be rank distributions. Let $P = (P_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a family of matrices satisfying $P_t \in \mathbb{R}^{K_{1,t}\times K_{2,t}}$ for all $t \in \mathcal{T}_{\mathcal{I}}$. Then P is called a *cluster operator* for K_1 and K_2.

We can see that $P_V := (P_{V,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ is a cluster operator for K and K_X, while $P_W := (P_{W,s})_{s\in\mathcal{T}_{\mathcal{J}}}$ is one for L and L_X.

Equation (5.7) allows us to compute the best approximation of X in the space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ efficiently if P_V and P_W are given. Therefore we now investigate the general problem of constructing the matrices from (5.6) efficiently.

Definition 5.11 (Cluster basis product). Let $V_1 = (V_{1,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $V_2 = (V_{2,t})_{t\in\mathcal{T}_\mathcal{I}}$ be nested cluster bases with rank distributions $K_1 = (K_{1,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $K_2 = (K_{2,t})_{t\in\mathcal{T}_\mathcal{I}}$. The cluster operator $P = (P_t)_{t\in\mathcal{T}_\mathcal{I}}$ defined by

$$P_t := V_{1,t}^* V_{2,t}$$

for all $t \in \mathcal{T}_\mathcal{I}$ is called the *cluster basis product* of V_1 and V_2.

According to (5.6), P_V is the cluster basis product of V and V_X, and P_W is the cluster basis product of W and W_X.

Constructing the cluster basis product for V_1 and V_2 directly would mean computing $P_t = V_{1,t}^* V_{2,t}$, which requires $\mathcal{O}((\#K_{1,t})(\#K_{2,t})\#\hat{t})$ operations, for all $t \in \mathcal{T}_\mathcal{I}$, and the resulting algorithm would have non-optimal complexity. In order to reach the optimal complexity, we have to rely on the fact that V_1 and V_2 are nested: equation (3.15) implies

$$\begin{aligned} P_t = V_{1,t}^* V_{2,t} &= \Big(\sum_{t'\in\operatorname{sons}(t)} V_{1,t'} E_{1,t'}\Big)^* \Big(\sum_{t'\in\operatorname{sons}(t)} V_{2,t'} E_{2,t'}\Big) \\ &\overset{(5.2)}{=} \sum_{t'\in\operatorname{sons}(t)} E_{1,t'}^* V_{1,t'}^* V_{2,t'} E_{2,t'} = \sum_{t'\in\operatorname{sons}(t)} E_{1,t'}^* P_{t'} E_{2,t'} \end{aligned}$$

for all $t \in \mathcal{T}_\mathcal{I}$ with $\operatorname{sons}(t) \neq \emptyset$, therefore we can compute the entire family P by the recursive approach given in Algorithm 13.

Algorithm 13. Compute $P_t := V_{1,t}^* V_{2,t}$ for all $t \in \mathcal{T}_\mathcal{I}$.

```
procedure ClusterBasisProduct(t, V1, V2, var P);
if sons(t) = ∅ then
  P_t ← V_{1,t}^* V_{2,t}
else
  P_t ← 0;
  for t' ∈ sons(t) do
    ClusterBasisProduct(t', V1, V2, P);
    P_t ← P_t + E_{1,t'}^* P_{t'} E_{2,t'}
  end for
end if
```

The complexity analysis of Algorithm 13 involves the two rank distributions K_1 and K_2 for the same cluster tree. We simplify the proof by introducing an auxiliary rank distribution:

Lemma 5.12 (Maximum of rank distributions). *Let $\kappa \in \mathbb{N}$, and let $K_1, \ldots, K_\kappa$ be $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded rank distributions for the cluster tree $\mathcal{T}_\mathcal{I}$. The rank distribution $\hat{K} := (\hat{K}_t)_{t\in\mathcal{T}_\mathcal{I}}$ defined by*

$$\hat{K}_t := \{1, \ldots, \max\{\#K_{1,t}, \ldots, \#K_{\kappa,t}\}\} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

is $(C_{\mathrm{bn}}\kappa, \alpha, \beta, r, \xi)$*-bounded and satisfies* $\#\hat{K}_t = \max\{\#K_{\iota,t} : \iota \in \{1, \ldots, \kappa\}\}$. *We call* $\hat{K}$ *the* ***maximum rank distribution*** *for* $K_1, \ldots, K_\kappa$.

Proof. In order to prove that $\hat{K}$ is $(C_{\mathrm{bn}}\kappa, \alpha, \beta, r, \xi)$-bounded, we have to investigate the cardinality of the sets

$$\mathcal{D}_\ell := \{t \in \mathcal{T}_\mathcal{I} \ : \ \#\hat{K}_t > (\alpha + \beta(\ell - 1))^r\}$$

for all $\ell \in \mathbb{N}_0$. We introduce

$$\mathcal{D}_{\iota,\ell} := \{t \in \mathcal{T}_\mathcal{I} \ : \ \#K_{\iota,t} > (\alpha + \beta(\ell - 1))^r\}$$

for all $\iota \in \{1, \ldots, \kappa\}$ and all $\ell \in \mathbb{N}_0$. Let $t \in \mathcal{T}_\mathcal{I}$. If $t \in \mathcal{D}_\ell$, we have

$$\max\{\#K_{\iota,t} \ : \ \iota \in \{1, \ldots, \kappa\}\} = \#\hat{K}_t > (\alpha + \beta(\ell - 1))^r,$$

and so we can find at least one $\iota \in \{1, \ldots, \kappa\}$ with $\#K_{\iota,t} > (\alpha + \beta(\ell - 1))^r$, i.e., $t \in \mathcal{D}_{\iota,\ell}$. We have proven

$$\mathcal{D}_\ell \subseteq \bigcup_{\iota=1}^{\kappa} \mathcal{D}_{\iota,\ell}$$

and therefore also

$$\#\mathcal{D}_\ell \leq \sum_{\iota=1}^{\kappa} \#\mathcal{D}_{\iota,\ell} \leq \sum_{\iota=1}^{\kappa} C_{\mathrm{bn}}\xi^{-\ell} c_\mathcal{I} = C_{\mathrm{bn}}\kappa\xi^{-\ell} c_\mathcal{I}. \qquad \square$$

The maximum rank distribution allows us to apply the crucial Lemma 3.45 also in situations involving multiple cluster bases: since $\#K_{1,t}$ and $\#K_{2,t}$ can be bounded by $\#\hat{K}_t$ for all $t \in \mathcal{T}_\mathcal{I}$, we also have $k_{1,t}, k_{2,t} \leq \hat{k}_t$ for all $t \in \mathcal{T}_\mathcal{I}$, where $\hat{k}_t$ is defined as in (3.16) for $\hat{K}$, and therefore can express complexity estimates in terms of $(\hat{k}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and use Lemma 3.45.

Lemma 5.13 (Cluster basis product). *Let* $V_1 = (V_{1,t})_{t\in\mathcal{T}_\mathcal{I}}$ *and* $V_2 = (V_{2,t})_{t\in\mathcal{T}_\mathcal{I}}$ *be nested cluster bases with rank distributions* $K_1 = (K_{1,t})_{t\in\mathcal{T}_\mathcal{I}}$ *and* $K_2 = (K_{2,t})_{t\in\mathcal{T}_\mathcal{I}}$. *Let* $(k_{1,t})_{t\in\mathcal{T}_\mathcal{I}}$ *and* $(k_{2,t})_{t\in\mathcal{T}_\mathcal{I}}$ *be defined as in* (3.16). *The computation of the cluster basis product* $P = (P_t)_{t\in\mathcal{T}_\mathcal{I}}$ *by Algorithm* 13 *requires not more than*

$$2\sum_{t\in\mathcal{T}_\mathcal{I}} (k_{1,t}^3 + k_{2,t}^3)$$

operations. If K_1 *and* K_2 *are* $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$*-bounded and if* $\mathcal{T}_\mathcal{I}$ *is* $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$*-regular, the number of operations is in* $\mathcal{O}(n_\mathcal{I}(\alpha + \beta)^{2r})$.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. If t is a leaf, the algorithm multiplies the adjoint of $V_{1,t} \in \mathbb{R}^{\mathcal{I}\times K_{t,1}}_{\hat{t}}$ and $V_{2,t} \in \mathbb{R}^{\mathcal{I}\times K_{t,2}}_{\hat{t}}$, and this requires not more than $2(\#\hat{t})(\#K_{1,t})(\#K_{2,t})$ operations.

If t is not a leaf, there are two ways of computing the product $E_{1,t'}^* P_{t'} E_{2,t'}$: we can first compute $\widehat{P} = P_{t'} E_{2,t'}$ and then $E_{1,t'}^* \widehat{P}$, or we can start with $\widehat{P} = E_{1,t'}^* P_{t'}$ and then compute $\widehat{P} E_{2,t'}$. We consider only the first case. Finding $\widehat{P}$ requires at most $2(\#K_{1,t'})(\#K_{2,t'})(\#K_{2,t})$ operations, the second product takes at most $2(\#K_{1,t})(\#K_{1,t'})(\#K_{2,t})$ operations, and the total number of operations for this cluster is bounded by

$$\begin{aligned}
&\sum_{t' \in \operatorname{sons}(t)} 2(\#K_{1,t'})(\#K_{2,t'})(\#K_{2,t}) + 2(\#K_{1,t})(\#K_{1,t'})(\#K_{2,t}) \\
&\quad \leq \Big(\sum_{t' \in \operatorname{sons}(t)} (\#K_{1,t'})^2 + \sum_{t' \in \operatorname{sons}(t)} (\#K_{2,t'})^2 \Big) \#K_{2,t} + 2(\#K_{1,t}) k_{1,t} (\#K_{2,t}) \\
&\quad \leq \Big(\Big(\sum_{t' \in \operatorname{sons}(t)} \#K_{1,t'} \Big)^2 + \Big(\sum_{t' \in \operatorname{sons}(t)} \#K_{2,t'} \Big)^2 \Big) \#K_{2,t} + 2(\#K_{1,t}) k_{1,t} (\#K_{2,t}) \\
&\quad \leq k_{1,t}^2 k_{2,t} + k_{2,t}^2 k_{2,t} + 2 k_{1,t}^2 k_{2,t} = (3k_{1,t}^2 + k_{2,t}^2) k_{2,t} \leq 2(k_{1,t}^3 + k_{2,t}^3).
\end{aligned}$$

In the last step, we have used the inequality

$$\begin{aligned}
3x^2 y &= 2x(2xy) - x^2 y \leq 2x(x^2 + y^2) - x^2 y = 2x^3 + y(2xy) - x^2 y \\
&\leq 2x^3 + y(x^2 + y^2) - x^2 y = 2x^3 + y^3 \quad \text{for all } x, y \in \mathbb{R}_{\geq 0}.
\end{aligned} \tag{5.8}$$

Let now K_1 and K_2 be $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and let $\mathcal{T}_I$ be $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular. By Lemma 5.12, we can find a rank distribution $(\widehat{K}_t)_{t \in \mathcal{T}_I}$ with $\#\widehat{k}_t \geq \max\{k_{1,t}, k_{2,t}\}$ that is $(2C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded. Our upper bound takes the form

$$2 \sum_{t \in \mathcal{T}_I} (k_{1,t}^3 + k_{2,t}^3) \leq 4 \sum_{t \in \mathcal{T}_I} \hat{k}_t^3,$$

and we can apply Lemma 3.45 and Lemma 3.48 again to complete the proof. □

Algorithm 14. Convert an $\mathcal{H}^2$-matrix into an $\mathcal{H}^2$-matrix.

```
procedure ConvertH2(S_X, V, W, var S);
ClusterBasisProduct(root(T_I), V, V_X, P_V);
ClusterBasisProduct(root(T_J), W, W_X, P_W);
for b = (t, s) ∈ L^+_{I×J} do
  S_b ← P_{V,t} S_{X,b} P^*_{W,s}
end for
```

Lemma 5.14 (Conversion of $\mathcal{H}^2$-matrices). *Let V and W be nested orthogonal cluster bases with rank distributions K and L, let V_X and W_X be nested cluster bases with rank distributions K_X and L_X. Let $X \in \mathcal{H}^2(\mathcal{T}_{I \times J}, V_X, W_X)$ be a matrix in $\mathcal{H}^2$-matrix*

representation. Let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$, $(k_{X,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$ *and* $(l_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ *be defined as in* (3.16) *and* (3.18)*. Algorithm* 14 *requires not more than*

$$2(C_{\mathrm{sp}}+2)\sum_{t\in\mathcal{T}_\mathcal{I}}\max\{k_t^3,k_{X,t}^3\}+2(C_{\mathrm{sp}}+2)\sum_{s\in\mathcal{T}_\mathcal{J}}\max\{l_s^3,l_{X,s}^3\}$$

operations. If K, K_X, L *and* L_X *are* $(C_{\mathrm{bn}},\alpha,\beta,r,\xi)$*-bounded and if* $\mathcal{T}_\mathcal{I}$ *and* $\mathcal{T}_\mathcal{J}$ *are* $(C_{\mathrm{rc}},\alpha,\beta,r,\xi)$*-regular, the number of operations is in* $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I}+n_\mathcal{J}))$.

Proof. Let $\hat{K}$ be the maximum rank distribution of K and K_X, and let $\hat{L}$ be the maximum rank distribution of L and L_X. Let $(\hat{k}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(\hat{l}_s)_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18) for $\hat{K}$ and $\hat{L}$, respectively. This implies $\hat{k}_t=\max\{k_t,k_{X,t}\}$ and $\hat{l}_s=\max\{l_s,l_{X,s}\}$ for all $t\in\mathcal{T}_\mathcal{I}$ and $s\in\mathcal{T}_\mathcal{J}$.

Let $b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. We can compute S_b in two ways depending on whether we perform the multiplication by $P^*_{W,s}$ or $P_{V,t}$ first. Here we restrict the investigation to the first case. Computing $S_{X,b}P^*_{W,s}$ requires $2(\#K_{X,t})(\#L_{X,s})(\#L_s)\leq 2k_{X,t}l_{X,s}l_s$ operations, and subsequently computing the matrix $S_b=P_{V,t}(S_{M,b}P^*_{W,s})$ requires $2(\#K_t)(\#K_{X,t})(\#L_s)\leq 2k_tk_{X,t}l_s$ operations. This means that the number of operations for one admissible block can be bounded by

$$2k_{X,t}l_{X,s}l_s+2k_tk_{X,t}l_s\leq 2\hat{k}_t\hat{l}_s^2+2\hat{k}_t^2\hat{l}_s.$$

Due to (5.8), we have

$$x^2y+xy^2\leq\frac{1}{3}(2x^3+y^3+2y^3+x^3)=x^3+y^3\qquad\text{for all } x,y\in\mathbb{R}_{\geq 0}\tag{5.9}$$

and can bound the number of operations for one admissible block by

$$2(\hat{k}_t\hat{l}_s^2+\hat{k}_t^2\hat{l}_s)\leq 2(\hat{k}_t^3+\hat{l}_s^3).\tag{5.10}$$

Once more we rely on C_{sp}-sparsity of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ in order to get the bound

$$\begin{aligned}\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}2(\hat{k}_t^3+\hat{l}_s^3)&=2\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{s\in\mathrm{row}^+(t)}\hat{k}_t^3+2\sum_{s\in\mathcal{T}_\mathcal{J}}\sum_{t\in\mathrm{col}^+(s)}\hat{l}_s^3\\&\leq 2C_{\mathrm{sp}}\sum_{t\in\mathcal{T}_\mathcal{I}}\hat{k}_t^3+2C_{\mathrm{sp}}\sum_{s\in\mathcal{T}_\mathcal{J}}\hat{l}_s^3\end{aligned}$$

for the number of operations.

Due to Lemma 5.13, the preparation of $(P_{V,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(P_{W,s})_{s\in\mathcal{T}_\mathcal{J}}$ requires not more than

$$4\sum_{t\in\mathcal{T}_\mathcal{I}}\hat{k}_t^3+4\sum_{s\in\mathcal{T}_\mathcal{J}}\hat{l}_s^3$$

operations, and adding both estimates yields our result.

Let now K, K_X, L and L_X be $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded, and let $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ be $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular. Then $\widehat{K}$ and $\widehat{L}$ are $(2C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded by Lemma 5.12, and we can once again apply the Lemmas 3.45 and 3.48 to complete the proof. □

Remark 5.15 (Conversion of submatrices). In many applications the cluster operators $(P_{V,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(P_{W,s})_{s\in\mathcal{T}_{\mathcal{J}}}$ are given and therefore do not have to be computed by Algorithm 13. In this situation, (5.10) yields the bound

$$2 \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} (\hat{k}_t^3 + \hat{l}_s^3) \leq 2C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} \max\{k_t^3, k_{X,t}^3\} + 2C_{\mathrm{sp}} \sum_{s\in\mathcal{T}_{\mathcal{J}}} \max\{l_s^3, l_{X,s}^3\}$$

for the number of operations of the projection into the $\mathcal{H}^2$-matrix space corresponding to the cluster bases V and W. □

5.4 Orthogonalization

Obviously, orthogonality is a very desirable property of a cluster basis: it ensures that no redundant columns are present and it facilitates the computation of matrix projections. We therefore now investigate an algorithm for constructing orthogonal cluster bases from general ones. If we do not require the resulting cluster basis to be nested, this is a straightforward task: we can simply compute the Householder factorizations $Q_t R_t$ of the cluster basis matrices V_t and use the orthogonal matrices Q_t to construct the desired cluster basis.

Lemma 5.16 (Householder factorization). *Let $\mathcal{I}' \subseteq \mathcal{I}$. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}'}$. Let $p := \min\{\#\mathcal{I}', \#\mathcal{J}\}$. There are an index set $\mathcal{K} \subseteq \mathcal{I}'$ with $\#\mathcal{K} = p$, an orthogonal matrix $Q \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ and a matrix $R \in \mathbb{R}^{\mathcal{K}\times\mathcal{J}}$ satisfying*

$$X = QR.$$

Proof. Let $m := \#\mathcal{I}'$ and $n := \#\mathcal{J}$. Let $\iota_m : \{1, \ldots, m\} \to \mathcal{I}'$ and $\iota_n : \{1, \ldots, n\} \to \mathcal{J}$ be arbitrary bijective mappings. Let $\widehat{X} \in \mathbb{R}^{m\times n}$ be defined by

$$(\widehat{X})_{\mu\nu} = X_{\iota_m(\mu),\iota_n(\nu)}$$

for all $\mu \in \{1, \ldots, m\}$ and all $\nu \in \{1, \ldots, n\}$.

We apply $p := \min\{m, n\}$ Householder transformations [48], Section 5.2, to the matrix $\widehat{X}$ in order to find an orthogonal matrix $\widehat{Q} \in \mathbb{R}^{m\times m}$ and an upper triangular matrix $\widehat{R} \in \mathbb{R}^{m\times n}$ with $\widehat{X} = \widehat{Q}\widehat{R}$. Since $\widehat{R}$ is upper triangular, its rows $p + 1, \ldots, m$ are zero.

We let $\mathcal{K} := \iota_m(\{1, \ldots, p\}) \subseteq \mathcal{I}'$ and define $Q \in \mathbb{R}^{\mathcal{I} \times \mathcal{K}}_{\mathcal{I}'}$ and $R \in \mathbb{R}^{\mathcal{K} \times \mathcal{J}}$ by

$$Q_{ik} = \begin{cases} \widehat{Q}_{\iota_m^{-1}(i), \iota_m^{-1}(k)} & \text{if } i \in \mathcal{I}', \\ 0 & \text{otherwise,} \end{cases} \qquad R_{kj} = \widehat{R}_{\iota_m^{-1}(k), \iota_n^{-1}(j)}$$

for $i \in \mathcal{I}$, $k \in \mathcal{K}$ and $j \in \mathcal{J}$.

Let $i \in \mathcal{I}$ and $j \in \mathcal{J}$. If $i \notin \mathcal{I}'$, we obviously have $(QR)_{ij} = 0 = M_{ij}$. If $i \in \mathcal{I}'$, we find

$$(QR)_{ij} = \sum_{k \in \mathcal{K}} Q_{ik} R_{kj} = \sum_{k=1}^{p} \widehat{Q}_{\iota_m^{-1}(i), \iota_m^{-1}(k)} \widehat{R}_{\iota_m^{-1}(k), \iota_n^{-1}(j)} = \widehat{X}_{\iota_m^{-1}(i), \iota_n^{-1}(j)} = X_{ij}.$$

□

Lemma 5.17. *There is a constant $C_{\mathrm{qr}} \in \mathbb{N}$ such that Algorithm 15 requires not more than $C_{\mathrm{qr}}(\#\mathcal{I}')n_{\mathcal{J}} \min\{\#\mathcal{I}', \#\mathcal{J}\}$ operations.*

Proof. See Section 5.2 in [48]. □

Algorithm 15. Householder factorization $QR = X$ of a matrix $X \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}_{\mathcal{I}'}$ with an orthogonal matrix $Q \in \mathbb{R}^{\mathcal{I} \times \mathcal{K}}_{\mathcal{I}'}$ and a weight matrix $R \in \mathbb{R}^{\mathcal{K} \times \mathcal{J}}$.

---

procedure Householder(X, $\mathcal{I}'$, **var** Q, R, $\mathcal{K}$)
$m \leftarrow \#\mathcal{I}'$; $n \leftarrow \#\mathcal{J}$; $p \leftarrow \min\{m, n\}$;
Fix arbitrary isomorphisms $\iota_m : \{1, \ldots, m\} \to \mathcal{I}'$ and $\iota_n : \{1, \ldots, n\} \to \mathcal{J}$;
$\widehat{X} \leftarrow 0 \in \mathbb{R}^{m \times n}$
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ **do**
  $\widehat{X}_{ij} \leftarrow X_{\iota_m(i), \iota_n(j)}$
end for;
Compute an orthogonal matrix $\widehat{Q} \in \mathbb{R}^{m \times m}$ and an upper triangular matrix
  $\widehat{R} \in \mathbb{R}^{m \times n}$ with $\widehat{X} \leftarrow \widehat{Q}\widehat{R}$;
$\mathcal{K} \leftarrow \iota_m(\{1, \ldots, p\})$;
$Q \leftarrow 0 \in \mathbb{R}^{\mathcal{I} \times \mathcal{K}}_{\mathcal{I}'}$; $R \leftarrow 0 \in \mathbb{R}^{\mathcal{K} \times \mathcal{J}}$;
for $i \in \{1, \ldots, m\}, k \in \{1, \ldots, p\}$ **do**
  $Q_{\iota_m(i), \iota_m(k)} \leftarrow \widehat{Q}_{ik}$
end for;
for $k \in \{1, \ldots, p\}$, $j \in \{1, \ldots, n\}$ **do**
  $R_{\iota_m(k), \iota_n(j)} \leftarrow \widehat{R}_{kj}$
end for

Using this lemma and the corresponding Algorithm 15, constructing an orthogonal cluster basis $Q = (Q_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ with rank distribution $L = (L_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ satisfying $\#L_t \leq \#K_t$ and $\operatorname{range}(V_t) \subseteq \operatorname{range}(Q_t)$ for all $t \in \mathcal{T}_{\mathcal{I}}$ is straightforward: let $t \in \mathcal{T}_{\mathcal{I}}$. Let

$l_t := \min\{\#\hat{t}, k_t\}$. Using Lemma 5.16, we can find an index set L_t with $\#L_t = l_t$, an orthogonal matrix $Q_t \in \mathbb{R}^{\mathcal{I}\times L_t}_{\hat{t}}$ and a matrix $R_t \in \mathbb{R}^{L_t\times K_t}$ satisfying $V_t = Q_t R_t$.

The family $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ constructed in this way is an orthogonal cluster basis with rank distribution $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$, and due to

$$V_t = Q_t R_t = Q_t(Q_t^* Q_t)R_t = Q_t Q_t^* V_t, \tag{5.11}$$

it can be used to express anything that can be expressed by V_t, and $R = (R_t)_{t\in\mathcal{T}_\mathcal{I}}$ is the cluster operator describing the change of basis from V to Q.

We have already seen that only *nested* cluster bases allow us to reach the optimal efficiency, therefore we are interested in finding an algorithm that turns a *nested* cluster basis V into an orthogonal *nested* cluster basis Q. For leaf clusters, we can still use the Householder factorization directly, but for the remaining clusters we have to find a way to work with transfer matrices instead of V_t and Q_t.

Let $t \in \mathcal{T}_\mathcal{I}$ with $\tau := \#\operatorname{sons}(t) > 0$. We denote the sons of t by $\{t_1, \ldots, t_\tau\} = \operatorname{sons}(t)$. We assume that the matrices $Q_{t'} \in \mathbb{R}^{\mathcal{I}\times L_{t'}}_{\hat{t}'}$, index sets $L_{t'}$ and cluster operator matrices $R_{t'} \in \mathbb{R}^{L_{t'}\times K_{t'}}$ with $V_{t'} = Q_{t'} R_{r'}$ have been constructed for all $t' \in \operatorname{sons}(t)$. Without loss of generality, we can also assume that the index sets $\{L_{t'} : t' \in \operatorname{sons}(t)\}$ are pairwise disjoint. Using the nested structure of V, this implies

$$V_t = \sum_{t'\in\operatorname{sons}(t)} V_{t'} E_{t'} = \sum_{t'\in\operatorname{sons}(t)} Q_{t'} R_{t'} E_{t'} = \begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} R_{t_1} E_{t_1} \\ \vdots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix}.$$

We introduce the auxiliary matrices

$$U_t := \begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix} \in \mathbb{R}^{\mathcal{I}\times M_t}_{\hat{t}} \quad \text{and} \quad \hat{V}_t := \begin{pmatrix} R_{t_1} E_{t_1} \\ \vdots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix} \in \mathbb{R}^{M_t\times K_t} \tag{5.12}$$

for the new index set

$$M_t := \bigcup_{t'\in\operatorname{sons}(t)} L_{t'}.$$

The matrix $\hat{V}_t$ contains the coefficients required to express the matrix V_t in terms of the cluster basis matrices $Q_{t_1}, \ldots, Q_{t_\tau}$, and U_t simply describes the transformation from these coefficients back to $\mathbb{R}^\mathcal{I}$, so that we have $V_t = U_t \hat{V}_t$ (cf. Figure 5.3).

Due to Lemma 5.2, we find

$$U_t^* U_t = \begin{pmatrix} Q_{t_1}^* \\ \vdots \\ Q_{t_\tau}^* \end{pmatrix} \begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix} = \begin{pmatrix} Q_{t_1}^* Q_{t_1} & \ldots & Q_{t_1}^* Q_{t_\tau} \\ \vdots & \ddots & \vdots \\ Q_{t_\tau}^* Q_{t_1} & \ldots & Q_{t_\tau}^* Q_{t_\tau} \end{pmatrix} = I$$

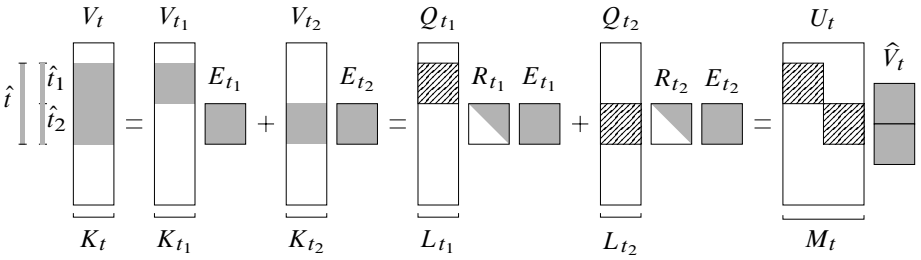


Figure 5.3. Construction of the representation $V_t = U_t \hat{V}_t$ used in the orthogonalization algorithm.

and conclude that U_t is an orthogonal matrix. The coefficient matrix $\hat{V}_t$ has only

$$\#M_t = \# \bigcup_{t' \in \operatorname{sons}(t)} L_{t'} = \sum_{t' \in \operatorname{sons}(t)} \#L_{t'}$$

rows and $\#K_t$ columns, therefore we can compute its Householder decomposition

$$\hat{V}_t = \hat{Q}_t R_t$$

efficiently using Algorithm 15, where $\hat{Q}_t \in \mathbb{R}^{M_t \times L_t}$ is an orthogonal matrix. Since U_t and $\hat{Q}_t$ are orthogonal, so is $Q_t := U_t \hat{Q}_t$, and we observe

$$V_t = U_t \hat{V}_t = U_t \hat{Q}_t R_t = Q_t R_t,$$

i.e., we have found a Householder decomposition of V_t.

For all $t' \in \operatorname{sons}(t)$, we let $F_{t'} := \hat{Q}_t|_{L_{t'} \times L_t}$, observe

$$\hat{Q}_t = \begin{pmatrix} F_{t_1} \\ \vdots \\ F_{t_\tau} \end{pmatrix} \tag{5.13}$$

and conclude

$$Q_t = U_t \hat{Q}_t = \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} F_{t_1} \\ \vdots \\ F_{t_\tau} \end{pmatrix} = \sum_{t' \in \operatorname{sons}(t)} Q_{t'} F_{t'}, \tag{5.14}$$

therefore the new orthogonal cluster basis is nested indeed. This construction is summarized in Algorithm 16.

Lemma 5.18 (Complexity of orthogonalization). *Let V be a nested cluster basis with rank distribution K, and let $(k_t)_{t \in \mathcal{T}_\mathcal{I}}$ be defined as in (3.16). Algorithm 16 requires not more than*

$$(C_{\mathrm{qr}} + 2) \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2 \#K_t \leq (C_{\mathrm{qr}} + 2) \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^3$$

operations. If K is $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r} n_{\mathcal{I}})$.

Algorithm 16. Orthogonalize a cluster basis.

```
procedure Orthogonalize(t, V, var Q, R, L)
if sons(t) = ∅ then
  Householder(V_t, t̂, Q_t, R_t, L_t)                    {Algorithm 15}
else
  M_t ← ∅;
  for t' ∈ sons(t) do
    Orthogonalize(t', V, Q, R, L);
    M_t ← M_t ∪̇ L_t'
  end for;
  V̂_t ← 0 ∈ R^{M_t × K_t};
  for t' ∈ sons(t) do
    V̂_t|_{L_t' × K_t} ← R_t' E_t'
  end for;
  Householder(V̂_t, M_t, Q̂_t, R_t, L_t);                {Algorithm 15}
  for t' ∈ sons(t) do
    F_t' ← Q̂_t|_{L_t' × L_t}
  end for
end if
```

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. If t is a leaf, we apply Algorithm 15 to the matrix $V_t \in \mathbb{R}^{\mathcal{I} \times K_t}_{\hat{t}}$, and according to Lemma 5.17, this takes not more than

$$C_{\mathrm{qr}}(\#\hat{t})(\#K_t)\min\{\#\hat{t}, \#K_t\} \leq C_{\mathrm{qr}}(\#\hat{t})(\#K_t)^2 \leq C_{\mathrm{qr}} k_t^2 \#K_t$$

operations. If t is not a leaf, we have to compute $R_{t'}E_{t'}$ for all $t' \in \mathrm{sons}(t)$, using not more than

$$2(\#L_{t'})(\#K_{t'})(\#K_t) \leq 2(\#K_{t'})^2(\#K_t)$$

operations per son and

$$2(\#K_t)\sum_{t' \in \mathrm{sons}(t)} (\#K_{t'})^2 \leq 2(\#K_t)\Big(\sum_{t' \in \mathrm{sons}(t)} \#K_{t'}\Big)^2 \leq 2(\#K_t)k_t^2$$

for all sons. Due to $\#L_{t'} \leq \#K_{t'}$, we have

$$\#M_t = \sum_{t' \in \mathrm{sons}(t)} \#L_{t'} \leq \sum_{t' \in \mathrm{sons}(t)} \#K_{t'} \leq k_t, \tag{5.15}$$

and so applying Algorithm 15 to $\hat{V}_t \in \mathbb{R}^{M_t \times K_t}$ takes not more than

$$C_{\mathrm{qr}} k_t (\#K_t)\min\{k_t, \#K_t\} \leq C_{\mathrm{qr}} k_t (\#K_t)^2 \leq C_{\mathrm{qr}} k_t^2 \#K_t$$

operations due to Lemma 5.17. We conclude that the number of operations for one cluster $t \in \mathcal{T}_{\mathcal{I}}$ is bounded by $(C_{\mathrm{qr}}+2)k_t^2 \# K_t$, and summing over all clusters completes the proof.

If K is $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, we can again use the Lemmas 3.45 and 3.48 to bound the asymptotic complexity. □

5.5 Truncation

The orthogonalization Algorithm 16 ensures that Q_t is orthogonal and satisfies $V_t = Q_t R_t$. By construction, the rank of Q_t is always equal to $\min\{k_t, \#K_t\}$, in particular it can be significantly larger than the rank of V_t.

Since the rank distribution determines the algorithmic complexity of most algorithms, we would like to reduce the rank as far as possible, i.e., to modify Algorithm 16 in such a way that it ensures $\#L_t \leq \operatorname{rank}(V_t)$.

In practical algorithms, computing the true rank of V_t can be impossible due to rounding errors, therefore we prefer to look for an orthogonal cluster basis Q with rank distribution L which can be used to approximate the original cluster basis $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$, i.e., which satisfies

$$\min\{\|V_t - Q_t Z_t\|_2 \,:\, Z_t \in \mathbb{R}^{L_t\times K_t}\} = \|V_t - Q_t Q_t^* V_t\|_2 \leq \epsilon \tag{5.16}$$

for a given tolerance $\epsilon \in \mathbb{R}_{>0}$ and all $t \in \mathcal{T}_{\mathcal{I}}$. We call the process of constructing an orthogonal cluster basis which approximates the original basis up to a certain error *truncation* and refer to the new basis Q satisfying a condition of the type (5.16) as the *truncated cluster basis*.

In order to reach optimal efficiency, we have to look for approximations of low rank. Approximations of this type can be constructed by the singular value decomposition:

Lemma 5.19 (Singular value decomposition). *Let $\mathcal{I}' \subseteq \mathcal{I}$. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}'}$. Let $p := \operatorname{rank}(X) \leq \min\{\#\mathcal{I}', \#\mathcal{J}\}$ with $p > 0$. Let $\sigma_1 \geq \cdots \geq \sigma_p > 0$ be the non-zero singular values of X. For each $l \in \{1, \ldots, p\}$, we can find an index set $\mathcal{K} \subseteq \mathcal{I}'$ with $\#\mathcal{K} = l$, an orthogonal matrix $Q \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ and a matrix $R \in \mathbb{R}^{\mathcal{K}\times\mathcal{J}}$ which satisfy*

$$\|X - QR\|_2 = \begin{cases} \sigma_{l+1} & \text{if } l < p, \\ 0 & \text{otherwise}, \end{cases} \tag{5.17a}$$

$$\|X - QR\|_F = \begin{cases} \left(\sum_{i=l+1}^{p} \sigma_i^2\right)^{1/2} & \text{if } l < p, \\ 0 & \text{otherwise}. \end{cases} \tag{5.17b}$$

The orthogonality of Q implies $R = Q^* X$.

Proof. Let $m := \#\mathcal{I}'$ and $n := \#\mathcal{J}$. Let ι_m, ι_n and $\widehat{X}$ be defined as in the proof of Lemma 5.16.

Due to Theorem 2.5.2 in [48], there is a singular value decomposition of $\hat{X}$, i.e., there are orthogonal matrices $\hat{U} \in \mathbb{R}^{m\times m}$ and $\hat{V} \in \mathbb{R}^{n\times n}$ such that

$$\hat{U}^* \hat{X} \hat{V} = \operatorname{diag}(\sigma_1, \ldots, \sigma_q) \in \mathbb{R}^{m\times n}$$

holds for the family $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q \geq 0$, $q := \min\{m, n\}$ of singular values. Since $\hat{U}$ and $\hat{V}$ are invertible, we have $p \leq q$ and $\sigma_{p+1} = \cdots = \sigma_q = 0$.

Let now $l \in \{1, \ldots, p\}$. We define the matrix $\hat{Q} \in \mathbb{R}^{m\times l}$ by setting $\hat{Q} := \hat{U}|_{m\times l}$, i.e., by using the first l columns of $\hat{U}$ to define the columns of $\hat{Q}$. Since $\hat{U}$ is orthogonal, the same holds for $\hat{Q}$.

The definition of $\hat{Q}$ implies

$$\begin{aligned}\hat{Q}^* \hat{X} &= (\hat{U}^* \hat{X})|_{l\times n} = (\hat{U}^* \hat{X} \hat{V})|_{l\times n} \hat{V}^* = \operatorname{diag}_{m\times n}(\sigma_1, \ldots, \sigma_q)|_{l\times n} \hat{V}^* \\ &= \operatorname{diag}_{l\times n}(\sigma_1, \ldots, \sigma_l) \hat{V}^*\end{aligned}$$

and therefore

$$\hat{Q} \hat{Q}^* \hat{X} = \hat{U} \operatorname{diag}_{m\times n}(\sigma_1, \ldots, \sigma_l, 0, \ldots, 0) \hat{V}^*.$$

Since $\hat{U}$ and $\hat{V}$ are orthogonal matrices, we find

$$\begin{aligned}&\|\hat{X} - \hat{Q} \hat{Q}^* \hat{X}\|_2 \\ &\quad = \|\hat{U} \operatorname{diag}_{n\times k}(\sigma_1, \ldots, \sigma_q) \hat{V}^* - \hat{U} \operatorname{diag}_{n\times k}(\sigma_1, \ldots, \sigma_l, 0, \ldots, 0) \hat{V}^*\|_2 \\ &\quad = \|\operatorname{diag}_{n\times k}(\sigma_1, \ldots, \sigma_q) - \operatorname{diag}_{n\times k}(\sigma_1, \ldots, \sigma_l, 0, \ldots, 0)\|_2 \\ &\quad = \|\operatorname{diag}_{n\times k}(0, \ldots, 0, \sigma_{l+1}, \ldots, \sigma_p)\|_2 \\ &\quad = \begin{cases} \sigma_{l+1} & \text{if } l < p \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{5.18}$$

This is the desired error bound for the spectral norm. The error bound for the Frobenius norm follows by a similar argument. We let $\mathcal{K} := \iota_m(\{1, \ldots, l\})$ and define $Q \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ by setting

$$Q_{ik} := \begin{cases} \hat{Q}_{\iota_m^{-1}(i), \iota_m^{-1}(k)} & \text{if } i \in \mathcal{I}', \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \mathcal{I}$ and $k \in \mathcal{K}$ and see that the error bound (5.18) is equivalent to (5.17a). □

The Algorithm 18 follows the structure of the proof of Lemma 5.19 in order to compute index sets $\mathcal{K} \subseteq \mathcal{I}'$ with $\#\mathcal{K} \leq \operatorname{rank}(X)$ and orthogonal matrices $Q \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ satisfying (5.17a) or (5.17b), depending on which strategy of the ones given in Algorithm 17 is used to determine the rank. We use a minor modification: we employ Householder factorizations in order to ensure that the singular value decomposition has only to be computed for a square matrix of dimension $q = \min\{n, m\}$.

Algorithm 17. Find the optimal rank l for approximating a matrix with singular values $\sigma_1 > \cdots > \sigma_q$ up to an accuracy of ϵ.

procedure FindRankEuclAbs (ϵ, q, $(\sigma_i)_{i=1}^q$, **var** l); $\{\|M - \widetilde{M}\|_2 \leq \epsilon\}$
$l \leftarrow 0$;
while $l < q$ and $\sigma_{l+1} \geq \epsilon$ **do**
  $l \leftarrow l + 1$
end while

procedure FindRankEuclRel (ϵ, q, $(\sigma_i)_{i=1}^q$, **var** l); $\{\|M - \widetilde{M}\|_2 \leq \epsilon\|M\|_2\}$
$l \leftarrow 1$;
while $l < q$ and $\sigma_{l+1} \geq \epsilon\sigma_1$ **do**
  $l \leftarrow l + 1$
end while

procedure FindRankFrobAbs (ϵ, q, $(\sigma_i)_{i=1}^q$, **var** l); $\{\|M - \widetilde{M}\|_F \leq \epsilon\}$
$l \leftarrow q$; $\delta \leftarrow 0$;
while $l > 0$ and $\delta + \sigma_l^2 \leq \epsilon^2$ **do**
  $\delta \leftarrow \delta + \sigma_l^2$; $l \leftarrow l - 1$
end while

procedure FindRankFrobRel (ϵ, q, $(\sigma_i)_{i=1}^q$, **var** l); $\{\|M - \widetilde{M}\|_F \leq \epsilon\|M\|_F\}$
$\mu \leftarrow 0$;
for $i \in \{1, \ldots, q\}$ **do**
  $\mu \leftarrow \mu + \sigma_i^2$
end for;
$l \leftarrow q$; $\delta \leftarrow 0$;
while $l > 0$ and $\delta + \sigma_l^2 \leq \epsilon^2\mu$ **do**
  $\delta \leftarrow \delta + \sigma_l^2$; $l \leftarrow l - 1$
end while

Remark 5.20 (Complexity of the SVD). A singular value decomposition is typically computed by an iteration, therefore the number of iteration steps depends on the desired precision. Due to [48], Section 5.4.5, there is a constant $C_{\text{svd}} \in \mathbb{N}$ such that the computation of the decomposition of the matrix $\widehat{Y} \in \mathbb{R}^{q\times q}$ to an arbitrary, but fixed, machine precision requires not more than $C_{\text{svd}}q^3$ operations. □

Lemma 5.21. *There is a constant $C_{\text{pr}} \in \mathbb{N}$ such that Algorithm 18 requires not more than $C_{\text{pr}}(\#\mathcal{I}')n_{\mathcal{J}} \min\{\#\mathcal{I}', \#\mathcal{J}\}$ operations.*

Proof. Let $m := \#\mathcal{I}'$, $n := \#\mathcal{J}$ and $q := \min\{m, n\}$.

If $m > n$ holds, we construct $\widehat{Y}$ by finding the Householder factorization $\widehat{Q}\widehat{R}$ of the $m \times n$ matrix $\widehat{X}$. According to [48], Section 5.2, this can be accomplished in not more than $C_{\text{qr}}mn^2 = C_{\text{qr}}mnq$ operations.

Algorithm 18. Construct an approximative factorization $QR \approx X$ of a matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}'}$ with an orthogonal matrix $Q \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ and a weight matrix $R \in \mathbb{R}^{\mathcal{K}\times\mathcal{J}}$.

procedure Lowrank(X, $\mathcal{I}'$, ϵ, **var** Q, R, $\mathcal{K}$)
$m \leftarrow \#\mathcal{I}'$; $n \leftarrow \#\mathcal{J}$; $q \leftarrow \min\{m,n\}$;
Fix arbitrary isomorphisms $\iota_m : \{1,\ldots,m\} \to \mathcal{I}'$ and $\iota_n : \{1,\ldots,n\} \to \mathcal{J}$;
$\hat{X} \leftarrow 0 \in \mathbb{R}^{m\times n}$;
for $i \in \{1,\ldots,m\}$, $j \in \{1,\ldots,n\}$ **do**
  $\hat{X}_{ij} \leftarrow X_{\iota_m(i),\iota_n(j)}$
end for;
if $m > n$ **then**
  Compute a Householder factorization $\hat{X} = \hat{Q}\hat{R}$ of $\hat{X}$;
  $\hat{Y} \leftarrow \hat{R} \in \mathbb{R}^{q\times q}$
else
  Compute a Householder factorization $\hat{X}^* = \hat{Q}\hat{R}$ of $\hat{X}^*$;
  $\hat{Y} \leftarrow \hat{R}^* \in \mathbb{R}^{q\times q}$
end if;
Compute a singular value decomposition $\hat{Y} = \hat{U}\,\mathrm{diag}(\sigma_1,\ldots,\sigma_q)\hat{V}^*$ of $\hat{Y}$;
if $m > n$ **then**
  $\hat{U} \leftarrow \hat{Q}\hat{U}$
end if;
FindRank(ϵ, q, $(\sigma_i)_{i=1}^q$, l); {cf. Algorithm 17}
$\mathcal{K} \leftarrow \iota_m(\{1,\ldots,l\})$; $Q \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}$;
for $i \in \{1,\ldots,m\}$, $k \in \{1,\ldots,l\}$ **do**
  $Q_{\iota_m(i),\iota_m(k)} \leftarrow \hat{U}_{ik}$
end for;
$R \leftarrow Q^*X \in \mathbb{R}^{\mathcal{K}\times\mathcal{J}}$

Otherwise, we construct $\hat{Y}$ be finding the Householder factorization $\hat{Q}\hat{R}$ of the $n \times m$ matrix $\hat{X}^*$. Again according to [48], Section 5.2, this requires not more than $C_{\mathrm{qr}}nm^2 = C_{\mathrm{qr}}mnq$ operations.

Due to Remark 5.20, we know that finding the singular values and left singular vectors of the matrix $\hat{Y} \in \mathbb{R}^{q\times q}$ up to machine precision requires not more than $C_{\mathrm{svd}}q^3$ operations. If $m > n$ holds, an additional multiplication is needed to construct $\hat{U}$, this takes not more than $2mq^2$ operations.

Filling the matrix R takes not more than $2qnm$ operations, so our claim holds with the constant $C_{\mathrm{pr}} = (C_{\mathrm{qr}} + C_{\mathrm{svd}} + 4)$. □

Remark 5.22 (Optimality of low-rank projections). Due to the optimality result [48], Theorem 2.5.3, the low-rank projection constructed by Algorithm 18 satisfies

$$\|X - QR\|_2 \leq \|X - AB\|_2 \quad \text{for all } A \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'},\ B \in \mathbb{R}^{\mathcal{K}\times\mathcal{J}},$$

i.e., the projection computed by using the singular value decomposition is at least as good as any other low-rank projection.

In this sense, Algorithm 18 provides us with an approximation with minimal rank $\#\mathcal{K}$. □

We can apply Algorithm 18 to the cluster basis matrices V_t in order to find an orthogonal cluster basis matrix Q_t satisfying (5.16) for any given precision $\epsilon \in \mathbb{R}_{>0}$ by ensuring that all but the first $\#L_t$ singular values of V_t are below ϵ. Picking the minimal rank with this property yields the *optimal* orthogonal cluster basis matrix $Q_t \in \mathbb{R}^{\mathcal{I}\times L_t}_{\hat{t}}$ for the given tolerance ϵ.

If V is nested, we again have to ensure that the new cluster basis Q has the same property. Let $t \in \mathcal{T}_\mathcal{I}$ be a cluster with $\tau := \#\operatorname{sons}(t) > 0$. We denote the sons of t by $\{t_1, \ldots, t_\tau\} = \operatorname{sons}(t)$. We assume that orthogonal matrices $Q_{t'} \in \mathbb{R}^{\mathcal{I}\times L_{t'}}_{\hat{t}'}$, index sets $L_{t'}$ and cluster operator matrices $R_{t'} = Q_{t'}^* V_{t'} \in \mathbb{R}^{L_{t'}\times K_{t'}}$ have already been constructed for all $t' \in \operatorname{sons}(t)$. Without loss of generality, we can also assume that the index sets $\{L_{t'} : t' \in \operatorname{sons}(t)\}$ are pairwise disjoint.

Since we want the new cluster basis to be nested, there have to be transfer matrices $F_{t'}$ for all $t' \in \operatorname{sons}(t)$ satisfying

$$Q_t = \sum_{t'\in\operatorname{sons}(t)} Q_{t'} F_{t'} = \underbrace{\begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix}}_{=U_t} \underbrace{\begin{pmatrix} F_{t_1} \\ \vdots \\ F_{t_\tau} \end{pmatrix}}_{=\widehat{Q}_t} = U_t \widehat{Q}_t. \tag{5.19}$$

This means that Q can only be nested if the range of the matrix Q_t is a subspace of the range of the matrix U_t.

Since U_t is orthogonal, the best approximation of V_t in the range of U_t is given by

$$\bar{V}_t := U_t U_t^* V_t = U_t \begin{pmatrix} Q_{t_1}^* V_t \\ \vdots \\ Q_{t_s}^* V_t \end{pmatrix},$$

and Lemma 5.2 and (3.15) imply

$$Q_{t_i}^* V_t = Q_{t_i}^* \sum_{t'\in\operatorname{sons}(t)} V_{t'} E_{t'} = Q_{t_i}^* V_{t_i} E_{t_i} = R_{t_i} E_{t_i}$$

for all $i \in \{1, \ldots, \tau\}$, so we find

$$\bar{V}_t = U_t \underbrace{\begin{pmatrix} R_{t_1} E_{t_1} \\ \vdots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix}}_{=\widehat{V}_t} = U_t \widehat{V}_t. \tag{5.20}$$

Here we can see the fundamental difference between the orthogonalization algorithm and the truncation algorithm: since the former is based on equation (5.11), no information is lost when switching from the original basis to the orthogonal one and we have $V_t = U_t \widehat{V}_t$. In the truncation algorithm, this equality no longer holds and we have to replace V_t by its projection $\overline{V}_t = U_t U_t^* V_t$ into the space prescribed by the sons of t.

In order to simplify the presentation, we extend the notation $\overline{V}_t$ to leaf clusters by setting

$$\overline{V}_t := \begin{cases} U_t U_t^* V_t & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ V_t & \text{otherwise,} \end{cases} \tag{5.21}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$.

We can now proceed as in the case of the orthogonalization algorithm: instead of computing a Householder factorization of $\widehat{V}_t$, we use a low-rank approximation of the $M_t \times K_t$-matrix $\widehat{V}_t$, which is provided efficiently by Algorithm 18, and observe that it gives rise to an orthogonal matrix $Q_t := U_t \widehat{Q}_t$ and by extension to an orthogonal cluster basis. The entire recursive procedure is given in Algorithm 19.

Algorithm 19. Truncate a cluster basis.

```
procedure Truncate(t, V, ε, var Q, R, L)
if sons(t) = ∅ then
  Lowrank(V_t, t̂, ε_t, Q_t, R_t, L_t)                    {Algorithm 18}
else
  M_t ← ∅;
  for t' ∈ sons(t) do
    Truncate(t', V, ε, Q, R, L)
    M_t ← M_t ∪̇ L_t'
  end for;
  V̂_t ← 0 ∈ ℝ^{M_t × K_t};
  for t' ∈ sons(t) do
    V̂_t|_{L_t' × K_t} ← R_t' E_t'
  end for;
  Lowrank(V̂_t, M_t, ε_t, Q̂_t, R_t, L_t);                {Algorithm 18}
  for t' ∈ sons(t) do
    F_t' ← Q̂_t|_{L_t' × L_t}
  end for
end if
```

Lemma 5.23 (Complexity of truncation). *Let V be a nested cluster basis with rank distribution K, and let $(k_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ be defined as in (3.16). Algorithm 19 requires not more than*

$$(C_{\mathrm{pr}} + 2) \sum_{t \in \mathcal{T}_{\mathcal{I}}} k_t^2 \# K_t$$

operations. If K is $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r} n_{\mathcal{I}})$.

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{sons}(t) = \emptyset$, the algorithm constructs a low-rank projection for V_t directly. According to Lemma 5.21, this requires not more than

$$C_{\mathrm{pr}}(\#\hat{t})(\#K_t)^2 \leq C_{\mathrm{pr}} k_t^2 \#K_t$$

operations.

If $\operatorname{sons}(t) \neq \emptyset$, we form the matrix $\widehat{V}_t$ by computing $R_{t'} E_{t'}$ for all sons t' of t, which requires $2(\#L_{t'})(\#K_{t'})(\#K_t) \leq 2(\#K_{t'})^2(\#K_t)$ operations for each $t' \in \operatorname{sons}(t)$, i.e., a total of

$$2(\#K_t) \sum_{t' \in \operatorname{sons}(t)} (\#K_{t'})^2 \leq 2(\#K_t)\Big(\sum_{t' \in \operatorname{sons}(t)} \#K_{t'} \Big)^2 \leq 2(\#K_t) k_t^2.$$

According to Lemma 5.21, finding the low-rank projection for $\widehat{V}_t$ by Algorithm 18 requires not more than

$$C_{\mathrm{pr}}(\#M_t)(\#K_t)^2 \overset{(5.15)}{\leq} C_{\mathrm{pr}} k_t^2 \#K_t$$

operations, and we can proceed as in the proof of Lemma 5.18. □

Remark 5.24. The truncation algorithm is related to the well-known Tausch–White construction [102] of wavelets on manifolds: instead of looking for a wavelet basis with vanishing moments, we are interested in finding a cluster basis Q that approximates the original cluster basis V. If V has been constructed using a polynomial basis, both techniques will yield similar results.

The advantage of the general truncation Algorithm 19 is that it also can be applied to general cluster bases, e.g., to the total cluster basis introduced in Chapter 6. □

Algorithm 19 guarantees the error bound (5.16) only for leaf clusters, since the construction in the remaining clusters involves the additional projection $U_t U_t^*$ used to ensure a nested structure. We now analyze the combined error of this projection and the truncated singular value decomposition. In order to be able to reach the optimal result, we first have to establish that the errors introduced by the singular value decompositions are pairwise perpendicular.

Lemma 5.25. *Let V be a nested cluster basis. Let Q be an orthogonal nested cluster basis. For each $t \in \mathcal{T}_{\mathcal{I}}$, we define the cluster-wise projection error*

$$D_t := \bar{V}_t - Q_t Q_t^* \bar{V}_t \tag{5.22}$$

where $\bar{V}_t$ is given by (5.21). The equation

$$Q_t^* D_s = 0 \tag{5.23}$$

holds for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{sons}^*(t)$.

Proof. We start with $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{sons}^*(t)$ satisfying $\operatorname{level}(s) - \operatorname{level}(t) = 0$. This implies $t = s$, and we find

$$Q_t^* D_t = Q_t^* \bar{V}_t - \underbrace{Q_t^* Q_t}_{=I} Q_t^* \bar{V}_t = Q_t^* \bar{V}_t - Q_t^* \bar{V}_t = 0.$$

Let $\ell \in \mathbb{N}_0$. We assume that equation (5.23) holds for all $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \operatorname{sons}^*(t)$ with $\operatorname{level}(s) - \operatorname{level}(t) \leq \ell$. Let $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{sons}^*(t)$ with $\operatorname{level}(s) - \operatorname{level}(t) = \ell + 1$. Since $t \neq s$, there is a cluster $t^- \in \operatorname{sons}(t)$ with $s \in \operatorname{sons}^*(t^-)$. The definition (5.22) implies $\chi_s D_s = D_s$, and since the sons of t correspond to disjoint subsets of $\hat{t}$, we have

$$\chi_{t'} \chi_s = \begin{cases} \chi_s & \text{if } t' = t^-, \\ 0 & \text{otherwise,} \end{cases} \tag{5.24}$$

for all $t' \in \operatorname{sons}(t)$, therefore we can apply (3.15) to Q and find

$$\begin{aligned} Q_t^* D_s &= \sum_{t' \in \operatorname{sons}(t)} F_{t'}^* Q_{t'}^* D_s \\ &= \sum_{t' \in \operatorname{sons}(t)} F_{t'}^* Q_{t'}^* \chi_{t'} \chi_s D_s = F_{t^-}^* Q_{t^-}^* \chi_s D_s = F_{t^-}^* Q_{t^-}^* D_s. \end{aligned}$$

Since $t^- \in \operatorname{sons}(t)$, we have $\operatorname{level}(t^-) = \operatorname{level}(t) + 1$. This implies $\operatorname{level}(s) - \operatorname{level}(t^-) = \ell$, so the induction assumption yields $Q_{t^-}^* D_s = 0$, which concludes the induction step. □

Lemma 5.26 (Error orthogonality). *Let V be a nested cluster basis. Let Q be an orthogonal nested cluster basis. Let $(D_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ be defined as in (5.22). The equation*

$$D_t^* D_s = 0 \tag{5.25}$$

holds for all $t, s \in \mathcal{T}_{\mathcal{I}}$ with $t \neq s$.

Proof. Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $t \neq s$. Due to symmetry, it is sufficient to restrict our attention to the case $\operatorname{level}(t) \leq \operatorname{level}(s)$.

Case 1: $\operatorname{level}(t) = \operatorname{level}(s)$. Lemma 3.8 implies $\hat{t} \cap \hat{s} = \emptyset$ and therefore $\chi_t \chi_s = 0$, which allows us to conclude

$$D_t^* D_s = D_t^* \chi_t \chi_s D_s = 0. \tag{5.26}$$

Case 2: $\operatorname{level}(t) < \operatorname{level}(s)$.

Case 2a: $s \notin \operatorname{sons}^*(t)$. Lemma 3.8 implies again $\hat{t} \cap \hat{s} = \emptyset$ and we proceed as in (5.26).

Case 2b: $s \in \operatorname{sons}^*(t)$. Due to $\operatorname{level}(t) < \operatorname{level}(s)$, we can infer $\tau := \#\operatorname{sons}(t) > 0$. We let $\{t_1, \ldots, t_\tau\} := \operatorname{sons}(t)$ and find

$$D_t = \bar{V}_t - Q_t Q_t^* \bar{V}_t = U_t U_t^* V_t - U_t \widehat{Q}_t Q_t^* \bar{V}_t = U_t \widehat{V}_t - U_t \widehat{Q}_t Q_t^* V_t$$

$$= \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} R_{t_1} E_{t_1} \\ \vdots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix} - \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} F_{t_1} \\ \vdots \\ F_{t_\tau} \end{pmatrix} Q_t^* V_t$$

$$= \sum_{t' \in \operatorname{sons}(t)} Q_{t'} (R_{t'} E_{t'} - F_{t'} Q_t^* V_t).$$

By definition of $\operatorname{sons}^*(t)$, there is a cluster $t^- \in \operatorname{sons}(t)$ with $s \in \operatorname{sons}^*(t^-)$, and (5.24) yields

$$\begin{aligned} D_t^* D_s &= \sum_{t' \in \operatorname{sons}(t)} (R_{t'} E_{t'} - F_{t'} Q_t^* V_t)^* Q_{t'}^* D_s \\ &= \sum_{t' \in \operatorname{sons}(t)} (R_{t'} E_{t'} - F_{t'} Q_t^* V_t)^* Q_{t'}^* \chi_{t'} \chi_s D_s \\ &= (R_{t^-} E_{t^-} - F_{t^-} Q_t^* V_t)^* Q_{t^-}^* D_s. \end{aligned}$$

Due to Lemma 5.25, we have $Q_{t^-}^* D_s = 0$. □

Since any descendant $r \in \operatorname{sons}^*(t)$ of a cluster t can be relevant to the error estimate, we require a method for investigating the interaction between t and all of its descendants. A straightforward approach is to introduce suitably generalized transfer matrices:

Definition 5.27 (Long-range transfer matrices). Let V be a nested cluster basis with rank distribution K and transfer matrices $(E_t)_{t \in \mathcal{T}_\mathcal{I}}$.

For all $t \in \mathcal{T}_\mathcal{I}$ and $r \in \operatorname{sons}^*(t)$, we define the matrix $E_{r,t} \in \mathbb{R}^{K_r \times K_t}$ by

$$E_{r,t} := \begin{cases} E_{r,t'} E_{t'} & \text{if there is a } t' \in \operatorname{sons}(t) \text{ with } r \in \operatorname{sons}^*(t') \\ I & \text{otherwise, i.e., if } t = r. \end{cases}$$

Lemma 5.28 (Transitivity). *Let $t, s, r \in \mathcal{T}_\mathcal{I}$ with $s \in \operatorname{sons}^*(t)$ and $r \in \operatorname{sons}^*(s)$. Then we have*

$$E_{r,t} = E_{r,s} E_{s,t}. \tag{5.27}$$

Proof. By induction on $\operatorname{level}(r) - \operatorname{level}(t)$. If $\operatorname{level}(r) - \operatorname{level}(t) = 0$, we have $t = s = r$ and the statement is trivial.

Let now $n \in \mathbb{N}_0$ be such that (5.27) holds for all $t, s, r \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(r) - \operatorname{level}(t) = n$, $s \in \operatorname{sons}^*(t)$ and $r \in \operatorname{sons}^*(s)$.

Let $t, s, r \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(r) - \operatorname{level}(t) = n + 1$, $s \in \operatorname{sons}^*(t)$ and $r \in \operatorname{sons}^*(s)$.

Case 1: $s = t$. This case is trivial, since we have

$$E_{r,t} = E_{r,s} = E_{r,s} E_{s,s} = E_{r,s} E_{s,t}.$$

Case 2: $s \neq t$. Since $s \in \operatorname{sons}^*(t)$, there is a $t' \in \operatorname{sons}(t)$ with $s \in \operatorname{sons}^*(t')$. Due to Lemma 3.7, $r \in \operatorname{sons}^*(s)$ implies $r \in \operatorname{sons}^*(t')$, and Definition 5.27 yields

$$E_{r,t} = E_{r,t'} E_{t'}. \tag{5.28}$$

We have $r \in \operatorname{sons}^*(s)$, $s \in \operatorname{sons}^*(t')$ and $\operatorname{level}(r) - \operatorname{level}(t') = n$, so we can apply the induction assumption to $E_{r,t'}$ and get

$$E_{r,t} = E_{r,t'} E_{t'} = E_{r,s} E_{s,t'} E_{t'},$$

and $s \in \operatorname{sons}^*(t')$ yields $E_{s,t'} E_{t'} = E_{s,t}$, which concludes the induction. □

The long-range transfer matrices give an explicit representation of the approximation error: the error matrix is the sum of local error matrices, which are weighted by the long-range transfer matrices.

Lemma 5.29 (Error decomposition). *Let V be a nested cluster basis. Let Q be an orthogonal nested cluster basis. Let $(\bar{V}_t)_{t \in \mathcal{T}_\mathcal{I}}$ be defined as in (5.21). Then the approximation error has the representation*

$$V_t - Q_t Q_t^* V_t = \sum_{r \in \operatorname{sons}^*(t)} (\bar{V}_r - Q_r Q_r^* \bar{V}_r) E_{r,t} \tag{5.29}$$

for all $t \in \mathcal{T}_\mathcal{I}$.

Proof. By induction on $\#\operatorname{sons}^*(t) \in \mathbb{N}$. We start by considering $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) = 1$. This implies $\operatorname{sons}(t) = \emptyset$ and (5.29) follows directly from Definition 5.27 and (5.21).

Let now $n \in \mathbb{N}$ be such that (5.29) holds for all $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) \leq n$. Let $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}^*(t) = n + 1$. This implies $\tau := \#\operatorname{sons}(t) > 0$. Let $\{t_1, \ldots, t_\tau\} := \operatorname{sons}(t)$. Since Q is nested, we have $Q_t = U_t \widehat{Q}_t$ (cf. (5.14)) and find

$$Q_t^* V_t = \widehat{Q}_t^* U_t^* V_t = \widehat{Q}_t^* (U_t^* U_t) U_t^* V_t = Q_t^* \bar{V}_t, \tag{5.30}$$

which together with (3.15) implies

$$\begin{aligned}
V_t - Q_t Q_t^* V_t &= V_t - \bar{V}_t + \bar{V}_t - Q_t Q_t^* \bar{V}_t = V_t - U_t \widehat{V}_t + D_t \\
&= D_t + \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'} - \begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} R_{t_1} E_{t_1} \\ \vdots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix} \\
&= D_t + \sum_{t' \in \operatorname{sons}(t)} (V_{t'} - Q_{t'} R_{r'}) E_{t'} \\
&= D_t + \sum_{t' \in \operatorname{sons}(t)} (V_{t'} - Q_{t'} Q_{t'}^* V_{t'}) E_{t'}.
\end{aligned}$$

Due to $\#\operatorname{sons}^*(t') < \#\operatorname{sons}^*(t) = n + 1$, we can apply the induction assumption to get

$$\begin{aligned}
V_t - Q_t Q_t^* V_t &= D_t + \sum_{t' \in \operatorname{sons}(t)} \sum_{r \in \operatorname{sons}^*(t')} D_r E_{r,t'} E_{t'} \\
&= D_t + \sum_{t' \in \operatorname{sons}(t)} \sum_{r \in \operatorname{sons}^*(t')} D_r E_{r,t} = \sum_{r \in \operatorname{sons}^*(t)} D_r E_{r,t},
\end{aligned}$$

which concludes the induction. □

In order to stress the similarities between leaf and non-leaf clusters and to simplify the notation, we extend the definitions of the matrices $\widehat{V}_t$ and $\widehat{Q}_t$ (cf. (5.12) and (5.13), respectively) to the case of leaf clusters: let

$$\widehat{V}_t := \begin{cases} \begin{pmatrix} R_{t_1} E_{t_1} \\ \dots \\ R_{t_\tau} E_{t_\tau} \end{pmatrix} & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ V_t & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_I, \tag{5.31a}$$

$$\widehat{Q}_t := \begin{cases} \begin{pmatrix} F_{t_1} \\ \dots \\ F_{t_\tau} \end{pmatrix} & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ Q_t & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_I. \tag{5.31b}$$

We also extend U_t by setting $U_t := I$ for leaf clusters $t \in \mathcal{L}_I$. Using these notations, the core of Algorithm 19 is the computation of a low-rank approximation of $\widehat{V}_t$, only the definition of this matrix and the interpretation of $\widehat{Q}_t$ vary depending on whether t is a leaf or not. We can also use the new notation to point out that the error estimate of the following Theorem 5.30 is based entirely on quantities available during the course of Algorithm 19:

Theorem 5.30 (Truncation error). *Let V be a nested cluster basis with rank distribution K. Let Q be an orthogonal nested cluster basis. Let $(\widehat{V}_t)_{t\in\mathcal{T}_I}$ and $(\widehat{Q}_t)_{t\in\mathcal{T}_I}$ be defined as in (5.31). The approximation error satisfies the equation*

$$\|V_t x - Q_t Q_t^* V_t x\|_2^2 = \sum_{r\in\operatorname{sons}^*(t)} \|(\widehat{V}_r - \widehat{Q}_r \widehat{Q}_r^* \widehat{V}_r) E_{r,t} x\|_2^2 \tag{5.32}$$

for all $t \in \mathcal{T}_I$ and all $x \in \mathbb{R}^{K_t}$.

Proof. Let $x \in \mathbb{R}^{\mathcal{J}}$ and $t \in \mathcal{T}_I$. We can combine Lemma 5.29 with Lemma 5.26 in order to get

$$\begin{aligned} \|V_t x - Q_t Q_t^* V_t x\|_2^2 &= \Big\| \sum_{r\in\operatorname{sons}^*(t)} D_r E_{r,t} x \Big\|_2^2 \\ &= \sum_{r_1\in\operatorname{sons}^*(t)} \sum_{r_2\in\operatorname{sons}^*(t)} \langle D_{r_1} E_{r_1,t} x, D_{r_2} E_{r_2,t} x \rangle \\ &= \sum_{r\in\operatorname{sons}^*(t)} \|D_r E_{r,t} x\|_2^2 \\ &= \sum_{r\in\operatorname{sons}^*(t)} \|(\overline{V}_r - Q_r Q_r^* \overline{V}_r) E_{r,t} x\|_2^2 \end{aligned} \tag{5.33}$$

for all $t \in \mathcal{T}_I$, $s \in \mathbb{R}^{K_t}$. If $\operatorname{sons}(t) = \emptyset$, equation (5.32) is the same as (5.33).

If $\operatorname{sons}(t) \neq \emptyset$, we have

$$\bar{V}_r = U_r U_r^* V_r = U_r \hat{V}_r, \quad Q_r = U_r \hat{Q}_r \quad \text{for all } r \in \operatorname{sons}^*(t)$$

and observe

$$\begin{aligned}\|(\bar{V}_r - Q_r Q_r^* \bar{V}_r) E_{r,t} x\|_2 &= \|(U_r \hat{V}_r - U_r \hat{Q}_r \hat{Q}_r^* U_r^* U_r \hat{V}_r) E_{r,t} x\|_2 \\ &= \|U_r (\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t} x\|_2 \quad \text{for all } r \in \operatorname{sons}^*(t).\end{aligned}$$

Since the matrices U_r are orthogonal, we get

$$\|(\bar{V}_r - Q_r Q_r^* \bar{V}_r) E_{r,t} x\|_2 = \|U_r (\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t} x\|_2 = \|(\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t} x\|_2,$$

and this is the desired equation. □

Corollary 5.31 (Spectral and Frobenius norm error). *Let V be a nested cluster basis. Let Q be an orthogonal nested cluster basis. We define*

$$\begin{aligned}\epsilon_{2,t} &:= \|\hat{V}_t - \hat{Q}_t \hat{Q}_t^* \hat{V}_t\|_2 \quad \text{for all } t \in \mathcal{T}_I, \\ \epsilon_{F,t} &:= \|\hat{V}_t - \hat{Q}_t \hat{Q}_t^* \hat{V}_t\|_F \quad \text{for all } t \in \mathcal{T}_I.\end{aligned}$$

Then the cluster basis Q satisfies the estimates

$$\|V_t - Q_t Q_t^* V_t\|_2^2 \leq \sum_{r \in \operatorname{sons}^*(t)} \epsilon_{2,r}^2 \|E_{r,t}\|_2^2 \quad \text{for all } t \in \mathcal{T}_I, \tag{5.34a}$$

$$\|V_t - Q_t Q_t^* V_t\|_F^2 \leq \sum_{r \in \operatorname{sons}^*(t)} \epsilon_{F,r}^2 \|E_{r,t}\|_2^2 \quad \text{for all } t \in \mathcal{T}_I. \tag{5.34b}$$

for all $t \in \mathcal{T}_I$.

Proof. Let K be the rank distribution of V. For the spectral norm estimate, we pick an arbitrary $x \in \mathbb{R}^{K_t}$ and see that Theorem 5.30 implies

$$\begin{aligned}&\|(V_t - Q_t Q_t^* V_t) x\|_2^2 \\ &\quad = \sum_{r \in \operatorname{sons}^*(t)} \|(\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t} x\|_2^2 \\ &\quad \leq \sum_{r \in \operatorname{sons}^*(t)} \|(\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r)\|_2^2 \|E_{r,t}\|_2^2 \|x\|_2^2 \\ &\quad = \sum_{r \in \operatorname{sons}^*(t)} \epsilon_{2,r}^2 \|E_{r,t}\|_2^2 \|x\|_2^2,\end{aligned}$$

and this implies (5.34a).

For $j \in K_t$, we define the canonical unit vectors $e^j \in \mathbb{R}^{K_t}$ by $e^j_i = \delta_{ij}$. The square of the Frobenius norm of a matrix equals the sum of the squares of the norms of its columns, so we have

$$\begin{aligned}
\|V_t - Q_t Q_t^* V_t\|_F^2 &= \sum_{j \in K_t} \|V_t e^j - Q_t Q_t^* V_t e^j\|_2^2 \\
&= \sum_{r \in \operatorname{sons}^*(t)} \sum_{j \in K_t} \|(\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t} e^j\|_2^2 \\
&= \sum_{r \in \operatorname{sons}^*(t)} \|(\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r) E_{r,t}\|_F^2 \\
&\leq \sum_{r \in \operatorname{sons}^*(t)} \|\hat{V}_r - \hat{Q}_r \hat{Q}_r^* \hat{V}_r\|_F^2 \|E_{r,t}\|_2^2 \\
&= \sum_{r \in \operatorname{sons}^*(t)} \epsilon_{F,r}^2 \|E_{r,t}\|_2^2
\end{aligned}$$

and this is (5.34b). □

Remark 5.32. According to Lemma 5.19, we can control the quantities $(\epsilon_{2,t})_{t \in \mathcal{T}_\mathcal{I}}$ and $(\epsilon_{F,t})_{t \in \mathcal{T}_\mathcal{I}}$ by choosing the ranks $(l_t)_{t \in \mathcal{T}_\mathcal{I}}$ appropriately. In order to be able to apply Corollary 5.31, we require bounds for the long-range transfer matrices $\|E_{r,t}\|_2$.

If the original cluster basis V is constructed by interpolation (cf. Section 4.4), we have $|(E_{r,t})_{\nu\mu}| = |\mathcal{L}_{t,\mu}(\xi_{r,\nu})| \leq \|\mathcal{L}_{t,\mu}\|_{\infty,\mathcal{Q}_t} \leq \Lambda_m$, i.e., each individual entry of the transfer matrix is bounded by the stability constant and we get $\|E_{r,t}\|_2 \leq \|E_{r,t}\|_F \leq (\#K_r)^{1/2}(\#K_t)^{1/2}\Lambda_m = m^d \Lambda_m$. For variable-order interpolation and Taylor expansion, similar estimates can be derived. □

Remark 5.33 (Practical error control). In the situation of Remark 5.32, i.e., if there is a constant $C_{\mathrm{lt}} \in \mathbb{R}_{>0}$ satisfying $\|E_{r,t}\|_2 \leq C_{\mathrm{lt}}$ for all $t \in \mathcal{T}_\mathcal{I}$ and $r \in \operatorname{sons}^*(t)$, we can choose $\hat{\epsilon} \in \mathbb{R}_{>0}$ and $p \in (0,1)$ and use

$$\epsilon_t := \hat{\epsilon} p^{\operatorname{level}(t)} \quad \text{for all } t \in \mathcal{T}_\mathcal{I} \tag{5.35}$$

to determine the error tolerances used in the truncation Algorithm 19. According to Lemma 5.19 and Corollary 5.31, we get

$$\begin{aligned}
\|V_t - Q_t Q_t^* V_t\|_2^2 &\leq \sum_{r \in \operatorname{sons}^*(t)} \epsilon_r^2 C_{\mathrm{lt}}^2 = C_{\mathrm{lt}}^2 \sum_{\ell=\operatorname{level}(t)}^{p_\mathcal{I}} \sum_{\substack{r \in \operatorname{sons}^*(t) \\ \operatorname{level}(r)=\ell}} \epsilon_r^2 \\
&\leq C_{\mathrm{lt}}^2 \hat{\epsilon}^2 \sum_{\ell=\operatorname{level}(t)}^{p_\mathcal{I}} p^{2\ell} \#\{r \in \operatorname{sons}^*(t) \,:\, \operatorname{level}(r) = \ell\}.
\end{aligned}$$

If we assume $\#\operatorname{sons}(t) \leq C_{\mathrm{bc}}$ (cf. Definition 3.43 of bounded cluster trees) for all $t \in \mathcal{T}_{\mathcal{I}}$, a simple induction yields

$$\#\{r \in \operatorname{sons}^*(t) \,:\, \operatorname{level}(r) = \ell\} \leq C_{\mathrm{bc}}^{\ell-\operatorname{level}(t)},$$

and assuming $p^2 < 1/C_{\mathrm{bc}}$ allows us to conclude

$$\begin{aligned}
\|V_t - Q_t Q_t^* V_t\|_2^2 &\leq C_{\mathrm{lt}}^2 \hat{\epsilon}^2 \sum_{\ell=\operatorname{level}(t)}^{p_{\mathcal{I}}} p^{2\ell} C_{\mathrm{bc}}^{\ell-\operatorname{level}(t)} \\
&= C_{\mathrm{lt}}^2 \hat{\epsilon}^2 p^{2\operatorname{level}(t)} \sum_{\ell=0}^{p_{\mathcal{I}}-\operatorname{level}(t)} (C_{\mathrm{bc}} p^2)^{\ell} \qquad (5.36) \\
&< C_{\mathrm{lt}}^2 \hat{\epsilon}^2 p^{2\operatorname{level}(t)} \sum_{\ell=0}^{\infty} (C_{\mathrm{bc}} p^2)^{\ell} = \frac{C_{\mathrm{lt}}^2}{1 - C_{\mathrm{bc}} p^2} \hat{\epsilon}^2 p^{2\operatorname{level}(t)}.
\end{aligned}$$

This means that we can reach any accuracy by choosing $\hat{\epsilon}$ small enough. □

Remark 5.34 (Matrix error control). In practical applications, we are not interested in bounds for $\|V_t - Q_t Q_t^* V_t\|_2$, i.e., the approximation error for the cluster basis, but in bounds for $\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\mathcal{Q},*}[X]\|_2$, the approximation error for the matrix.

We assume that X is an $\mathcal{H}^2$-matrix constructed by constant-order interpolation (cf. Section 4.4) and that we have used Algorithm 19 with the error tolerances (5.35).

For a single admissible block $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+$, we have

$$\begin{aligned}
\|\chi_t(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\mathcal{Q},*}[X])\chi_s\|_2 &= \|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2 \\
&= \|V_t S_b W_s^* - Q_t Q_t^* V_t S_b W_s^*\|_2 = \|(V_t - Q_t Q_t^* V_t) S_b W_s^*\|_2 \\
&\leq \|V_t - Q_t Q_t^* V_t\|_2 \|S_b W_s^*\|_2.
\end{aligned}$$

Remark 5.33 allows us to bound the first factor, so we only have to find a bound for the second one. The definitions (4.20b) and (4.20c) yield

$$\begin{aligned}
|(S_b W_s^* u)_\nu| &= \Big| \sum_{j\in\hat{s}} \sum_{\mu\in L_s} (S_b)_{\nu\mu} (W_s)_{j\mu} u_j \Big| \\
&= \Big| \sum_{j\in\hat{s}} \sum_{\mu\in L_s} g(\xi_{t,\nu}, \xi_{s,\mu}) \int_\Omega \mathcal{L}_{s,\mu}(y) \psi_j(y)\, dy\, u_j \Big| \\
&= \Big| \int_\Omega \mathfrak{I}_s[g(\xi_{t,\nu}, \cdot)](y) \sum_{j\in\hat{s}} u_j \psi_j(y)\, dy \Big| \\
&\leq \|\mathfrak{I}_s[g(\xi_{t,\nu}, \cdot)]\|_{\infty,\Omega_s} \int_\Omega \Big| \sum_{j\in\hat{s}} u_j \psi_j(y) \Big| \, dy
\end{aligned}$$

for all $u \in \mathbb{R}^{\mathcal{J}}$. Using the stability of the interpolation scheme, the asymptotic smoothness of g and the admissibility condition (4.49), we prove

$$\|\mathfrak{I}_s[g(\xi_{t,\nu},\cdot)]\|_{\infty,\mathcal{Q}_s} \leq \Lambda_m \|g(\xi_{t,\nu},\cdot)\|_{\infty,\mathcal{Q}_s} \leq \frac{C_{\mathrm{as}}\Lambda_m}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^\sigma} \leq \frac{C_{\mathrm{as}}\Lambda_m(2\eta)^\sigma}{\operatorname{diam}(\mathcal{Q}_s)^\sigma},$$

and assuming that the supports are C_{ov}-overlapping and that the "curvature bound" (4.58) holds implies

$$\begin{aligned}
\int_\Omega \Big|\sum_{j\in\hat{s}} u_j \psi_j(y)\,dy\Big| &\leq \sum_{j\in\hat{s}} \int_\Omega |\psi_j(y)|\,dy\,|u_j| \\
&\leq \Big(\sum_{j\in\hat{s}} u_j^2\Big)^{1/2}\Big(\sum_{j\in\hat{s}}\Big(\int_\Omega |\psi_j(y)|\,dy\Big)^2\Big)^{1/2} \\
&\leq \|\chi_s u\|_2\Big(\sum_{j\in\hat{s}} |\Omega_j| \int_\Omega \psi_j^2(y)\,dy\Big)^{1/2} \leq C_{\mathcal{J}}\|\chi_s u\|_2\Big(\sum_{j\in\hat{s}} |\Omega_j|\Big)^{1/2} \\
&\leq C_{\mathcal{J}}\|\chi_s u\|_2 C_{\mathrm{ov}}^{1/2}|\Omega_s|^{1/2} \leq C_{\mathcal{J}}\|\chi_s u\|_2 C_{\mathrm{ov}}^{1/2} C_{\mathrm{cu}}^{1/2}\operatorname{diam}(\mathcal{Q}_s)^{d_\Omega/2}
\end{aligned}$$

for all $u \in \mathbb{R}^{\mathcal{J}}$. Combining these estimates gives us

$$\|S_b W_s^*\|_2 \leq C_{\mathrm{as}}\Lambda_m(2\eta)^\sigma C_{\mathrm{ov}}^{1/2} C_{\mathrm{cu}}^{1/2} C_{\mathcal{J}} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega/2-\sigma}. \tag{5.37}$$

In practical applications, $C_{\mathcal{J}}$ and $\operatorname{diam}(\mathcal{Q}_s)$ can be bounded, and combining the resulting estimate with (5.36) allows us to ensure that the error is small enough in each block, and a bound for the global error is provided by Lemma 4.46.

Finding estimates of the type (5.36) and (5.37) for more general approximation schemes can be complicated. In these situation, the algorithms presented in Chapter 6, especially Algorithm 30, are more attractive, since these take the matrices $E_{r,t}$ and $S_b W_s^*$ directly into account and guarantee a given accuracy without the need for additional estimates. □

The orthogonalization Algorithm 16 only guarantees $\operatorname{range}(V_t) \subseteq \operatorname{range}(Q_t)$ for all $t \in \mathcal{T}_{\mathcal{I}}$. With the truncation algorithm 19, we can reach equality of these spaces, which is a useful property for theoretical investigations.

Corollary 5.35 (Lossless truncation). *Let V be a nested cluster basis. There is an orthogonal nested cluster basis Q satisfying* $\operatorname{range}(Q_t) = \operatorname{range}(V_t)$ *and, consequently,* $\operatorname{rank}(Q_t) = \operatorname{rank}(V_t)$ *for all $t \in \mathcal{T}_{\mathcal{I}}$.*

Proof. We construct Q by using Algorithm 19 with an error tolerance of $\epsilon = 0$. Let $t \in \mathcal{T}_{\mathcal{I}}$. Due to Remark 5.32 and Theorem 5.30, we find $V_t = Q_t Q_t^* V_t$, i.e., $\operatorname{range}(V_t) \subseteq \operatorname{range}(Q_t)$ and $\operatorname{rank}(V_t) \leq \operatorname{rank}(Q_t)$. According to the construction of Lemma 5.19, we have $\operatorname{rank}(Q_t) \leq \operatorname{rank}(V_t)$, which implies $\operatorname{rank}(Q_t) = \operatorname{rank}(V_t)$ and therefore $\operatorname{range}(V_t) = \operatorname{range}(Q_t)$. □

5.6 Computation of the Frobenius norm of the projection error

In order to compute the Frobenius norm

$$\|X\|_F = \Big(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} X_{ij}^2\Big)^{1/2}$$

of the error of an $\mathcal{H}^2$-matrix approximation explicitly, we use Lemma 4.39 to express the global approximation error by local errors:

Lemma 5.36 (Frobenius approximation error). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, and let V and W be orthogonal. We have*

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F = \Big(\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \|\chi_t X \chi_s\|_F^2 - \|V_t^* X W_s\|_F^2\Big)^{1/2}.$$

Proof. Due to Lemma 4.39 and Lemma 5.4, we only have to investigate

$$\begin{aligned}&\|\chi_t(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X])\chi_s\|_F^2\\ &\qquad= \begin{cases}\|\chi_t X\chi_s - V_t V_t^* X W_s W_s^*\|_F^2 & \text{if } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},\\ \|\chi_t X\chi_s - \chi_t X\chi_s\|_F^2 & \text{otherwise,}\end{cases}\end{aligned}$$

for all $b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$. Since the case $b \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} = \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ is trivial, we focus on the admissible blocks and pick $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. The definition of the Frobenius norm implies

$$\begin{aligned}\|\chi_t X\chi_s - V_t V_t^* X W_s W_s^*\|_F^2 = \|\chi_t X\chi_s\|_F^2 + \|V_t V_t^* X W_s W_s^*\|_F^2\\ - 2\langle \chi_t X\chi_s, V_t V_t^* X W_s W_s^*\rangle_F,\end{aligned}$$

and the orthogonality of V_t and W_s yields

$$\begin{aligned}\langle \chi_t X\chi_s, V_t V_t^* X W_s W_s^*\rangle_F &= \langle \chi_t X\chi_s, V_t(V_t^*V_t)V_t^* X W_s(W_s^*W_s)W_s^*\rangle_F\\ &= \langle V_t V_t^* X W_s W_s^*, V_t V_t^* X W_s W_s^*\rangle_F\\ &= \|V_t V_t^* X W_s W_s^*\|_F^2,\end{aligned}$$

so we find

$$\|\chi_t X\chi_s - V_t V_t^* X W_s W_s^*\|_F^2 = \|\chi_t X\chi_s\|_F^2 - \|V_t V_t^* X W_s W_s^*\|_F^2$$

and observing $\|V_t V_t^* X W_s W_s^*\|_F^2 = \|V_t^* X W_s\|_F^2$ concludes the proof. □

Using this lemma, we can embed the efficient computation of the approximation error directly into Algorithm 11: after computing $S_b = V_t^* X W_s$ for an admissible

block $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we determine $\|S_b\|_F^2$ and $\|\chi_t X \chi_s\|_F^2$ and accumulate the differences of these norms. The result is the modified Algorithm 20, which computes the coupling matrices $(S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of the best approximation $\widetilde{X} := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]$ of X and returns the approximation error $\epsilon := \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F$ in the Frobenius norm.

Algorithm 20. Convert a dense matrix into an $\mathcal{H}^2$-matrix and compute the resulting approximation error.

procedure ConvertDenseAndError(X, V, W, **var** S, ϵ);
$\epsilon \leftarrow 0$;
for $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ **do**
  BlockForwardTransformation(s, W, $(\chi_t X \chi_s)^*$, $\widehat{X}$);
  BlockForwardTransformation(t, V, $\widehat{X}_s^*$, $\widehat{Y}$);
  $S_b \leftarrow \widehat{Y}_t$;
  $\epsilon \leftarrow \epsilon + (\|\chi_t X \chi_s\|_F^2 - \|S_b\|_F^2)$
end for;
$\epsilon \leftarrow \epsilon^{1/2}$

Finding the approximation error when converting a hierarchical matrix into the $\mathcal{H}^2$-matrix format is slightly more complicated since we need an efficient way of computing $\|\chi_t X \chi_s\|_F = \|A_b B_b^*\|_F$ for admissible blocks $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ given in factorized form. Due to

$$\begin{aligned}
\|A_b B_b^*\|_F^2 &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} (A_b B_b)_{ij}^2 = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} \Big(\sum_{\nu=1}^{k} (A_b)_{i\nu}(B_b)_{j\nu}\Big)^2 \\
&= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{\nu=1}^{k}\sum_{\mu=1}^{k} (A_b)_{i\nu}(B_b)_{j\nu}(A_b)_{i\mu}(B_b)_{j\mu} \\
&= \sum_{\nu=1}^{k}\sum_{\mu=1}^{k} \Big(\sum_{i\in\mathcal{I}} (A_b)_{i\nu}(A_b)_{i\mu}\Big)\Big(\sum_{j\in\mathcal{J}} (B_b)_{j\nu}(B_b)_{j\mu}\Big) \\
&= \sum_{\nu=1}^{k}\sum_{\mu=1}^{k} (A_b^* A_b)_{\nu\mu}(B_b^* B_b)_{\nu\mu},
\end{aligned}$$

this task is reduced to the computation of the k-by-k Gram matrices $A_b^* A_b$ and $B_b^* B_b$ and the summation of k^2 products. We can exploit the symmetry of the Gram matrices in order to derive Algorithm 21.

While finding the approximation error for matrices in dense and hierarchical matrix representation is straightforward, carrying out this computation in the case of $\mathcal{H}^2$-matrices presents us with a challenge: we have to compute the norm $\|\chi_t X \chi_s\|_F =$

Algorithm 21. Convert an $\mathcal{H}$-matrix into an $\mathcal{H}^2$-matrix and compute the resulting approximation error.

procedure ConvertHAndError(A, B, V, W, **var** S, ϵ);
$\epsilon \leftarrow 0$;
for $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ **do**
  BlockForwardTransformation(t, V, A_b, $\widehat{X}$);
  BlockForwardTransformation(s, W, B_b, $\widehat{Y}$);
  $S_b \leftarrow \widehat{X}_t \widehat{Y}_s^*$;
  $\epsilon_b \leftarrow 0$;
  for $\nu \in \{1,\ldots,k\}$ **do**
    $\epsilon_b \leftarrow \epsilon_b + (A_b^* A_b)_{\nu\nu}(B_b^* B_b)_{\nu\nu}$;
    for $\mu \in \{\nu+1,\ldots,k\}$ **do**
      $\epsilon_b \leftarrow \epsilon_b + 2(A_b^* A_b)_{\nu\mu}(B_b^* B_b)_{\nu\mu}$
    end for
  end for;
  $\epsilon := \epsilon + (\epsilon_b - \|S_b\|_F^2)$
end for;
$\epsilon := \epsilon^{1/2}$

$\|V_{X,t} S_{X,b} W_{X,s}^*\|_F$, and in order to reach optimal efficiency, we cannot use more than $\mathcal{O}(k_t^3 + l_s^3)$ operations. Our approach relies on Algorithm 16: applied to the cluster bases V_X and W_X, it provides us with orthogonal cluster bases Q_V and Q_W and cluster operators R_V and R_W satisfying $V_{X,t} = Q_{V,t} R_{V,t}$ and $W_{X,s} = Q_{W,s} R_{W,s}$ for all $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$. For an admissible block $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, the orthogonality of $Q_{V,t}$ and $Q_{W,s}$ implies

$$\begin{aligned}\|\chi_t X \chi_s\|_F &= \|V_{X,t} S_{X,b} W_{X,s}^*\|_F \\ &= \|Q_{V,t} R_{V,t} S_{X,b} R_{W,s}^* Q_{W,s}^*\|_F \\ &= \|R_{V,t} S_{X,b} R_{W,s}^*\|_F,\end{aligned}$$

and this norm can be computed in the desired complexity. We can see that in this case the orthogonal cluster bases Q_V and Q_W are not required, we are only interested in the cluster operators R_V and R_W, which are used as *weights* in place of V_X and W_X. Combining this approach with Algorithm 14 yields Algorithm 22.

5.7 Numerical experiments

Let us now investigate how closely the theoretical estimates for the complexity and the approximation error correspond to practical properties of our algorithms.

Algorithm 22. Convert an $\mathcal{H}^2$-matrix into an $\mathcal{H}^2$-matrix and compute the resulting approximation error.

procedure ConvertH2(S_X, V, W, **var** S, ϵ);
ClusterBasisProduct(root($\mathcal{T}_\mathcal{I}$), V, V_X, P_V);
ClusterBasisProduct(root($\mathcal{T}_\mathcal{J}$), W, W_X, P_W);
Orthogonalize(root($\mathcal{T}_\mathcal{I}$), V, Q_V, R_V);
Orthogonalize(root($\mathcal{T}_\mathcal{J}$), W, Q_W, R_W);
$\epsilon \leftarrow 0$;
for $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ **do**
  $S_b \leftarrow P_{V,t} S_{X,b} P^*_{W,s}$;
  $\epsilon \leftarrow \epsilon + (\|R_{V,t} S_{X,b} R^*_{W,s}\|_F^2 - \|S_b\|_F^2)$
end for
$\epsilon \leftarrow \epsilon^{1/2}$

Orthogonalization

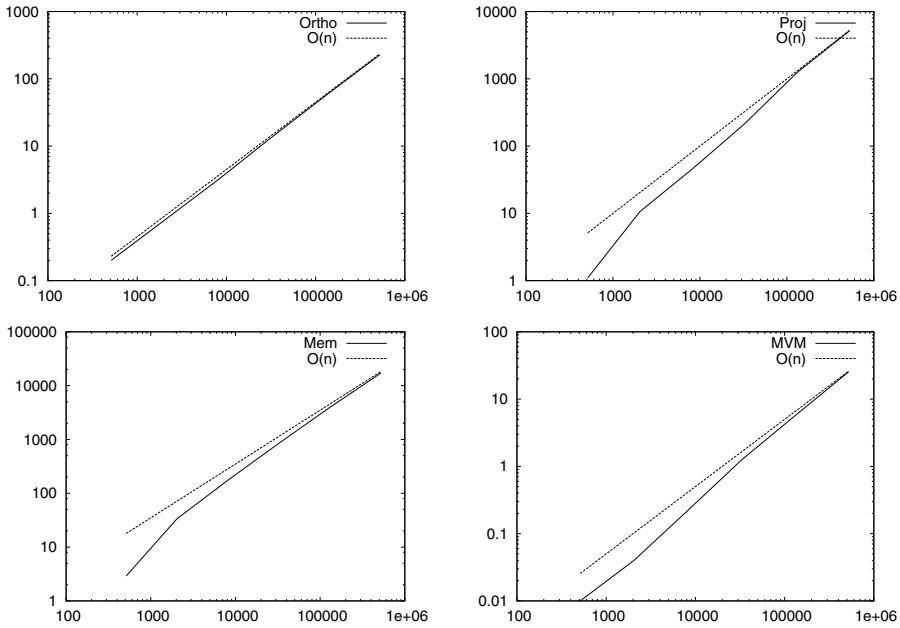
In a first experiment, we apply the orthogonalization Algorithm 16 to the $\mathcal{H}^2$-matrix approximation of the single layer potential V introduced in Section 4.9. We first consider a fixed interpolation order $m = 4$ and different matrix dimensions $n \in \{512, 2048, \ldots, 524288\}$, similar to Table 4.1. The results of this series of experiments are given in Table 5.1. According to the Lemmas 5.18 and 5.14, we expect that the time required for the orthogonalization of the cluster bases and the projection into the corresponding $\mathcal{H}^2$-matrix space is proportional to n, and this is indeed the case. Since the orthogonalization does not change the $\mathcal{H}^2$-matrix space, the approximation errors in Table 5.1 and Table 4.1 are identical.

Now let us investigate the influence of the interpolation order. This time, we fix $n = 32768$ and consider different orders $m \in \{1, 2, \ldots, 7\}$. The results are given in Table 5.2 for the admissibility parameter $\eta = 2$ and in Table 5.3 for the parameter $\eta = 1$. Lemma 5.18 predicts that the time for the computation will be proportional to m^6, and the prediction fits the experimental results. For the change of cluster bases, Lemma 5.14 also predicts a complexity proportional to m^6, but the experiments suggest that m^3 is a better fit for the practical results. One possible explanation for these surprisingly good execution times is that the matrices appearing in the matrix-matrix multiplications of Algorithm 14 are so small that the complexity is dominated by the $\mathcal{O}(m^6)$ write accesses instead of the $\mathcal{O}(m^9)$ read accesses to storage.

In the next experiment, we consider a variable-order approximation of the matrix V. Our theory predicts a complexity of $\mathcal{O}(n)$ both in time and in storage requirements, and the results in Table 5.4 match these predictions. Comparing this table with Table 4.6 shows that the orthogonalization procedure increases the runtime only by approximately 25% and reduces the storage requirements by 14%. Since the orthogonalization leaves the underlying $\mathcal{H}^2$-matrix approximation unchanged (only the cluster bases are

Table 5.1. Approximation of the three-dimensional single layer potential by orthogonalized polynomial interpolation of order $m = 4$.

n	Ortho	Proj	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	0.2	1.1	2.9	5.8	< 0.01	0
2048	0.8	10.7	33.8	16.9	0.04	3.6_{-7}
8192	3.2	45.7	176.1	22.0	0.22	1.5_{-7}
32768	13.8	215.6	854.0	26.7	1.24	3.5_{-8}
131072	56.8	1241.3	3968.2	31.0	5.67	9.0_{-9}
524288	226.8	5150.9	17263.8	33.7	25.53	2.2_{-9}



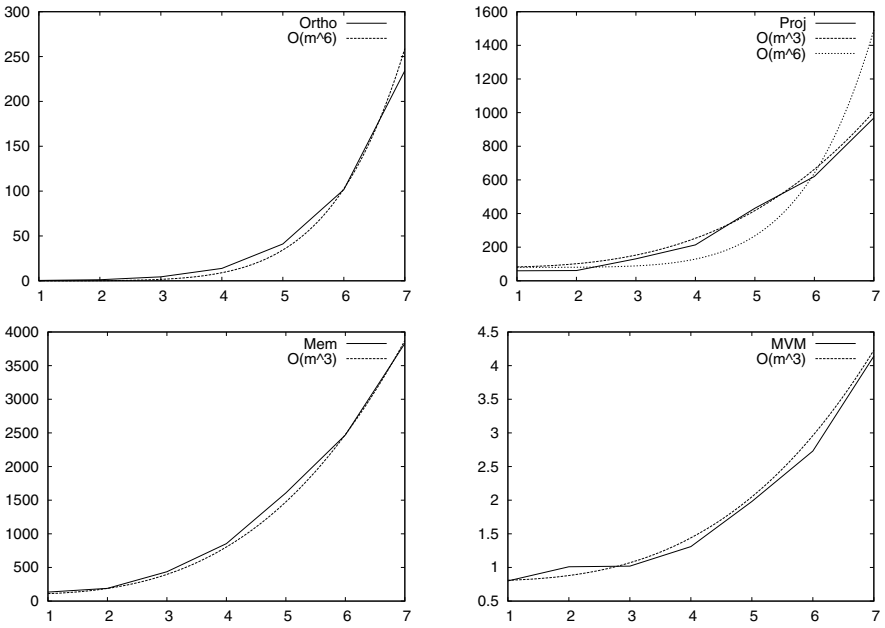
swapped), the approximation errors of the orthogonalized matrix are the same as for the original matrix.

The orthogonalization algorithm reduces the storage requirements only if $\#\hat{t} \leq \#K_t$ holds for a cluster $t \in \mathcal{T}_\mathcal{I}$, i.e., if the matrix V_t has more columns than rows. In the previous experiments, we have constructed cluster trees satisfying $\#\hat{t} \geq \#K_t$ for most clusters, therefore the orthogonalization algorithm could not reduce the storage requirements by a significant amount.

Let us now consider a situation in which there are a large number of clusters with $\#\hat{t} \leq \#K_t$: we stop subdividing clusters if they contain less than $C_{\mathrm{lf}} = 16$ indices, therefore the average size of leaf clusters will be close to 8. We once more compute an approximation of V for different interpolation orders $m \in \{1, 2, \ldots, 7\}$. Without

Table 5.2. Approximation of the three-dimensional single layer potential by orthogonalized polynomial interpolation with $\eta = 2$ and leaf bound $C_{\mathrm{lf}} = 2m^3$.

m	Ortho	Proj	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
1	0.4	59.5	134.2	4.2	0.80	1.7_{-5}
2	1.2	61.0	188.9	5.9	1.01	2.9_{-6}
3	4.4	130.8	436.2	13.6	1.02	2.3_{-7}
4	13.8	213.3	854.0	26.7	1.31	3.6_{-8}
5	41.1	431.4	1607.7	50.2	1.98	5.1_{-9}
6	101.7	620.4	2469.3	77.2	2.73	6.9_{-10}
7	234.3	968.1	3830.7	119.7	4.14	1.9_{-10}



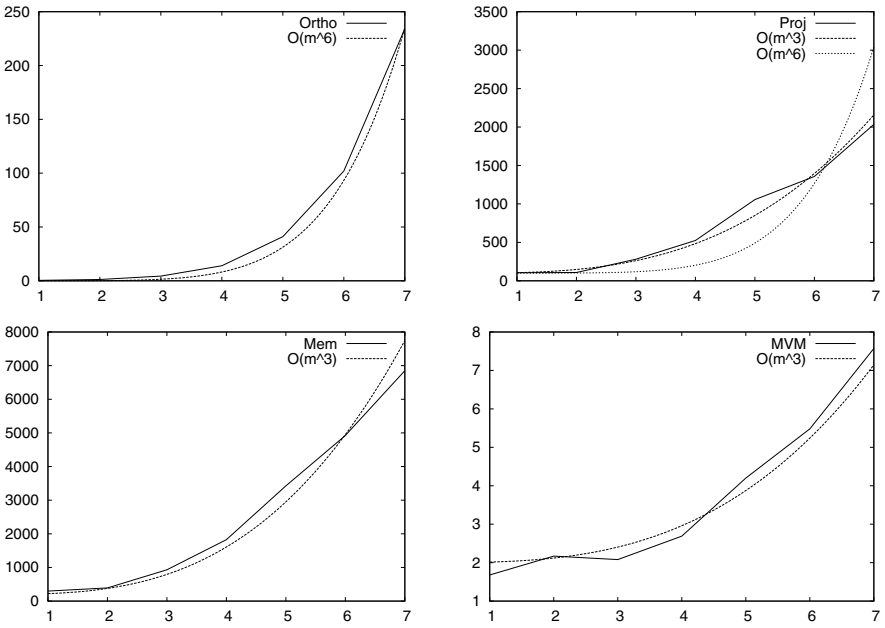
orthogonalization, the storage requirements would grow like m^6, since the number of clusters now no longer depends on m, and each cluster requires $\mathcal{O}(m^6)$ units of storage.

Table 5.5 shows that orthogonalization can be used to prevent this: the storage requirements are comparable to the standard case described in Tables 5.2 and 4.2, i.e., they grow like m^3 despite the unsuitable choice of C_{lf}, since the orthogonality of the matrices Q_t ensure that they cannot have more columns than rows.

This is a very important application of the orthogonalization Algorithm 16: in many applications, the cluster basis is constructed adaptively by applying truncation or the compression techniques introduced in Chapter 6 to an initial approximation of the matrix. In order to reach the optimal performance, we have to choose the cluster

Table 5.3. Approximation of the three-dimensional single layer potential by orthogonalized polynomial interpolation with $\eta = 1$ and leaf bound $C_{\mathrm{lf}} = 2m^3$.

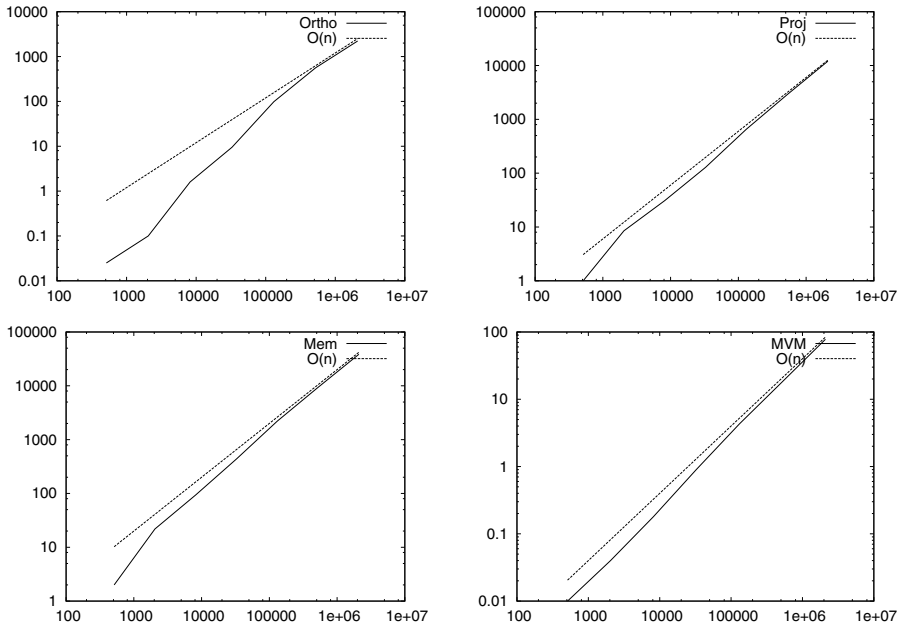
m	Ortho	Proj	Mem	Mem$/n$	MVM	$\|X - \tilde{X}\|_2$
1	0.4	104.5	297.0	9.3	1.68	1.3_{-5}
2	1.2	107.8	392.2	12.3	2.17	1.2_{-6}
3	4.4	280.7	930.4	29.1	2.08	5.4_{-8}
4	14.0	524.4	1825.0	57.0	2.69	4.6_{-9}
5	41.0	1056.4	3425.3	107.0	4.20	2.9_{-10}
6	102.0	1356.4	4923.7	153.9	5.48	2.8_{-11}
7	234.2	2034.2	6850.3	214.1	7.57	4.0_{-12}



tree and the block cluster tree in such a way that the amount of storage required for the nearfield is approximately proportional to the amount of storage required for the farfield, i.e., if the ranks of the final cluster bases are small, the leaf clusters also have to be small, usually far smaller than the number of expansion functions used for the initial approximation. This means that we are in the situation described above: the matrices V_t corresponding to the initial approximation have far more columns than rows, therefore we can apply orthogonalization in order to reduce both storage and time requirements by a large amount, especially for large problems with initial approximations of high order.

Table 5.4. Approximation of the three-dimensional single layer potential by orthogonalized polynomial interpolation of variable order with $\alpha = 2$ and $\beta = 1$.

n	Ortho	Proj	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	< 0.1	1.0	2.0	4.1	< 0.01	9.1_{-5}
2048	0.1	8.5	21.9	11.0	0.04	1.4_{-5}
8192	1.6	30.9	93.8	11.7	0.18	7.3_{-6}
32768	9.5	128.1	438.7	13.7	0.91	8.7_{-7}
131072	98.7	654.1	2210.2	17.3	4.31	1.0_{-7}
524288	564.5	2867.4	9353.5	18.3	18.29	1.5_{-8}
2097152	2221.7	12052.1	38480.8	18.8	77.88	1.9_{-9}



Truncation

Now we apply the truncation Algorithm 19 to the cluster bases constructed by interpolation. Since the truncated cluster bases only approximate the original ones, we have to be able to control the error.

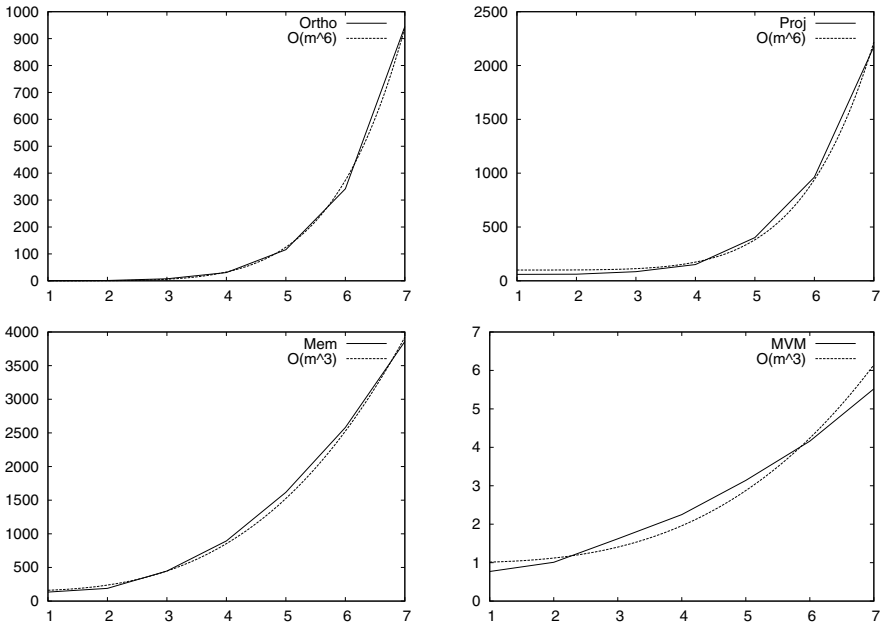
We use constant-order interpolation and base our investigation on the Remarks 5.33 and 5.34: we have $\|E_{r,t}\|_2 \leq \Lambda_m(\#K_t) = \Lambda_m m^d$ and

$$\|S_b W_s^*\|_2 \lesssim \Lambda_m C_{\mathcal{J}} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega/2-\sigma}.$$

We use Lagrangian basis functions on a two-dimensional surface, hence $C_{\mathcal{J}}$ is in $\mathcal{O}(h)$.

Table 5.5. Approximation of the three-dimensional single layer potential by orthogonalized polynomial interpolation with $\eta = 2$ and fixed leaf bound $C_{\mathrm{lf}} = 16$.

m	Ortho	Proj	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
1	0.4	59.5	134.2	4.2	0.77	1.8_{-5}
2	1.2	61.3	188.9	5.9	1.01	2.8_{-6}
3	7.6	84.6	446.8	14.0	1.62	2.6_{-7}
4	30.7	150.7	894.4	27.9	2.25	3.6_{-8}
5	115.6	401.4	1617.2	50.5	3.14	5.1_{-9}
6	341.8	963.0	2580.8	80.7	4.16	7.1_{-10}
7	944.8	2175.2	3858.5	120.6	5.52	1.3_{-10}

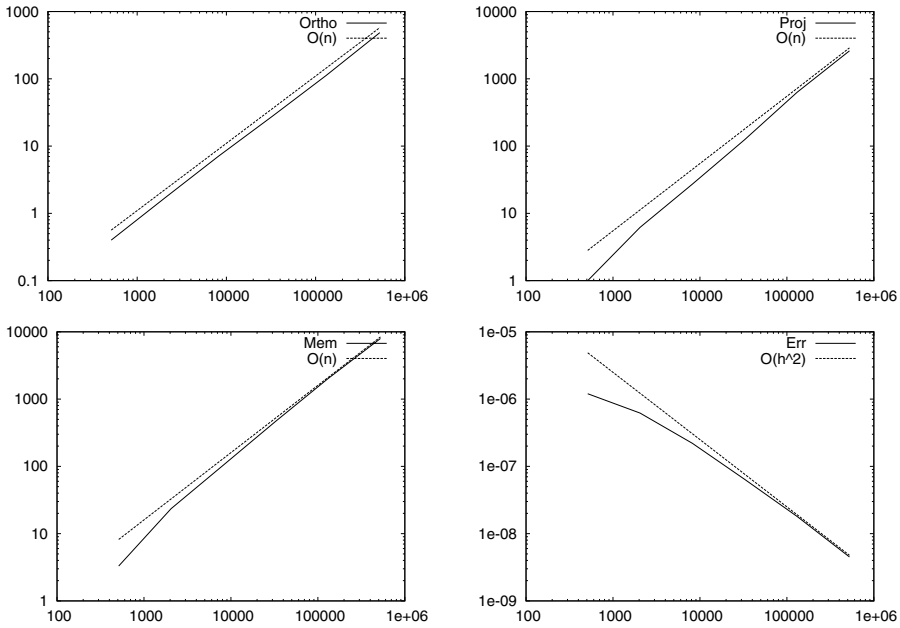


For the single layer potential, we get $d_\Omega = 2$ and $\sigma = 1$, i.e., $\|S_b W_s^*\|_2$ is in $\mathcal{O}(h)$ and we have to ensure $\|V_t - Q_t Q_t^* V_t\|_2 \lesssim h$ in order to reach the full accuracy of $\mathcal{O}(h^2)$.

Choosing the local error tolerances according to (5.35) with $p = 2/3$ yields the results given in Table 5.6: the approximation error is doubled compared to the original approximation (cf. Table 4.1), but the storage requirements are halved. The loss of accuracy is not critical, since we can always increase the order of the interpolation and get "exponential convergence at polynomial cost". The reduction of storage is very important, since most computers provide only a (relatively) small amount of storage, and the storage efficiency of algorithms determines whether a certain computation can

Table 5.6. Approximation of the three-dimensional single layer potential by truncated polynomial interpolation with $\eta = 2$, fixed leaf bound $C_{\mathrm{lf}} = 32$ and $\hat{\epsilon} = 2 \times 10^{-3} h$.

n	Trunc	Proj	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	0.4	1.0	3.3	6.6	< 0.01	1.2_{-6}
2048	1.7	6.2	23.5	11.8	0.05	6.2_{-7}
8192	7.1	27.2	104.4	13.0	0.25	2.2_{-7}
32768	27.8	124.9	468.1	14.6	1.23	6.4_{-8}
131072	111.8	630.9	2018.9	15.8	5.39	1.8_{-8}
524288	486.0	2602.5	7965.0	15.6	22.15	4.5_{-9}



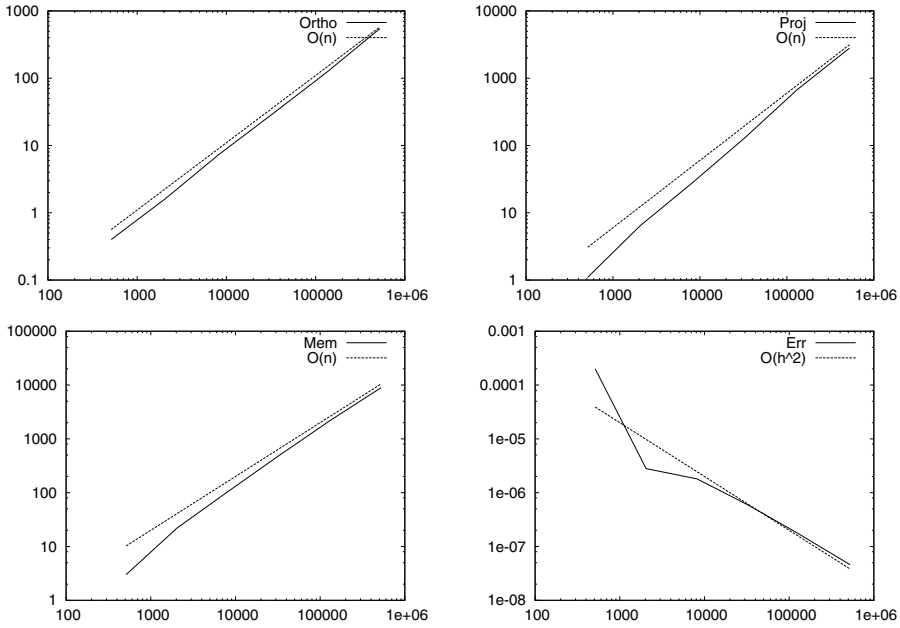
be performed or not.

For the double layer potential, we find $\sigma = 2$, i.e., $\|S_b W_s^*\|_2$ is in $\mathcal{O}(1)$ and we have to ensure $\|V_t - Q_t Q_t^* V_t\|_2 \lesssim h^2$ for the row cluster basis $V = (V_t)_{t \in \mathcal{T}_\mathcal{I}}$ in order to reach the full accuracy. For the column cluster basis $W = (W_s)_{s \in \mathcal{T}_\mathcal{J}}$, we still have $\|V_t S_b\|_2$ in $\mathcal{O}(h)$ and an accuracy of $\mathcal{O}(h)$ is sufficient for the approximation of W.

We once again choose the local error tolerances according to (5.35) with $p = 2/3$ and obtain the results given in Table 5.7. As expected, the approximation quality is reduced by one order (due to the normal derivative used to define K), but otherwise the errors behave as expected. The storage requirements and execution times are comparable to the case of the single layer potential.

Table 5.7. Approximation of the three-dimensional double layer potential by truncated polynomial interpolation with $\eta = 2$, fixed leaf bound $C_{\mathrm{lf}} = 32$ and $\hat{\epsilon}_{\mathrm{row}} = 5 \times 10^{-2} h^2$, $\hat{\epsilon}_{\mathrm{col}} = 2 \times 10^{-2} h$.

n	Trunc	Proj	Mem	Mem$/n$	MVM	$\|X - \tilde{X}\|_2$
512	0.4	1.1	3.0	6.0	< 0.01	2.0_{-4}
2048	1.6	6.4	22.1	11.1	0.05	2.8_{-6}
8192	7.3	27.9	105.7	13.2	0.27	1.8_{-6}
32768	29.4	129.9	490.9	15.3	1.29	5.9_{-7}
131072	120.4	674.2	2191.3	17.1	5.70	1.7_{-7}
524288	546.1	2799.1	8936.3	17.5	23.58	4.6_{-8}



Chapter 6
Compression

We have seen that we can compute $\mathcal{H}^2$-matrix approximations of matrices efficiently if suitable nested cluster bases are given, and that we can improve $\mathcal{H}^2$-matrix approximations by removing redundant information from the cluster bases.

Now we investigate the approximability of general matrices and address the following three questions:

- What are the basic structural properties of $\mathcal{H}^2$-matrices?
- Which types of matrices can be approximated by $\mathcal{H}^2$-matrices?
- If a matrix can be approximated, how can we find the approximation efficiently?

In order to simplify the investigation of the first question, we introduce the space of semi-uniform matrices and demonstrate that a matrix X is an $\mathcal{H}^2$-matrix if and only if X and X^* are semi-uniform matrices, therefore we only have to investigate this more general class of matrices.

The answer to the first question is then rather simple: a matrix is semi-uniform if and only if all elements of a family of submatrices, called the *total cluster basis*, have low rank.

The second question can also be answered by referring to total cluster bases: a matrix can be approximated by a semi-uniform matrix if and only if the total cluster basis matrices can be approximated by low-rank matrices.

To answer the third question, we derive an efficient algorithm which uses recursively constructed low-rank approximations of the total cluster basis matrices in order to find an orthogonal nested cluster basis.

This chapter is organized as follows:

- Section 6.1 introduces the spaces of left and right semi-uniform matrices for a given block cluster tree and given cluster bases.
- Section 6.2 contains the definition of total cluster bases and the proof that a matrix is semi-uniform if and only if the total cluster bases have low rank.
- Section 6.3 investigates the *approximation* of general matrices by semi-uniform matrices. The fundamental result is the explicit error representation given by Theorem 6.16 (cf. [13]).
- Section 6.4 is devoted to the introduction and analysis of an algorithm for finding good cluster bases, and thereby good $\mathcal{H}^2$-matrix approximations, for arbitrary matrices (cf. [64] for a less robust predecessor of this algorithm).

- Section 6.5 introduces a modification of the basic compression algorithm that allows us to approximate $\mathcal{H}$-matrices efficiently (cf. [64] for a less robust predecessor of this algorithm).
- Section 6.6 is devoted to another modification of the algorithm that allows us to approximate $\mathcal{H}^2$-matrices in *optimal* complexity (cf. [10] for a less robust predecessor of this algorithm).
- Section 6.7 describes a variant of this algorithm to find cluster bases that approximate multiple $\mathcal{H}^2$-matrices simultaneously, the resulting *hierarchical compression* strategy is of crucial importance for several advanced applications including the matrix arithmetic operations presented in Chapter 8.
- Section 6.8 provides refined strategies for controlling the error introduced by the $\mathcal{H}^2$-matrix approximation algorithm (cf. [12]).
- Section 6.9 concludes this chapter by presenting numerical results that show that the compression algorithm reaches optimal complexity and that the resulting approximation error behaves as expected.

Assumptions in this chapter: We assume that cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ for the finite index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, are given. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let $n_\mathcal{I} := \#\mathcal{I}$ and $n_\mathcal{J} := \#\mathcal{J}$ denote the number of indices in $\mathcal{I}$ and $\mathcal{J}$, and let $c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}$ and $c_\mathcal{J} := \#\mathcal{T}_\mathcal{J}$ denote the number of clusters in $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let $p_\mathcal{I}$, $p_\mathcal{J}$ and $p_{\mathcal{I}\times\mathcal{J}}$ be the depths of $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ and $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

6.1 Semi-uniform matrices

Let V and W be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. Let K and L be the rank distributions of V and W, and let $E = (E_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $F = (F_s)_{s\in\mathcal{T}_\mathcal{J}}$ be their transfer matrices.

Our goal is to investigate the approximation properties of $\mathcal{H}^2$-matrices. Instead of taking the influence of both row and column cluster bases into account simultaneously, we consider only "half" $\mathcal{H}^2$-matrices, i.e., matrices which depend only on row *or* column cluster bases.

Definition 6.1 (Semi-uniform hierarchical matrices). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. X is a *left semi-uniform matrix* for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and V if there is a family $B = (B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of matrices satisfying $B_b \in \mathbb{R}^{\hat{\mathcal{J}}\times K_t}_{\hat{s}}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t B_b^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{6.1}$$

The matrices $B = (B_b)_{b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ are called *right coefficient matrices*.

X is a *right semi-uniform matrix* for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and W if there is a family $A = (A_b)_{b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of matrices satisfying $A_b \in \mathbb{R}^{\mathcal{I}\times L_s}_{\hat{t}}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} A_b W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{6.2}$$

The matrices $A = (A_b)_{b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ are called *left coefficient matrices*.

Semi-uniform hierarchical matrices and $\mathcal{H}^2$-matrices share an important property: unlike hierarchical matrices, both are subspaces of a matrix space.

Remark 6.2 (Matrix subspaces). The sets

$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *) := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} : X \text{ is a left semi-uniform matrix for } \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \text{ and } V\},$$
$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W) := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} : X \text{ is a right semi-uniform matrix for } \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \text{ and } W\}$$

are subspaces of $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

Proof. We introduce the trivial cluster bases $\chi_{\mathcal{I}} := (\chi_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $\chi_{\mathcal{J}} := (\chi_s)_{s\in\mathcal{T}_{\mathcal{J}}}$, observe

$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *) = \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, \chi_{\mathcal{J}}),$$
$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W) = \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, \chi_{\mathcal{I}}, W),$$

and apply Remark 3.37. □

Our goal is to establish a connection between left and right semi-uniform matrices and $\mathcal{H}^2$-matrices. The key result is that a matrix is an $\mathcal{H}^2$-matrix if and only if it is both left and right semi-uniform.

Lemma 6.3 ($\mathcal{H}^2$- and semi-uniform matrices). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. We have $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ if and only if X is both left and right semi-uniform, i.e., if $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *)$ and $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W)$ hold.*

Proof. Let X be left and right semi-uniform. Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Due to Definition 6.1, there are matrices $A_b \in \mathbb{R}^{\mathcal{I}\times L_s}_{\hat{t}}$ and $B_b \in \mathbb{R}^{\mathcal{J}\times K_t}_{\hat{s}}$ with

$$\chi_t X \chi_s = V_t B_b^* = A_b W_s^*.$$

Let $n := \#\hat{t}$ and $k := \dim(\operatorname{range}(V_t)) \leq n$. Due to $\operatorname{range}(V_t) \subseteq \mathbb{R}^{\mathcal{I}}_{\hat{t}}$, we can find a basis $(v_i)_{i=1}^n$ of $\mathbb{R}^{\mathcal{I}}_{\hat{t}}$ which satisfies $\operatorname{range}(V_t) = \operatorname{span}\{v_1, \ldots, v_k\}$. For each $i \in \{1, \ldots, k\}$, we fix a vector $x_i \in \mathbb{R}^{K_t}$ with $v_i = V_t x_i$. Since the vectors $(v_i)_{i=1}^n$ are linear independent, the same holds for the vectors $(x_i)_{i=1}^n$.

We define the linear operator $R \in \mathbb{R}^{K_t \times \mathcal{I}}_{\hat{t}}$ by

$$Rv_i = \begin{cases} x_i & \text{if } i \leq k, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \{1, \ldots, n\}$. This definition implies $V_t R v_i = V_t x_i = v_i$ for all $i \in \{1, \ldots, k\}$, and since $(v_i)_{i=1}^k$ is a basis of range(V_t), we can conclude $V_t R V_t = V_t$.

Due to $\chi_t X \chi_s = V_t B_b^*$, we have

$$\chi_t X \chi_s = V_t B_b^* = V_t R V_t B_b^* = V_t R \chi_t X \chi_s,$$

and due to $\chi_t X \chi_s = A_b W_s^*$, we have

$$V_t R \chi_t X \chi_s = V_t R A_b W_s^*,$$

so we can set $S_b := R A_b$ and find

$$\chi_t X \chi_s = V_t R \chi_t X \chi_s = V_t R A_b W_s^* = V_t S_b W_s^*,$$

which proves that X is an $\mathcal{H}^2$-matrix. The rest of the proof is trivial. □

If V and W are orthogonal, the best approximations of a given matrix in the spaces $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *)$ and $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W)$ are given by orthogonal projections (with respect to the Frobenius norm).

Lemma 6.4 (Semi-uniform matrix projections). *Let V and W be orthogonal. The operators*

$$\begin{aligned} \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*} \colon \mathbb{R}^{\mathcal{I}\times\mathcal{J}} &\to \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *), \\ X &\mapsto \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_t V_t^* X \chi_s + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s, \\ \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W} \colon \mathbb{R}^{\mathcal{I}\times\mathcal{J}} &\to \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W), \\ X &\mapsto \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \chi_t X W_s W_s^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s, \end{aligned}$$

are orthogonal projections into $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *)$ and $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W)$. They satisfy

$$\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*} \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W} = \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W} = \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W} \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}.$$

Proof. For the first part of the proof, we again use the trivial cluster bases

$$\chi_{\mathcal{I}} := (\chi_t)_{t\in\mathcal{T}_{\mathcal{I}}}, \quad \chi_{\mathcal{J}} := (\chi_s)_{s\in\mathcal{T}_{\mathcal{J}}},$$

observe

$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *) = \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, \chi_{\mathcal{J}}),$$
$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W) = \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, \chi_{\mathcal{I}}, W)$$

as well as $\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*} = \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,\chi_{\mathcal{J}}}$ and $\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W} = \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\chi_{\mathcal{I}},W}$, and apply Lemma 5.5.

The second part of the proof is a simple consequence of Lemma 5.4. □

We would like to restrict our attention to left semi-uniform matrices and handle right semi-uniform matrices by switching to adjoint matrices. In order to be able to switch from a matrix X to its adjoint X^* in a meaningful way, we require the block cluster tree corresponding to X^*:

Definition 6.5 (Transposed block cluster tree). Let $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ be a block cluster tree for $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{I}}$. $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ is the *transposed block cluster tree* of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ if the following conditions hold:

- For all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$, we have $b = (s,t) \in \mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ if and only if $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.
- For all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$, we have $b = (s,t) \in \mathcal{L}^+_{\mathcal{J}\times\mathcal{I}}$ if and only if $(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

The transposed block cluster tree of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is unique and denoted by $\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}}$.

Using the transposed block cluster tree, we can replace the projection into the space of right semi-uniform matrices by the one into the space of left semi-uniform matrices:

Lemma 6.6. *Let W be orthogonal. For all $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, we have*

$$(\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X])^* = \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*].$$

Proof. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{\mathcal{J}\times\mathcal{I}} = \mathcal{T}^*_{\mathcal{I}\times\mathcal{J}}$. We denote the admissible and inadmissible leaves of $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ by $\mathcal{L}^+_{\mathcal{J}\times\mathcal{I}}$ and $\mathcal{L}^-_{\mathcal{J}\times\mathcal{I}}$. Due to the definitions in Lemma 6.4, we find

$$\begin{aligned}
(\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X])^* &= \Big(\sum_{(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \chi_t X W_s W_s^* + \sum_{(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s\Big)^* \\
&= \sum_{(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} W_s W_s^* X^* \chi_t + \sum_{(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_s X^* \chi_t \\
&= \sum_{(s,t)\in\mathcal{L}^+_{\mathcal{J}\times\mathcal{I}}} W_s W_s^* X^* \chi_t + \sum_{(s,t)\in\mathcal{L}^-_{\mathcal{J}\times\mathcal{I}}} \chi_s X^* \chi_t \\
&= \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*].
\end{aligned}$$

□

This lemma allows us to formulate estimates for the operator norm of the $\mathcal{H}^2$-matrix approximation error which depend only on row matrix projections.

Lemma 6.7 (Spectral norm estimate). *Let V and W be orthogonal cluster bases. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. Introducing $X_l := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]$ and $X_r := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X] = (\Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*])^*$, we find*

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_2 \leq \begin{cases} \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_2 + \|X_l^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X_l^*]\|_2, \\ \|X^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*]\|_2 + \|X_r - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X_r]\|_2. \end{cases}$$

Proof. Combining the triangle inequality and Lemma 6.6, we find

$$\begin{aligned} \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_2 &\leq \|X - X_l\|_2 + \|X_l - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X_l]\|_2 \\ &= \|X - X_l\|_2 + \|X_l^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X_l^*]\|_2 \end{aligned}$$

and

$$\begin{aligned} \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_2 &\leq \|X - X_r\|_2 + \|X_r - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X_r]\|_2 \\ &= \|X^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*]\|_2 + \|X_r - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X_r]\|_2. \end{aligned}$$

□

Differently from the spectral norm, the Frobenius norm of a matrix can be expressed in terms of the Frobenius norms of the subblocks of the matrix. This allows us to derive error estimates which are significantly more elegant than the ones for the operator norm.

Lemma 6.8 (Frobenius norm estimate). *Let V and W be orthogonal. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. Introducing $X_l := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}X$ and $X_r := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}X = (\Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}X^*)^*$, we find*

$$\begin{aligned} \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2 &= \|X - X_l\|_F^2 + \|X_l - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X_l]\|_F^2 \\ &\leq \|X - X_l\|_F^2 + \|X - X_r\|_F^2. \end{aligned} \tag{6.3}$$

Proof. Let $\Pi := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}$, $\Pi_l := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}$ and $\Pi_r := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}$. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. Due to Lemma 6.4, Π_l is an orthogonal projection, and we find

$$\begin{aligned} \|X - \Pi[X]\|_F^2 &= \|X - \Pi_l[X] + \Pi_l[X] - \Pi_l\Pi_r[X]\|_F^2 \\ &= \|X - \Pi_l[X]\|_F^2 + 2\langle X - \Pi_l[X], \Pi_l[X - \Pi_r X]\rangle_F \\ &\quad + \|\Pi_l[X - \Pi_r[X]]\|_F^2 \\ &= \|X - \Pi_l[X]\|_F^2 + 2\langle \Pi_l[X] - \Pi_l^2[X], X - \Pi_r[X]\rangle_F \\ &\quad + \|\Pi_l[X - \Pi_r[X]]\|_F^2 \end{aligned}$$

$$\begin{aligned}
&= \|X - \Pi_l[X]\|_F^2 + \|\Pi_l[X] - \Pi_l\Pi_r[X]\|_F^2 \\
&= \|X - \Pi_l[X]\|_F^2 + \|\Pi_l[X] - \Pi_r\Pi_l[X]\|_F^2,
\end{aligned}$$

which is the first half of our claim. For $X_e := X - \Pi_r[X]$, we have

$$\begin{aligned}
\|\Pi_l[X] - \Pi_l\Pi_r[X]\|_F^2 &= \|\Pi_l[X_e]\|_F^2 \leq \|\Pi_l[X_e]\|_F^2 + \|X_e - \Pi_l[X_e]\|_F^2 \\
&= \|\Pi_l[X_e]\|_F^2 + \|X_e\|_F^2 - 2\langle X_e, \Pi_l[X_e]\rangle_F + \|\Pi_l[X_e]\|_F^2 \\
&= 2\|\Pi_l[X_e]\|_F^2 + \|X_e\|_F^2 - 2\langle \Pi_l[X_e], \Pi_l[X_e]\rangle_F \\
&= \|X_e\|_F^2 = \|X - \Pi_r[X]\|_F^2,
\end{aligned}$$

and this completes the proof. □

Now we can conclude that, as far as error estimates in the Frobenius norm are concerned, investigating only the left matrix projections of a matrix and its transposed provides us with enough information to bound the $\mathcal{H}^2$-matrix approximation error.

Theorem 6.9 (Separation of cluster bases). *Let V and W be orthogonal. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree. For all $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, we have*

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2 \leq \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 + \|X^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*]\|_F^2.$$

Proof. Combine Lemma 6.8 with Lemma 6.6. □

We can even demonstrate that treating a matrix and its transposed separately increases the error estimate only by a small factor:

Remark 6.10. Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. We have

$$\begin{aligned}
\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 &= \|X\|_F^2 - 2\langle X, \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\rangle_F + \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 \\
&= \|X\|_F^2 - 2\langle \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X], \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\rangle_F \\
&\quad + \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 \\
&= \|X\|_F^2 - \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 \\
&\leq \|X\|_F^2 - \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 \\
&= \|X\|_F^2 - \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2 \\
&= \|X\|_F^2 - 2\langle X, \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\rangle_F + \|\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2 \\
&= \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2.
\end{aligned}$$

The same upper bound holds for $\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},*,W}[X]\|_F^2$, and we can conclude

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,*}[X]\|_F^2 + \|X^* - \Pi_{\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}},W,*}[X^*]\|_F^2 \leq 2\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},V,W}[X]\|_F^2.$$

This means that Theorem 6.9 only slightly overestimates the true approximation error. □

6.2 Total cluster bases

Due to Theorem 6.9, we can investigate row and column cluster bases independently, and it is sufficient to consider only the left semi-uniform matrices. Our goal is to find an alternative characterization of these matrices which can be used for our theoretical investigations.

We start by fixing an arbitrary matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. We intend to express it as a left semi-uniform matrix, therefore we have to construct a suitable cluster basis. If we define

$$X_{0,t} := \sum_{r\in\operatorname{row}(t)} \chi_t X \chi_r \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

the resulting family $(X_{0,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ satisfies

$$\begin{aligned}\chi_t X \chi_s &= \chi_t X \sum_{r\in\operatorname{row}(t)} \chi_r \chi_s \\ &= \Big(\sum_{r\in\operatorname{row}(t)} \chi_t X \chi_r\Big)\chi_s = X_{0,t}\chi_s \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\end{aligned}$$

due to Lemma 5.7, i.e., the condition (6.1) can be fulfilled by letting $B_b := \chi_s$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. The family $(X_{0,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ is a cluster basis with the rank distribution $(\mathcal{J})_{t\in\mathcal{T}_{\mathcal{I}}}$, but it is not nested.

Fortunately, this can be remedied by a simple trick: a cluster basis is nested (cf. (3.15)), if the cluster basis matrix for a non-leaf cluster t can be expressed in terms of the cluster basis matrices of its sons. Due to Lemma 5.7, all matrices $X_{0,t}$ correspond to disjoint blocks of X, therefore the condition (3.15) can be fulfilled by adding suitable restrictions of all predecessors of a cluster t to $X_{0,t}$ (cf. Figure 6.1). This idea gives rise to the following definition:

Definition 6.11 (Total cluster basis). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. The family $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ defined by

$$X_t := \sum_{s\in\operatorname{row}^*(t)} \chi_t X \chi_s = \sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} \chi_t X \chi_s \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}$$

is called the *total cluster basis* corresponding to X.

Using Lemma 5.7, we can easily verify that $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ is indeed a nested cluster basis and that the matrix X is a left semi-uniform matrix with respect to this cluster basis:

Lemma 6.12 (Properties of the total cluster basis). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. The total cluster basis $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ for X is a nested cluster basis with rank distribution $(\mathcal{J})_{t\in\mathcal{T}_{\mathcal{I}}}$, and its transfer matrices are given by*

$$E_t := \begin{cases} \sum_{s\in\operatorname{row}^*(t^+)} \chi_s & \text{if there exists } t^+ \in \mathcal{T}_{\mathcal{I}} \text{ with } t \in \operatorname{sons}(t^+), \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

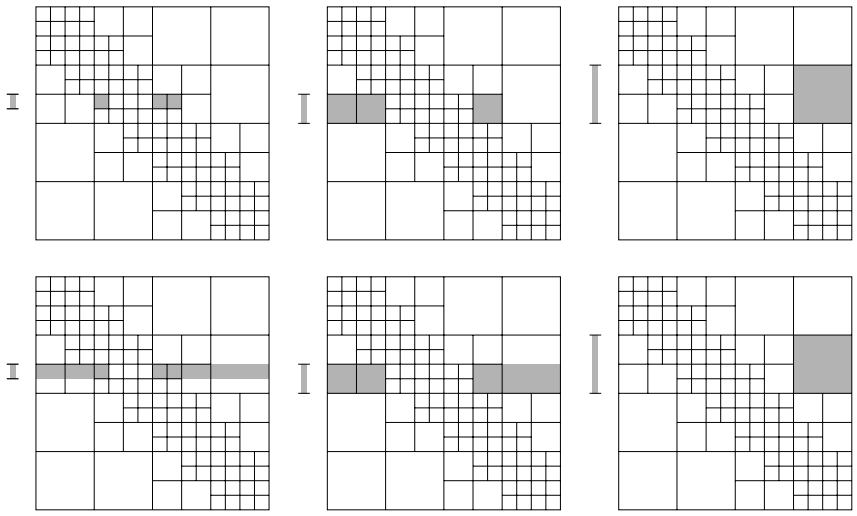


Figure 6.1. Matrices $X_{0,t}$ (top) and X_t (bottom) for the model problem.

Using the total cluster basis, X can be represented as

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} X_t \chi_s + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s, \tag{6.5}$$

i.e., X is a left semi-uniform matrix for the cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$. The long-range transfer matrices (cf. Definition 5.27) satisfy

$$E_{r,t} = \begin{cases} \sum_{s\in\operatorname{row}^*(t)} \chi_s & \text{if } r \neq t, \\ I & \text{otherwise}, \end{cases} \tag{6.6}$$

for $t \in \mathcal{T}_\mathcal{I}$ and $r \in \operatorname{sons}^*(t)$.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$ with $\operatorname{sons}(t) \neq \emptyset$. Definition 3.4 and Definition 3.24 imply

$$\chi_t = \sum_{t'\in\operatorname{sons}(t)} \chi_{t'},$$

and due to Lemma 5.7, we have $\operatorname{row}^*(t') \supseteq \operatorname{row}^*(t)$ and

$$\chi_r \chi_s = \delta_{s,r} \chi_s$$

for $t' \in \operatorname{sons}(t)$, $s \in \operatorname{row}^*(t) \subseteq \operatorname{row}^*(t')$ and $r \in \operatorname{row}^*(t')$. This means

$$\begin{aligned} X_t &= \sum_{s\in\operatorname{row}^*(t)} \chi_t X \chi_s = \sum_{t'\in\operatorname{sons}(t)} \sum_{s\in\operatorname{row}^*(t)} \chi_{t'} X \chi_s \\ &= \sum_{t'\in\operatorname{sons}(t)} \sum_{s\in\operatorname{row}^*(t)} \sum_{r\in\operatorname{row}^*(t')} \chi_{t'} X \chi_r \chi_s \end{aligned}$$

$$= \sum_{t' \in \operatorname{sons}(t)} \sum_{r \in \operatorname{row}^*(t')} \chi_{t'} X \chi_r \sum_{s \in \operatorname{row}^*(t)} \chi_s$$
$$= \sum_{t' \in \operatorname{sons}(t)} X_{t'} \sum_{s \in \operatorname{row}^*(t)} \chi_s = \sum_{t' \in \operatorname{sons}(t)} X_{t'} E_{t'}.$$

Due to Lemma 5.7 and $t \in \operatorname{pred}(t)$, the second part of the proof is trivial.

We prove (6.6) by induction on $\operatorname{level}(r) - \operatorname{level}(t) \in \mathbb{N}_0$. Let $t \in \mathcal{T}_I$ and $r \in \operatorname{sons}^*(t)$ with $\operatorname{level}(r) - \operatorname{level}(t) = 0$. This implies $t = r$, and (6.6) follows directly. Let now $n \in \mathbb{N}_0$ be such that (6.6) holds for all $t \in \mathcal{T}_I$ and all $r \in \operatorname{sons}^*(t)$ with $\operatorname{level}(r) - \operatorname{level}(t) = n$. Let $t \in \mathcal{T}_I$ and $r \in \operatorname{sons}^*(t)$ with $\operatorname{level}(r) - \operatorname{level}(t) = n + 1$. Due to the definition of $\operatorname{sons}^*(t)$, we can find $t^- \in \operatorname{sons}(t)$ such that $r \in \operatorname{sons}^*(t^-)$ holds. We have $\operatorname{level}(r) - \operatorname{level}(t^-) = n$, so we can apply the induction assumption in order to find

$$E_{r,t^-} = \begin{cases} \sum_{s \in \operatorname{row}^*(t^-)} \chi_s & \text{if } s \neq t^-, \\ I & \text{otherwise,} \end{cases}$$

and (6.4) implies

$$E_{t^-} = \sum_{s \in \operatorname{row}^*(t)} \chi_s.$$

Lemma 5.7 yields $\operatorname{row}^*(t^-) \supseteq \operatorname{row}^*(t)$, and we can conclude

$$E_{r,t} = E_{r,t^-} E_{t^-} = \sum_{s \in \operatorname{row}^*(t)} \chi_s. \qquad \square$$

In general, the total cluster basis is not useful in practical applications, since its rank is too large by far (although there are some efficient algorithms that are based on working *implicitly* with the total cluster basis, cf. Section 6.4).

On the other hand, the total cluster basis is a useful tool when investigating whether an arbitrary matrix can be approximated by an $\mathcal{H}^2$-matrix. In order to illustrate this, we require the following general result concerning restrictions of cluster basis matrices:

Lemma 6.13 (Restriction). *Let $V = (V_t)_{t \in \mathcal{T}_I}$ be a nested cluster basis. Let $t \in \mathcal{T}_I$ and $r \in \operatorname{sons}^*(t)$. With the long-range transfer matrix $E_{r,t}$ introduced in Definition 5.27, we have*

$$\chi_r V_t = V_r E_{r,t}. \tag{6.7}$$

Proof. By induction on $\#\operatorname{sons}^*(t) \in \mathbb{N}$. Let $t \in \mathcal{T}_I$ and $r \in \operatorname{sons}^*(t)$. If $\#\operatorname{sons}^*(t) = 1$ holds, we have $\operatorname{sons}(t) = \emptyset$ and therefore $t = r$. The definition of the cluster basis implies $\chi_r V_t = \chi_t V_t = V_t = V_t E_{t,t}$.

Let $n \in \mathbb{N}$ be such that (6.7) holds for all $t \in \mathcal{T}_I$ with $\#\operatorname{sons}^*(t) \leq n$. Let $t \in \mathcal{T}_I$ with $\#\operatorname{sons}^*(t) = n + 1$. Let $r \in \operatorname{sons}^*(t)$. If $r = t$, we can proceed as before. If $r \neq t$, there is a $t^- \in \operatorname{sons}(t)$ with $r \in \operatorname{sons}^*(t^-)$, and we find

$$\chi_r V_t = \chi_r \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'} = \chi_r V_{t^-} E_{t^-}.$$

Due to $\#\operatorname{sons}(t^-) \leq \#\operatorname{sons}(t) - 1 = n$, we can apply the induction assumption to find $\chi_r V_{t^-} = V_r E_{r,t^-}$, which implies

$$\chi_r V_t = \chi_r V_{t^-} E_{t^-} = V_r E_{r,t^-} E_{t^-} = V_r E_{r,t},$$

and concludes the induction. □

Using this result, total cluster bases can be used to characterize left semi-uniform matrices without the need for explicitly given cluster bases. The basic result is that the rank of the total cluster basis matrices is bounded if the corresponding matrix is left semi-uniform:

Lemma 6.14. *Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a nested cluster basis. We fix a left semi-uniform matrix $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *)$ and its total cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$. Then we have* $\operatorname{range}(X_t) \subseteq \operatorname{range}(V_t)$ *and therefore* $\operatorname{rank}(X_t) \leq \operatorname{rank}(V_t)$ *for all $t \in \mathcal{T}_\mathcal{I}$.*

Proof. Let $t \in \mathcal{T}_\mathcal{I}$, let $t^+ \in \operatorname{pred}(t)$ and $s \in \operatorname{row}^+(t^+)$. We have $b := (t^+, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ by definition, and since X is left semi-uniform, we can find a matrix $B_b \in \mathbb{R}^{\mathcal{J}\times K_t}_{\hat{s}}$ satisfying $\chi_{t^+} X \chi_s = V_{t^+} B_b^*$. Lemma 6.13 implies

$$\chi_t X \chi_s = \chi_t \chi_{t^+} X \chi_s = \chi_t V_{t^+} B_b^* = V_t E_{t,t^+} B_b^*. \tag{6.8}$$

For X_t, this means

$$X_t = \sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} \chi_t X \chi_s = V_t \Big(\sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} E_{t,t^+} B^*_{(t^+,s)} \Big), \tag{6.9}$$

i.e., the range of X_t is a subspace of the range of V_t. □

The following lemma can be considered the counterpart of the previous one: it states that the matrix X is left semi-uniform if the ranks of the total cluster basis matrices are bounded.

Lemma 6.15. *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. There exists an orthogonal nested cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ with rank distribution $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ satisfying $\#L_t = \operatorname{rank}(Q_t) = \operatorname{rank}(X_t)$ for all $t \in \mathcal{T}_\mathcal{I}$ such that $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, Q, *)$ holds.*

Proof. Due to Lemma 6.12, $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ is a nested cluster basis. Therefore we can apply Corollary 5.35 to it in order to find a nested orthogonal cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ with $Q_t Q_t^* X_t = X_t$ and $\#L_t = \operatorname{rank}(Q_t) = \operatorname{rank}(X_t)$ for all $t \in \mathcal{T}_\mathcal{I}$. We use (6.5) to conclude

$$\begin{aligned} X &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} X_t \chi_s + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s \\ &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} Q_t Q_t^* X_t \chi_s + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s \end{aligned}$$

$$= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} Q_t B_b^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s$$

with $B_b := (Q_t^* X_t \chi_s)^* \in \mathbb{R}^{\mathcal{J}\times L_t}_{\hat{s}}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. □

In the standard situation, i.e., if each cluster t corresponds to a subdomain $\Omega_t \subseteq \Omega \subseteq \mathbb{R}^d$ and each admissible block $b = (t,s)$ satisfies the condition

$$\max\{\operatorname{diam}(\Omega_t), \operatorname{diam}(\Omega_s)\} \leq 2\eta \operatorname{dist}(\Omega_t, \Omega_s),$$

the total cluster basis has a geometric interpretation: for $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{row}^*(t)$, we can find a cluster $t^+ \in \operatorname{pred}(t)$ with $s \in \operatorname{row}^+(t^+)$. Due to the admissibility condition, this means

$$\operatorname{diam}(\Omega_t) \leq \operatorname{diam}(\Omega_{t^+}) \leq 2\eta \operatorname{dist}(\Omega_{t^+}, \Omega_s) \leq 2\eta \operatorname{dist}(\Omega_t, \Omega_s),$$

i.e., for all $s \in \operatorname{row}^+(t)$, the set Ω_s is contained in the set

$$\Omega_t^c := \{x \in \Omega \ : \ \operatorname{diam}(\Omega_t) \leq 2\eta \operatorname{dist}(\Omega_t, x)\}.$$

In this sense, the matrix X_t describes the transfer of information from the "geometric farfield" Ω_t^c (cf. Figure 6.2) of the cluster t into the domain Ω_t.

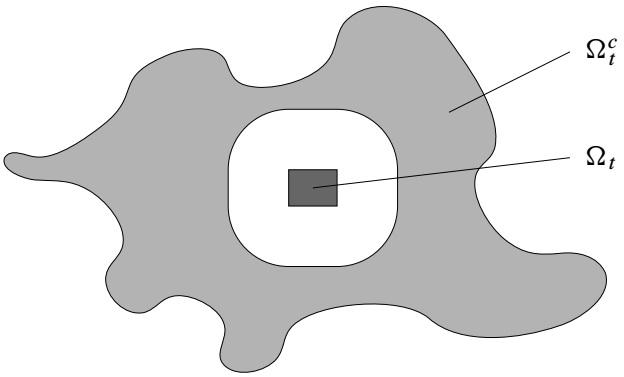


Figure 6.2. Geometric interpretation of total cluster bases: the matrix X_t describes the transfer of information from the farfield Ω_t^c to the cluster Ω_t.

6.3 Approximation by semi-uniform matrices

We have seen that a matrix X is left semi-uniform if and only if the ranks of the total cluster basis matrices $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ are bounded. In practice, we frequently encounter

matrices which do not have low rank, but which can be *approximated* by low-rank matrices, therefore we have to investigate whether the existence of low-rank approximations of the matrices $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ implies that we can approximate X by a left semi-uniform matrix.

Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, and let $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ be its total cluster basis. Let $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a rank distribution, and let $A = (A_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(B_t)_{t\in\mathcal{T}_\mathcal{I}}$ be families of matrices satisfying

$$A_t \in \mathbb{R}^{\mathcal{I}\times K_t}_{\hat{t}} \quad \text{and} \quad B_t \in \mathbb{R}^{\mathcal{J}\times K_t} \tag{6.10}$$

for all $t \in \mathcal{T}_\mathcal{I}$.

Let us assume that the low-rank matrices $A_t B_t^*$ approximate X_t, i.e., that a suitable norm of $X_t - A_t B_t^*$ is small enough for all $t \in \mathcal{T}_\mathcal{I}$.

In order to find a left semi-uniform approximation of X, we have to construct an orthogonal nested cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$. Once we have this basis, we can find the best approximation of X in $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, Q, *)$ by applying the orthogonal projection $\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}$. Obviously, the challenging task is the construction of Q.

We use a variant of the orthogonalization Algorithm 16: in leaf clusters $t \in \mathcal{T}_\mathcal{I}$, we construct the orthogonal cluster basis matrices by computing the Householder factorization $A_t = Q_t R_t$ of A_t.

Let now $t \in \mathcal{T}_\mathcal{I}$ with $\tau := \#\operatorname{sons}(t) > 0$. We let $\{t_1, \ldots, t_\tau\} := \operatorname{sons}(t)$ and assume that we have already constructed orthogonal cluster basis matrices $Q_{t'} \in \mathbb{R}^{\mathcal{I}\times L_{t'}}_{\hat{t}'}$ with disjoint index sets $L_{t'}$ for all $t' \in \operatorname{sons}(t)$.

As in the case of the truncation Algorithm 19, we have to replace A_t by its projection

$$\bar{A}_t := U_t U_t^* A_t = U_t \begin{pmatrix} Q_{t_1}^* A_t \\ \vdots \\ Q_{t_\tau}^* A_t \end{pmatrix} = \sum_{t'\in\operatorname{sons}(t)} Q_{t'} Q_{t'}^* A_t$$

using the orthogonal matrix

$$U_t := \begin{pmatrix} Q_{t_1} & \ldots & Q_{t_\tau} \end{pmatrix} \in \mathbb{R}^{\mathcal{I}\times M_t}_{\hat{t}} \quad \text{for} \quad M_t := \bigcup_{t'\in\operatorname{sons}(t)} L_{t'}$$

we have introduced in (5.12). Using

$$\hat{A}_t := \begin{pmatrix} Q_{t_1}^* A_t \\ \vdots \\ Q_{t_\tau}^* A_t \end{pmatrix} \in \mathbb{R}^{M_t\times K_t},$$

we find $\bar{A}_t = U_t \hat{A}_t$. Now we can proceed as in the orthogonalization algorithm: the Householder factorization $\hat{A}_t = \hat{Q}_t R_t$ of $\hat{A}_t$ yields an index set L_t and an orthogonal matrix $\hat{Q}_t \in \mathbb{R}^{M_t\times L_t}$, and the orthogonality of U_t implies that $Q_t := U_t \hat{Q}_t$ is also

orthogonal. The corresponding transfer matrices can be defined by $F_{t'} := \widehat{Q}_t|_{L_{t'} \times L_t}$, since this yields

$$Q_t = U_t \widehat{Q}_t = \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau} \end{pmatrix} \begin{pmatrix} F_{t_1} \\ \vdots \\ F_{t_\tau} \end{pmatrix} = \sum_{t' \in \operatorname{sons}(t)} Q_{t'} F_{t'}.$$

The resulting recursive procedure is given in Algorithm 23.

Algorithm 23. Converting a general cluster basis into a nested one.

```
procedure Nestify(t, A, var Q, R, L)
if sons(t) = ∅ then
  Householder(A_t, t̂, Q_t, R_t, L_t)
else
  M_t ← ∅;
  for t' ∈ sons(t) do
    Nestify(t', A, Q, R, L);
    M_t ← M_t ∪̇ L_t'
  end for;
  for t' ∈ sons(t) do
    Â_t|_{L_t' × K_t} ← Q_t'^* A_t
  end for;
  Householder(Â_t, M_t, Q̂_t, R_t, L_t);
  for t' ∈ sons(t) do
    F_t' ← Q̂_t|_{L_t' × L_t}
  end for
end if
```

Since Algorithm 23 uses a projection to ensure that the resulting cluster basis is nested, we have to analyze the resulting error.

Theorem 6.16 (Matrix approximation). *Let $Q = (Q_t)_{t \in \mathcal{T}_I}$ be a nested orthogonal cluster basis for $\mathcal{T}_I$. Let $X \in \mathbb{R}^{I \times J}$. Let $(X_t)_{t \in \mathcal{T}_I}$ be the total cluster basis for X, and let*

$$\bar{X}_t := \begin{cases} U_t U_t^* X_t & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ X_t & \text{otherwise}, \end{cases}$$

for all $t \in \mathcal{T}_I$ (compare (5.21)). The error of the left matrix projection is given by

$$X - \Pi_{\mathcal{T}_{I \times J}, Q, *}[X] = \sum_{t \in \mathcal{T}_I} \bar{X}_t - Q_t Q_t^* \bar{X}_t \tag{6.11}$$

and satisfies

$$\|(X - \Pi_{\mathcal{T}_{I \times J}, Q, *}[X])x\|_2^2 = \sum_{t \in \mathcal{T}_I} \|(\bar{X}_t - Q_t Q_t^* \bar{X}_t)x\|_2^2 \quad \text{for all } x \in \mathbb{R}^J. \tag{6.12}$$

Proof. We have

$$X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X] = \sum_{(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s - Q_t Q_t^* X \chi_s$$
$$= \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^+(t)} (\chi_t X - Q_t Q_t^* X)\chi_s = \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^+(t)} (X_t - Q_t Q_t^* X_t)\chi_s.$$

As in Lemma 5.25, we let

$$D_t := \bar{X}_t - Q_t Q_t^* \bar{X}_t$$

for all $t \in \mathcal{T}_\mathcal{I}$ and apply Lemma 5.29 to each term in the above sum in order to find

$$X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X] = \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^+(t)} \sum_{r\in\operatorname{sons}^*(t)} D_r E_{r,t} \chi_s$$
$$= \sum_{r\in\mathcal{T}_\mathcal{I}} \sum_{t\in\operatorname{pred}(r)} \sum_{s\in\operatorname{row}^+(t)} D_r E_{r,t} \chi_s,$$

where $E_{r,t}$ are the long-range transfer matrices of the total cluster basis (cf. (6.6)).

Let $r \in \mathcal{T}_\mathcal{I}$, $t \in \operatorname{pred}(r)$ and $s \in \operatorname{row}^+(t)$. Due to Lemma 6.12, we have

$$E_{r,t}\chi_s = \chi_s E_{r,t} = \begin{cases} \chi_s \sum_{s'\in\operatorname{row}^*(t)} \chi_{s'} & \text{if } r \neq t, \\ \chi_s & \text{otherwise,} \end{cases}$$

and $s \in \operatorname{row}^+(t) \subseteq \operatorname{row}^*(t)$ implies $E_{r,t}\chi_s = \chi_s$, which leads to

$$X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X] = \sum_{r\in\mathcal{T}_\mathcal{I}} \sum_{t\in\operatorname{pred}(r)} \sum_{s\in\operatorname{row}^+(t)} D_r \chi_s = \sum_{r\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^*(r)} D_r \chi_s = \sum_{r\in\mathcal{T}_\mathcal{I}} D_r.$$

This is the desired equation (6.11). According to Lemma 5.26, the ranges of the matrices D_r are pairwise orthogonal and we get

$$\|(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X])x\|_2^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \|D_t x\|_2^2,$$

which concludes the proof. □

With this general approximation result, we can now investigate the error introduced by Algorithm 23.

Corollary 6.17 (Approximation error bound). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, and let $(A_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(B_t)_{t\in\mathcal{T}_\mathcal{I}}$ be as in (6.10). Let $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ be the nested orthogonal cluster basis constructed by Algorithm 23. We have*

$$\|(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X])x\|_2^2 \leq \sum_{t\in\mathcal{T}_\mathcal{I}} \|(X_t - A_t B_t^*)x\|_2^2 \tag{6.13}$$

for all $x \in \mathbb{R}^\mathcal{J}$.

Proof. Let $X_l := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\mathcal{Q},*}[X]$ and let $x \in \mathbb{R}^{\mathcal{J}}$. According to Theorem 6.16, we have to find bounds for $\|(\bar{X}_t - Q_t Q_t^* \bar{X}_t)x\|_2^2$.

Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{sons}(t) = \emptyset$, we have $\bar{X}_t = X_t$ and $Q_t R_t = A_t$ by definition and find

$$\begin{aligned} X_t - Q_t Q_t^* X_t &= X_t - A_t B_t^* + A_t B_t^* - Q_t Q_t^* X_t \\ &= X_t - A_t B_t^* + Q_t (R_t B_t^* - Q_t^* X_t) \\ &= X_t - A_t B_t^* - Q_t Q_t^* (X_t - A_t B_t^*) \\ &= (I - Q_t Q_t^*)(X_t - A_t B_t^*). \end{aligned}$$

Since $Q_t Q_t^*$ is an orthogonal projection, so is $I - Q_t Q_t^*$, and we conclude

$$\|(X_t - Q_t Q_t^* X_t)x\|_2 \leq \|(X_t - A_t B_t^*)x\|_2. \tag{6.14}$$

If $\operatorname{sons}(t) \neq \emptyset$, we have $\bar{X}_t = U_t U_t^* X_t$ and $Q_t R_t = U_t \widehat{Q}_t R_t = U_t \widehat{A}_t = \bar{A}_t$ and observe

$$\begin{aligned} \bar{X}_t - Q_t Q_t^* \bar{X}_t &= \bar{X}_t - \bar{A}_t B_t^* + \bar{A}_t B_t^* - Q_t Q_t^* \bar{X}_t \\ &= \bar{X}_t - \bar{A}_t B_t^* + Q_t (R_t B_t^* - Q_t^* \bar{X}_t) \\ &= \bar{X}_t - \bar{A}_t B_t^* - Q_t Q_t^* (\bar{X}_t - \bar{A}_t B_t^*) \\ &= (I - Q_t Q_t^*)(\bar{X}_t - \bar{A}_t B_t^*). \end{aligned}$$

Using the fact that $I - Q_t Q_t^*$ and $U_t U_t^*$ are orthogonal projections, we find

$$\begin{aligned} \|(\bar{X}_t - Q_t Q_t^* \bar{X}_t)x\|_2 &= \|(I - Q_t Q_t^*)(\bar{X}_t - \bar{A}_t B_t^*)x\|_2 \leq \|(\bar{X}_t - \bar{A}_t B_t^*)x\|_2 \\ &= \|U_t U_t^* (X_t - A_t B_t^*)x\|_2 \leq \|(X_t - A_t B_t^*)x\|_2. \end{aligned} \tag{6.15}$$

Combining (6.14) and (6.15) with Theorem 6.16 yields (6.13). □

Corollary 6.18 (Approximation error). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, and let $(A_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(B_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be as in (6.10). There is a nested orthogonal cluster basis $\mathcal{Q} = (Q_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ (the one defined by Algorithm 23) such that we have*

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\mathcal{Q},*}[X]\|_2^2 \leq \sum_{t\in\mathcal{T}_{\mathcal{I}}} \|X_t - A_t B_t^*\|_2^2 \tag{6.16a}$$

and

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},\mathcal{Q},*}[X]\|_F^2 \leq \sum_{t\in\mathcal{T}_{\mathcal{I}}} \|X_t - A_t B_t^*\|_F^2. \tag{6.16b}$$

Proof. Let $\mathcal{Q} = (Q_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be the orthogonal nested cluster basis defined by applying Algorithm 23 to the family $(A_t)_{t\in\mathcal{T}_{\mathcal{I}}}$.

We denote the best left semi-uniform approximation of the matrix X in the space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, Q, *)$ by $X_l := \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]$. Due to Corollary 6.17, we have

$$\|(X - X_l)x\|_2^2 \overset{(6.13)}{\leq} \sum_{t\in\mathcal{T}_\mathcal{I}} \|(X_t - A_t B_t^*)x\|_2^2 \leq \sum_{t\in\mathcal{T}_\mathcal{I}} \|X_t - A_t B_t^*\|_2^2 \|x\|_2^2$$

for all $x \in \mathbb{R}^\mathcal{J}$, and this implies (6.16a).

In order to prove (6.16b), we proceed as in Theorem 5.30: for all $j \in \mathcal{J}$, we introduce the canonical unit vectors $e^j \in \mathbb{R}^\mathcal{J}$ by $e_i^j = \delta_{ij}$. The Frobenius norm is given by

$$\begin{aligned}\|X - X_l\|_F^2 &= \sum_{j\in\mathcal{J}} \|(X - X_l)e^j\|_2^2 \overset{(6.13)}{\leq} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}_\mathcal{I}} \|(X_t - A_t B_t^*)e^j\|_2^2 \\ &= \sum_{t\in\mathcal{T}_\mathcal{I}} \|X_t - A_t B_t^*\|_F^2,\end{aligned}$$

and this is the desired estimate. □

We can combine this result with Theorem 6.9 in order to prove that a matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ can be approximated efficiently by an $\mathcal{H}^2$-matrix if the total cluster bases of X and X^* can be approximated by low-rank matrices.

6.4 General compression algorithm

Up to this point, we have only investigated the approximability of an arbitrary matrix from a theoretical point of view: we know that a matrix can be approximated by a left semi-uniform matrix if the corresponding total cluster bases can be approximated by low-rank matrices.

Let us now consider the practical side of the problem: how can we construct a good left semi-uniform approximation of an arbitrary matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$? A straightforward approach could be based on the construction of the previous section: we construct low-rank approximations of the total cluster basis matrices $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ by the singular value decomposition (cf. Lemma 5.19) or a similar technique and then apply Algorithm 23 to create a nested cluster basis.

The major disadvantage of this procedure is that finding low-rank approximations of all total cluster basis matrices leads to a rather high algorithmic complexity: constructing optimal low-rank approximations by singular value decompositions typically involves a *cubic* complexity in the number $n_\mathcal{I}$ of degrees of freedom, and although more efficient and less accurate techniques like rank-revealing factorization [47], [33] or cross-approximation [104], [7] schemes can reduce the complexity, they will also not reach the optimal order of complexity.

Lemma 6.12 suggests a more efficient approach: the total cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ is a nested cluster basis, and due to Theorem 6.16, a good approximation of this cluster basis by an orthogonal nested cluster basis will yield a good approximation of the matrix X in left semi-uniform representation.

The problem of approximating a nested cluster basis is solved by the truncation Algorithm 19, therefore applying this procedure to the total cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ can be expected to yield good results.

According to Lemma 6.12, the transfer matrices of the total cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ are given by

$$E_t := \begin{cases} \sum_{s\in\operatorname{row}^*(t^+)} \chi_s & \text{if there exists } t^+ \in \mathcal{T}_\mathcal{I} \text{ with } t \in \operatorname{sons}(t^+), \\ 0 & \text{otherwise,} \end{cases}$$

for all $t \in \mathcal{T}_\mathcal{I}$, therefore applying Algorithm 19 to the total cluster basis leads to Algorithm 24. We recall that in this algorithm, as in Algorithm 19, the matrices $R_t := Q_t^* X_t$ describe the cluster operator used to switch from the original total cluster basis X_t to the truncated orthogonal cluster basis Q_t and that the matrices $\widehat{X}_t$ are given by

$$\widehat{X}_t = U_t^* X_t = \begin{pmatrix} R_{t_1} \\ \vdots \\ R_{t_\tau} \end{pmatrix} \sum_{s\in\operatorname{row}^*(t)} \chi_s \in \mathbb{R}^{M_t\times\mathcal{J}}$$

for $\tau := \#\operatorname{sons}(t)$, $\{t_1,\ldots,t_\tau\} := \operatorname{sons}(t)$ and $M_t := L_{t_1} \dot\cup \cdots \dot\cup L_{t_\tau}$.

Algorithm 24 looks fairly simple from a mathematical point of view, but its implementation can be challenging since the total cluster basis matrices X_t is usually not directly available.

Let us therefore now consider a more practical variant of the algorithm. We introduce block-related counterparts of the matrices X_t, $\widehat{X}_t$ and R_t by defining

$$X_{t,s} := \chi_t X \chi_s, \quad \widehat{X}_{t,s} := \widehat{X}_t \chi_s, \quad R_{t,s} := R_t \chi_s \quad \text{for all } s \in \operatorname{row}^*(t).$$

Due to the definition of X_t, we have

$$X_t = \sum_{s\in\operatorname{row}^*(t)} X_{t,s},$$

and this implies

$$\widehat{X}_t = \sum_{s\in\operatorname{row}^*(t)} \widehat{X}_{t,s}, \quad R_t = \sum_{s\in\operatorname{row}^*(t)} R_{t,s}$$

In the data structures typically used when implementing $\mathcal{H}$- and $\mathcal{H}^2$-matrix algorithms, the set

$$\operatorname{row}^*(t) = \bigcup_{t^+\in\operatorname{pred}(t)} \operatorname{row}^+(t^+)$$

Algorithm 24. Theoretical algorithm for finding adaptive cluster bases.

```
procedure AdaptiveDenseSimple(t, X, ϵ, var Q, R)
if sons(t) = ∅ then
  Lowrank(X_t, t̂, ϵ_t, Q_t, R_t, L_t)                    {Algorithm 18}
else
  M_t ← ∅;
  for t' ∈ sons(t) do
    AdaptiveDenseSimple(t', X, ϵ, Q, R);
    M_t ← M_t ∪̇ L_t'
  end for;
  X̂_t ← 0 ∈ ℝ^{M_t × 𝒥};
  for t' ∈ sons(t) do
    X̂_t|_{L_t' × 𝒥} ← R_t' Σ_{s∈row*(t)} χ_s
  end for;
  Lowrank(X̂_t, t̂, ϵ_t, Q̂_t, R_t, L_t);                  {Algorithm 18}
  for t' ∈ sons(t) do
    F_t' ← Q̂_t|_{L_t' × L_t}
  end for
end if
```

is not directly available, therefore we have to construct it from $\operatorname{row}^+(t)$ by using the recursion

$$r_t := \begin{cases} r_{t^+} \cup \operatorname{row}^+(t) & \text{if there is a cluster } t^+ \in \mathcal{T}_{\mathcal{I}} \text{ with } t \in \operatorname{sons}(t^+), \\ \operatorname{row}^+(t) & \text{otherwise, i.e., if } t \text{ is the root of } \mathcal{T}_{\mathcal{I}}. \end{cases} \tag{6.17}$$

We can see that $r_t = \operatorname{row}^*(t)$ holds for all $t \in \mathcal{T}_{\mathcal{I}}$.

We express X_t, $\widehat{X}_t$ and R_t by their block counterparts $X_{t,s}$, $\widehat{X}_{t,s}$ and $R_{t,s}$ and use the recursive definition (6.17) to determine r_t. This results in the practical Algorithm 25 for the construction of an adaptive cluster basis.

Remark 6.19. Algorithm 25 actually computes not only the orthogonal nested cluster basis $Q = (Q_t)_{t\in\mathcal{T}_{\mathcal{I}}}$, but also the matrices $B = (B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ corresponding to the best approximation of X in $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, Q, *)$: for $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we have

$$R_{t,s} = Q_t^* X_{t,s} = Q_t^* X \chi_s$$

and conclude

$$Q_t Q_t^* X \chi_s = Q_t R_{t,s},$$

i.e., $B_b = R_{t,s}^*$. □

Algorithm 25. Algorithm for finding adaptive cluster bases for dense matrices.

procedure AdaptiveDense(t, r_{t^+}, X, ϵ, **var** Q, R, L)
$r_t \leftarrow r_{t^+} \cup \operatorname{row}^+(t)$;
if $\operatorname{sons}(t) = \emptyset$ **then**
  $X_t \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\hat{t}}$;
  for $s \in r_t$ **do**
    $X_{t,s} \leftarrow \chi_t X \chi_s$;
    $X_t \leftarrow X_t + X_{t,s}$
  end for;
  Lowrank(X_t, $\hat{t}$, ϵ_t, Q_t, R_t, L_t); {Algorithm 18}
  for $s \in r_t$ **do**
    $R_{t,s} \leftarrow R_t \chi_s$
  end for
else
  $M_t \leftarrow \emptyset$;
  for $t' \in \operatorname{sons}(t)$ **do**
    AdaptiveDense(t', r_t, X, ϵ, Q, R, L);
    $M_t \leftarrow M_t \dot{\cup} L_{t'}$
  end for;
  $\widehat{X}_t \leftarrow 0 \in \mathbb{R}^{M_t\times\mathcal{J}}$;
  for $s \in r_t$ **do**
    $\widehat{X}_{t,s} \leftarrow 0 \in \mathbb{R}^{M_t\times\mathcal{J}}$;
    for $t' \in \operatorname{sons}(t)$ **do**
      $\widehat{X}_{t,s}|_{L_{t'}\times\mathcal{J}} \leftarrow R_{t',s}$
    end for;
    $\widehat{X}_t \leftarrow \widehat{X}_t + \widehat{X}_{t,s}$
  end for;
  Lowrank($\widehat{X}_t$, $\hat{t}$, ϵ_t, $\widehat{Q}_t$, R_t, L_t); {Algorithm 18}
  for $t' \in \operatorname{sons}(t)$ **do**
    $F_{t'} \leftarrow \widehat{Q}_t|_{L_{t'}\times L_t}$
  end for;
  for $s \in r_t$ **do**
    $R_{t,s} \leftarrow R_t \chi_s$
  end for
end if

Lemma 6.20 (Complexity). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, let $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ be the rank distribution computed by Algorithm 25, and let*

$$l_t := \begin{cases} \max\{\#\hat{t}, \#L_t\} & \text{if } \operatorname{sons}(t) = \emptyset, \\ \max\left\{\sum_{t'\in\operatorname{sons}(t)} \#L_{t'}, \#L_t\right\} & \text{otherwise} \end{cases} \tag{6.18}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$. Algorithm 25 requires not more than

$$C_{\mathrm{pr}} n_{\mathcal{J}} \sum_{t \in \mathcal{T}_{\mathcal{I}}} l_t^2$$

operations. If L is $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha + \beta)^r n_{\mathcal{I}} n_{\mathcal{J}})$.

Proof. Lemma 5.7 states that all $s \in \operatorname{row}^*(t)$ correspond to disjoint index sets $\hat{s}$, therefore constructing the matrices X_t and $\widehat{X}_t$ requires no arithmetic operations, we only have to copy coefficients.

Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{sons}(t) = \emptyset$, Algorithm 25 computes the singular value decomposition of the matrix X_t and determines Q_t, R_t and L_t. Due to Lemma 5.21, this takes not more than

$$C_{\mathrm{pr}}(\#\hat{t}) n_{\mathcal{J}} \min\{\#\hat{t}, n_{\mathcal{J}}\} \leq C_{\mathrm{pr}}(\#\hat{t})^2 n_{\mathcal{J}} \leq C_{\mathrm{pr}} l_t^2 n_{\mathcal{J}}$$

operations.

If $\operatorname{sons}(t) \neq \emptyset$, the algorithm computes the singular value decomposition of the matrix $\widehat{X}_t$ and determines $\widehat{Q}_t$, R_t and L_t. Since the matrix $\widehat{X}_t$ has only M_t rows and since

$$\#M_t = \sum_{t' \in \operatorname{sons}(t)} \#L_{t'} \leq l_t$$

holds by definition, the computation requires not more than

$$C_{\mathrm{pr}}(\#M_t) n_{\mathcal{J}} \min\{\#M_t, n_{\mathcal{J}}\} \leq C_{\mathrm{pr}}(\#M_t)^2 n_{\mathcal{J}} \leq C_{\mathrm{pr}} l_t^2 n_{\mathcal{J}}$$

operations. Once again we can use the Lemmas 3.45 and 3.48 to complete the proof. □

As in the case of the conversion of dense matrices to $\mathcal{H}^2$-matrices considered in Lemma 5.8, the number of operations required for constructing the cluster basis grows quadratically with the matrix dimension. Since the matrix is represented by $n_{\mathcal{I}} n_{\mathcal{J}}$ coefficients, this can be considered to be the optimal rate.

Let us now investigate the error introduced by the compression Algorithm 25. As in Theorem 5.30, we extend the notation to cover leaf as well as non-leaf clusters by introducing

$$\widehat{X}_t := \begin{cases} \begin{pmatrix} R_{t_1} \\ \dots \\ R_{t_\tau} \end{pmatrix} \sum_{s \in \operatorname{row}^*(t)} \chi_s = U_t^* X_t & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ X_t & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

$$\widehat{Q}_t := \begin{cases} \begin{pmatrix} F_{t_1} \\ \dots \\ F_{t_\tau} \end{pmatrix} & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ Q_t & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Since the matrices $(\widehat{X}_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(\widehat{Q}_t)_{t\in\mathcal{T}_\mathcal{I}}$ are computed explicitly during the course of Algorithm 25, an error estimate based on these quantities is desirable, since this gives us the possibility to ensure that the error is below a given tolerance.

Theorem 6.21 (Compression error). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. Let $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ be the nested orthogonal cluster basis constructed by Algorithm 25. We have*

$$\|(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X])x\|_2^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \|(\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t^*\widehat{X}_t)x\|_2^2 \quad \text{for all } x \in \mathbb{R}^{\mathcal{J}}.$$

Proof. In order to be able to apply Theorem 6.16, we have to derive bounds for the error terms appearing on the right-hand side of equation (6.12). Let $x \in \mathbb{R}^{\mathcal{J}}$ and $t \in \mathcal{T}_\mathcal{I}$. If $\operatorname{sons}(t) = \emptyset$ holds, we have

$$\|(\bar{X}_t - Q_tQ_t^*\bar{X}_t)x\|_2 = \|(X_t - Q_tQ_t^*X_t)x\|_2 = \|(\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t^*\widehat{X}_t)x\|_2 \tag{6.19}$$

by definition. If, on the other hand, $\operatorname{sons}(t) \neq \emptyset$ holds, we find

$$\begin{aligned}\|(\bar{X}_t - Q_tQ_t^*\bar{X}_t)x\|_2 &= \|(U_tU_t^*X_t - U_t\widehat{Q}_t\widehat{Q}_t^*U_t^*U_tU_t^*X_t)x\|_2\\ &= \|U_t(U_t^*X_t - \widehat{Q}_t\widehat{Q}_t^*U_t^*X_t)x\|_2\end{aligned}$$

for all $x \in \mathbb{R}^{\mathcal{J}}$. The orthogonality of U_t and the equation $U_t^*X_t = \widehat{X}_t$ imply

$$\|(\bar{X}_t - Q_tQ_t^*\bar{X}_t)x\|_2 = \|(U_t^*X_t - \widehat{Q}_t\widehat{Q}_t^*U_t^*X_t)x\|_2 = \|(\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t^*\widehat{X}_t)x\|_2 \tag{6.20}$$

for all $x \in \mathbb{R}^{\mathcal{J}}$. Combining (6.19) and (6.20) with Theorem 6.16 concludes the proof. □

Corollary 6.22 (Error control). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$. Let $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ be the nested orthogonal cluster basis constructed by Algorithm 25. For all $t \in \mathcal{T}_\mathcal{I}$, we let*

$$\epsilon_{2,t} := \|\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t\widehat{X}_t\|_2, \quad \epsilon_{F,t} := \|\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t\widehat{X}_t\|_F.$$

Then we have

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_2^2 \leq \sum_{t\in\mathcal{T}_\mathcal{I}} \epsilon_{2,t}^2, \quad \|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_F^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \epsilon_{F,t}^2. \tag{6.21}$$

Proof. Theorem 6.21 yields

$$\|(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}X)x\|_2^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \|(\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t^*\widehat{X}_t)x\|_2^2 \leq \sum_{t\in\mathcal{T}_\mathcal{I}} \epsilon_{2,t}^2\|x\|_2^2$$

for all $x \in \mathbb{R}^{\mathcal{J}}$. This implies the left part of (6.21).

For the proof of the right part, we once again use the canonical unit vectors $e^j \in \mathbb{R}^{\mathcal{J}}$ defined by $e^j_i := \delta_{ij}$ for all $i, j \in \mathcal{J}$ and observe

$$\begin{aligned}\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}X\|_F^2 &= \sum_{j\in\mathcal{J}} \|(X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}X)e^j\|_2^2 \\ &= \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}_{\mathcal{I}}} \|(\hat{X}_t - \hat{Q}_t\hat{Q}_t^*\hat{X}_t)e^j\|_2^2 \\ &= \sum_{t\in\mathcal{T}_{\mathcal{I}}} \|\hat{X}_t - \hat{Q}_t\hat{Q}_t^*\hat{X}_t)\|_F^2 = \sum_{t\in\mathcal{T}_{\mathcal{I}}} \epsilon_{F,t}^2. \end{aligned}$$ □

Lemma 6.23 (Quasi-optimality). *Let $Q = (Q_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be the nested orthogonal cluster basis with rank distribution $L = (L_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ computed by Algorithm 25.*

Let $A = (A_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $B = (B_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be families satisfying

$$A_t \in \mathbb{R}^{\mathcal{I}\times L_t}_{\hat{t}} \quad \text{and} \quad B_t \in \mathbb{R}^{\mathcal{J}\times L_t}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$ (compare (6.10)).

Then we have

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_2^2 \leq \sum_{t\in\mathcal{T}_{\mathcal{I}}} \|X_t - A_tB_t^*\|_2^2,$$

*i.e., the approximation constructed by Algorithm 25 is at least as good as the one resulting from applying Algorithm 23 to **arbitrary** low-rank approximations of the total cluster basis.*

Proof. In order to apply Corollary 6.22, we have to find bounds for the quantities $\epsilon_{2,t}$.

Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{sons}(t) = \emptyset$, we can use Theorem 2.5.3 in [48] to prove

$$\epsilon_{2,t} = \|X_t - Q_tQ_t^*X_t\|_2 \leq \|X_t - A_tB_t^*\|_2.$$

If $\operatorname{sons}(t) \neq \emptyset$, the same theorem implies

$$\begin{aligned}\epsilon_{2,t} &= \|\hat{X}_t - \hat{Q}_t\hat{Q}_t^*\hat{X}_t\|_2 \leq \|\hat{X}_t - \hat{A}_tB_t^*\|_2 = \|U_t(\hat{X}_t - \hat{A}_tB_t^*\|_2 \\ &= \|U_t(U_t^*X_t - U_t^*A_tB_t^*\|_2 = \|U_tU_t^*(X_t - A_tB_t^*)\|_2 \leq \|X_t - A_tB_t\|_2.\end{aligned}$$

Combining these estimates with Corollary 6.22 yields the desired result. □

Remark 6.24 (Adaptivity). Due to Lemma 5.19, we can ensure that the quantities $(\epsilon_{2,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(\epsilon_{F,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ used in Corollary 6.22 are arbitrarily small by choosing suitable ranks l_t for the clusters adaptively, e.g., by using Algorithm 18. □

6.5 Compression of hierarchical matrices

Algorithm 25 can be used to find a left semi-uniform approximation of an arbitrary matrix, but the number of operations required to do this will usually behave like $n_{\mathcal{I}} n_{\mathcal{J}}$, which is optimal considering the fact that every single entry of the matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ has to be taken into account.

Sometimes the matrix X is not given in the standard representation, but in a data-sparse format. In this case we can take advantage of the compact representation of the input matrix in order to reach a better than quadratic complexity.

Let us assume that the matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ is an $\mathcal{H}$-matrix for the admissible block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with local rank $k_{\mathcal{H}} \in \mathbb{N}$. And let us also assume that X is given in hierarchical matrix representation (3.12), i.e., that there are a family $(K_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of index sets and families $(A_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ and $(B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ satisfying $A_b \in \mathbb{R}^{\mathcal{I}\times K_b}_{\hat{t}}$, $B_b \in \mathbb{R}^{\mathcal{J}\times K_b}_{\hat{s}}$, $\#K_b \leq k_{\mathcal{H}}$ and

$$X_{t,s} = \chi_t X \chi_s = A_b B_b^*$$

for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Our goal is to take advantage of the factorized representation in order to reduce the complexity of the basic Algorithm 25.

Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ be an admissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. We apply the Householder factorization as introduced in Lemma 5.16 to the matrix B_b in order to find an index set $\hat{K}_b \subseteq \hat{s}$, an orthogonal matrix $P_b \in \mathbb{R}^{\mathcal{J}\times\hat{K}_b}_{\hat{s}}$ and a matrix $C_b \in \mathbb{R}^{\hat{K}_b\times K_b}$ with

$$B_b = P_b C_b.$$

According to the definition, we have $\#\hat{K}_b = \min\{\#\hat{s}, \#K_b\} \leq k_{\mathcal{H}}$.

We define the matrix

$$X^c_{t,s} := A_b C_b^* \in \mathbb{R}^{\mathcal{I}\times\hat{K}_b}_{\hat{t}}$$

and observe that

$$X_{t,s} = \chi_t X \chi_s = A_b B_b^* = (A_b C_b^*) P_b^* = X^c_{t,s} P_b^* \tag{6.22}$$

is a factorization of the submatrix into a left factor $X^c_{t,s}$ with only $\#\hat{K}_t = \min\{\#\hat{s}, \#K_b\}$ columns and an *orthogonal* right factor P_b (cf. Figure 6.3). Since the construction of the cluster basis $Q = (Q_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ is based only on the *left* singular vectors and the non-zero singular values, multiplications by orthogonal matrices from the right do not change the result, therefore our algorithm has to consider only the matrix $X^c_{t,s}$ instead of $X_{t,s}$. While the latter matrix has $\#\hat{s}$ columns, the former cannot have more than $\min\{\#K_b, \#\hat{s}\} \leq k_{\mathcal{H}}$, i.e., it can be handled more efficiently.

The process of using an orthogonal matrix in this way to translate a matrix into a smaller matrix that contains the same information is called *condensation*, and we refer to $X^c_{t,s}$ as a *condensed* counterpart of $X_{t,s}$.

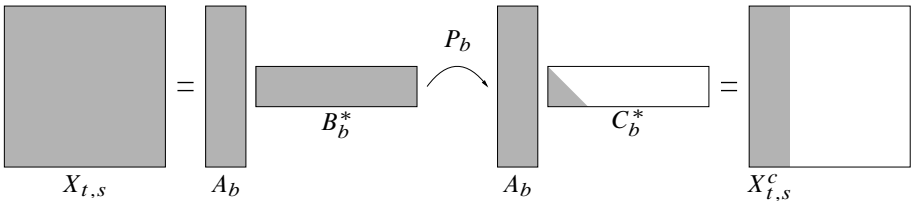


Figure 6.3. Condensation of a low-rank block.

Due to (6.22), we only have to consider the condensed matrices $X_{t,s}^c$ when constructing the adaptive cluster basis. In order to reach an efficient algorithm, it is not sufficient to replace $X_{t,s}$ by $X_{t,s}^c$ in Algorithm 25, we also have to replace $\widehat{X}_{t,s}$ and $R_{t,s}$ by suitable condensed counterparts defined by

$$\widehat{X}_{t,s}^c := \widehat{X}_{t,s} P_b \in \mathbb{R}^{M_t \times \widehat{K}_b}, \quad R_{t,s}^c := R_{t,s} P_b \in \mathbb{R}^{L_t \times \widehat{K}_b}$$

for all $b = (t,s)$ with $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{row}^*(t)$. Combining all of these condensed matrices, we can construct condensed counterparts of X_t and $\widehat{X}_t$ by

$$X_t^c := \begin{pmatrix} X_{t,s_1}^c & \dots & X_{t,s_\sigma}^c \end{pmatrix} \in \mathbb{R}^{\mathcal{I} \times N_t}_{\hat{t}},$$
$$\widehat{X}_t^c := \begin{pmatrix} \widehat{X}_{t,s_1}^c & \dots & \widehat{X}_{t,s_\sigma}^c \end{pmatrix} \in \mathbb{R}^{M_t \times N_t}_{\hat{t}},$$

where we let

$$\sigma := \#\operatorname{row}^*(t), \quad \{s_1, \dots, s_\sigma\} := \operatorname{row}^*(t), \quad N_t := \widehat{K}_{t,s_1} \mathbin{\dot\cup} \cdots \mathbin{\dot\cup} \widehat{K}_{t,s_\sigma}.$$

The set N_t is the *disjoint* union of the sets $\widehat{K}_{t,s}$ since we have $\widehat{K}_{t,s} \subseteq \hat{s}$ by definition and since Lemma 5.7 guarantees that all $\hat{s}$ with $s \in \operatorname{row}^*(t)$ are disjoint.

The sets N_t can be constructed by a recursion that closely resembles the one applied to r_t in (6.17):

$$N_t = \begin{cases} N_{t^+} \cup \bigcup_{s \in \operatorname{row}^+(t)} \widehat{K}_{t,s} & \text{if there is a cluster } t^+ \in \mathcal{T}_{\mathcal{I}} \\ & \text{with } t \in \operatorname{sons}(t^+), \\ \bigcup_{s \in \operatorname{row}^+(t)} \widehat{K}_{t,s} & \text{otherwise.} \end{cases} \tag{6.23}$$

Applying these modifications to Algorithm 25 yields Algorithm 26.

Before Algorithm 26 can begin its work, we have to prepare the matrices C_b for all admissible leaves $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. This is a simple task that can easily be accomplished by a loop applying the Householder Algorithm 15 to all matrices B_b. Combining this loop with the initial call to the recursive Algorithm 26 yields the complete compression Algorithm 27.

Since Algorithm 27 computes the same orthogonal nested cluster basis as the general Algorithm 25, the error analysis of Theorem 6.21 and particularly the Corollary 6.22 also hold for the new, more efficient algorithm.

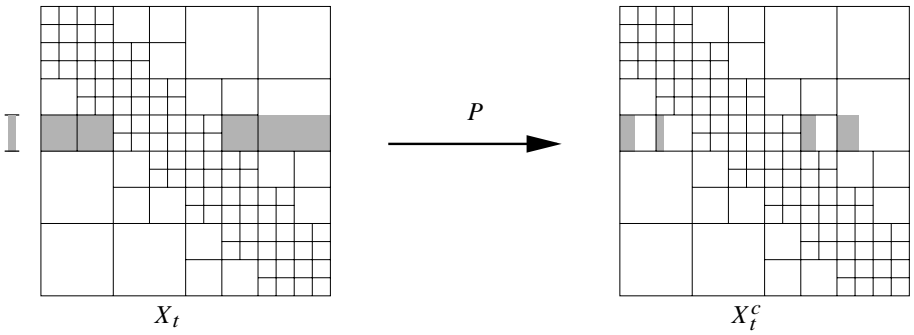


Figure 6.4. Condensation of X_t for the model problem.

Lemma 6.25 (Complexity). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be a hierarchical matrix for the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and the local rank $k_{\mathcal{H}} \in \mathbb{N}$, given in $\mathcal{H}$-matrix representation (3.12). Let $p_{\mathcal{I}\times\mathcal{J}}$ be the depth of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Let $L = (L_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be the rank distribution computed by Algorithm 27, and let $(l_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be defined as in (6.18). Algorithm 27 requires not more than*

$$C_{\mathrm{sp}}C_{\mathrm{qr}}(p_{\mathcal{I}\times\mathcal{J}}+1)k_{\mathcal{H}}^2 n_{\mathcal{J}} + C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1)\sum_{t\in\mathcal{T}_{\mathcal{I}}}(C_{\mathrm{pr}}k_{\mathcal{H}}l_t^2 + 2k_{\mathcal{H}}^2 l_t)$$

operations. If L is $(C_{\mathrm{bn}},\alpha,\beta,r,\xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}},\alpha,\beta,r,\xi)$-regular, the number of operations is in $\mathcal{O}(k_{\mathcal{H}}^2(p_{\mathcal{I}\times\mathcal{J}}+1)(n_{\mathcal{I}}+n_{\mathcal{J}}) + k_{\mathcal{H}}(p_{\mathcal{I}\times\mathcal{J}}+1)(\alpha+\beta)^r n_{\mathcal{I}})$.

Proof. Algorithm 27 starts by computing the weight matrices C_b for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ using Algorithm 15.

According to Lemma 5.17, computing C_b for an admissible leaf $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ requires not more than

$$C_{\mathrm{qr}}(\#\hat{s})(\#K_b)\min\{\#\hat{s},\#K_b\} \leq C_{\mathrm{qr}}(\#\hat{s})(\#K_b)^2 \leq C_{\mathrm{qr}}(\#\hat{s})k_{\mathcal{H}}^2$$

operations. Since the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse and since Definition 3.12 implies $\mathrm{level}(s) \leq \mathrm{level}(b) \leq p_{\mathcal{I}\times\mathcal{J}}$ for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we get the bound

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} C_{\mathrm{qr}}k_{\mathcal{H}}^2\#\hat{s} &= C_{\mathrm{qr}}k_{\mathcal{H}}^2 \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ \mathrm{level}(s)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{t\in\mathrm{col}^+(s)} \#\hat{s} \\
&\leq C_{\mathrm{sp}}C_{\mathrm{qr}}k_{\mathcal{H}}^2 \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ \mathrm{level}(s)\leq p_{\mathcal{I}\times\mathcal{J}}}} \#\hat{s} = C_{\mathrm{sp}}C_{\mathrm{qr}}k_{\mathcal{H}}^2 \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} \sum_{s\in\mathcal{T}_{\mathcal{J}}^{\ell}} \#\hat{s} \\
&\overset{\text{Cor. 3.10}}{\leq} C_{\mathrm{sp}}C_{\mathrm{qr}}k_{\mathcal{H}}^2 \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} n_{\mathcal{J}} = C_{\mathrm{sp}}C_{\mathrm{qr}}k_{\mathcal{H}}^2(p_{\mathcal{I}\times\mathcal{J}}+1)n_{\mathcal{J}}
\end{aligned}$$

Algorithm 26. Recursion for finding adaptive cluster bases for $\mathcal{H}$-matrices.

procedure AdaptiveHRec($t, r_{t+}, N_{t+}, \epsilon, A, C, \widehat{K}$, **var** Q, R, L)
$r_t \leftarrow r_{t+} \cup \operatorname{row}^+(t)$; $N_t \leftarrow N_{t+}$;
for $s \in \operatorname{row}^+(t)$ **do**
  $N_t \leftarrow N_t \dot\cup \widehat{K}_{t,s}$
end for;
if $\operatorname{sons}(t) = \emptyset$ **then**
  $X_t^c \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times N_t}_{\hat{t}}$;
  for $s \in r_t$ **do**
    $X_t^c|_{\mathcal{I}\times\widehat{K}_{t,s}} \leftarrow \chi_t A_b C_b^*$
  end for;
  Lowrank($X_t^c, \hat{t}, \epsilon_t, Q_t, R_t^c, L_t$); {Algorithm 18}
  for $s \in r_t$ **do**
    $R_{t,s}^c \leftarrow R_t^c|_{L_t\times\widehat{K}_{t,s}}$
  end for
else
  $M_t \leftarrow \emptyset$;
  for $t' \in \operatorname{sons}(t)$ **do**
    AdaptiveHRec($t', r_t, N_t, \epsilon, A, C, \widehat{K}, Q, R, L$);
    $M_t \leftarrow M_t \dot\cup L_{t'}$
  end for;
  $\widehat{X}_t^c \leftarrow 0 \in \mathbb{R}^{M_t\times N_t}$;
  for $s \in r_t$ **do**
    $\widehat{X}_{t,s}^c \leftarrow 0 \in \mathbb{R}^{M_t\times\widehat{K}_{t,s}}$;
    for $t' \in \operatorname{sons}(t)$ **do**
      $\widehat{X}_{t,s}^c|_{L_{t'}\times\widehat{K}_{t,s}} \leftarrow R_{t',s}^c$
    end for;
    $\widehat{X}_t^c|_{M_t\times\widehat{K}_{t,s}} \leftarrow \widehat{X}_{t,s}^c$
  end for;
  Lowrank($\widehat{X}_t^c, M_t, \epsilon_t, \widehat{Q}_t, \widehat{R}_t^c, L_t$); {Algorithm 18}
  for $t' \in \operatorname{sons}(t)$ **do**
    $F_{t'} \leftarrow \widehat{Q}_t|_{L_{t'}\times L_t}$
  end for;
  for $s \in r_t$ **do**
    $\widehat{R}_{t,s}^c \leftarrow \widehat{R}_t^c|_{L_t\times\widehat{K}_{t,s}}$
  end for
end if

for the number of operations required to prepare all matrices $(C_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$. Once these matrices are available, the recursive Algorithm 26 is called.

Algorithm 27. Adaptive cluster bases for $\mathcal{H}$-matrices.

procedure AdaptiveH(A, B, ϵ, **var** Q, R, L)
for $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ **do**
  Householder(B_b, $\hat{s}$, P_b, C_b, $\widehat{K}_b$) {Algorithm 15}
end for;
AdaptiveHRec(root($\mathcal{T}_\mathcal{I}$), $\emptyset$, $\emptyset$, ϵ, A, C, $\widehat{K}$, **var** Q, R, L) {Algorithm 26}

Let $t \in \mathcal{T}_\mathcal{I}$. We can bound the number of columns of X_t^c by

$$\#\widehat{N}_t = \sum_{s\in\operatorname{row}^*(t)} \#\widehat{K}_{t,s} = \sum_{\substack{t^+\in\operatorname{pred}(t)\\ \operatorname{level}(t^+)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{s\in\operatorname{row}^+(t^+)} \#\widehat{K}_{t,s} \leq C_{\operatorname{sp}} k_\mathcal{H} (p_{\mathcal{I}\times\mathcal{J}} + 1).$$

If $\operatorname{sons}(t) = \emptyset$, Algorithm 26 first constructs the matrix X_t^c by multiplying $\chi_t A_b$ and C_b^*, which requires not more than $2(\#\hat{t})k_\mathcal{H}\#\widehat{K}_{t,s}$ operations for each $s \in \operatorname{row}^*(t)$, and a total of

$$\sum_{s\in\operatorname{row}^*(t)} 2(\#\hat{t})k_\mathcal{H}\#\widehat{K}_{t,s} = 2(\#\hat{t})k_\mathcal{H}\#\widehat{N}_t \leq 2C_{\operatorname{sp}} l_t k_\mathcal{H}^2 (p_{\mathcal{I}\times\mathcal{J}} + 1).$$

Finding the low-rank factorization of this matrix requires not more than

$$C_{\operatorname{pr}}(\#\hat{t})^2\#N_t \leq C_{\operatorname{pr}}(\#\hat{t})^2 C_{\operatorname{sp}} k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1) \leq C_{\operatorname{sp}} C_{\operatorname{pr}} l_t^2 k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1)$$

operations according to Lemma 5.21.

If $\operatorname{sons}(t) \neq \emptyset$, the algorithm uses Algorithm 18 to find a low-rank approximation of the matrix $\widehat{X}_t^c \in \mathbb{R}^{M_t\times N_t}$. According to Lemma 5.21, this requires not more than

$$C_{\operatorname{pr}}(\#M_t)^2\#N_t \leq C_{\operatorname{pr}}(\#M_t)^2 C_{\operatorname{sp}} k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1) \leq C_{\operatorname{pr}} C_{\operatorname{sp}} l_t^2 k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1)$$

operations. Combining the estimates for leaf and non-leaf clusters yields the bound

$$C_{\operatorname{sp}}(p_{\mathcal{I}\times\mathcal{J}} + 1)(2l_t k_\mathcal{H}^2 + C_{\operatorname{pr}} k_\mathcal{H} l_t^2)$$

for the number of operations for one cluster t, and summing over all clusters proves our claim.

If L is $(C_{\operatorname{bn}}, \alpha, \beta, r, \xi)$-bounded and $\mathcal{T}_\mathcal{I}$ is $(C_{\operatorname{rc}}, \alpha, \beta, r, \xi)$-regular, we can again use the Lemmas 3.45 and 3.48 to bound the asymptotic complexity. □

According to Lemma 3.31, the $\mathcal{H}$-matrix representation of a matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ requires $\mathcal{O}(k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_\mathcal{I} + n_\mathcal{J}))$ units of storage. The preparation of X_t^c (or $\widehat{X}_t^c$) in Algorithm 27 requires $\mathcal{O}(k_\mathcal{H}^2(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_\mathcal{I} + n_\mathcal{J}))$ operations, since approximately $\mathcal{O}(k_\mathcal{H})$ operations are performed for each coefficient of the input representation. The computation of the cluster bases Q then takes additional $\mathcal{O}(k_\mathcal{H}(\alpha+\beta)^r(p_{\mathcal{I}\times\mathcal{J}} + 1)n_\mathcal{I})$ operations, this reminds us of the bound of Lemma 6.20 for the original algorithm, only that $n_\mathcal{J}$ in the original estimate is replaced by $k_\mathcal{H}(p_{\mathcal{I}\times\mathcal{J}} + 1)$ due to condensation.

6.6 Recompression of $\mathcal{H}^2$-matrices

We have seen that we can improve the efficiency of the general Algorithm 25 if the matrix X we intend to convert to the $\mathcal{H}^2$-matrix format is already given in a data-sparse representation, e.g., as an $\mathcal{H}$-matrix. The complexity of the modified Algorithm 27 is essentially proportional to the amount of storage required to represent the matrix X.

Let us now consider an even more compact representation of X: we assume that X is given in $\mathcal{H}^2$-matrix form and intend to compute the adapted cluster bases $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ defined by Algorithm 25 efficiently. At first glance, this task makes no sense: if X is already an $\mathcal{H}^2$-matrix, why should we convert it into an $\mathcal{H}^2$-matrix? The answer is given by Lemma 6.23: the cluster bases computed by the adaptive algorithm have almost optimal rank, i.e., we could create an initial $\mathcal{H}^2$-matrix approximation of a matrix by polynomial interpolation or a similar, probably non-optimal, approach and then construct adaptive cluster bases with lower storage requirements.

Let $V_X = (V_{X,t})_{t\in\mathcal{T}_\mathcal{I}}$ be a nested cluster basis with rank distribution $(K_{X,t})_{t\in\mathcal{T}_\mathcal{I}}$ for $\mathcal{T}_\mathcal{I}$. Let $W_X = (W_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ be a nested cluster basis with rank distribution $(L_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ for $\mathcal{T}_\mathcal{J}$. Let $X \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_X, W_X)$. Let $S = (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be the coefficient matrices of X, i.e., let

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} V_{X,t} S_b W_{X,s}^* + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{6.24}$$

As in the case of hierarchical matrices, the key to finding an efficient algorithm is to use suitable condensed matrices. We consider an admissible leaf $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Due to Lemma 5.4, we have

$$X_{t,s} = \chi_t X \chi_s = V_{X,t} S_b W_{X,s}^*. \tag{6.25}$$

Let now $t \in \mathcal{T}_\mathcal{I}$ and $s \in \operatorname{row}^*(t)$. Due to (5.4), we can find a predecessor $t^+ \in \operatorname{pred}(t)$ of t and a cluster $s \in \operatorname{row}^+(t^+)$ such that $(t^+, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ is an admissible leaf. Lemma 6.13 yields

$$X_{t,s} = \chi_t X \chi_s = \chi_t \chi_{t^+} X \chi_s = \chi_t V_{X,t^+} S_{t^+,s} W_{X,s}^* = V_{X,t} E_{X,t,t^+} S_{t^+,s} W_{X,s}^*,$$

where E_{X,t,t^+} denotes a long-range transfer matrix (cf. Definition 5.27) for the cluster basis V_X, and applying this equation to all $s \in \operatorname{row}^*(t)$ gives us the representation

$$X_t = V_{X,t} \sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} E_{X,t,t^+} S_{t^+,s} W_{X,s}^*$$

(cf. (6.9) for a similar result for left semi-uniform matrices). Introducing the matrix

$$Y_t := \sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} W_{X,s} S_{t^+,s}^* E_{X,t,t^+}^* \in \mathbb{R}^{\mathcal{J}\times K_{X,t}},$$

we conclude that

$$X_t = V_{X,t} Y_t^*$$

holds, i.e., that we have a factorized representation of the *entire* total cluster basis matrix X_t, not only of one block $X_{t,s}$. We apply Lemma 5.16 in order to construct an index set $\widehat{K}_t \subseteq \mathcal{J}$ with $\#\widehat{K}_t \leq \#K_{X,t}$, an orthogonal matrix $P_t \in \mathbb{R}^{\mathcal{J} \times \widehat{K}_t}$ and a weight matrix $Z_t \in \mathbb{R}^{\widehat{K}_t \times K_{X,t}}$ satisfying

$$Y_t = P_t Z_t.$$

Using this factorization, we conclude

$$X_t = V_{X,t} Y_t^* = (V_{X,t} Z_t^*) P_t^*, \tag{6.26}$$

i.e., we can replace the *entire* total cluster basis matrix X_t by the condensed matrix

$$X_t^c := V_{X,t} Z_t^*$$

with not more than $\#K_{X,t}$ columns. This is a significant improvement compared to the case of $\mathcal{H}$-matrices, where the condensed total cluster basis matrix has $\#N_t \approx C_{\mathrm{sp}} k_{\mathcal{H}} (p_{\mathcal{I} \times \mathcal{J}} + 1)$ columns. Obviously, working with the condensed matrix X_t^c is *far* more efficient than working directly with X_t.

Computing Z_t in the way described here would involve the factorization of the matrix $Y_t \in \mathbb{R}^{\mathcal{J} \times K_{X,t}}$ and lead immediately to an algorithm of quadratic complexity. Instead of computing the factorization of Y_t directly, we have to use a recursive construction: if we assume that Y_t has already been computed and that we want to compute $Y_{t'}$ for a son $t' \in \operatorname{sons}(t)$, Lemma 5.28 implies

$$E_{X,t',t^+} = E_{X,t'} E_{X,t,t^+} \quad \text{for all } t^+ \in \operatorname{pred}(t)$$

and we find

$$\begin{aligned} Y_{t'} &= \sum_{t^+ \in \operatorname{pred}(t')} \sum_{s \in \operatorname{row}^+(t^+)} W_{X,s} S_{t^+,s}^* E_{X,t',t^+}^* \\ &= \sum_{s \in \operatorname{row}^+(t')} W_{X,s} S_{t',s}^* + \sum_{t^+ \in \operatorname{pred}(t)} \sum_{s \in \operatorname{row}^+(t^+)} W_{X,s} S_{t^+,s}^* E_{X,t,t^+}^* E_{X,t'}^* \\ &= \sum_{s \in \operatorname{row}^+(t')} W_{X,s} S_{t',s}^* + Y_t E_{X,t'}^*. \end{aligned} \tag{6.27}$$

This equation allows us to compute all matrices Y_t by the recursive relationship

$$Y_t = \begin{cases} Y_{t^+} E_{X,t}^* + \sum_{s \in \operatorname{row}^+(t)} W_{X,s} S_{t,s}^* & \text{if there exists a } t^+ \in \mathcal{T}_{\mathcal{I}} \\ & \text{with } t \in \operatorname{sons}(t^+), \\ \sum_{s \in \operatorname{row}^+(t)} W_{X,s} S_{t,s}^* & \text{otherwise.} \end{cases} \tag{6.28}$$

We first consider the second case. Finding a factorization of

$$Y_t = \sum_{s \in \operatorname{row}^+(t)} W_{X,s} S_{t,s}^* \tag{6.29}$$

directly leads to a complexity of $\mathcal{O}((\#\hat{t})(\#L_{X,s})(\#K_{X,t}))$ for each term in the sum, and the resulting total complexity would not be optimal.

We avoid this problem by applying the orthogonalization Algorithm 16 to the cluster basis $W_X = (W_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$. This procedure is of optimal complexity and yields an orthogonal cluster basis $Q_X = (Q_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$ with rank distribution $\hat{L}_X = (\hat{L}_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$ and a family of weight matrices $R_X = (R_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$ satisfying

$$W_{X,s} = Q_{X,s} R_{X,s} \quad \text{for all } s \in \mathcal{T}_\mathcal{J}. \tag{6.30}$$

Using these factorizations allows us to write the sum (6.29) in the form

$$\begin{aligned} Y_t &= \sum_{s \in \operatorname{row}^+(t)} W_{X,s} S_{t,s}^* \\ &= \sum_{s \in \operatorname{row}^+(t)} Q_{X,s} R_{X,s} S_{t,s}^* \\ &= \begin{pmatrix} Q_{X,s_1} & \dots & Q_{X,s_\sigma} \end{pmatrix} \begin{pmatrix} R_{X,s_1} S_{t,s_1}^* \\ \vdots \\ R_{X,s_\sigma} S_{t,s_\sigma}^* \end{pmatrix} = U_{X,t} \hat{Y}_t \end{aligned}$$

for $\sigma := \#\operatorname{row}^+(t)$ and $\{s_1, \dots, s_\sigma\} := \operatorname{row}^+(t)$ and the matrices

$$U_{X,t} := \begin{pmatrix} Q_{X,s_1} & \dots & Q_{X,s_\sigma} \end{pmatrix} \in \mathbb{R}^{\mathcal{J} \times N_t}, \quad \hat{Y}_t := \begin{pmatrix} R_{X,s_1} S_{t,s_1}^* \\ \vdots \\ R_{X,s_\sigma} S_{t,s_\sigma}^* \end{pmatrix} \in \mathbb{R}^{N_t \times K_{X,t}}$$

with the index set

$$N_t := \bigcup_{s \in \operatorname{row}^+(t)} \hat{L}_{X,s}.$$

Due to Lemma 5.7, the matrix $U_{X,t}$ is orthogonal. The matrix $\hat{Y}_t$ has only

$$\#N_t = \sum_{s \in \operatorname{row}^+(t)} \#\hat{L}_{X,s}$$

rows, therefore we can afford to apply the Householder factorization Algorithm 15 in order to construct an index set $\hat{K}_t \subseteq N_t$, an orthogonal matrix $\hat{P}_t \in \mathbb{R}^{N_t \times \hat{K}_t}$ and a weight matrix $Z_t \in \mathbb{R}^{\hat{K}_t \times K_{X,t}}$ satisfying

$$\hat{Y}_t = \hat{P}_t Z_t.$$

Since $U_{X,t}$ is orthogonal, the same holds for the matrix

$$P_t := U_t \widehat{P}_t,$$

and we see that we have *efficiently* constructed the desired factorization

$$Y_t = \sum_{s \in \mathrm{row}^+(t)} W_{X,s} S_{t,s}^* = U_{X,t} \widehat{Y}_t = U_{X,t} \widehat{P}_t Z_t = P_t Z_t.$$

Now let us consider the first line in (6.28). We assume that a factorization

$$Y_{t+} = P_{t+} Z_{t+}$$

of Y_{t+} has already been computed. Using the weight matrices $R_{X,s}$ provided by the orthogonalization Algorithm 16, we find

$$\begin{aligned} Y_t &= Y_{t+} E_{X,t}^* + \sum_{s \in \mathrm{row}^+(t)} W_{X,s} S_{t,s}^* = P_{t+} Z_{t+} E_{X,t}^* + \sum_{s \in \mathrm{row}^+(t)} Q_{X,s} R_{X,s} S_{t,s}^* \\ &= \begin{pmatrix} P_{t+} & Q_{X,s_1} & \dots Q_{X,s_\sigma} \end{pmatrix} \begin{pmatrix} Z_{t+} E_{X,t}^* \\ R_{X,s_1} S_{t,s_1}^* \\ \vdots \\ R_{X,s_\sigma} S_{t,s_\sigma}^* \end{pmatrix} = U_{X,t} \widehat{Y}_t \end{aligned}$$

for $\sigma := \#\,\mathrm{row}^+(t)$, $\{s_1, \dots, s_\sigma\} := \mathrm{row}^+(t)$ and the matrices

$$U_{X,t} := \begin{pmatrix} P_{t+} & Q_{X,s_1} & \dots & Q_{X,s_\sigma} \end{pmatrix} \in \mathbb{R}^{\mathcal{I} \times N_t}, \quad \widehat{Y}_t := \begin{pmatrix} Z_{t+} E_{X,t}^* \\ R_{X,s_1} S_{t,s_1}^* \\ \vdots \\ R_{X,s_\sigma} S_{t,s_\sigma}^* \end{pmatrix} \in \mathbb{R}^{N_t \times K_{X,t}} \tag{6.31}$$

with the index set

$$N_t := \widehat{K}_{t+} \cup \bigcup_{s \in \mathrm{row}^+(t)} \widehat{L}_{X,s}.$$

We apply Algorithm 15 to the matrix $\widehat{Y}_t$ in order to find an index set $\widehat{K}_t \subseteq N_t$, an orthogonal matrix $\widehat{P}_t \in \mathbb{R}^{N_t \times \widehat{K}_t}$ and a weight matrix $Z_t \in \mathbb{R}^{\widehat{K}_t \times K_{X,t}}$ with

$$\widehat{Y}_t = \widehat{P}_t Z_t. \tag{6.32}$$

Due to Lemma 5.7, the matrix $U_{X,t}$ is orthogonal, and since $\widehat{P}_t$ is also orthogonal, the same holds for

$$P_t := U_{X,t} \widehat{P}_t,$$

and we find

$$Y_t = Y_{t^+} E_{X,t}^* + \sum_{s \in \operatorname{row}^+(t)} W_{X,s} S_{t,s}^* = U_{X,t} \widehat{Y}_t = U_{X,t} \widehat{P}_t Z_t = P_t Z_t.$$

Fortunately, we are only interested in the weight matrices $(Z_t)_{t \in \mathcal{T}_I}$, therefore we do not have to construct P_t, $U_{X,t}$ and $Q_{X,s}$ explicitly. The fairly small matrices $R_{X,s}$ and $\widehat{Y}_t$ are sufficient, and these matrices we can handle efficiently.

We construct adaptive cluster bases for $\mathcal{H}^2$-matrices in three phases: the weights $R_X = (R_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$ are computed by applying the orthogonalization Algorithm 16 to the column cluster basis W_X.

Once the weights for the column cluster basis have been prepared, we have to compute the weights $(Z_t)_{t \in \mathcal{T}_I}$ for the total cluster basis are computed using (6.31) and (6.32). The corresponding top-down recursion is summarized in Algorithm 28.

Algorithm 28. Recursive construction of weight matrices for the total cluster basis.

procedure TotalWeights(t, $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\widehat{K}_{t^+}$, Z_{t^+}, V_X, R_X, S, **var** Z, $\widehat{K}$)
$N_t \leftarrow \widehat{K}_{t^+}$;
for $s \in \operatorname{row}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ **do**
  $N_t \leftarrow N_t \dot\cup \widehat{L}_{X,s}$
end for;
$\widehat{Y}_t \leftarrow 0 \in \mathbb{R}^{N_t \times K_{X,t}}$;
if $\widehat{K}_{t^+} \neq \emptyset$ **then**
  $\widehat{Y}_t|_{\widehat{K}_{t^+} \times K_{X,t}} \leftarrow Z_{t^+} E_{X,t}^*$; {$t^+ = \operatorname{father}(t)$}
end if;
for $s \in \operatorname{row}^+(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, t)$ **do**
  $\widehat{Y}_t|_{\widehat{L}_{X,s} \times K_{X,t}} \leftarrow R_{X,s} S_{t,s}^*$
end for;
Householder($\widehat{Y}_t$, N_t, $\widehat{P}_t$, Z_t, $\widehat{K}_t$); {Algorithm 15}
for $t' \in \operatorname{sons}(t)$ **do**
  TotalWeights(t', $\widehat{K}_t$, Z_t, V_X, R_X, S, Z, $\widehat{K}$)
end for

Lemma 6.26 (Complexity of finding weights). *Let $(k_{X,t})_{t \in \mathcal{T}_I}$ and $(l_{X,s})_{s \in \mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18) for the rank distributions K_X and L_X. Algorithm 28 requires not more than*

$$\max\{2, C_{\mathrm{qr}}\}\Big(2 \sum_{t \in \mathcal{T}_I} k_{X,t}^3 + \sum_{b=(t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+} (k_{X,t}^3 + l_{X,s}^3)\Big)$$

operations.

Proof. Since Algorithm 28 uses Algorithm 15 to construct $\widehat{K}_t$ by splitting the matrix $\widehat{Y}_t \in \mathbb{R}^{N_t \times K_{X,t}}$, we have $\#\widehat{K}_t \leq \#K_{X,t}$ for all $t \in \mathcal{T}_\mathcal{I}$.

Let now $t \in \mathcal{T}_\mathcal{I} \setminus \{\operatorname{root}(\mathcal{T}_\mathcal{I})\}$, and let $t^+ \in \mathcal{T}_\mathcal{I}$ be its father. The computation of $Z_{t^+} E^*_{X,t}$ requires not more than

$$2(\#\widehat{K}_{t^+})(\#K_{X,t^+})(\#K_{X,t}) \leq 2(\#K_{X,t^+})^2(\#K_{X,t})$$

operations, the computation of $R_{X,s} S^*_{t,s}$ requires not more than

$$2(\#\widehat{L}_{X,s})(\#L_{X,s})(\#K_{X,t}) \leq 2(\#L_{X,s})^2(\#K_{X,t}) \leq 2l^2_{X,s} l_{X,t}$$

operations, and therefore

$$2\Big((\#K_{X,t})(\#K_{X,t^+})^2 + \sum_{s \in \operatorname{row}^+(t)} k_{X,t} l^2_{X,s}\Big)$$

operations are sufficient to set up the matrix $\widehat{Y}_t \in \mathbb{R}^{K_{X,t} \times N_t}$. Due to Lemma 5.17, computing its Householder factorization using Algorithm 15 requires not more than

$$\begin{aligned} C_{\mathrm{qr}}(\#K_{X,t})(\#N_t)\min\{\#K_{X,t}, \#N_t\} &\leq C_{\mathrm{qr}}(\#K_{X,t})^2(\#N_t) \\ &\leq C_{\mathrm{qr}}(\#K_{X,t})^2\Big(\#\widehat{K}_{t^+} + \sum_{s \in \operatorname{row}^+(t)} \#\widehat{L}_{X,s}\Big) \\ &\leq C_{\mathrm{qr}}\Big((\#K_{X,t})^2(\#K_{X,t^+}) + \sum_{s \in \operatorname{row}^+(t)} k^2_{X,t} l_{X,s}\Big) \end{aligned}$$

operations. Using (5.9), we get the bound

$$\begin{aligned} &\max\{2, C_{\mathrm{qr}}\}\Big((\#K_{X,t})(\#K_{X,t^+})^2 + (\#K_{X,t})^2(\#K_{X,t^+}) + \sum_{s \in \operatorname{row}^+(t)} k_{X,t} l^2_{X,s} + k^2_{X,t} l_{X,s}\Big) \\ &\leq \max\{2, C_{\mathrm{qr}}\}\Big((\#K_{X,t})(\#K_{X,t^+})^2 + (\#K_{X,t})^2(\#K_{X,t^+}) + \sum_{s \in \operatorname{row}^+(t)} k^3_{X,t} + l^3_{X,s}\Big) \end{aligned}$$

for the number of operations required in one cluster $t \in \mathcal{T}_\mathcal{I} \setminus \{\operatorname{root}(\mathcal{T}_\mathcal{I})\}$. For the root cluster $r := \operatorname{root}(\mathcal{T}_\mathcal{I})$ we get the same estimate without the terms involving $\#K_{X,t^+}$, i.e.,

$$\max\{2, C_{\mathrm{qr}}\} \sum_{s \in \operatorname{row}^+(r)} k^3_{X,r} + l^3_{X,s}.$$

For the number of operations for all clusters except for the root r we get the bound

$$\max\{2, C_{\mathrm{qr}}\}\Big(\sum_{t^+ \in \mathcal{T}_\mathcal{I}} \sum_{t \in \operatorname{sons}(t^+)} (\#K_{X,t})^2(\#K_{X,t^+}) + \sum_{t^+ \in \mathcal{T}_\mathcal{I}} \sum_{t \in \operatorname{sons}(t^+)} (\#K_{X,t})(\#K_{X,t^+})^2$$

$$
\begin{aligned}
&+ \sum_{t\in\mathcal{T}_\mathcal{I}\setminus\{r\}} \sum_{s\in\operatorname{row}^+(t)} k_{X,t}^3 + l_{X,s}^3 \Big) \\
&\leq \max\{2, C_{\mathrm{qr}}\} \Big(\sum_{t^+\in\mathcal{T}_\mathcal{I}} k_{X,t^+}^2 (\#K_{X,t^+}) + \sum_{t^+\in\mathcal{T}_\mathcal{I}} k_{X,t^+} (\#K_{X,t^+})^2 \\
&\quad + \sum_{t\in\mathcal{T}_\mathcal{I}\setminus\{r\}} \sum_{s\in\operatorname{row}^+(t)} k_{X,t}^3 + l_{X,s}^3 \Big) \\
&\leq \max\{2, C_{\mathrm{qr}}\} \Big(2 \sum_{t^+\in\mathcal{T}_\mathcal{I}} k_{X,t^+}^3 + \sum_{t\in\mathcal{T}_\mathcal{I}\setminus\{r\}} \sum_{s\in\operatorname{row}^+(t)} k_{X,t}^3 + l_{X,s}^3 \Big),
\end{aligned}
$$

and adding the bound for the root cluster yields the bound

$$
\begin{aligned}
&2\max\{2, C_{\mathrm{qr}}\} \sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \max\{2, C_{\mathrm{qr}}\} \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^+(t)} k_{X,t}^3 + l_{X,s}^3 \\
&\qquad = 2\max\{2, C_{\mathrm{qr}}\} \sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \max\{2, C_{\mathrm{qr}}\} \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} k_{X,t}^3 + l_{X,s}^3
\end{aligned}
$$

for the *total* number of operations. □

In the last step, we compute the adaptive cluster basis by using the weights $(Z_t)_{t\in\mathcal{T}_\mathcal{I}}$ and equation (6.26) to replace the total cluster basis matrices X_t of the original Algorithm 25 by the condensed matrices $X_t^c = V_{X,t} Z_t^*$. Since the handling of the total cluster bases has been almost completely moved into Algorithm 28, the resulting Algorithm 29 closely resembles the truncation Algorithm 19 up to an important difference: the singular value decompositions are not computed for $V_{X,t}$ and $\widehat{V}_{X,t}$, but for the weighted matrices $X_t^c = V_{X,t} Z_t^*$ and $\widehat{X}_t^c = \widehat{V}_{X,t} Z_t^*$. Due to this modification, the matrices $\widetilde{R}_t = Q_t^* \widehat{V}_{X,t} Z_t^*$ do not, as in Algorithm 19, describe the change of basis from $V_{X,t}$ to Q_t, and we have to compute $R_t = Q_t^* V_{X,t}$ explicitly.

All three phases of the procedure are collected in the "front end" Algorithm 30, which calls the orthogonalization Algorithm 16, the weighting Algorithm 28 and the modified truncation Algorithm 29 in order to construct an adaptive orthogonal cluster basis for an $\mathcal{H}^2$-matrix X.

We can avoid the preparation step of calling Algorithm 16 to get the matrices $(R_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ if we know that the cluster basis W is orthogonal, since then we have $R_{X,s} = I$ for all $s \in \mathcal{T}_\mathcal{J}$ (cf. Section 6.7).

Theorem 6.27 (Complexity). *Let $(k_{X,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18) for the rank distributions K_X and L_X. There is a constant $C_{\mathrm{ad}} \in \mathbb{R}_{\geq 3}$ such that Algorithm 30 requires not more than*

$$
C_{\mathrm{ad}}(C_{\mathrm{sp}} + 2) \Big(\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_{X,s}^3 \Big)
$$

Algorithm 29. Recursion for finding adaptive cluster bases for $\mathcal{H}^2$-matrices.

procedure AdaptiveH2Rec(t, V_X, Z, $\widehat{K}$, ϵ, **var** Q, R, L)
if sons$(t) = \emptyset$ **then**
  $X_t^c \leftarrow V_{X,t} Z_t^* \in \mathbb{R}_{\hat{t}}^{\mathcal{I}\times\widehat{K}_t}$;
  Lowrank(X_t^c, $\hat{t}$, ϵ_t, Q_t, $\widetilde{R}_t$, L_t); {Algorithm 18}
  $R_t \leftarrow Q_t^* V_{X,t}$
else
  $M_t \leftarrow \emptyset$;
  for $t' \in$ sons(t) **do**
    AdaptiveH2Rec(t', V_X, Z, $\widehat{K}$, ϵ, Q, R, L);
    $M_t \leftarrow M_t \dot\cup L_{t'}$
  end for;
  $\widehat{V}_{X,t} \leftarrow 0 \in \mathbb{R}^{M_t\times K_{X,t}}$;
  for $t' \in$ sons(t) **do**
    $\widehat{V}_{X,t}|_{L_{t'}\times\widehat{K}_{X,t}} \leftarrow R_{t'} E_{X,t'}$;
  end for;
  $\widehat{X}_t^c \leftarrow \widehat{V}_{X,t} Z_t^* \in \mathbb{R}^{M_t\times\widehat{K}_t}$;
  Lowrank($\widehat{X}_t^c$, M_t, ϵ_t, $\widehat{Q}_t$, $\widetilde{R}_t$, L_t); {Algorithm 18}
  for $t' \in$ sons(t) **do**
    $F_{t'} \leftarrow \widehat{Q}_t|_{L_{t'}\times L_t}$
  end for;
  $R_t \leftarrow \widehat{Q}_t^* \widehat{V}_{X,t}$
end if

Algorithm 30. Adaptive cluster bases for $\mathcal{H}^2$-matrices.

procedure AdaptiveH2(V_X, W_X, S, ϵ, **var** Q, R, L)
Orthogonalize(root$(\mathcal{T}_\mathcal{J})$, W_X, Q_X, R_X, $\widehat{L}_X$); {Algorithm 16}
TotalWeights(root$(\mathcal{T}_\mathcal{I})$, $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\emptyset$, 0, V_X, R_X, S, Z, $\widehat{K}$); {Algorithm 28}
AdaptiveH2Rec(root$(\mathcal{T}_\mathcal{I})$, V_X, Z, $\widehat{K}$, ϵ, Q, R, L) {Algorithm 29}

operations. If the rank distributions K_X and L_X are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if the trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I} + n_\mathcal{J}))$.

Proof. According to Lemma 5.18, the computation of the weight matrices $(R_{X,s})_{s\in\mathcal{T}_\mathcal{J}}$ requires not more than

$$(C_{\mathrm{qr}} + 2) \sum_{s\in\mathcal{T}_\mathcal{J}} l_{X,s}^3 \tag{6.33}$$

operations. According to Lemma 6.26, the computation of the total weight matrices

$(Z_t)_{t\in\mathcal{T}_\mathcal{I}}$ requires not more than

$$\begin{aligned}
&\max\{2,C_{\mathrm{qr}}\}\Big(2\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} k_{X,t}^3 + l_{X,s}^3\Big)\\
&\quad= \max\{2,C_{\mathrm{qr}}\}\Big(2\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{s\in\mathrm{row}^+(t)} k_{X,t}^3 + \sum_{s\in\mathcal{T}_\mathcal{J}}\sum_{t\in\mathrm{col}^+(s)} l_{X,s}^3\Big)\\
&\quad\leq \max\{2,C_{\mathrm{qr}}\}\Big(2\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + C_{\mathrm{sp}}\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + C_{\mathrm{sp}}\sum_{s\in\mathcal{T}_\mathcal{J}} l_{X,s}^3\Big)\\
&\quad\leq (C_{\mathrm{sp}}+2)\max\{2,C_{\mathrm{qr}}\}\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_{X,s}^3\Big) \qquad (6.34)
\end{aligned}$$

operations. Once the weights $(Z_t)_{t\in\mathcal{T}_\mathcal{I}}$ have been computed, Algorithm 30 starts the main recursion by calling Algorithm 29. Let $t\in\mathcal{T}_\mathcal{I}$. If $\mathrm{sons}(t)=\emptyset$, we compute $X_t^c = V_{X,t}Z_t^*$ in $2(\#\hat{t})(\#K_{X,t})(\#\hat{K}_t) \leq 2k_{X,t}^3$ operations, construct Q_t using Algorithm 18 in not more than $C_{\mathrm{pr}}((\#\hat{t})(\#K_{X,t})^2) \leq C_{\mathrm{pr}}k_{X,t}^3$ operations (cf. Lemma 5.21) and $R_t = Q_t^* V_{X,t}$ in $2(\#L_t)(\#\hat{t})(\#K_{X,t}) \leq 2k_{X,t}^3$ operations, which leads to a total complexity of

$$(C_{\mathrm{pr}}+4)k_{X,t}^3$$

for each leaf cluster.

If $\mathrm{sons}(t)\neq\emptyset$, (5.15) implies that we can compute the matrix $\hat{V}_{X,t}$ in

$$\begin{aligned}
\sum_{t'\in\mathrm{sons}(t)} 2(\#L_{t'})(\#K_{X,t'})(\#K_{X,t}) &\leq 2(\#K_{X,t})\sum_{t'\in\mathrm{sons}(t)}(\#K_{X,t'})^2\\
&\leq 2(\#K_{X,t})\Big(\sum_{t'\in\mathrm{sons}(t)}\#K_{X,t'}\Big)^2 \leq 2k_{X,t}^3
\end{aligned}$$

operations, construct $\hat{X}_t^c = \hat{V}_{X,t}Z_t^*$ in $2(\#M_t)(\#K_{X,t})(\#\hat{K}_t) \leq 2k_{X,t}^3$ operations, apply Algorithm 18 to find $\hat{Q}_t$ in not more than $C_{\mathrm{pr}}(\#M_t)(\#\hat{K}_t)^2 \leq C_{\mathrm{pr}}k_{X,t}^3$ operations and then compute R_t in $2(\#L_t)(\#M_t)(\#K_{X,t}) \leq 2k_{X,t}^3$ operations. Therefore we also have a complexity of

$$(C_{\mathrm{pr}}+4)k_{X,t}^3$$

for each non-leaf cluster. Adding the bounds for all $t\in\mathcal{T}_\mathcal{I}$ yields

$$(C_{\mathrm{pr}}+4)\sum_{t\in\mathcal{T}_\mathcal{I}} k_{X,t}^3. \qquad (6.35)$$

Adding (6.33), (6.34) and (6.35) gives us the desired estimate with

$$C_{\mathrm{ad}} := \max\{2,C_{\mathrm{qr}}\} + \max\{C_{\mathrm{qr}}/2+1, C_{\mathrm{pr}}/2+2\}.$$

If K_X and L_X are $(C_{\mathrm{bn}},\alpha,\beta,r,\xi)$-bounded and if $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}},\alpha,\beta,r,\xi)$-regular, we can again use the Lemmas 3.45 and 3.48 to complete the proof. □

6.7 Unification and hierarchical compression

The Algorithm 27 for the construction of an adaptive row cluster basis for an $\mathcal{H}$-matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ computes the same cluster basis as Algorithm 25, and according to Lemma 6.23, this means that we achieve an almost optimal result.

The disadvantage of this procedure is that the entire matrix X has to be given in $\mathcal{H}$-matrix representation before Algorithm 27 can begin to construct the cluster basis. In some applications, e.g., if we are approximating a boundary element matrix by adaptive [4], [7] or hybrid cross approximation [17], if the block cluster tree is determined by an adaptive coarsening strategy [50] or if the blocks result from an a posteriori matrix product (cf. Chapter 8), we would like to convert submatrices into an $\mathcal{H}^2$-matrix format as soon as possible in order to save storage.

Turning a single low-rank matrix block $A_b B_b^*$ into an $\mathcal{H}^2$-matrix with adaptive cluster bases is a simple task: we can use $S_b = I$, $V_b = A_b$ and $W_b = B_b$ in order to get

$$A_b B_b^* = V_b W_b^* = V_b S_b W_b^*.$$

This means that we can convert the submatrices corresponding to the leaf blocks of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ into individual $\mathcal{H}^2$-matrices, but each of these $\mathcal{H}^2$-matrices will have its own cluster bases, so we cannot simply add them to construct a global $\mathcal{H}^2$-matrix approximation.

We solve this problem by a blockwise recursion: let $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. If b is a leaf, the construction of cluster bases $V_b = (V_{b,t^*})_{t^*\in\operatorname{sons}^*(t)}$, $W_b = (W_{b,s^*})_{s^*\in\operatorname{sons}^*(s)}$ is straightforward. Otherwise, we assume that cluster bases $V_{b'}$ and $W_{b'}$ and $\mathcal{H}^2$-matrix approximations $X_{b'} \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_{b'}, W_{b'})$ have been computed for all submatrices $\chi_{t'} X \chi_{s'}$ corresponding to sons $b' = (t', s') \in \operatorname{sons}(b)$ of b and aim to combine these submatrices into a new $\mathcal{H}^2$-matrix approximation $X_b \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_b, W_b)$ of $\chi_t X \chi_s$.

Construction of unified cluster bases

Let us first consider the task in a more general setting. We assume that we have a finite index set $\mathcal{B}$ and a family $(X_b)_{b\in\mathcal{B}}$ of $\mathcal{H}^2$-matrices. For each $b \in \mathcal{B}$, there are a row cluster basis $V_b = (V_{b,t})_{t\in\mathcal{T}_\mathcal{I}}$ with rank distribution $K_b = (K_{b,t})_{t\in\mathcal{T}_\mathcal{I}}$ and a column cluster basis $W_b = (W_{b,s})_{s\in\mathcal{T}_\mathcal{J}}$ with rank distribution $L_b = (L_{b,s})_{s\in\mathcal{T}_\mathcal{J}}$ such that

$$X_b \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_b, W_b) \quad \text{holds for all } b \in \mathcal{B}.$$

We are looking for a row cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ such that the combined matrix given by

$$\widetilde{X} = \begin{pmatrix} X_{b_1} & \dots & X_{b_\beta} \end{pmatrix}$$

for $\beta := \#\mathcal{B}$ and $\{b_1, \dots, b_\beta\} := \mathcal{B}$ can be approximated, i.e., such that each of the matrices X_b can be approximated in the $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, Q, W_b)$.

Since this cluster basis Q can be used for all matrices X_b, we call it a *unified* row cluster basis for the family $(X_b)_{b\in\mathcal{B}}$.

For each $b \in \mathcal{B}$, we introduce the corresponding total cluster basis $(X_{b,t})_{t\in\mathcal{T}_\mathcal{I}}$ given by

$$X_{b,t} := \sum_{t^+\in\operatorname{pred}(t)} \sum_{s\in\operatorname{row}^+(t^+)} \chi_t X_b \chi_s \quad \text{for all } t \in \mathcal{T}_\mathcal{I}. \tag{6.36}$$

Let $t \in \mathcal{T}_\mathcal{I}$. Since X_b is an $\mathcal{H}^2$-matrix in the space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_b, W_b)$, we can proceed as in the previous section to find a matrix $Y_{b,t} \in \mathbb{R}^{\mathcal{J}\times K_{b,t}}$ with

$$X_{b,t} = V_{b,t} Y_{b,t}^*,$$

and applying Lemma 5.16 to $Y_{b,t}$ yields an index set $\hat{K}_{b,t} \subseteq \mathcal{J}$ with $\#\hat{K}_{b,t} \leq \#K_{b,t}$, an orthogonal matrix $P_{b,t} \in \mathbb{R}^{\mathcal{J}\times\hat{K}_{b,t}}$ and a weight matrix $Z_{b,t} \in \mathbb{R}^{\hat{K}_{b,t}\times K_{b,t}}$ such that

$$Y_{b,t} = P_{b,t} Z_{b,t}$$

holds and we get

$$X_{b,t} = V_{b,t} Z_{b,t}^* P_{b,t}^*, \tag{6.37}$$

i.e., we can replace the total cluster basis $X_{b,t}$ by the condensed matrix $X_{b,t}^c := V_{b,t} Z_{b,t}^*$.

The total cluster basis of $\widetilde{X}$ is given by

$$\widetilde{X}_t := \begin{pmatrix} X_{b_1,t} & \dots & X_{b_\beta,t} \end{pmatrix} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},$$

and using equation (6.37) allows us to represent it in the condensed form

$$\begin{aligned} \widetilde{X}_t &= \begin{pmatrix} X_{b_1,t} & \dots & X_{b_\beta,t} \end{pmatrix} \\ &= \begin{pmatrix} V_{b_1,t} Z_{b_1,t}^* & \dots & V_{b_\beta,t} Z_{b_\beta,t}^* \end{pmatrix} \begin{pmatrix} P_{b_1,t}^* & & \\ & \ddots & \\ & & P_{b_\beta,t}^* \end{pmatrix} \\ &= \widetilde{X}_t^c \begin{pmatrix} P_{b_1,t}^* & & \\ & \ddots & \\ & & P_{b_\beta,t}^* \end{pmatrix} \end{aligned}$$

for the matrix

$$\widetilde{X}_t^c := \begin{pmatrix} V_{b_1,t} Z_{b_1,t}^* & \dots & V_{b_\beta,t} Z_{b_\beta,t}^* \end{pmatrix} \in \mathbb{R}^{\mathcal{I}\times N_t}_{\hat{\mathcal{I}}}$$

and the index set

$$N_t := \bigcup_{b\in\mathcal{B}} \hat{K}_{b,t}.$$

Due to $\#N_t \leq (\#K_{b_1,t}) + \dots + (\#K_{b_\beta,t})$, the condensed matrix $\widetilde{X}_t^c$ will typically be significantly smaller than the original total cluster basis matrix $\widetilde{X}_t$.

We can proceed as in Algorithm 30: in leaf clusters, a singular value decomposition of $\widetilde{X}_t^c$ yields the cluster basis matrix Q_t and $R_{b,t} := Q_t^* V_{b,t}$, in non-leaf clusters we let $\tau = \#\operatorname{sons}(t)$ and $\{t_1, \ldots, t_\tau\} = \operatorname{sons}(t)$ and use

$$\widehat{V}_{b,t} = \begin{pmatrix} R_{b_1,t_1} E_{b,t_1} \\ \vdots \\ R_{b_1,t_\tau} E_{b,t_\tau} \end{pmatrix} \in \mathbb{R}^{M_t \times K_{b,t}},$$
$$\widehat{X}_t^c = \begin{pmatrix} \widehat{V}_{b_1,t} Z_{b_1,t}^* & \cdots & \widehat{V}_{b_\beta,t} Z_{b_\beta,t}^* \end{pmatrix} \in \mathbb{R}^{M_t \times N_t}$$

to construct the matrix $\widehat{Q}_t$ and $R_{b,t} := \widehat{Q}_t^* \widehat{V}_{b,t}$. Essentially the only difference between the new algorithm and the original Algorithm 30 is that the new basis is required to approximate multiple total cluster bases simultaneously, and that we therefore have to work with multiple original cluster bases and multiple weights in parallel.

Lemma 6.28 (Complexity). *Let $(l_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(k_{b,t})_{b\in\mathcal{B},t\in\mathcal{T}_\mathcal{I}}$ be defined as in (6.18) and (3.16) for the rank distributions L and K_b, $b \in \mathcal{B}$. We assume that the weight matrices $(Z_{b,t})_{t\in\mathcal{T}_\mathcal{I}}$ have already been computed. Algorithm 31 requires not more than*

$$\max\{4, C_{\mathrm{pr}} + 2\} \sum_{b\in\mathcal{B}} \sum_{t\in\mathcal{T}_\mathcal{I}} (l_t k_{b,t}^2 + l_t^2 k_{b,t})$$

operations. If the rank distributions L and K_b, $b \in \mathcal{B}$ are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(\#\mathcal{B})n_\mathcal{I})$.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$.

If $\operatorname{sons}(t) = \emptyset$, Algorithm 31 computes $\widetilde{X}_t^c$ by multiplying $V_{b,t}$ and $Z_{b,t}^*$ for all $b \in \mathcal{B}$, and this requires not more than $2(\#\hat{t})(\#K_{b,t})(\#\widehat{K}_{b,t}) \leq 2(\#\hat{t})(\#K_{b,t})^2 \leq 2l_t k_{b,t}^2$ operations for each $b \in \mathcal{B}$.

Then Algorithm 18 is called to compute Q_t. According to Lemma 5.21, this can be accomplished in $C_{\mathrm{pr}}(\#\hat{t})^2(\#N_t) \leq C_{\mathrm{pr}} l_t^2 (\#N_t)$ operations.

Finally the matrices $R_{b,t}$ are computed by not more than $2(\#L_t)(\#\hat{t})(\#K_{b,t}) \leq 2l_t^2 k_{b,t}$ operations for each $b \in \mathcal{B}$.

We conclude that a leaf cluster requires not more than

$$\sum_{b\in\mathcal{B}} 2l_t k_{b,t}^2 + (C_{\mathrm{pr}} + 2) l_t^2 k_{b,t} \quad \text{operations.} \tag{6.38}$$

If $\operatorname{sons}(t) \neq \emptyset$, Algorithm 31 computes $\widehat{V}_{b,t}$ by multiplying $R_{b,t'}$ and $E_{b,t'}$ for all blocks $b \in \mathcal{B}$ and all sons $t' \in \operatorname{sons}(t)$. This requires $2(\#L_{t'})(\#K_{b,t'})(\#K_{b,t})$ operations per block and son, leading to a total of

$$\sum_{b\in\mathcal{B}} \sum_{t'\in\operatorname{sons}(t)} 2(\#L_{t'})(\#K_{b,t'})(\#K_{b,t})$$

$$\leq 2 \sum_{b \in \mathcal{B}} \Big(\sum_{t' \in \operatorname{sons}(t)} (\#L_{t'})^2 \Big)^{1/2} \Big(\sum_{t' \in \operatorname{sons}(t)} (\#K_{b,t'})^2 \Big)^{1/2} (\#K_{b,t})$$
$$\leq 2 \sum_{b \in \mathcal{B}} \Big(\sum_{t' \in \operatorname{sons}(t)} \#L_{t'} \Big) \Big(\sum_{t' \in \operatorname{sons}(t)} \#K_{b,t'} \Big) (\#K_{b,t}) \leq 2 \sum_{b \in \mathcal{B}} l_t k_{b,t}^2$$

operations.

Once the matrices $\widehat{V}_{b,t}$ are available, the matrix $\widehat{X}_t^c$ is computed by multiplying $\widehat{V}_{b,t}$ by $Z_{b,t}^*$, which takes $2(\#M_t)(\#K_{b,t})(\widehat{K}_{b,t}) \leq 2(\#M_t)(\#K_{b,t})^2 \leq 2 l_t k_{b,t}^2$ operations per block and a total of

$$2 \sum_{b \in \mathcal{B}} l_t k_{b,t}^2 \quad \text{operations.}$$

We use Algorithm 18 to compute the matrix $\widehat{Q}_t$. Due to Lemma 5.21, this takes not more than

$$C_{\mathrm{pr}} (\#M_t)^2 (\#N_t) = C_{\mathrm{pr}} \Big(\sum_{t' \in \operatorname{sons}(t)} \#L_{t'} \Big)^2 (\#N_t) \leq C_{\mathrm{pr}} l_t^2 \sum_{b \in \mathcal{B}} \#\widehat{K}_{b,t}$$
$$\leq C_{\mathrm{pr}} l_t^2 \sum_{b \in \mathcal{B}} \#K_{b,t} \leq C_{\mathrm{pr}} l_t^2 \sum_{b \in \mathcal{B}} k_{b,t} \quad \text{operations.}$$

Finally the matrices $R_{b,t}$ are computed by $2(\#L_t)(\#M_t)(\#K_{b,t}) \leq 2 l_t^2 k_{b,t}$ operations per block, so we can conclude that a non-leaf cluster requires not more than

$$\sum_{b \in \mathcal{B}} 4 l_t k_{b,t}^2 + (C_{\mathrm{pr}} + 2) l_t^2 k_{b,t} \quad \text{operations.} \tag{6.39}$$

We can complete the proof by combining the estimates (6.38) and (6.39) to get the bound

$$\max\{4, C_{\mathrm{pr}} + 2\} \sum_{b \in \mathcal{B}} l_t k_{b,t}^2 + l_t^2 k_{b,t} \tag{6.40}$$

for one cluster, and summing the bounds for all clusters completes the first part of the proof. If L and K_b, $b \in \mathcal{B}$, are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, we can apply the Lemmas 3.45 and 3.48 to prove the asymptotic bound. □

The complexity estimate of Lemma 6.28 for Algorithm 31 requires explicit bounds for the rank distribution L of the new cluster basis Q, while the complexity estimate of Theorem 6.27 for the closely related Algorithm 30 uses only bounds for the original cluster bases. Assuming that the cardinality of $\mathcal{B}$ is bounded, we can derive bounds for L from bounds for the rank distributions $(K_b)_{b \in \mathcal{T}_{\mathcal{I}}}$ and find the following result, which closely resembles the one of Theorem 6.27.

Corollary 6.29. *Let $k_b = (k_{b,t})_{t \in \mathcal{T}_{\mathcal{I}}}$ be defined as in (3.16) for the rank distributions K_b, $b \in \mathcal{B}$. We assume that the weight matrices $(Z_{b,t})_{t \in \mathcal{T}_{\mathcal{I}}}$ have already been*

computed. Algorithm 31 requires not more than

$$\max\{4, C_{\mathrm{pr}} + 2\}((\#\mathcal{B})^3 + 1) \sum_{b\in\mathcal{B}} \sum_{t\in\mathcal{T}_\mathcal{I}} k_{b,t}^3$$

operations.

Algorithm 31. Recursion for finding unified cluster bases for multiple $\mathcal{H}^2$-matrices.

procedure UnifiedBasesRec(t, $(V_b)_{b\in\mathcal{B}}$, $(Z_b)_{b\in\mathcal{B}}$, $(\widehat{K}_b)_{b\in\mathcal{B}}$, ϵ, **var** Q, $(R_b)_{b\in\mathcal{B}}$, L)
  $N_t \leftarrow \emptyset$;
  for $b \in \mathcal{B}$ **do**
    $N_t \leftarrow N_t \dot\cup \widehat{K}_{b,t}$
  end for;
  if sons$(t) = \emptyset$ **then**
    $\widetilde{X}_t^c \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times N_t}$;
    for $b \in \mathcal{B}$ **do**
      $\widetilde{X}_t^c|_{\mathcal{I}\times\widehat{K}_{b,t}} \leftarrow V_{b,t} Z_{b,t}^*$
    end for;
    Lowrank($\widetilde{X}_t^c$, $\hat{t}$, ϵ_t, Q_t, $\widetilde{R}_t$, L_t); {Algorithm 18}
    for $b \in \mathcal{B}$ **do**
      $R_{b,t} \leftarrow Q_t^* V_{b,t}$
    end for
  else
    $M_t \leftarrow \emptyset$;
    for $t' \in$ sons(t) **do**
      UnifiedBasesRec(t', $(V_b)_{b\in\mathcal{B}}$, $(Z_b)_{b\in\mathcal{B}}$, $(\widehat{K}_b)_{b\in\mathcal{B}}$, ϵ, Q, $(R_b)_{b\in\mathcal{B}}$, L);
      $M_t \leftarrow M_t \dot\cup L_{t'}$
    end for;
    $\widehat{X}_t^c \leftarrow 0 \in \mathbb{R}^{M_t\times N_t}$;
    for $b \in \mathcal{B}$ **do**
      $\widehat{V}_{b,t} \leftarrow 0 \in \mathbb{R}^{M_t\times\widehat{K}_{b,t}}$;
      for $t' \in$ sons(t) **do**
        $\widehat{V}_{b,t}|_{L_{t'}\times\widehat{K}_{b,t}} \leftarrow R_{b,t'} E_{b,t'}$
      end for;
      $\widehat{X}_t^c|_{M_t\times\widehat{K}_{b,t}} \leftarrow \widehat{V}_{b,t} Z_{b,t}^*$
    end for;
    Lowrank($\widehat{X}_t^c$, M_t, ϵ_t, $\widehat{Q}_t$, $\widetilde{R}_t$, L_t); {Algorithm 18}
    for $b \in \mathcal{B}$ **do**
      $R_{b,t} \leftarrow \widehat{Q}_t^* \widehat{V}_{b,t}$
    end for
  end if

Proof. Let $l = (l_t)_{t\in\mathcal{T}_\mathcal{I}}$ be defined as in (6.18) for the rank distribution $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ of the cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$.

Lemma 5.19 implies that $\#L_t \leq \min\{\#M_t, \#N_t\}$ holds for all $t \in \mathcal{T}_\mathcal{I}$. The definition of N_t yields

$$\#L_t \leq \#N_t = \sum_{b\in\mathcal{B}} \#K_{b,t} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},$$

and this gives us

$$l_t \leq \sum_{b\in\mathcal{B}} k_{b,t} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}.$$

According to (6.40) and the simple estimate (5.9), the number of operations for a cluster $t \in \mathcal{T}_\mathcal{I}$ is bounded by

$$\max\{4, C_{\mathrm{pr}} + 2\} \sum_{b\in\mathcal{B}} k_{b,t}^2 l_t + k_{b,t} l_t^2 \leq \max\{4, C_{\mathrm{pr}} + 2\} \sum_{b\in\mathcal{B}} (k_{b,t}^3 + l_t^3).$$

We apply Hölder's inequality to the second term and find

$$l_t^3 \leq \Big(\sum_{b\in\mathcal{B}} k_{b,t}\Big)^3 \leq \Big(\sum_{b\in\mathcal{B}} 1^{3/2}\Big)^2 \Big(\sum_{b\in\mathcal{B}} k_{b,t}^3\Big) = (\#\mathcal{B})^2 \sum_{b\in\mathcal{B}} k_{b,t}^3, \tag{6.41}$$

therefore we can bound the number of operations for $t \in \mathcal{T}_\mathcal{I}$ by

$$\begin{aligned}\max\{4, C_{\mathrm{pr}} + 2\} &\sum_{b\in\mathcal{B}} \Big(k_{b,t}^3 + (\#\mathcal{B})^2 \sum_{b'\in\mathcal{B}} k_{b',t}^3\Big) \\ &= \max\{4, C_{\mathrm{pr}} + 2\}((\#\mathcal{B})^3 + 1) \sum_{b\in\mathcal{B}} k_{b,t}^3,\end{aligned}$$

and summing these bounds for all $t \in \mathcal{T}_\mathcal{I}$ completes the proof. □

Unification of submatrices

Now we can return our attention to the problem of constructing a unified cluster basis for a matrix consisting of several submatrices. Submatrices correspond to subtrees of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. The elements of these subtrees are formed from clusters contained in subtrees of the cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$. In order to handle these subtrees efficiently, we introduce the following notation:

Definition 6.30 (Subtrees). For $t \in \mathcal{T}_\mathcal{I}$, we denote the subtree of $\mathcal{T}_\mathcal{I}$ with root t by $\mathcal{T}_t$.

For $s \in \mathcal{T}_\mathcal{J}$, we denote the subtree of $\mathcal{T}_\mathcal{J}$ with root s by $\mathcal{T}_s$.

For $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we denote the subtree of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with root b by $\mathcal{T}_b$. We denote its leaves by $\mathcal{L}_b$ and its admissible leaves by $\mathcal{L}_b^+$.

Let $b_0 = (t_0, s_0) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a non-leaf block in $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. We assume that $X_b = \chi_t X_b \chi_s \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_b, W_b)$ holds for all $b = (t, s) \in \operatorname{sons}(b_0)$ with a row basis $(V_{b,t^*})_{t^*\in\mathcal{T}_t}$ and a column basis $(W_{b,s^*})_{s^*\in\mathcal{T}_s}$, i.e., that all submatrices are $\mathcal{H}^2$-matrices and represented accordingly.

Let $t \in \operatorname{sons}^+(t_0)$, and let

$$\mathcal{B} := \{(t, s) : s \in \operatorname{sons}^+(s_0)\} \subseteq \operatorname{sons}(b_0)$$

(with $\operatorname{sons}^+(s_0)$ as in Definition 3.13). We use Algorithm 28 to compute weight matrices $(Z_{b,t})_{t\in\mathcal{T}_\mathcal{I}}$ for all $b \in \mathcal{B}$, then we apply Algorithm 31 to determine the new orthogonal cluster basis V_{b_0} for all descendants of t (cf. Figure 6.5).

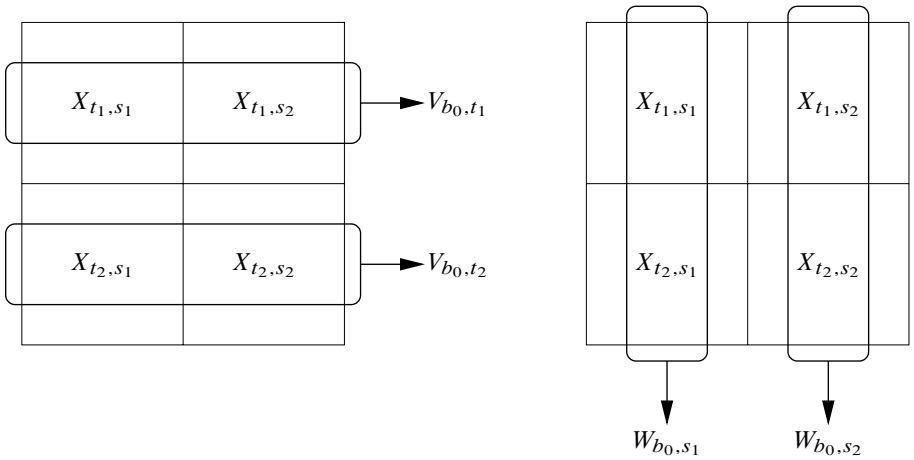


Figure 6.5. Computation of unified cluster bases V_{b_0} and W_{b_0} for a block matrix $b_0 = (t_0, s_0)$ with $\operatorname{sons}(t_0) = \{t_1, t_2\}$ and $\operatorname{sons}(s_0) = \{s_1, s_2\}$.

Once we have done this for all $t \in \operatorname{sons}^+(t_0)$, we have determined the cluster basis matrices $V_{b_0,t}$ for all $t \in \operatorname{sons}^*(t_0) \setminus \{t_0\}$. Since no admissible block uses V_{b_0,t_0}, we can set it equal to zero and complete the construction of the new cluster basis V_{b_0}.

We repeat this procedure for the column cluster basis W_{b_0} and project all submatrices X_b into the new $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V_{b_0}, W_{b_0})$ by using the corresponding cluster operators provided by Algorithm 31. The procedure is summarized in Algorithm 32.

To keep the notation simple, we denote the coupling matrices of X_b^* by $S_b^* = ((S_b^*)_{s^*,t^*})_{b^*=(t^*,s^*)\in\mathcal{L}_b^+}$, defined as

$$(S_b^*)_{s^*,t^*} := (S_{b,(t^*,s^*)})^*$$

for all $b^* = (t^*, s^*) \in \mathcal{L}_b^+$. This allows us to compute total weights for X_b^* by applying Algorithm 28 to the transposed block cluster tree $\mathcal{T}_b^*$ and the coupling matrices S_b^*.

Algorithm 32. Construct an approximation $X_{b_0} \in \mathcal{H}^2(\mathcal{T}_{b_0}, V_{b_0}, W_{b_0})$ by combining approximations $X_b \in \mathcal{H}^2(\mathcal{T}_b, V_b, W_b)$ of the subblocks $b \in \operatorname{sons}(b_0)$.

procedure UnifySubmatrices(b_0, $(V_b)_{b\in\operatorname{sons}(b_0)}$, $(W_b)_{b\in\operatorname{sons}(b_0)}$, $(S_b)_{b\in\operatorname{sons}(b_0)}$, ϵ, **var** V_{b_0}, W_{b_0}, S_{b_0})
$(t_0, s_0) \leftarrow b_0$;
{Construct row cluster basis V_{b_0}}
for $t \in \operatorname{sons}^+(t_0)$ **do**
  $\mathcal{B} \leftarrow \emptyset$;
  for $s \in \operatorname{sons}^+(s_0)$ **do**
    $b \leftarrow (t, s)$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{b\}$;
    Orthogonalize(t, W_b, Q, R_b, L_b); {Algorithm 16, Q is not used}
    TotalWeights(t, $\mathcal{T}_b$, $\emptyset$, 0, V_b, R_b, S_b, Z_b, $\widehat{K}_b$) {Algorithm 28}
  end for;
  UnifiedBasesRec(t, $(V_b)_{b\in\mathcal{B}}$, $(Z_b)_{b\in\mathcal{B}}$, $(\widehat{K}_b)_{b\in\mathcal{B}}$, ϵ, V_{b_0}, $(R_{V,b})_{b\in\mathcal{B}}$, K_{b_0})
end for;
{Construct column cluster basis W_{b_0}}
for $s \in \operatorname{sons}^+(s_0)$ **do**
  $\mathcal{B} \leftarrow \emptyset$;
  for $t \in \operatorname{sons}^+(t_0)$ **do**
    $b \leftarrow (t, s)$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{b\}$;
    Orthogonalize(t, V_b, Q, R_b, L_b); {Algorithm 16, Q is not used}
    TotalWeights(s, $\mathcal{T}_b^*$, $\emptyset$, 0, W_b, R_b, S_b^*, Z_b, $\widehat{K}_b$) {Algorithm 28}
  end for;
  UnifiedBasesRec(s, $(W_b)_{b\in\mathcal{B}}$, $(Z_b)_{b\in\mathcal{B}}$, $(\widehat{K}_b)_{b\in\mathcal{B}}$, ϵ, W_{b_0}, $(R_{W,b})_{b\in\mathcal{B}}$, L_{b_0})
end for;
{Switch coupling matrices to new bases}
for $b \in \operatorname{sons}(b_0)$ **do**
  for $b^* = (t^*, s^*) \in \mathcal{L}_b^+$ **do**
    $S_{b_0,b^*} \leftarrow R_{V,b,t^*} S_{b,b^*} R_{W,b,s^*}^*$
  end for
end for

Lemma 6.31 (Complexity). *Let $b_0 \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and let row cluster bases V_b with rank distributions K_b and columns cluster bases W_b with rank distributions L_b be given for all $b \in \operatorname{sons}(b_0)$. Let k_b and l_b be defined as in (3.16) and (3.18) for K_b and L_b.*

Let K_{b_0} and L_{b_0} be the rank distributions computed by Algorithm 32, and let k_{b_0} and l_{b_0} be defined as before. Let

$$\hat{k}_{t^*} := \max\{k_{b_0,t^*},\ k_{b,t^*} \ :\ b = (t,s) \in \operatorname{sons}(b_0) \text{ with } t^* \in \mathcal{T}_t\},$$
$$\hat{l}_{s^*} := \max\{l_{b_0,s^*},\ l_{b,s^*} \ :\ b = (t,s) \in \operatorname{sons}(b_0) \text{ with } s^* \in \mathcal{T}_s\}$$

for all $t^* \in \mathcal{T}_t$, $s^* \in \mathcal{T}_s$, $(t,s) \in \operatorname{sons}(b_0)$. *Algorithm 32 requires not more than*

$$\sum_{b=(t,s)\in\operatorname{sons}(b_0)} \Bigl(C_{\mathrm{uc}} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + C_{\mathrm{uc}} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 + C_{\mathrm{ub}} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \Bigr)$$

operations, where $C_{\mathrm{uc}}, C_{\mathrm{ub}} \in \mathbb{N}$ *are constants that depend only on* C_{qr} *and* C_{pr}.

Proof. We first consider the construction of the unified row basis V_{b_0}. According to Lemma 5.18, the weights R_b for each $b = (t,s) \in \operatorname{sons}(b_0)$ can be computed in

$$2\max\{2, C_{\mathrm{qr}}\} \sum_{s\in\mathcal{T}_s} \hat{l}_{s^*}^3$$

operations, and Lemma 6.26 states that the weights of the total cluster basis for the block b can be computed in

$$\max\{2, C_{\mathrm{qr}}\} \Bigl(2 \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \Bigr)$$

operations. Together we get a bound of

$$\max\{2, C_{\mathrm{qr}}\} \Bigl(2 \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + 2 \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 + \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \Bigr) \tag{6.42}$$

for each block. Once the total weights for all blocks have been computed, Algorithm 31 is applied to each $t \in \operatorname{sons}^+(t_0)$, and Lemma 6.28 yields the bound

$$2\max\{4, C_{\mathrm{pr}} + 2\} \sum_{\substack{b=(t,s)\\ s\in\operatorname{sons}^+(s_0)}} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3$$

for each $t \in \operatorname{sons}^+(t_0)$. Adding up gives us the bound

$$2\max\{4, C_{\mathrm{pr}} + 2\} \sum_{b=(t,s)\in\operatorname{sons}(b_0)} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3. \tag{6.43}$$

By similar arguments, (6.42) is also a bound for the construction of the weights for the unified column basis W_{b_0}, and the construction of the basis itself requires not more than

$$2\max\{4, C_{\mathrm{pr}} + 2\} \sum_{b=(t,s)\in\operatorname{sons}(b_0)} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 \tag{6.44}$$

operations. We assume that the computation of the new coupling matrices S_{b_0,b^*} for all $b \in \operatorname{sons}(b_0)$ and $b^* \in \mathcal{L}_b^+$ is performed from right to left: the product $S_{b,b^*} R_{W,b,s^*}^*$

can be computed in $2(\#K_{b,t^*})(\#L_{b,s^*})(\#L_{b_0,s^*}) \leq 2\hat{k}_{t^*}\hat{l}_{s^*}^2$ operations, and the product $R_{V,b,t^*} S_{b,b^*} R_{W,b,s^*}^*$ can be computed in $2(\#K_{b_0,t^*})(\#K_{b,t^*})(\#L_{b_0,s^*}) \leq 2\hat{k}_{t^*}^2\hat{l}_{s^*}$ additional operations. Constructing all coupling matrices therefore takes not more than

$$\sum_{b\in\operatorname{sons}(b_0)} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} 2\hat{k}_{t^*}\hat{l}_{s^*}^2 + 2\hat{k}_{t^*}^2\hat{l}_{s^*} \leq 2 \sum_{b\in\operatorname{sons}(b_0)} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \tag{6.45}$$

operations. Here we have used (5.9) again to separate the factors in the sum.

Adding (6.42) twice, (6.43), (6.44) and (6.45) yields the upper bound

$$\sum_{b=(t,s)\in\operatorname{sons}(b_0)} \left(C_{\mathrm{uc}} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + C_{\mathrm{uc}} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 + C_{\mathrm{ub}} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \right)$$

with the constants

$$C_{\mathrm{uc}} := 4\max\{2, C_{\mathrm{qr}}\} + 2\max\{4, C_{\mathrm{pr}} + 2\}, \quad C_{\mathrm{ub}} := 2\max\{2, C_{\mathrm{qr}}\} + 2. \qquad \square$$

Hierarchical compression

Now that we have the unification Algorithm 32 at our disposal, we can convert an $\mathcal{H}$-matrix into an $\mathcal{H}^2$-matrix by the blockwise approach of Algorithm 33: we recursively merge submatrices until we have found an approximation of the entire matrix [14].

Algorithm 33. Blockwise conversion of an $\mathcal{H}$-matrix into an $\mathcal{H}^2$-matrix with adaptively chosen cluster bases.

procedure ConvertHtoH2(b, X, ϵ, **var** V_b, W_b, S_b)
$(t,s) \leftarrow b$;
if $b \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^-$ **then**
  {Nothing to do for inadmissible leaves}
else if $b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+$ **then**
  $V_{b,t} \leftarrow A_b \in \mathbb{R}^{\mathcal{I}\times K_b}_{\hat{t}}$; $W_{b,s} \leftarrow B_b \in \mathbb{R}^{\mathcal{J}\times K_b}_{\hat{s}}$; $S_{b,b} \leftarrow I \in \mathbb{R}^{K_b\times K_b}$
  $K_{b,t} \leftarrow K_b$; $L_{b,s} \leftarrow K_b$
else
  for $b' \in \operatorname{sons}(b)$ **do**
    $(t',s') \leftarrow b'$; $X_{b'} \leftarrow \chi_{t'} X \chi_{s'}$;
    ConvertHtoH2(b', $X_{b'}$, ϵ, $V_{b'}$, $W_{b'}$, $S_{b'}$)
  end for;
  UnifySubmatrices(b, $(V_{b'})_{b'\in\operatorname{sons}(b)}$, $(W_{b'})_{b'\in\operatorname{sons}(b)}$, $(S_{b'})_{b'\in\operatorname{sons}(b)}$, ϵ_b, V_b, W_b, S_b)
end if

Theorem 6.32 (Complexity). *Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be a hierarchical matrix for the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, given in the form (3.12). Similar to Lemma 5.12, we let*

$$\widehat{K}_t := \{1,\ldots,\max\{k_{b,t} \,:\, b = (t^+,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},\ t^+ \in \operatorname{pred}(t)\}\} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$
$$\widehat{L}_s := \{1,\ldots,\max\{l_{b,s} \,:\, b = (t,s^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},\ s^+ \in \operatorname{pred}(s)\}\} \quad \text{for all } s \in \mathcal{T}_{\mathcal{J}}$$

denote index sets for the maximal ranks. Let $(k_{b,t^*})_{t^*\in\mathcal{T}_t}$ and $(l_{b,s^*})_{s^*\in\mathcal{T}_s}$ be defined as in (3.16) and (3.18) for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and let $(\hat{k}_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(\hat{l}_s)_{s\in\mathcal{T}_{\mathcal{J}}}$ be defined similarly for $\widehat{K}$ and $\widehat{L}$. Let $p_{\mathcal{I}\times\mathcal{J}}$ be the depth of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Algorithm 33 requires not more than

$$C_{\mathrm{sp}}(C_{\mathrm{uc}} + C_{\mathrm{ub}})(p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3\Big)$$

operations. If $\widehat{K}_t$, $\widehat{L}_s$ are $(C_{\mathrm{bn}},\alpha,\beta,r,\xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ are $(C_{\mathrm{rc}},\alpha,\beta,r,\xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(p_{\mathcal{I}\times\mathcal{J}}+1)(n_{\mathcal{I}}+n_{\mathcal{J}}))$.

Proof. The algorithm only performs work for blocks $b_0 \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ that are not leaves, and the number of operations for each of these blocks can be bounded by Lemma 6.31. Adding these bounds yields

$$\begin{aligned}
&\sum_{b_0\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}\setminus\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \sum_{b=(t,s)\in\operatorname{sons}(b_0)} \Big(C_{\mathrm{uc}} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + C_{\mathrm{uc}} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 \\
&\qquad\qquad + C_{\mathrm{ub}} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} \hat{k}_{t^*}^3 + \hat{l}_{s^*}^3\Big) \\
&\leq \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \Big(C_{\mathrm{uc}} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + C_{\mathrm{uc}} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 \\
&\qquad\qquad + C_{\mathrm{ub}} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} \hat{k}_{t^*}^3 + \hat{l}_{s^*}^3\Big) \\
&= C_{\mathrm{uc}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{s\in\operatorname{row}(t)} \sum_{t^*\in\mathcal{T}_t} \hat{k}_{t^*}^3 + C_{\mathrm{uc}} \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ \operatorname{level}(s)\leq p_{\mathcal{I}\times\mathcal{J}}}} \sum_{t\in\operatorname{col}(s)} \sum_{s^*\in\mathcal{T}_s} \hat{l}_{s^*}^3 \\
&\quad + C_{\mathrm{ub}} \sum_{b\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} \hat{k}_{t^*}^3 + \hat{l}_{s^*}^3 \\
&\leq C_{\mathrm{uc}} C_{\mathrm{sp}} \sum_{t^*\in\mathcal{T}_{\mathcal{I}}} \hat{k}_{t^*}^3 \#\{t \in \operatorname{pred}(t^*) \,:\, \operatorname{level}(t) \leq p_{\mathcal{I}\times\mathcal{J}}\} \\
&\quad + C_{\mathrm{uc}} C_{\mathrm{sp}} \sum_{s^*\in\mathcal{T}_{\mathcal{J}}} \hat{l}_{s^*}^3 \#\{s \in \operatorname{pred}(s^*) \,:\, \operatorname{level}(s) \leq p_{\mathcal{I}\times\mathcal{J}}\} \\
&\quad + C_{\mathrm{ub}} \sum_{b^*\in\mathcal{L}_b^+} (\hat{k}_{t^*}^3 + \hat{l}_{s^*}^3) \#\{b \in \operatorname{pred}(b^*)\}
\end{aligned}$$

$$\begin{aligned}
&\leq C_{\mathrm{uc}} C_{\mathrm{sp}} (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \Big) \\
&\quad + C_{\mathrm{ub}} (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\mathrm{row}^+(t)} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \sum_{t\in\mathrm{col}^+(s)} \hat{l}_s^3 \Big) \\
&\leq C_{\mathrm{uc}} C_{\mathrm{sp}} (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \Big) \\
&\quad + C_{\mathrm{ub}} C_{\mathrm{sp}} (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \Big) \\
&= C_{\mathrm{sp}} (C_{\mathrm{uc}} + C_{\mathrm{ub}}) (p_{\mathcal{I}\times\mathcal{J}} + 1)\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \Big),
\end{aligned}$$

where we have used again the C_{sp}-sparsity of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and the fact that we have $\mathrm{level}(t)$, $\mathrm{level}(s) \leq \mathrm{level}(b) \leq p_{\mathcal{I}\times\mathcal{J}}$ for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ due to Definition 3.12.

If $\widehat{K}$ and $\widehat{L}$ are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, we can again apply Lemmas 3.45 and 3.48 to complete the proof. □

Remark 6.33 (Error control). Since the Algorithm 31 used to construct the unified cluster bases is only an efficient realization of the original compression Algorithm 25, the error estimates provided by Corollary 6.22 and the comments of Remark 6.24 apply, i.e., we can guarantee any given accuracy by choosing the error tolerances used in the truncation procedures appropriately.

Using the simple recursive estimate

$$\begin{aligned}
\|\chi_t X \chi_s - X_b\| &= \Big\| \chi_t X \chi_s - \sum_{b'\in\mathrm{sons}(b)} X_{b'} + \sum_{b'\in\mathrm{sons}(b)} X_{b'} - X_b \Big\| \\
&\leq \sum_{b'=(t',s')\in\mathrm{sons}(b)} \|\chi_{t'} X \chi_{s'} - X_{b'}\| + \Big\| \sum_{b'\in\mathrm{sons}(b)} X_{b'} - X_b \Big\|
\end{aligned}$$

for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we can see that the total error resulting from Algorithm 33 can be bounded by the sum of the errors introduced by the unification Algorithm 32 for all non-leaf blocks of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, i.e., we can guarantee a given accuracy of the result by ensuring that the intermediate approximation steps are sufficiently accurate. □

6.8 Refined error control and variable-rank approximation

The compression Algorithms 25, 27, 30 and 32 allow us to control only the total error by means of Corollary 6.22. In some situations, it is desirable to ensure that the error *in individual blocks* can be controlled. Important applications include estimates for the

relative error and the algebraic counterpart of the variable-order schemes introduced in Section 4.7.

We assume that an error tolerance $\epsilon_b \in \mathbb{R}_{>0}$ is given for all blocks $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, and our goal is to ensure

$$\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2 \leq \epsilon_b \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, \tag{6.46}$$

i.e., that the spectral error in each *individual* admissible block $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ is bounded by the corresponding parameter ϵ_b.

Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. According to Lemma 5.7, we have $\chi_t X \chi_s = X_t \chi_s$. Applying Theorem 5.30 yields

$$\|(X_t - Q_t Q_t^* X_t)\chi_s\|_2^2 \leq \sum_{r\in\operatorname{sons}^*(t)} \|(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r) E_{r,t} \chi_s\|_2^2,$$

where the transfer matrix $E_{r,t}$ for the total cluster basis is given by equation (6.6) of Lemma 6.12. Since $s \in \operatorname{row}^*(r)$ holds for all $r \in \operatorname{sons}^*(t)$, we can use Lemma 5.7 again to prove $E_{r,t}\chi_s = \chi_s$ and get the estimate

$$\|(X_t - Q_t Q_t^* X_t)\chi_s\|_2^2 \leq \sum_{r\in\operatorname{sons}^*(t)} \|(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r)\chi_s\|_2^2. \tag{6.47}$$

The quantities on the right-hand side of this estimate appear explicitly during the course of the compression Algorithm 25, and Lemma 5.19 allows us to ensure that they are below a prescribed bound.

Therefore we now have to find bounds for the right-hand side that ensure that the left-hand side of the estimate is bounded by ϵ_b^2. As in Remark 5.33, we achieve this by turning the sum into a geometric sum: we fix $p \in (0,1)$ and assume

$$\|(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r)\chi_s\|_2 \leq \hat{\epsilon}_b p^{\operatorname{level}(r)-\operatorname{level}(t)} \quad \text{for all } r \in \operatorname{sons}^*(t), \tag{6.48}$$

for a constant $\hat{\epsilon}_b \in \mathbb{R}_{>0}$ we will specify later. Under this assumption, we find

$$\begin{aligned}
\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2^2 &\leq \sum_{r\in\operatorname{sons}^*(t)} \|(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r)\chi_s\|_2^2 \\
&\leq \hat{\epsilon}_b^2 \sum_{r\in\operatorname{sons}^*(t)} p^{2(\operatorname{level}(r)-\operatorname{level}(t))} \\
&= \hat{\epsilon}_b^2 \sum_{\ell=\operatorname{level}(t)}^{p_{\mathcal{I}}} p^{2(\ell-\operatorname{level}(t))} \#\{r \in \operatorname{sons}^*(t) \,:\, \operatorname{level}(r) = \ell\}.
\end{aligned} \tag{6.49}$$

In order to control the right-hand side, we have to assume that the cluster tree $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{bc}}, \alpha, \beta, r, \xi)$-bounded, since this implies $\#\operatorname{sons}(t) \leq C_{\mathrm{bc}}$ for all $t \in \mathcal{T}_{\mathcal{I}}$, and a simple induction yields

$$\#\{r \in \operatorname{sons}^*(t) \,:\, \operatorname{level}(r) = \ell\} \leq C_{\mathrm{bc}}^{\ell-\operatorname{level}(t)}.$$

Assuming $p^2 < 1/C_{\mathrm{bc}} \leq 1$, we can combine the estimate with (6.49) and find

$$\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2^2 \leq \hat{\epsilon}_b^2 \sum_{\ell=\mathrm{level}(t)}^{p_{\mathcal{I}}} (C_{\mathrm{bc}} p^2)^{\ell-\mathrm{level}(t)}$$
$$< \hat{\epsilon}_b^2 \sum_{\ell=0}^{\infty} (C_{\mathrm{bc}} p^2)^{\ell} = \hat{\epsilon}_b^2 \frac{1}{1 - C_{\mathrm{bc}} p^2}.$$

If we ensure $\hat{\epsilon}_b \leq \epsilon_b \sqrt{1 - C_{\mathrm{bc}} p^2}$, we get

$$\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2^2 < \hat{\epsilon}_b^2 \frac{1}{1 - C_{\mathrm{bc}} p^2} \leq \epsilon_b^2, \tag{6.50}$$

i.e., we have reached our goal provided that we can ensure (6.48). For a single block, this is straightforward. For *all* blocks, we have to use a simple reformulation: we introduce the weights

$$\omega_{r,s} := \hat{\epsilon}_b p^{\mathrm{level}(r)-\mathrm{level}(t)} \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},\ r \in \mathrm{sons}^*(t), \tag{6.51}$$

which are well-defined since due to Corollary 3.15 there is exactly one $t \in \mathrm{pred}(r)$ with $(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, and observe that the inequalities

$$\|(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r)\chi_s\|_2^2 \leq \omega_{r,s}^2 = \hat{\epsilon}_b^2 p^{2(\mathrm{level}(r)-\mathrm{level}(t))}$$

and

$$\|\omega_{r,s}^{-1}(\hat{X}_r - \hat{Q}_r \hat{Q}_r^* \hat{X}_r)\chi_s\|_2^2 \leq 1 \tag{6.52}$$

are equivalent. The second inequality has the advantage that we can "hide" the additional weight in the matrix $\hat{X}_r$ by introducing a suitable generalization of a total cluster basis:

Definition 6.34 (Weighted total cluster basis). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be a matrix and $\omega = (\omega_{r,s})_{r\in\mathcal{T}_{\mathcal{I}}, s\in\mathrm{row}^*(r)}$ a family of weights. The family $(X_{\omega,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ defined by

$$X_{\omega,t} := \sum_{t^+\in\mathrm{pred}(t)} \sum_{s\in\mathrm{row}^+(t^+)} \omega_{r,s}^{-1} \chi_t X \chi_s \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}$$

is called the *weighted total cluster basis* corresponding to X and ω.

The relation between weighted and original total cluster basis is given by

$$X_{\omega,r}\chi_s = \omega_{r,s}^{-1}\chi_r X \chi_s = \omega_{r,s}^{-1} X_r \chi_s \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, r \in \mathrm{sons}^*(t),$$

and the inequality (6.52) takes the form

$$\|(\hat{X}_{\omega,r} - \hat{Q}_r \hat{Q}_r^* \hat{X}_{\omega,r})\chi_s\|_2^2 \leq 1 \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, r \in \mathrm{sons}^*(t). \tag{6.53}$$

Since we have to treat all blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ simultaneously, we use the condition

$$\|\widehat{X}_{\omega,r} - \widehat{Q}_r \widehat{Q}_r^* \widehat{X}_{\omega,r}\|_2^2 \leq 1 \quad \text{for all } r \in \mathcal{T}_{\mathcal{I}}, \tag{6.54}$$

which, by definition, implies (6.53) for all admissible blocks and fits perfectly into the context of Algorithm 25.

Due to (6.51), the weights $\omega_{t,s}$ can be computed by the recursion

$$\omega_{t,s} := \begin{cases} \hat{\epsilon}_b & \text{if } (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, \\ p\omega_{t^+,s} & \text{otherwise with } t^+ = \operatorname{father}(t) \end{cases} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}, s \in \operatorname{row}^*(t).$$

Including this recursion in Algorithm 25 yields the new Algorithm 34 for computing an adaptive cluster basis for weighted matrices. The additional complexity compared to Algorithm 25 is negligible, therefore the complexity estimate of Lemma 6.20 also holds for Algorithm 34.

Of course, the same modifications can also be included in the Algorithms 27 and 30 for converting $\mathcal{H}$- and $\mathcal{H}^2$-matrices.

Now let us consider two important applications of the new algorithm. The first is the construction of cluster bases ensuring that the *blockwise relative error* of the matrix approximation is bounded, i.e., that

$$\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2 \leq \epsilon \|\chi_t X \chi_s\|_2 \quad \text{holds for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

This property is very important for adaptive-rank arithmetic operations (cf. [49]). Obviously, the condition fits well into the framework presented here: we only have to apply Algorithm 34 to $\epsilon_b := \epsilon \|\chi_t X \chi_s\|_2$, i.e., to

$$\hat{\epsilon}_b := \epsilon \|\chi_t X \chi_s\|_2 \sqrt{1 - C_{\mathrm{bc}} p^2}.$$

In practice, the local norms $\|\chi_t X \chi_s\|_2$ can be approximated by a power iteration. For dense matrices, this is straightforward, for $\mathcal{H}$- and $\mathcal{H}^2$-matrices, we can use the techniques introduced in Section 5.6 to reduce the complexity: by Householder factorizations or Algorithm 16, we find orthogonal transformations which reduce $\chi_t X \chi_s$ to a small matrix, and applying the power iteration to this matrix is very efficient.

Remark 6.35 (Frobenius norm). The construction also applies to the Frobenius norm: Lemma 5.19 allows us to control the approximation error with respect to this norm in each step of the compression algorithm using the weighted matrix, i.e., to ensure

$$\|\widehat{X}_{\omega,r} - \widehat{Q}_r \widehat{Q}_r^* \widehat{X}_{\omega,r}\|_F^2 \leq 1 \quad \text{for all } r \in \mathcal{T}_{\mathcal{I}},$$

and using $\epsilon_b := \epsilon \|\chi_t X \chi_s\|_F$ and $\hat{\epsilon}_b := \epsilon \|\chi_t X \chi_s\|_F \sqrt{1 - C_{\mathrm{bc}} p^2}$ yields

$$\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_F \leq \epsilon \|\chi_t X \chi_s\|_F \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Algorithm 34. Algorithm for finding adaptive cluster bases for weighted dense matrices.

procedure AdaptiveDenseWeighted(t, r_{t^+}, X, ϵ, **var** Q, R, L)
$r_t \leftarrow \emptyset$;
for $s \in r_{t^+}$ **do**
  $\omega_{t,s} \leftarrow p\omega_{t^+,s}$; $r_t \leftarrow r_t \cup \{s\}$
end for;
for $s \in \operatorname{row}^+(t)$ **do**
  $b \leftarrow (t, s)$; $\omega_{t,s} \leftarrow \hat{\epsilon}_b$; $r_t \leftarrow r_t \cup \{s\}$
end for;
if $\operatorname{sons}(t) = \emptyset$ **then**
  $X_{\omega,t} \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\hat{t}}$;
  for $s \in r_t$ **do**
    $X_{t,s} \leftarrow \chi_t X \chi_s$; $X_{\omega,t} \leftarrow X_{\omega,t} + \omega_{t,s}^{-1} X_{t,s}$
  end for;
  Lowrank($X_{\omega,t}$, $\hat{t}$, 1, Q_t, $R_{\omega,t}$, L_t); {Algorithm 18}
  for $s \in r_t$ **do**
    $R_{t,s} \leftarrow \omega_{t,s} R_{\omega,t} \chi_s$
  end for
else
  $M_t \leftarrow \emptyset$;
  for $t' \in \operatorname{sons}(t)$ **do**
    AdaptiveDenseWeighted(t', r_t, X, ϵ, Q, R, L); $M_t \leftarrow M_t \dot{\cup} L_{t'}$
  end for;
  $\widehat{X}_{\omega,t} \leftarrow 0 \in \mathbb{R}^{M_t\times\mathcal{J}}$;
  for $s \in r_t$ **do**
    $\widehat{X}_{t,s} \leftarrow 0 \in \mathbb{R}^{M_t\times\mathcal{J}}$;
    for $t' \in \operatorname{sons}(t)$ **do**
      $\widehat{X}_{t,s}|_{L_{t'}\times\mathcal{J}} \leftarrow R_{t',s}$
    end for;
    $\widehat{X}_{\omega,t} \leftarrow \widehat{X}_t + \omega_{t,s}^{-1} \widehat{X}_{t,s}$
  end for;
  Lowrank($\widehat{X}_{\omega,t}$, $\hat{t}$, 1, $\widehat{Q}_t$, $R_{\omega,t}$, L_t); {Algorithm 18}
  for $t' \in \operatorname{sons}(t)$ **do**
    $F_{t'} \leftarrow \widehat{Q}_t|_{L_{t'}\times L_t}$
  end for;
  for $s \in r_t$ **do**
    $R_{t,s} \leftarrow \omega_{t,s} R_{\omega,t} \chi_s$
  end for
end if

Due to the special properties of the Frobenius norm, we can even prove

$$\begin{aligned}\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_F^2 &= \sum_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_F^2 \\ &\leq \sum_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \epsilon_b^2 = \sum_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \epsilon^2 \|\chi_t X \chi_s\|_F^2 \\ &\leq \sum_{b\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon^2 \|\chi_t X \chi_s\|_F^2 = \epsilon^2 \|X\|_F^2,\end{aligned}$$

i.e., the *global* relative error is also bounded. □

The second application of Algorithm 34 is related to the variable-order approximation schemes introduced in Section 4.7. We have seen that these schemes can significantly improve the approximation quality while retaining the optimal order of complexity. In general situations, e.g., for kernel functions involving derivatives and for complicated geometries, the construction of a good variable-order interpolation scheme can be fairly complicated.

Fortunately, the variable-order scheme is only a means to an end: we are only interested in the resulting $\mathcal{H}^2$-matrix, not in the way it was constructed. By choosing the weights for Algorithm 34 correctly, we can ensure that the resulting $\mathcal{H}^2$-matrix has the same properties as one constructed by a variable-order interpolation scheme without actually using this scheme.

Given an error tolerance $\epsilon \in \mathbb{R}_{>0}$, we want to ensure

$$\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_2 \leq \epsilon, \tag{6.55}$$

and we want to keep the ranks of the cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ as low as possible. To reach this goal, we combine the spectral norm estimate of Theorem 4.47 with another geometric sum: we let $\hat{\epsilon} \in \mathbb{R}_{>0}$ and $\xi \in (0,1)$ and use the weights

$$\hat{\epsilon}_b := \hat{\epsilon}\xi^{(p_\mathcal{I}-\operatorname{level}(t))/2}\xi^{(p_\mathcal{J}-\operatorname{level}(s))/2} \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

For these weights, (6.50) gives us

$$\begin{aligned}&\|\chi_t X \chi_s - Q_t Q_t^* X \chi_s\|_2 \\ &\quad\leq \hat{\epsilon}\frac{\xi^{(p_\mathcal{I}-\operatorname{level}(t))/2}\xi^{(p_\mathcal{J}-\operatorname{level}(s))/2}}{\sqrt{1-C_{\mathrm{bc}}p^2}} \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},\end{aligned}$$

and Theorem 4.47 yields

$$\begin{aligned}\|X - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}},Q,*}[X]\|_2 &\leq \frac{C_{\mathrm{sp}}\hat{\epsilon}}{\sqrt{1-C_{\mathrm{bc}}p^2}}\Big(\sum_{\ell=0}^{p_\mathcal{I}}\xi^{p_\mathcal{I}-\ell}\Big)^{1/2}\Big(\sum_{\ell=0}^{p_\mathcal{J}}\xi^{p_\mathcal{J}-\ell}\Big)^{1/2} \\ &< \frac{C_{\mathrm{sp}}\hat{\epsilon}}{\sqrt{1-C_{\mathrm{bc}}p^2}}\sum_{\ell=0}^{\infty}\xi^\ell = \frac{C_{\mathrm{sp}}\hat{\epsilon}}{\sqrt{1-C_{\mathrm{bc}}p^2}}\frac{1}{1-\xi},\end{aligned}$$

i.e., the estimate (6.55) holds if we can ensure

$$\hat{\epsilon} \leq \frac{(1-\xi)\sqrt{1-C_{\mathrm{bc}}p^2}}{C_{\mathrm{sp}}}\epsilon.$$

Choosing values for $\hat{\epsilon}_b$ and $\omega_{t,s}$ is simple, the interesting part is to demonstrate that these choices lead to a reasonable complexity of the resulting approximation.

As an example, we consider the variable-order approximation of matrices corresponding to integral operators introduced in Section 4.7: we consider the matrix $G \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ defined by

$$G_{ij} := \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\psi_j(y)\,dy\,dx \quad \text{for all } i \in \mathcal{I}, j \in \mathcal{J}, \tag{6.56}$$

where the subdomain or submanifold $\Omega \in \mathbb{R}^d$, the kernel function g and the finite element basis functions $(\varphi_i)_{i\in\mathcal{I}}$ and $(\psi_j)_{j\in\mathcal{J}}$ are as in Chapter 4.

Our goal is to prove that applying Algorithm 34 to this matrix leads to cluster bases that are at least as efficient as the ones constructed in Section 4.7: a quasi-optimality property (cf. Lemma 6.23) allows us to use general results of polynomial interpolation theory to derive upper bounds for the rank, the complicated analysis of the stability of re-interpolation carried out in Section 4.7 can be avoided, since Algorithm 34 replaces the re-interpolation operators by simple orthogonal projections that are always stable.

For variable-order parameters $\alpha, \beta \in \mathbb{N}$ – which we will specify later – let the order vectors $(m_t)_{t\in\mathcal{T}_\mathcal{I}}$ be defined as in (4.61) and the corresponding interpolation operators $(\mathfrak{I}_t)_{t\in\mathcal{T}_\mathcal{I}}$ as in (4.17).

Assuming that the cluster tree $\mathcal{T}_\mathcal{I}$ is quasi-balanced, Lemma 4.52 guarantees that the rank distribution $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ corresponding to the order vectors $(m_t)_{t\in\mathcal{T}_\mathcal{I}}$ is $(C_{\mathrm{bn}}, \alpha, \beta, d, \xi)$-bounded.

We require the following generalization of Lemma 6.23:

Lemma 6.36 (Weighted quasi-optimality). *Let $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ be the rank distribution of the orthogonal cluster basis Q computed by Algorithm 34. Let $(A_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(B_t)_{t\in\mathcal{T}_\mathcal{I}}$ be families of matrices satisfying $A_t \in \mathbb{R}^{\hat{\mathcal{I}}_t\times K_t}$, $B_t \in \mathbb{R}^{\mathcal{J}\times K_t}$, and let $(G_{\omega,t})_{t\in\mathcal{T}_\mathcal{I}}$ be the weighted total cluster bases corresponding to G. If*

$$\|G_{\omega,t} - A_t B_t^*\|_2 \leq 1 \quad \text{holds for all } t \in \mathcal{T}_\mathcal{I}, \tag{6.57}$$

we have $\#L_t \leq \#K_t$ for all $t \in \mathcal{T}_\mathcal{I}$, i.e., L is also $(C_{\mathrm{bn}}, \alpha, \beta, d, \xi)$-bounded.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. We define

$$\widehat{A}_t := \begin{cases} U_t^* A_t & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ A_t & \text{otherwise} \end{cases} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

and the orthogonality of U_t yields

$$\|\widehat{G}_{\omega,t} - \widehat{A}_t B_t^*\|_2 \leq \|G_{\omega,t} - A_t B_t^*\|_2 \leq 1 \quad \text{for all } t \in \mathcal{T}_\mathcal{I}.$$

This means that $\widehat{A}_t B_t^*$ is an approximation of $\widehat{G}_{\omega,t}$ with rank $\leq \#K_t$. Since the matrices $\widehat{Q}_t$ and the corresponding rank distribution $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ are constructed by singular value decompositions in Algorithm 18, the quasi-optimality stated in Remark 5.22 implies that the rank of the matrices $\widehat{Q}_t$ cannot be larger than $\#K_t$. Since the matrices $\widehat{Q}_t$ are orthogonal, this implies $\#L_t \leq \#K_t$. □

In order to be able to apply this lemma, we have to construct the matrix families $(A_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(B_t)_{t\in\mathcal{T}_\mathcal{I}}$. Among several possible approaches, including Taylor and multipole expansion, we choose interpolation to keep the presentation consistent with Section 4.7: let $t \in \mathcal{T}_\mathcal{I}$. We interpolate the kernel function in the first argument, i.e., we approximation g by the degenerate function

$$\tilde{g}_t(x,y) := \sum_{\nu\in K_t} g(\xi_{t,\nu}, y)\mathcal{L}_{t,\nu}(x) \quad \text{for all } x \in \mathcal{Q}_t,\ y \in \Omega \setminus \mathcal{Q}_t,$$

where $(\xi_{t,\nu})_{\nu\in K_t}$ and $(\mathcal{L}_{t,\nu})_{\nu\in K_t}$ are the interpolation points and Lagrange polynomials corresponding to the bounding box $\mathcal{Q}_t$ and the order vector m_t.

As in Section 4.4, we define the low-rank approximation of $A_t B_t^*$ of $G_{\omega,t}$ by replacing g with $\tilde{g}_t$:

$$(A_t)_{i\nu} := \begin{cases} \int_\Omega \varphi_i(x)\mathcal{L}_{t,\nu}(x)\,dx & \text{if } i \in \hat{t}, \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in \mathcal{I}, \nu \in K_t,$$

$$(B_{t,s})_{j\nu} := \begin{cases} \int_\Omega \psi_j(y) g(\xi_{t,\nu}, y)\,dy & \text{if } j \in \hat{s}, \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } j \in \mathcal{J}, \nu \in K_t,$$

$$B_t := \sum_{s\in\operatorname{row}^*(t)} \omega_{t,s}^{-1} B_{t,s}.$$

We can see that $A_t B_t^*$ corresponds to a blockwise approximation of $G_{\omega,t}$:

$$\begin{aligned} G_{\omega,t} - A_t B_t^* &= \sum_{s\in\operatorname{row}^*(t)} \omega_{t,s}^{-1} \chi_t G \chi_s - \sum_{s\in\operatorname{row}^*(t)} \omega_{t,s}^{-1} A_t B_{t,s}^* \\ &= \sum_{s\in\operatorname{row}^*(t)} \omega_{t,s}^{-1} (\chi_t G \chi_s - A_t B_{t,s}^*). \end{aligned}$$

Due to the different weights involved in this sum, we cannot bound its norm directly. Instead, we break it down into individual blocks:

Lemma 6.37. *Let $p \in (0, C_{\mathrm{bc}}^{-1/2})$ and $\xi \in (0,1)$. Let $(\omega_{r,s})_{r\in\mathcal{T}_\mathcal{I}, s\in\operatorname{row}^*(r)}$ be as in Definition 6.34. If*

$$\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 \leq \omega_{t,s} \sqrt{\frac{1-\xi}{C_{\mathrm{sp}}}} \xi^{(\operatorname{level}(t) - \operatorname{level}(t^+))/2} \tag{6.58}$$

holds for all $t^+ \in \operatorname{pred}(t)$, $s \in \operatorname{row}^+(t^+)$, *we have*

$$\|G_{\omega,t} - A_t B_t^*\|_2 \leq 1.$$

Proof. We apply Lemma 4.45 to the equation

$$G_{\omega,t} - A_t B_t^* = \sum_{s \in \operatorname{row}^*(t)} \omega_{t,s}^{-1} (\chi_t G \chi_s - A_t B_{t,s}^*)$$

and get

$$\begin{aligned}
\|G_{\omega,t} - A_t B_t^*\|_2^2 &\leq \sum_{s \in \operatorname{row}^*(t)} \omega_{t,s}^{-2} \|\chi_t G \chi_s - A_t B_{t,s}^*\|_2^2 \\
&\leq \frac{1-\xi}{C_{\mathrm{sp}}} \sum_{t^+ \in \operatorname{pred}(t)} \sum_{s \in \operatorname{row}^+(t^+)} \omega_{t,s}^{-2} \omega_{t,s}^2 \xi^{\operatorname{level}(t) - \operatorname{level}(t^+)} \\
&= \frac{1-\xi}{C_{\mathrm{sp}}} \sum_{t^+ \in \operatorname{pred}(t)} \sum_{s \in \operatorname{row}^+(t^+)} \xi^{\operatorname{level}(t) - \operatorname{level}(t^+)} \\
&\leq (1-\xi) \sum_{t^+ \in \operatorname{pred}(t)} \xi^{\operatorname{level}(t) - \operatorname{level}(t^+)} = (1-\xi) \sum_{\ell=0}^{\operatorname{level}(t)} \xi^\ell \\
&< (1-\xi) \sum_{\ell=0}^{\infty} \xi^\ell = (1-\xi) \frac{1}{1-\xi} = 1.
\end{aligned}$$

□

As in Chapter 4, the approximation properties of $A_t B_{t,s}^*$ are determined by the interpolation error. We apply the results of Section 4.4 to the interpolant g_t:

Lemma 6.38. *Let* g *be* $(C_{\mathrm{as}}, \sigma, c_0)$*-asymptotically smooth with* $\sigma > 0$. *Let the family of interpolation operators* $(\mathfrak{I}_m)_{m=0}^\infty$ *be* (Λ, λ)*-stable. Let* $\theta \in (0,1)$ *be such that*

$$\operatorname{diam}(\mathcal{Q}_{t'}) \leq \theta \operatorname{diam}(\mathcal{Q}_t) \quad \text{holds for all } t \in \mathcal{T}_{\mathcal{I}}, t' \in \operatorname{sons}(t). \tag{6.59}$$

As in Remark 4.23, we denote the convergence rate of the interpolation scheme by

$$q := \min\left\{\frac{c_0\eta}{c_0\eta+1}, \frac{c_0\eta}{2}\right\} \in (0,1).$$

There are constants $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ *and* $\hat{\theta} \in (0,1)$ *which only depend on* d, g, η, θ, Λ *and* λ *such that*

$$\|g - \tilde{g}_t\|_{\infty, \mathcal{Q}_t \times \mathcal{Q}_s} \leq \frac{C_{\mathrm{in}}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \hat{\theta}^{\alpha(\operatorname{level}(t) - \operatorname{level}(t^+))} q^{\alpha+\beta(p_{\mathcal{I}} - \operatorname{level}(t))} \tag{6.60}$$

holds for all $t \in \mathcal{T}_{\mathcal{I}}$, $t^+ \in \operatorname{pred}(t)$ *and* $s \in \operatorname{row}^+(t^+)$ *satisfying the admissibility condition*

$$\operatorname{diam}(\mathcal{Q}_{t^+}) \leq 2\eta \operatorname{dist}(\mathcal{Q}_{t^+}, \mathcal{Q}_s)$$

(*which is implied by the standard admissibility condition* (4.49)).

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. We start by observing that

$$C_f := \frac{C_{\mathrm{as}}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma}, \quad \gamma_f := \frac{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)}{c_0}$$

fulfill the requirements of Theorem 4.20, which gives us

$$\|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq 2edC_f\Lambda_m^d \left(1 + \frac{\operatorname{diam}(\mathcal{Q}_t)}{\gamma_f}\right)(n_t+1)^\sigma \varrho\left(\frac{2\gamma_f}{\operatorname{diam}(\mathcal{Q}_t)}\right)^{-n_t}.$$

We apply (6.59) to bound the diameter of $\mathcal{Q}_t$ by that of $\mathcal{Q}_{t^+}$:

$$\varrho\left(\frac{2\gamma_f}{\operatorname{diam}(\mathcal{Q}_t)}\right)^{-n_t} \leq \varrho\left(\frac{2\gamma_f}{\theta^{\operatorname{level}(t)-\operatorname{level}(t^+)}\operatorname{diam}(\mathcal{Q}_{t^+})}\right)^{-n_t}.$$

The admissibility condition yields

$$\begin{aligned}
\frac{2\gamma_f}{\operatorname{diam}(\mathcal{Q}_{t^+})} &= \frac{2\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)}{c_0\operatorname{diam}(\mathcal{Q}_{t^+})} \geq \frac{2\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)}{c_0\operatorname{diam}(\mathcal{Q}_{t^+})} \geq \frac{2\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)}{2\eta c_0\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)} = \frac{1}{c_0\eta},\\
\frac{\operatorname{diam}(\mathcal{Q}_t)}{\gamma_f} &= \frac{c_0\operatorname{diam}(\mathcal{Q}_t)}{\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)} \leq \frac{c_0\operatorname{diam}(\mathcal{Q}_{t^+})}{\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)} \leq \frac{2\eta c_0\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)}{\operatorname{dist}(\mathcal{Q}_{t^+},\mathcal{Q}_s)} = 2c_0\eta
\end{aligned}$$

and we conclude

$$\|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq 2edC_f\Lambda_m^d(1+2c_0\eta)(n_t+1)^\sigma \varrho\left(\frac{1}{\theta^{\operatorname{level}(t)-\operatorname{level}(t^+)}c_0\eta}\right)^{-n_t}.$$

Now we can use Lemma 4.78 to find a $\hat{\theta} \in (0,1)$ such that $\varrho(1/(\theta c_0\eta)) \geq \varrho(1/(c_0\eta))/\hat{\theta}$ holds, and applying this estimate $\operatorname{level}(t) - \operatorname{level}(t^+)$ times yields

$$\|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s} \leq 2edC_f\Lambda_m^d(1+2c_0\eta)(n_t+1)^\sigma \hat{\theta}^{n_t(\operatorname{level}(t)-\operatorname{level}(t^+))}\varrho\left(\frac{1}{c_0\eta}\right)^{-n_t}.$$

Remark 4.17 implies $\varrho(1/(c_0\eta)) > \max\{1/(c_0\eta)+1, 2/(c_0\eta)\} = 1/q$, so we have $\gamma := q\varrho(1/(c_0\eta)) \in \mathbb{R}_{>1}$, and Lemma 3.50 implies that there is a constant $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ such that

$$2edC_{\mathrm{as}}(1+2c_0\eta)\Lambda^d(x+2)^{d\lambda}(x+1)^\sigma\gamma^{-x} \leq C_{\mathrm{in}} \quad \text{holds for all } x \in \mathbb{N},$$

therefore we find

$$\begin{aligned}
&\|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t\times\mathcal{Q}_s}\\
&\quad\leq 2edC_f\Lambda^d(n_t+2)^{d\lambda}(1+2c_0\eta)(n_t+1)^\sigma\hat{\theta}^{n_t(\operatorname{level}(t)-\operatorname{level}(t^+))}\varrho\left(\frac{1}{c_0\eta}\right)^{-n_t}\\
&\quad= \frac{2edC_{\mathrm{as}}(1+2c_0\eta)\Lambda^d}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^\sigma}(n_t+2)^{d\lambda}(n_t+1)^\sigma\gamma^{-n_t}\hat{\theta}^{n_t(\operatorname{level}(t)-\operatorname{level}(t^+))}q^{n_t}\\
&\quad\leq \frac{C_{\mathrm{in}}}{\operatorname{dist}(\mathcal{Q}_t,\mathcal{Q}_s)^\sigma}\hat{\theta}^{n_t(\operatorname{level}(t)-\operatorname{level}(t^+))}q^{n_t}.
\end{aligned}$$

Observing $n_t = \alpha + \beta(p_{\mathcal{I}} - \operatorname{level}(t)) \geq \alpha$ completes the proof. □

It is important to note that estimates similar to (6.60) also hold for Taylor or multipole expansion schemes. Based on this estimate, we now derive the required bound for the blockwise spectral error:

Lemma 6.39. *Let $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ and $\hat{\theta} \in (0,1)$ be such that (6.60) holds for all α, β, t, t^+ and s. Let $\hat{\epsilon} \in \mathbb{R}_{>0}$, $p \in (0, C_{\mathrm{bc}}^{-1/2})$ and $\xi \in (0,1)$. We require the following assumptions:*

- *The measure of the support of a cluster can be bounded by a power of its diameter, i.e., there are constants $C_{\mathrm{cu}} \in \mathbb{R}_{\geq 1}$ and $d_\Omega \in \mathbb{N}$ satisfying*

$$|\Omega_t| \leq C_{\mathrm{cu}} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{6.61a}$$

  This assumption defines the "effective dimension" d_Ω of the subdomain or submanifold Ω (cf. (4.58)).

- *The order σ of the singularity of g is not too high, i.e., we have $\sigma \leq d_\Omega$.*

- *The growth of bounding boxes can be controlled, and the diameters of bounding boxes of leaf clusters can be bounded by a multiple of the grid parameter h, i.e., there are constants $C_{\mathrm{gr}} \in \mathbb{R}_{>0}$ and $\zeta \in (0,1)$ satisfying*

$$\zeta^{p_{\mathcal{I}} - \operatorname{level}(t)} \operatorname{diam}(\mathcal{Q}_t) \leq C_{\mathrm{gr}} h \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{6.61b}$$

  This assumption is related to the quasi-uniformity of the underlying grid and the regularity of the clustering scheme (cf. (4.59)).

- *The levels of row and column clusters of admissible blocks are not too far apart, i.e., there is a constant $C_{\mathrm{cn}} \in \mathbb{N}_0$ satisfying*

$$|(p_{\mathcal{I}} - \operatorname{level}(t)) - (p_{\mathcal{J}} - \operatorname{level}(s))| \leq C_{\mathrm{cn}} \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}. \tag{6.61c}$$

  This assumption is related to the compatibility of the cluster tree $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ and to the "level-consistency" (cf. [18]) of the resulting block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

We assume that the rank parameters are large enough, i.e., that

$$\alpha \geq \frac{\log(p\xi\zeta^{1/2})}{\log\hat{\theta}}, \quad \beta \geq \frac{\log(\xi\zeta)}{\log q},$$
$$\alpha \geq \frac{-\log(C_{\mathrm{in}} C_{\mathrm{ov}} C_{\mathrm{cu}} C_{\mathrm{gr}} C_{\mathrm{sp}}^{1/2} (2\eta)^\sigma (1-\xi)^{-1/2} (\xi\zeta)^{-C_{\mathrm{cn}}/2} C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma}/\hat{\epsilon})}{\log q}$$

hold. This ensures

$$\|\chi_t G \chi_s - A_t B_{t,s}\|_2 \leq \hat{\epsilon} \sqrt{\frac{1-\xi}{C_{\mathrm{sp}}}} (p\xi^{1/2})^{\operatorname{level}(t) - \operatorname{level}(t^+)} \xi^{(p_{\mathcal{I}} - \operatorname{level}(t))/2} \xi^{(p_{\mathcal{J}} - \operatorname{level}(s))/2} \tag{6.62}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$, $t^+ \in \operatorname{pred}(t)$ and $s \in \operatorname{row}^+(t^+)$.

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$, $t^+ \in \operatorname{pred}(t)$ and $s \in \operatorname{row}^+(t^+)$. We denote the levels of t, t^+ and s by $l_t := \operatorname{level}(t)$, $l_{t^+} := \operatorname{level}(t^+)$ and $l_s := \operatorname{level}(s)$. Lemma 4.44 yields

$$\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 \leq C_{\mathrm{ov}} C_{\mathcal{I}} C_{\mathcal{J}} |\Omega_t|^{1/2} |\Omega_s|^{1/2} \|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t \times \mathcal{Q}_s}.$$

We apply (6.61a) and get

$$\begin{aligned}\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 &\\ &\leq C_{\mathrm{ov}} C_{\mathrm{cu}} C_{\mathcal{I}} C_{\mathcal{J}} \operatorname{diam}(\mathcal{Q}_t)^{d_\Omega/2} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega/2} \|g - \tilde{g}_t\|_{\infty,\mathcal{Q}_t \times \mathcal{Q}_s}.\end{aligned}$$

Now we use the interpolation error estimate (6.60) in order to find

$$\begin{aligned}\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 &\\ &\leq C_{\mathrm{in}} C_{\mathrm{ov}} C_{\mathrm{cu}} C_{\mathcal{I}} C_{\mathcal{J}} \frac{\operatorname{diam}(\mathcal{Q}_t)^{d_\Omega/2} \operatorname{diam}(\mathcal{Q}_s)^{d_\Omega/2}}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma} \hat{\theta}^{\alpha(l_t - l_{t^+})} q^{\alpha+\beta(p_{\mathcal{I}} - l_t)}.\end{aligned}$$

Since $(t^+, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ is admissible, we have

$$\begin{aligned}\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma &\geq \operatorname{dist}(\mathcal{Q}_{t^+}, \mathcal{Q}_s)^\sigma \geq \left(\frac{1}{2\eta}\right)^\sigma \operatorname{diam}(\mathcal{Q}_{t^+})^\sigma \geq \left(\frac{1}{2\eta}\right)^\sigma \operatorname{diam}(\mathcal{Q}_t)^\sigma,\\ \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^\sigma &\geq \operatorname{dist}(\mathcal{Q}_{t^+}, \mathcal{Q}_s)^\sigma \geq \left(\frac{1}{2\eta}\right)^\sigma \operatorname{diam}(\mathcal{Q}_s)^\sigma,\end{aligned}$$

and these inequalities can be used to eliminate the denominator of our estimate and get

$$\begin{aligned}\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 \leq C_{\mathrm{in}} C_{\mathrm{ov}} C_{\mathrm{cu}} C_{\mathcal{I}} C_{\mathcal{J}} (2\eta)^\sigma &\\ \operatorname{diam}(\mathcal{Q}_t)^{(d_\Omega-\sigma)/2} \operatorname{diam}(\mathcal{Q}_s)^{(d_\Omega-\sigma)/2} \hat{\theta}^{\alpha(l_t - l_{t^+})} q^{\alpha+\beta(p_{\mathcal{I}} - l_t)}.&\end{aligned}$$

Now we use (6.61b) in order to prove

$$\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 \leq C_1 C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma} \zeta^{(l_t - p_{\mathcal{I}})/2} \zeta^{(l_s - p_{\mathcal{J}})/2} \hat{\theta}^{\alpha(l_t - l_{t^+})} q^{\alpha+\beta(p_{\mathcal{I}} - l_t)},$$

where the constants are collected in $C_1 := C_{\mathrm{in}} C_{\mathrm{ov}} C_{\mathrm{cu}} C_{\mathrm{gr}} (2\eta)^\sigma$ for better readability.

Due to our choice of α and β, we have $q^\beta \leq \xi\zeta$ and $\hat{\theta}^\alpha \leq p\xi\zeta^{1/2}$, and the consistency assumption (6.61c) yields

$$\begin{aligned}\hat{\theta}^{\alpha(l_t - l_{t^+})} q^{\beta(p_{\mathcal{I}} - l_t)} &\leq (p\xi\zeta^{1/2})^{l_t - l_{t^+}} (\xi\zeta)^{p_{\mathcal{I}} - l_t}\\ &= (p\xi^{1/2})^{l_t - l_{t^+}} (\xi\zeta)^{(l_t - l_{t^+})/2} (\xi\zeta)^{(p_{\mathcal{I}} - l_t)/2} (\xi\zeta)^{(p_{\mathcal{I}} - l_t)/2}\\ &= (p\xi^{1/2})^{l_t - l_{t^+}} (\xi\zeta)^{(p_{\mathcal{I}} - l_t)/2} (\xi\zeta)^{(p_{\mathcal{I}} - l_{t^+})/2}\\ &\leq (p\xi^{1/2})^{l_t - l_{t^+}} (\xi\zeta)^{(p_{\mathcal{I}} - l_t)/2} (\xi\zeta)^{(p_{\mathcal{J}} - l_s)/2} (\xi\zeta)^{-C_{\mathrm{cn}}/2},\end{aligned}$$

therefore our error estimate now takes the form

$$\begin{aligned}\|\chi_t G \chi_s - A_t B_{t,s}^*\|_2 &\\ &\leq C_1 C_{\mathcal{I}} C_{\mathcal{J}} (\xi\zeta)^{-C_{\mathrm{cn}}/2} h^{d_\Omega - \sigma} q^\alpha (p\xi^{1/2})^{l_t - l_{t^+}} \xi^{(p_{\mathcal{I}} - l_t)/2} \xi^{(p_{\mathcal{J}} - l_s)/2}.\end{aligned}$$

Observing that our choice of α implies

$$q^\alpha \leq \frac{(\xi\zeta)^{C_{\mathrm{cn}}/2}}{C_1 C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma}} \sqrt{\frac{1-\xi}{C_{\mathrm{sp}}}} \hat{\epsilon}$$

concludes the proof. □

It is important to note that the rank parameter β depends only on the constants ξ, ζ and q, but not on $\hat{\epsilon}$, while α depends on the quotient

$$\frac{\hat{\epsilon}}{C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma}},$$

i.e., if $\hat{\epsilon}$ decays not faster than $C_{\mathcal{I}} C_{\mathcal{J}} h^{d_\Omega - \sigma}$, both α and β can be chosen as constants independent of h, i.e., the corresponding rank distributions K and L will be bounded independently of the mesh parameter, while the discretization error converges like $h^{d_\Omega - \sigma}$.

This means that the approximation resulting from Algorithm 34 will have the same advantages as the variable-order interpolation scheme introduced in Section 4.7, while requiring weaker assumptions and no complicated analysis of re-interpolation operators: the *existence* of low-rank approximations $A_t \; B_t^*$ for each cluster is sufficient, they do not have to be nested. Other panel clustering schemes, e.g., Taylor expansion or multipole expansion, can also be used to construct these low-rank approximations, and the quasi-optimality result of Lemma 6.36 implies that the adaptively constructed $\mathcal{H}^2$-matrix approximation will be at least as good as any of these.

6.9 Numerical experiments

The theoretical estimates for the complexity of the compression algorithms require the rank distribution of the *resulting* cluster basis to be bounded. In the case of the recompression of an existing $\mathcal{H}^2$-matrix, this requirement is easily fulfilled, in the general case we have to rely on the quasi-optimality estimate of Lemma 6.23: if the total cluster bases can be approximated by bounded-rank matrices, then the rank distribution of the resulting cluster basis will also be bounded.

In a first series of experiments, we consider the compression of dense matrices in standard array representation. The dense matrices $S, D \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ are given by

$$S_{ij} := -\frac{1}{2\pi} \int_\gamma \varphi_i(x) \int_\Omega \varphi_j(y) \log \|x-y\|_2 \, dy \, dx \qquad \text{for all } i, j \in \mathcal{I},$$

$$D_{ij} := \frac{1}{2\pi} \int_\gamma \varphi_i(x) \int_\Omega \varphi_j(y) \frac{\langle x-y, n(y)\rangle}{\|x-y\|_2^2} \, dy \, dx \qquad \text{for all } i, j \in \mathcal{I},$$

where γ is a polygonal curve in two-dimensional space and $(\varphi_i)_{i\in\mathcal{I}}$ is a family of piecewise constant basis functions in $L^2(\Omega)$. The matrices S and D correspond to the two-dimensional single and double layer potential operators. The entries of the matrices S and D are approximated by hierarchical quadrature [21].

In the first experiment, we apply the weighted compression Algorithm 34 to the matrix S, where γ is a regular polygonal approximation of the unit circle $\gamma_C := \{x \in \mathbb{R}^2 : \|x\|_2 = 1\}$. We use the weighting parameters $p = 2/3 < \sqrt{1/2}$ and $\xi = 5/6 < 1$ and observe the results given in Table 6.1: the times for the construction of the original matrix ("Matrix"), of the adaptive cluster basis ("Basis") and of the corresponding

Table 6.1. Variable-order recompression of the two-dimensional single layer potential operator on the circle with $\hat{\epsilon} = h^2$, based on a dense matrix.

n	Matrix	Basis	Proj	Mem	Mem/n	$\|X - \tilde{X}\|_2$
256	0.1	< 0.1	< 0.1	109.2	0.43	1.0_{-4}
512	0.6	0.1	< 0.1	223.0	0.44	4.1_{-5}
1024	2.2	0.4	0.2	446.4	0.44	1.3_{-5}
2048	8.9	1.8	0.7	896.6	0.44	3.7_{-6}
4096	36.0	7.8	3.1	1794.3	0.44	9.3_{-7}
8192	144.2	38.3	12.9	3590.4	0.44	2.5_{-7}
16384	580.6	164.1	55.6	7176.0	0.44	6.3_{-8}
32768	2318.5	688.3	236.3	14369.1	0.44	1.7_{-8}

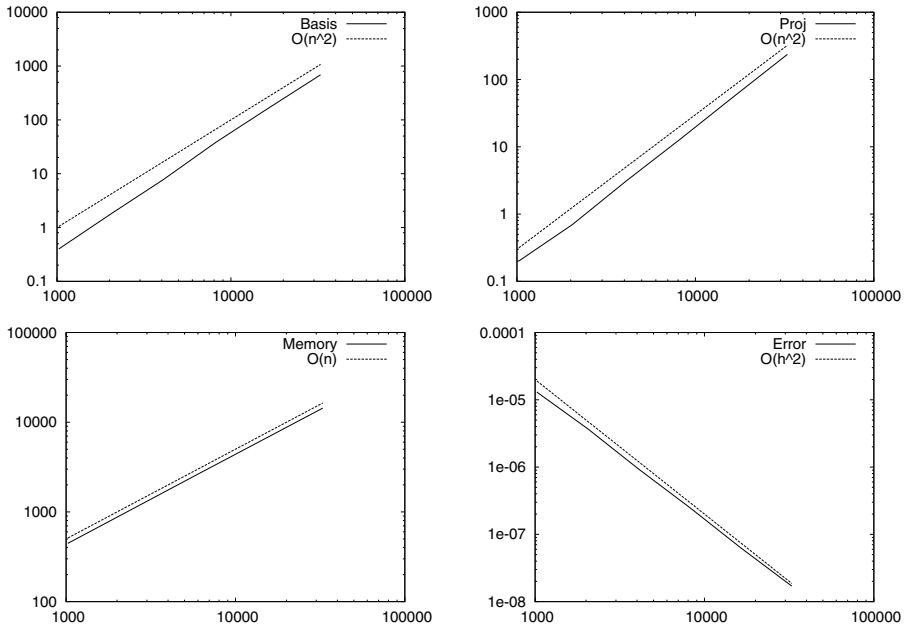
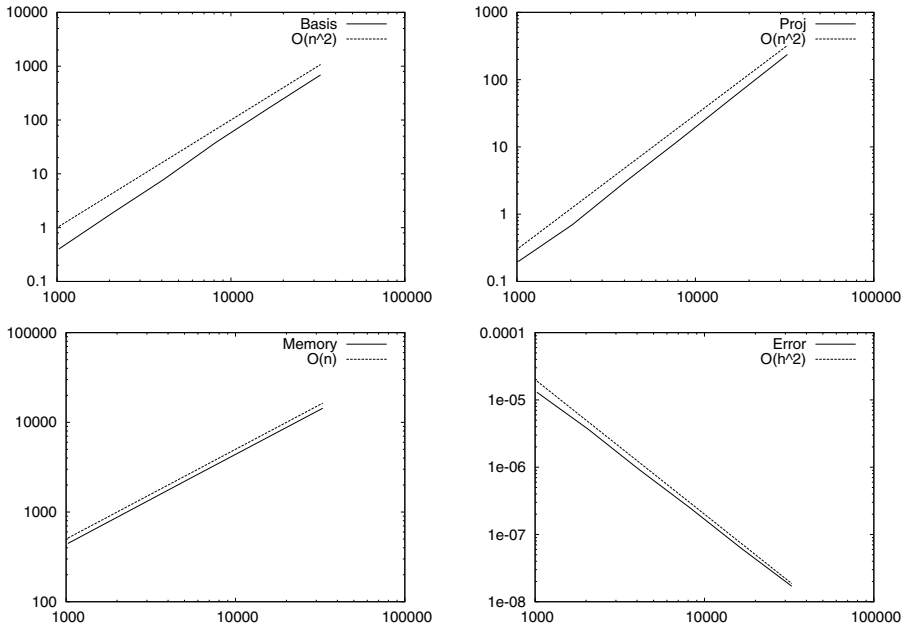


Table 6.2. Variable-order recompression of the two-dimensional single layer potential operator on the square with $\hat{\epsilon} = h^2$, based on a dense matrix.

n	Matrix	Basis	Proj	Mem	Mem/n	$\|X - \tilde{X}\|_2$
256	0.1	< 0.1	< 0.1	146.1	0.57	2.1_{-4}
512	0.6	0.1	< 0.1	283.7	0.55	2.4_{-5}
1024	2.2	0.4	0.1	553.6	0.54	8.2_{-6}
2048	8.9	1.9	0.6	1094.7	0.53	3.2_{-6}
4096	36.0	7.6	2.4	2170.8	0.53	8.6_{-7}
8192	144.2	37.5	10.1	4318.5	0.53	2.0_{-7}
16384	576.7	154.1	43.4	8569.3	0.52	5.0_{-8}
32768	2320.5	654.4	187.0	17040.3	0.52	1.3_{-8}



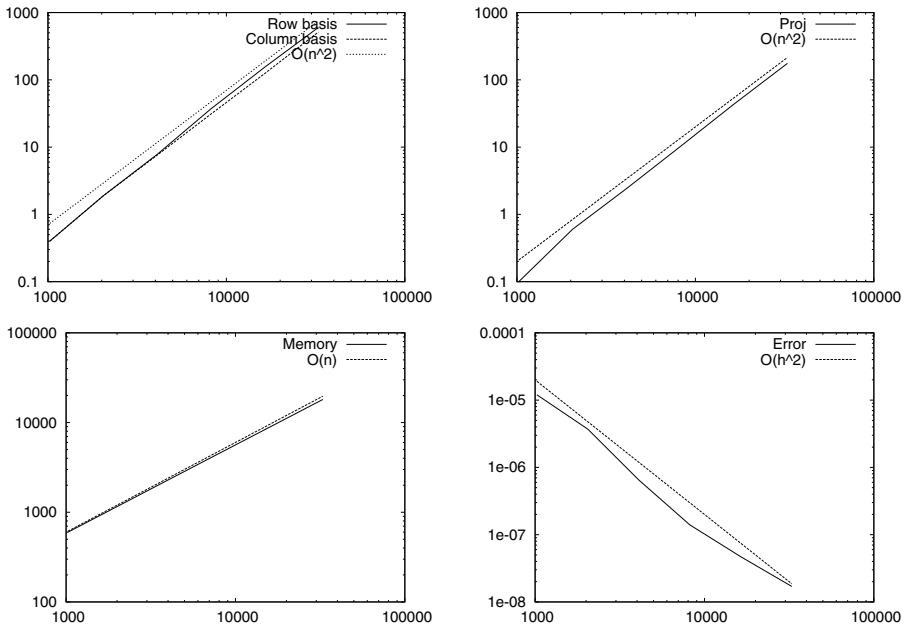
$\mathcal{H}^2$-matrix representation ("Proj") scale like n^2, which can be considered optimal for a method based on dense matrices and is in accordance with the Lemmas 6.20 and 5.8. The storage requirements scale like n and the approximation error scales like n^{-2}, therefore we can conclude that the weighted compression algorithm indeed finds a variable-order approximation of the dense matrix.

In the next experiment, we apply the algorithm to a regular triangulation of the unit square[1] $\gamma_S := \{-1, 1\} \times [-1, 1] \cup [-1, 1] \times \{-1, 1\}$. The results given in Table 6.2 are

[1] In practical applications, a graded triangulation would be more appropriate, but the discussion of the

Table 6.3. Variable-order recompression of the two-dimensional double layer potential operator on the square with $\hat{\epsilon} = h^2$, based on a dense matrix.

n	Matrix	Row	Column	Proj	Mem	Mem/n	$\|X - \tilde{X}\|_2$
256	0.1	< 0.1	< 0.1	< 0.1	153.8	0.60	2.4_{-4}
512	0.2	0.1	0.1	< 0.1	302.9	0.59	4.3_{-5}
1024	0.9	0.4	0.4	0.1	598.5	0.58	1.2_{-5}
2048	3.7	1.9	1.9	0.6	1185.5	0.58	3.7_{-6}
4096	15.1	7.8	7.6	2.4	2346.5	0.57	6.5_{-7}
8192	60.4	37.1	31.0	10.1	4620.8	0.56	1.4_{-7}
16384	243.3	154.0	123.9	43.2	9132.9	0.56	4.7_{-8}
32768	971.6	601.8	514.0	175.6	18058.7	0.55	1.7_{-8}



very similar to those observed for the unit sphere: again the time for the construction scales like n^2, while the storage requirements of the resulting $\mathcal{H}^2$-matrix scale like n and the error scales like n^{-2}.

The previous two cases are covered by the theory presented in Sections 4.7 and 6.8. The situation is more complicated for the double layer potential operator D on the unit square: a variable-order approximation exists [81], but its construction is

technical details connected to this approach is beyond the scope of this book, cf. [66].

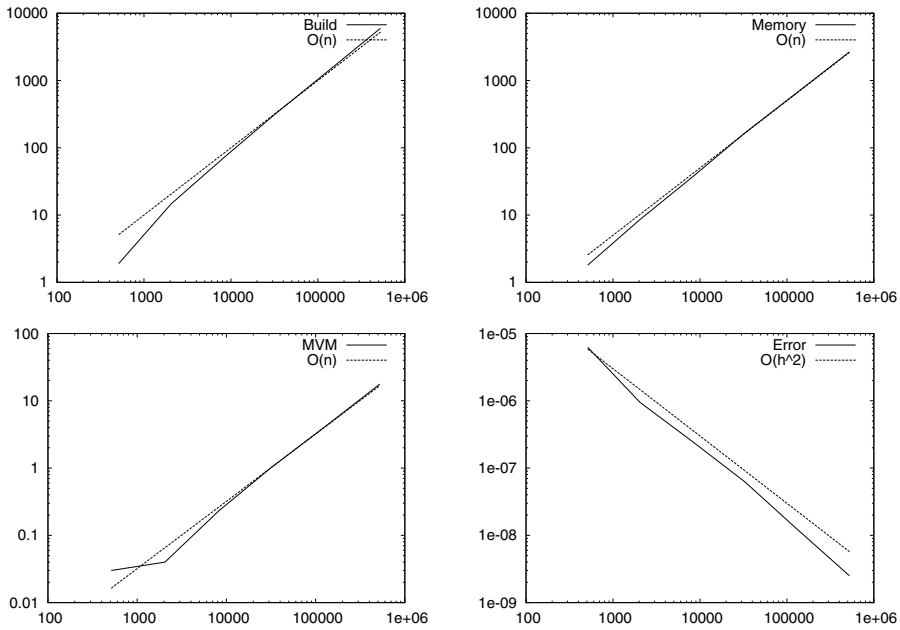
complicated. Fortunately, the existence of an approximation is already sufficient for Algorithm 34, and the numerical results in Table 6.3 show that it finds an efficient $\mathcal{H}^2$-matrix approximation.

Let us now consider the three-dimensional case. Since the matrix dimensions tend to grow rapidly due to the higher spatial dimension, we can no longer base our experiments on dense matrices, but construct an initial $\mathcal{H}^2$-matrix and apply recompression to improve its efficiency. We construct this initial approximation by the interpolation approaches discussed in the Sections 4.4 and 4.5.

We start with a simple example: the matrix V corresponding to the single layer operator (cf. Section 4.9) is approximated by interpolation with constant order $m = 4$

Table 6.4. Recompression of the single layer potential operator with $\hat{\epsilon} = 4h^2 \times 10^{-5}$, based on an initial approximation by constant-order interpolation with $m = 4$.

n	Build	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	1.9	1.8	3.6	< 0.01	6.2_{-6}
2048	14.6	8.6	4.3	0.04	9.5_{-7}
8192	70.4	36.8	4.6	0.23	2.5_{-7}
32768	322.6	165.5	5.2	1.05	6.3_{-8}
131072	1381.5	664.5	5.2	4.25	1.2_{-8}
524288	5975.2	2662.6	5.2	17.72	2.5_{-9}



on a cluster tree with $C_{\mathrm{lf}} = 16$, then we use Algorithm 30 with the variable-order error control scheme (cf. Section 6.8) to construct a more efficient approximation. According to Section 4.6, it is reasonable to use an error tolerance $\hat{\epsilon} \approx h^2$, where h is again the minimal meshwidth of the underlying triangulation. The results are given in Table 6.4. We can see that the computation time, the storage requirements and the time for the matrix-vector multiplication are in $\mathcal{O}(n)$, as predicted by our theory, and that the error decreases at a rate of approximately h^2. Compared to the uncompressed case (cf. Table 4.1), we can see that the storage requirements are reduced by a factor of more than six, while the approximation error and the computation time are only slightly increased.

Table 6.5. Recompression of the double layer potential operator with $\hat{\epsilon} = 4h^2 \times 10^{-4}$, based on an initial approximation by constant-order interpolation with $m = 4$.

n	Build	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	1.9	1.8	3.5	< 0.01	9.1_{-5}
2048	14.2	7.6	3.8	0.03	9.2_{-6}
8192	68.6	31.0	3.9	0.18	2.2_{-6}
32768	312.8	143.3	4.5	0.84	6.5_{-7}
131072	1360.5	585.2	4.6	3.45	1.7_{-7}
524288	5937.8	2388.1	4.7	15.00	4.7_{-8}

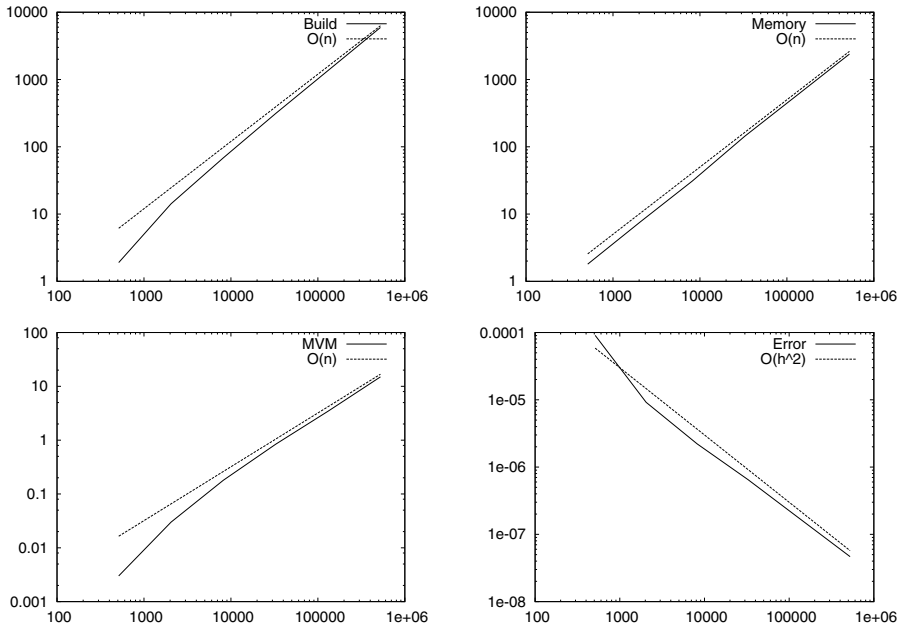
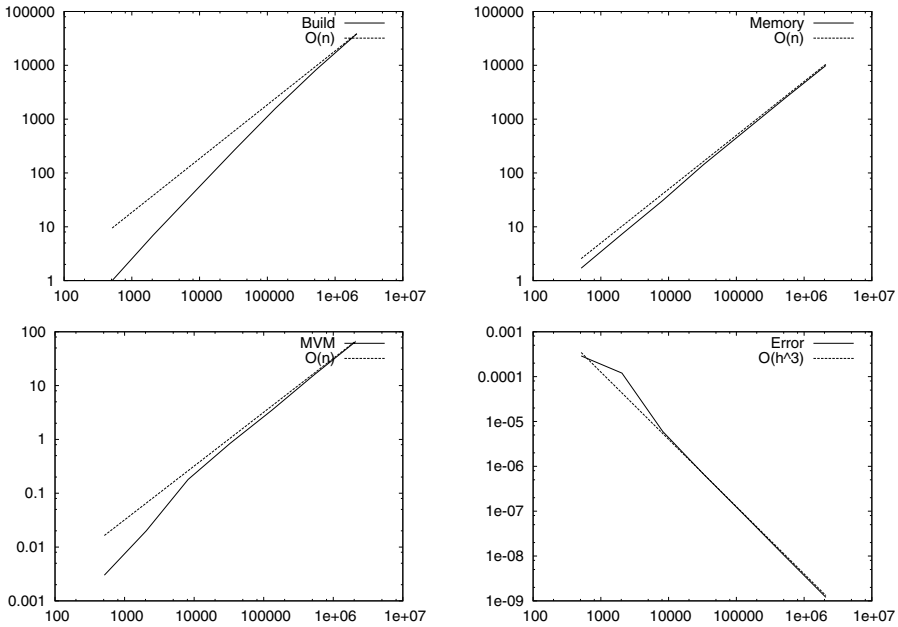


Table 6.6. Variable-order recompression of the single layer potential operator with $\hat{\epsilon} = 2h^3 \times 10^{-2}$, based on an initial approximation by variable-order interpolation with $\alpha = 2$ and $\beta = 1$.

n	Build	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	1.0	1.7	3.5	< 0.01	2.9_{-4}
2048	6.9	7.3	3.6	0.02	1.2_{-4}
8192	43.1	30.7	3.8	0.18	6.0_{-6}
32768	267.9	142.9	4.5	0.84	6.7_{-7}
131072	1574.9	590.5	4.6	3.49	8.2_{-8}
524288	8271.4	2449.4	4.8	15.60	9.8_{-9}
2097152	38640.7	9921.5	4.8	65.74	1.2_{-9}

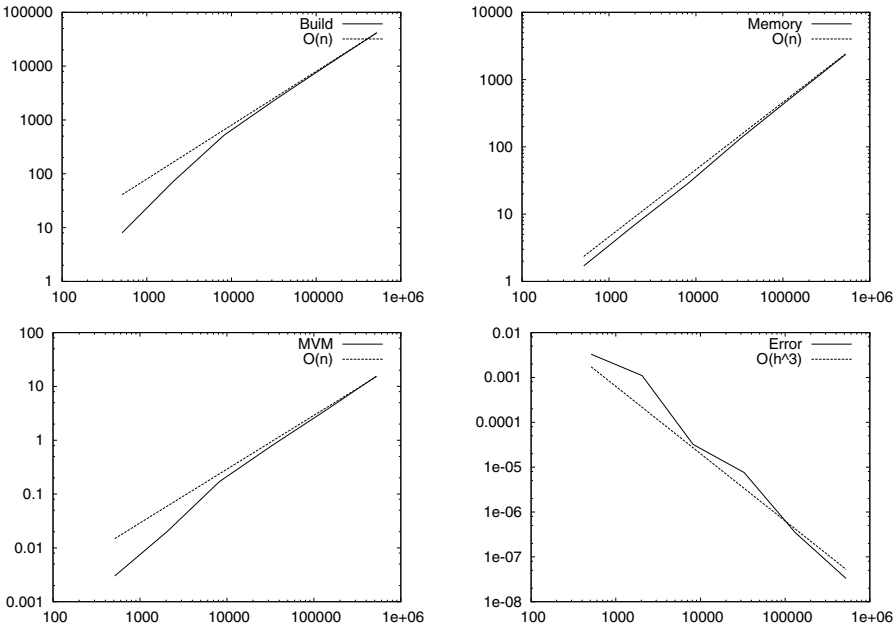


In the next experiment, we construct an approximation of the matrix K corresponding to the double layer operator (cf. Section 4.9) by using the derivatives of local interpolants (cf. Section 4.5) and again use Algorithm 30 with variable-order error control and a tolerance of $\hat{\epsilon} \approx h^2$. The results given in Table 6.5 are comparable to the previous experiment: the computation times are almost identical, and the storage requirements for K are slightly lower than for V.

We have seen in Chapter 4 that variable-order interpolation schemes yield a very good approximation quality compared to their time and storage requirements, and of course we would like to ensure that these properties are preserved by the recompression algorithm. Combining an initial variable-order approximation with the variable-order

Table 6.7. Variable-order recompression of the double layer potential operator with $\hat{\epsilon} = 2h^3 \times 10^{-1}$, based on an initial approximation by constant-order interpolation with $m = 6$.

n	Build	Mem	Mem/n	MVM	$\|X - \tilde{X}\|_2$
512	8.0	1.7	3.4	< 0.01	3.3_{-3}
2048	71.4	7.2	3.6	0.02	1.1_{-3}
8192	521.4	28.9	3.6	0.17	3.2_{-5}
32768	2373.9	134.5	4.2	0.79	7.6_{-6}
131072	10102.2	567.8	4.4	3.43	3.5_{-7}
524288	41980.5	2353.6	4.6	15.52	3.3_{-8}



recompression algorithm yields the results reported in Table 6.6: compared to the uncompressed case (cf. Table 4.6), the storage requirements are reduced by a factor of four, the approximation error is even smaller than before, only the computation time is increased. This latter effect can be explained by a closer look at the proof of Theorem 6.27: we have to bound the product of a *ninth*-order polynomial and an exponential term, and the corresponding constant is so large that the asymptotic behaviour is not yet visible.

As a final example, we consider the variable-order recompression of the matrix K corresponding to the double layer potential operator. It is possible to base the recompression on variable-order interpolation schemes [23] or on Taylor expansions [90], [100], but the implementation would be relatively complicated and not very

efficient. Instead, we simply use an initial approximation constructed by constant-order interpolation with $m = 6$ and compress it using the strategy described in Section 6.8. The experimental results given in Table 6.7 show that the compression strategy is successful: the error decays like h^3, the storage requirements increase only like n.

Chapter 7
A priori matrix arithmetic

The structure of $\mathcal{H}$- and $\mathcal{H}^2$-matrices is purely algebraic, therefore it is straightforward to wonder whether it is possible to perform matrix arithmetic operations like addition, multiplication, inversion or factorizations efficiently in these data-sparse formats. Factorizations could be used to construct efficient solvers for systems of linear equations, the matrix inverse would be a useful tool for the approximation of matrix functions or the solution of matrix equations like Lyapunov's or Riccati's.

Both the inversion and the basic Cholesky and LU-factorizations can be expressed by means of sums and products of matrices, therefore we focus on algorithms for performing these fundamental operations efficiently.

We consider three ways of handling sums and products of two $\mathcal{H}^2$-matrices A and B with different structure:

- We can prescribe an $\mathcal{H}^2$-matrix space and compute the best approximations of $A + B$ and $AB + C$ in this space. Using orthogonal cluster bases, we can use the results of Chapter 5 in order to handle these computations very efficiently.

- The exact sum $A + B$ and the exact product AB are elements of $\mathcal{H}^2$-matrix spaces with suitably chosen block cluster trees and cluster bases.

- We can compute an auxiliary approximation of a sum or a product in the form of a hierarchical matrix, which can then be approximated by an $\mathcal{H}^2$-matrix with adaptively chosen cluster bases by applying Algorithm 33.

The three methods have different advantages and disadvantages: the first approach computes the best approximation of the result in a *prescribed* $\mathcal{H}^2$-matrix space, i.e., in order to reach a given precision, the space has to be chosen correctly. Constructing a suitable matrix space can be quite complicated in practical applications, but once it is available, the algorithm reaches the *optimal* order of complexity.

The second approach yields *exact* sums and products, but at the cost of a significant increase both in the number of nodes in the block cluster tree and in the rank. Therefore it is, at least at the moment, only interesting for theoretical investigations, but computationally too complex to be used in practical applications.

The third approach combines both methods: it computes an intermediate approximation by a technique similar to the second approach, relying on simple blockwise low-rank approximations instead of an exact representation in an $\mathcal{H}^2$-matrix format. Since this approach is only loosely related to the other two, it is discussed in the separate Chapter 8.

The current chapter is organized as follows:

- Section 7.1 introduces the matrix forward transformation algorithm, a variant of the forward transformation Algorithm 6 that applies to matrices and is of crucial importance for the algorithms introduced in the following Sections 7.4 and 7.7.
- Section 7.2 is devoted to its counterpart, the matrix backward transformation algorithm that is also required in Sections 7.4 and 7.7.
- Section 7.3 contains the definitions and notations required to investigate the matrix addition.
- Section 7.4 considers an algorithm for computing the best approximation of the sum of two matrices in a given $\mathcal{H}^2$-matrix space (cf. [11], Section 3.3).
- Section 7.5 considers the computation of the exact sum of two matrices in an extended $\mathcal{H}^2$-matrix space that will usually be too large for practical applications.
- Section 7.6 contains the definitions and notations required to investigate the matrix multiplication.
- Section 7.7 is devoted to an algorithm for computing the best approximation of the product of two matrices in a given $\mathcal{H}^2$-matrix space (cf. [11], Section 4). This algorithm and the proof of its optimal complexity are the main results of this chapter.
- Section 7.8 considers the computation of the exact product of two matrices in an extended $\mathcal{H}^2$-matrix space that will usually be *far* too large for practical applications.
- Section 7.9 presents numerical experiments that show that the complexity estimates for the algorithm introduced in Section 7.7 are of optimal order.

Assumptions in this chapter: We assume that cluster trees T_I, T_J and T_K for the finite index sets I, J and K, respectively, are given. Let $n_I := \#I$, $n_J := \#J$ and $n_K := \#K$ denote the number of indices in each of the sets, and let $c_I := \#T_I$, $c_J := \#T_J$ and $c_K := \#T_K$ denote the number of clusters in each of the cluster trees.

7.1 Matrix forward transformation

In this chapter, we frequently have to compute representations of blocks of an $\mathcal{H}^2$-matrix in varying cluster bases. Differently from the simple techniques introduced in Section 5.3, which only handle transformations between admissible leaves of a block cluster tree, we require transformations between admissible leaves, inadmissible leaves and non-leaf blocks.

We now introduce algorithms for handling these transformations efficiently. Let $A \in \mathcal{H}^2(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, V_A, W_A)$, where $V_A = (V_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $W_A = (W_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$ are nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively, and $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ is an admissible block cluster tree.

Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively, and let $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $L = (L_s)_{s\in\mathcal{T}_\mathcal{J}}$ be the rank distributions for V and W, respectively.

We consider the computation of the matrix family $\widehat{S}_A = (\widehat{S}_{A,b})_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}$ (cf. Figure 7.1) defined by

$$\widehat{S}_{A,b} := V_t^* A W_s \in \mathbb{R}^{K_t\times L_s} \quad \text{for all } b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}. \tag{7.1}$$

If V and W are orthogonal cluster bases, the matrix $V_t \widehat{S}_{A,b} W_s^* = V_t V_t^* A W_s W_s^*$ is the best approximation of the block $\chi_t A \chi_s$ with respect to the Frobenius norm (cf. Lemma 5.5), but the matrices $\widehat{S}_A = (\widehat{S}_{A,b})_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}$ are also useful in the non-orthogonal case (cf. equation (7.17) used for computing matrix-matrix products).

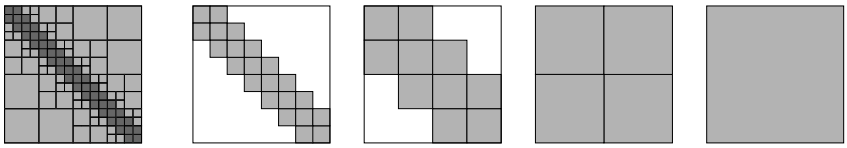


Figure 7.1. Blocks involved in the matrix forward transformation.

If $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ is an inadmissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, both t and s have to be leaves of $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively, and we can compute $\widehat{S}_{A,b}$ directly by using its definition (7.1).

If b is an admissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, we have

$$\chi_t A \chi_s = V_{A,t} S_{A,b} W_{A,s}^*$$

and conclude

$$\widehat{S}_{A,b} = V_t^* A W_s = V_t^* V_{A,t} S_{A,b} W_{A,s}^* W_s = P_{V,t} S_{A,b} P_{W,s}, \tag{7.2}$$

where the cluster operators $P_V = (P_{V,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $P_W = (P_{W,s})_{s\in\mathcal{T}_\mathcal{J}}$ are defined by

$$P_{V,t} := V_t^* V_{A,t}, \quad P_{W,s} := W_{A,s}^* W_s, \quad \text{for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}.$$

We can see that P_V and P_W are cluster basis products of the cluster bases V, V_A, W and W_A, so we can use Algorithm 13 to compute these cluster operators efficiently.

This leaves us with the case that b is not a leaf, i.e., that $\operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b) \neq \emptyset$ holds. According to Definition 3.12, we have

$$\operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b) = \begin{cases} \{t\} \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) = \emptyset, \operatorname{sons}(s) \neq \emptyset, \\ \operatorname{sons}(t) \times \{s\} & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) = \emptyset, \\ \operatorname{sons}(t) \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) \neq \emptyset. \end{cases}$$

Using Definition 3.13, we can see that

$$\operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$$

holds for all $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ with $\operatorname{sons}(b) \neq \emptyset$.

For all $t' \in \operatorname{sons}^+(t)$, the Definition 5.27 of long-range transfer matrices yields

$$E_{t',t} = \begin{cases} E_{t'} & \text{if } \operatorname{sons}(t) \neq \emptyset, \\ I & \text{otherwise}, \end{cases} \tag{7.3}$$

and since V is nested, equation (3.15) takes the form

$$V_t = \sum_{t' \in \operatorname{sons}^+(t)} V_{t'} E_{t',t}.$$

Obviously, the same holds for the cluster basis W, i.e., we have

$$W_s = \sum_{s' \in \operatorname{sons}^+(s)} W_{s'} F_{s',s}.$$

Applying these equations to (7.1) yields

$$\begin{aligned} \widehat{S}_{A,b} = V_t^* A W_s &= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{s' \in \operatorname{sons}^+(s)} E_{t',t}^* V_{t'}^* A W_{s'} F_{s',s} \\ &= \sum_{b'=(t',s') \in \operatorname{sons}(b)} E_{t',t}^* V_{t'}^* A W_{s'} F_{s',s} = \sum_{b'=(t',s') \in \operatorname{sons}(b)} E_{t',t}^* \widehat{S}_{A,b'} F_{s',s}. \end{aligned}$$

Using the definitions of $E_{t',t}$ and $F_{s',s}$ (e.g., $E_{t',t} = I$ for $t' = t$ and $E_{t',t} = E_{t'}$ for $t' \in \operatorname{sons}(t)$), we can return this equation to the more explicit form

$$\widehat{S}_{A,b} = \begin{cases} \sum_{b'=(t,s') \in \operatorname{sons}(b)} \widehat{S}_{A,b'} F_{s'} & \text{if } \operatorname{sons}(t) = \emptyset \text{ and } \operatorname{sons}(s) \neq \emptyset, \\ \sum_{b'=(t',s) \in \operatorname{sons}(b)} E_{t'}^* \widehat{S}_{A,b'} & \text{if } \operatorname{sons}(t) \neq \emptyset \text{ and } \operatorname{sons}(s) = \emptyset, \\ \sum_{b'=(t',s') \in \operatorname{sons}(b)} E_{t'}^* \widehat{S}_{A,b'} F_{s'} & \text{if } \operatorname{sons}(t) \neq \emptyset \text{ and } \operatorname{sons}(s) \neq \emptyset, \end{cases} \tag{7.4}$$

which we can use to determine $\widehat{S}_{A,b}$ by an efficient recursion (cf. Figure 7.2).

The step (7.4) is handled by Algorithm 35. In case that both t and s are not leaf clusters, we employ auxiliary matrices $X_{t'}$ to accumulate the results corresponding to one son $t' \in \operatorname{sons}(t)$ of the row cluster t and all sons $s' \in \operatorname{sons}(s)$ of the column cluster s. This leads to a more efficient implementation since each of the transfer matrices $E_{t'}$ has to be applied only once.

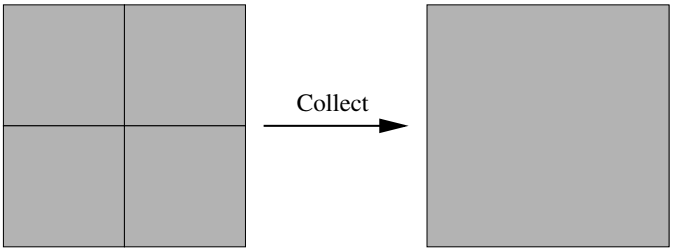


Figure 7.2. Collect matrices $\widehat{S}_{A,b'}$ corresponding to subblocks of b in order to form $\widehat{S}_{A,b}$.

Lemma 7.1 (Complexity of collecting submatrices). *Let V and W be nested cluster bases with rank distributions K and L, and let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18). Let $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Algorithm 35 requires not more than*

$$2(k_t^2 l_s + k_t l_s^2)$$

operations.

Proof. Depending on whether t and s are leaves, we have to distinguish four cases: if t and s are leaves, no operations are performed. If t is a leaf and s is not, the matrix $\widehat{S}_{A,b}$ can be computed in

$$\sum_{s'\in\operatorname{sons}(s)} 2(\#K_t)(\#L_{s'})(\#L_s) \leq 2(\#K_t)l_s(\#L_s) \leq 2k_t l_s^2$$

operations. If s is a leaf and t is not, the matrix can be computed in

$$\sum_{t'\in\operatorname{sons}(t)} 2(\#K_t)(\#K_{t'})(\#L_s) \leq 2(\#K_t)k_t(\#L_s) \leq 2k_t^2 l_s$$

operations. If both t and s are not leaves, the matrix $X_{t'}$ can be computed in

$$\sum_{s'\in\operatorname{sons}(s)} 2(\#K_{t'})(\#L_{s'})(\#L_s) \leq 2(\#K_{t'})l_s(\#L_s) \leq 2(\#K_{t'})l_s^2$$

operations for each $t' \in \operatorname{sons}(t)$, and using these matrices, $\widehat{S}_{A,b}$ can be computed in

$$\sum_{t'\in\operatorname{sons}(t)} 2(\#K_t)(\#K_{t'})(\#L_s) \leq 2(\#K_t)k_t(\#L_s) \leq 2k_t^2 l_s.$$

The total number of operations is bounded by

$$2k_t^2 l_s + \sum_{t'\in\operatorname{sons}(t)} 2(\#K_{t'})l_s^2 \leq 2k_t^2 l_s + 2k_t l_s^2. \qquad \square$$

Algorithm 35. Collect son blocks to compute their father.

```
procedure Collect(b, V, W, var ŜA);
(t, s) ← b;
Ŝ_{A,b} ← 0;
if sons(t) = ∅ then
  if sons(s) = ∅ then
    Do nothing                                  {Leaf blocks are handled differently}
  else
    for s' ∈ sons(s) do
      b' ← (t, s');                             {sons(t) = ∅, sons(s) ≠ ∅}
      Ŝ_{A,b} ← Ŝ_{A,b} + Ŝ_{A,b'} F_{s'}
    end for
  end if
else
  if sons(s) = ∅ then
    for t' ∈ sons(t) do
      b' ← (t', s);                             {sons(t) ≠ ∅, sons(s) = ∅}
      Ŝ_{A,b} ← Ŝ_{A,b} + E*_{t'} Ŝ_{A,b'}
    end for
  else
    for t' ∈ sons(t) do
      X_{t'} ← 0 ∈ R^{K_{t'} × L_s};            {sons(t) ≠ ∅, sons(s) ≠ ∅}
      for s' ∈ sons(s) do
        b' ← (t', s');
        X_{t'} ← X_{t'} + Ŝ_{A,b'} F_{s'}
      end for
      Ŝ_{A,b} ← Ŝ_{A,b} + E*_{t'} X_{t'}
    end for
  end if
end if
```

Combining (7.4), as implemented in Algorithm 35, with (7.1) for inadmissible leaves and (7.2) for admissible leaves yields the recursive Algorithm 36 for computing the entire family $\widehat{S}_A = (\widehat{S}_{A,b})_{b \in \mathcal{T}_{A,\mathcal{I} \times \mathcal{J}}}$ efficiently.

It closely resembles the forward transformation Algorithm 6: instead of computing coefficient *vectors* corresponding to an input *vector* by proceeding upwards through a cluster tree, it computes coefficient *matrices* of an input *matrix* by proceeding upwards through a block cluster tree. Due to this similarity, we call it the *matrix forward transformation*.

Algorithm 36. Matrix forward transformation.

procedure MatrixForward(b, V, W, P_V, P_W, S_A, **var** $\widehat{S}_A$);
$(t,s) \leftarrow b$;
if $\operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b) \neq \emptyset$ **then**
  for $b' \in \operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b)$ **do**
    MatrixForward(b', V, W, P_V, P_W, S_A, $\widehat{S}_A$)
  end for;
  Collect(b, V, W, $\widehat{S}_A$) {Algorithm 35}
else if b is admissible **then**
  $\widehat{S}_{A,b} \leftarrow P_{V,t} S_{A,b} P_{W,s}$ {Admissible leaf}
else
  $\widehat{S}_{A,b} \leftarrow V_t^* A W_s$ {Inadmissible leaf}
end if

Remark 7.2. If the cluster bases $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ are orthogonal, for all blocks $b = (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ the matrix

$$V_t \widehat{S}_{A,b} W_s^* = V_t V_t^* A W_s W_s^*$$

is the best approximation of the subblock $\chi_t A \chi_s$ by the cluster bases V_t and W_s. In this sense, the matrix forward transformation constructs a hierarchy of approximations of the matrix A, and this hierarchy is of crucial importance for matrix arithmetic operations. □

Lemma 7.3 (Complexity). *Let V and W be nested cluster bases with rank distributions K and L, let V_A and W_A be nested cluster bases with rank distributions K_A and L_A. Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree. Let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$, $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$, $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18). Let*

$$\hat{k}_t := \max\{k_t, k_{A,t}\}, \quad \hat{l}_s := \max\{l_s, l_{A,s}\}$$

for all $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$. Algorithm 36, started with $b = \operatorname{root}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}})$, requires not more than

$$2 \sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} (k_t^2 l_s + k_t l_s^2) \leq 2 \sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} (k_t^3 + l_s^3)$$

operations. If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse, this can be bounded by

$$2C_{\mathrm{sp}}\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_s^3\Big).$$

If in addition K, L, K_A and L_A are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I} + n_\mathcal{J}))$.

Proof. Let $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$.

If b is not a leaf, the algorithm handles the subblocks $b' \in \operatorname{sons}(b)$ by recursion and then uses Algorithm 35. According to Lemma 7.1, this requires not more than

$$2(\hat{k}_t^2 \hat{l}_s + \hat{k}_t \hat{l}_s^2)$$

operations.

If b is an admissible leaf, the algorithm can compute the product $P_{V,t} S_{A,b} P_{W,s}$ "from right to left", i.e., by first computing $S_{A,b} P_{W,s}$ in $2(\#K_{A,t})(\#L_{A,s})(\#L_s) \leq 2\hat{k}_t \hat{l}_s^2$ operations, and then computing $\widehat{S}_{A,b}$ in $2(\#K_t)(\#K_{A,t})(\#L_s) \leq 2\hat{k}_t^2 \hat{l}_s$ operations. The entire computation again takes not more than

$$2(\hat{k}_t^2 \hat{l}_s + \hat{k}_t \hat{l}_s^2)$$

operations.

If b is an inadmissible leaf, both t and s have to be leaves, since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible. This implies $\#\hat{t} \leq k_t$ and $\#\hat{s} \leq l_s$. If we compute $V_t^* A W_s$ again "from right to left", the first product AW_s takes not more than $2(\#\hat{t})(\#\hat{s})(\#L_s) \leq 2\hat{k}_t \hat{l}_s^2$ operations, and the second takes not more than $2(\#K_t)(\#\hat{t})(\#L_s) \leq 2\hat{k}_t^2 \hat{l}_s$ operations, and we again get the upper bound

$$2(\hat{k}_t^2 \hat{l}_s + \hat{k}_t \hat{l}_s^2).$$

Adding these bounds for all $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and applying (5.9) yields the bound

$$2 \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \hat{k}_t^2 \hat{l}_s + \hat{k}_t \hat{l}_s^2 \leq 2 \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \hat{k}_t^3 + \hat{l}_s^3$$

for the total number of operations. If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse, we get

$$\begin{aligned} 2 \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} \hat{k}_t^3 + \hat{l}_s^3 &= 2 \sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\operatorname{row}(t)} \hat{k}_t^3 + 2 \sum_{s\in\mathcal{T}_{\mathcal{J}}} \sum_{t\in\operatorname{col}(s)} \hat{l}_s^3 \\ &\leq 2C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + 2C_{\mathrm{sp}} \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 = 2C_{\mathrm{sp}} \Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \Big), \end{aligned}$$

and we can complete the proof using the Lemmas 3.45 and 3.48. □

7.2 Matrix backward transformation

We have seen that the matrix forward transformation can be used to approximate subblocks of an $\mathcal{H}^2$-matrix by an admissible block. Its counterpart is the *matrix backward transformation*, which approximates an admissible block by an $\mathcal{H}^2$-matrix.

Let $V_C = (V_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $W_C = (W_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ be orthogonal nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively, and let $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree.

Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively, and let $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $L = (L_s)_{s\in\mathcal{T}_\mathcal{J}}$ be the corresponding rank distributions.

Let $b = (t,s) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, and let $S_b \in \mathbb{R}^{K_t\times L_s}$. We consider the approximation of the matrix

$$C := V_t S_b W_s^*$$

in the space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$. According to Lemma 5.5, the best approximation is given by

$$\begin{aligned}\widetilde{C} := \Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}},V_C,W_C} C &= \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} V_{C,t^*} V_{C,t^*}^* C W_{C,s^*} W_{C,s^*}^* \\ &\quad + \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^-} \chi_{t^*} C \chi_{s^*} \\ &= \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^+} V_{C,t^*} V_{C,t^*}^* V_t S_b W_s^* W_{C,s^*} W_{C,s^*}^* \\ &\quad + \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b^-} \chi_{t^*} V_t S_b W_s^* \chi_{s^*}.\end{aligned} \tag{7.5}$$

Handling the nearfield blocks is straightforward: for all $b^* = (t^*,s^*) \in \mathcal{L}_b^-$, Definition 3.33 implies

$$\chi_{t^*} V_t S_b W_s^* \chi_{s^*} = \chi_{t^*}\chi_t V_t S_b W_s^* \chi_s \chi_{s^*},$$

and due to Lemma 3.8, this matrix can only differ from zero if $t^* \in \operatorname{sons}^*(t)$ and $s^* \in \operatorname{sons}^*(s)$ hold. In this case, Lemma 6.13 implies

$$\chi_{t^*} V_t = V_{t^*} E_{t^*,t}, \quad \chi_{s^*} W_s = W_{s^*} F_{s^*,s},$$

and we can conclude

$$\chi_{t^*} C \chi_{s^*} = \begin{cases} V_{t^*} E_{t^*,t} S_b F_{s^*,s}^* W_{s^*}^* & \text{if } b^* = (t^*,s^*) \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b), \\ 0 & \text{otherwise.} \end{cases} \tag{7.6}$$

Here $E_{t^*,t}$ and $F_{s^*,s}$ denote the long-range transfer matrices introduced in Definition 5.27.

For the farfield blocks, we have to compute

$$S_{C,b^*} := V_{C,t^*}^* V_t S_b W_s^* W_{C,s^*}$$

for all $b^* = (t^*,s^*) \in \mathcal{L}_b^+$. According to Definition 3.33, we have

$$S_{C,b^*} = V_{C,t^*}^* (\chi_{t^*}\chi_t) V_t S_b W_s^* (\chi_s \chi_{s^*}) W_{C,s^*},$$

and Lemma 3.8 yields that S_{C,b^*} can only be non-zero if $t^* \in \operatorname{sons}^*(t)$ and $s^* \in \operatorname{sons}^*(s)$. In this case, we can again use Lemma 6.13 to find

$$S_{C,b^*} = \begin{cases} V_{C,t^*}^* V_{t^*} E_{t^*,t} S_b F_{s^*,s}^* W_{s^*}^* W_{C,s^*} & \text{if } b^* = (t^*, s^*) \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b), \\ 0 & \text{otherwise.} \end{cases}$$

We once more employ cluster operators to simplify this equation: we introduce $P_V := (P_{V,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $P_W := (P_{W,s})_{s\in\mathcal{T}_\mathcal{J}}$ by

$$P_{V,t} := V_{C,t}^* V_t, \quad P_{W,s} := W_s^* W_{C,s}$$

for all $t \in \mathcal{T}_\mathcal{I}$, $s \in \mathcal{T}_\mathcal{J}$, and find

$$S_{C,b^*} = \begin{cases} P_{V,t^*} E_{t^*,t} S_b F_{s^*,s}^* P_{W,s^*} & \text{if } b^* = (t^*, s^*) \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b), \\ 0 & \text{otherwise.} \end{cases} \tag{7.7}$$

Both equations (7.6) and (7.7) require matrices of the type

$$\widehat{S}_{C,b^*} := \begin{cases} E_{t^*,t} S_b F_{s^*,s}^* & \text{if } b^* = (t^*, s^*) \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b), \\ 0 & \text{otherwise,} \end{cases} \tag{7.8}$$

and the efficient computation of these matrices is the key component of the matrix backward transformation.

Let $b^* \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b)$, and let $b' \in \operatorname{sons}(b^*)$. According to Lemma 5.28, we have

$$E_{t',t} = E_{t',t^*} E_{t^*,t}, \quad F_{s',s} = F_{s',s^*} F_{s^*,s},$$

and (7.8) yields

$$\widehat{S}_{C,b'} = E_{t',t} S_b F_{s',s}^* = E_{t',t^*} E_{t^*,t} S_b F_{s,s^*}^* F_{s',s^*}^* = E_{t',t^*} \widehat{S}_{C,b^*} F_{s',s^*}^*.$$

Due to Definition 3.12, we have either $t' \in \operatorname{sons}(t)$ or $t' = t$ and either $s' \in \operatorname{sons}(s)$ or $s' = s$, so Definition 5.27 implies

$$\widehat{S}_{C,b'} = \begin{cases} \widehat{S}_{C,b^*} F_{s'}^* & \text{if } \operatorname{sons}(t) = \emptyset, \operatorname{sons}(s) \neq \emptyset, \\ E_{t'} \widehat{S}_{C,b^*} & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) = \emptyset, \\ E_{t'} \widehat{S}_{C,b^*} F_{s'}^* & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) \neq \emptyset. \end{cases} \tag{7.9}$$

Lemma 7.4 (Complexity of splitting). *Let V and W be nested cluster bases with rank distributions K and L, and let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18). Let $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Algorithm 37 requires not more than*

$$2(k_t^2 l_s + k_t l_s^2)$$

operations.

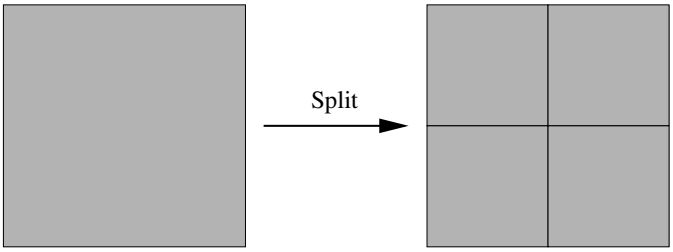


Figure 7.3. Split a matrix $\widehat{S}_{C,b}$ into matrices $\widehat{S}_{C,b'}$ corresponding to subblocks b' of b.

Algorithm 37. Split a block into son blocks.

procedure Split(b, V, W, **var** $\widehat{S}_C$);
$(t, s) \leftarrow b$;
if sons$(t) = \emptyset$ **then**
  if sons$(s) = \emptyset$ **then**
    Do nothing {Leaf blocks are handled differently}
  else
    for $s' \in$ sons(s) **do**
      $b' \leftarrow (t, s')$; {sons$(t) = \emptyset$, sons$(s) \neq \emptyset$}
      $\widehat{S}_{C,b'} \leftarrow \widehat{S}_{C,b'} + \widehat{S}_{C,b} F_{s'}^*$
    end for
  end if
else
  if sons$(s) = \emptyset$ **then**
    for $t' \in$ sons(t) **do**
      $b' \leftarrow (t', s)$; {sons$(t) \neq \emptyset$, sons$(s) = \emptyset$}
      $\widehat{S}_{C,b'} \leftarrow \widehat{S}_{C,b'} + E_{t'} \widehat{S}_{C,b}$
    end for
  else
    for $t' \in$ sons(t) **do**
      $X_{t'} \leftarrow E_{t'} \widehat{S}_{C,b} \in \mathbb{R}^{K_{t'} \times L_s}$; {sons$(t) \neq \emptyset$, sons$(s) \neq \emptyset$}
      for $s' \in$ sons(s) **do**
        $b' \leftarrow (t', s')$;
        $\widehat{S}_{C,b'} \leftarrow \widehat{S}_{C,b'} + X_{t'} F_{s'}^*$
      end for
    end for
  end if
end if

Proof. Similar to the proof of Lemma 7.1. □

The step (7.9) is handled by Algorithm 37, which translates a coupling matrix corresponding to the father block $b = (t, s)$ into coupling matrices corresponding to son blocks b'. If t and s are not leaf clusters, we employ auxiliary matrices $X_{t'}$ to store the intermediate results corresponding to sons t' of t. As in Algorithm 35, this improves the efficiency of the implementation.

Using (7.9), as implemented in Algorithm 37, to construct $\widehat{S}_C = (\widehat{S}_{C,b})_{b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}}$ for all blocks by a top-down recursion and noting that (7.6) takes the form

$$\chi_t C \chi_s = V_t \widehat{S}_{C,b} W_s$$

for inadmissible leaves $b = (t, s) \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ and that (7.7) takes the form

$$S_{C,b} = P_{V,t} \widehat{S}_{C,b} P_{W,s}$$

for admissible leaves $b = (t, s) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$ leads to the recursive Algorithm 38 for constructing the best approximation $\widetilde{C}$ of $C = V_t S_b W_s^*$ in the $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$.

Algorithm 38. Matrix backward transformation.

procedure MatrixBackward(b, V, W, P_V, P_W, **var** $\widehat{S}_C$, S_C);
$(t, s) := b$;
if sons($\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b$) $\neq \emptyset$ **then**
  Split(b, V, W, $\widehat{S}_C$); {Algorithm 37}
  for $b' \in$ sons($\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b$) **do**
    MatrixBackward(b', V, W, P_V, P_W, $\widehat{S}_C$, S_C)
  end for
else if b is admissible **then**
  $S_{C,b} \leftarrow S_{C,b} + P_{V,t} \widehat{S}_{C,b} P_{W,s}$ {Admissible leaf}
else
  $\chi_t C \chi_s \leftarrow \chi_t C \chi_s + V_t \widehat{S}_{C,b} W_s^*$ {Inadmissible leaf}
end if

This algorithm closely resembles the backward transformation Algorithm 7, but works on matrices instead of vectors, therefore it is called the *matrix backward transformation*. As in the case of the former, it is a good idea to ensure that the algorithm adds the contribution of a father block to the son blocks instead of simply replacing the latter, since this leads to a significant reduction in complexity for the matrix-matrix multiplication discussed in Secion 7.7.

The matrix backward transformation is the counterpart of the matrix forward transformation: the latter performs a bottom-up recursion to transform leaf blocks into larger blocks, while the matrix backward transformation uses a top-down recursion to transform larger blocks into leaf blocks.

Lemma 7.5 (Complexity). *Let V and W be nested cluster bases with rank distributions K and L, let V_C and W_C be nested cluster bases with rank distributions K_C and L_C. Let $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree. Let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$, $(l_s)_{s\in\mathcal{T}_\mathcal{J}}$, $(k_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.16) and (3.18). Let*

$$\hat{k}_t := \max\{k_t, k_{C,t}\}, \quad \hat{l}_s := \max\{l_s, l_{C,s}\}$$

for all $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$. Algorithm 38, started with $b = \operatorname{root}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}})$, requires not more than

$$2 \sum_{b=(t,s)\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}} (k_t^2 l_s + k_t l_s^2) \leq 2 \sum_{b=(t,s)\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}} (k_t^3 + l_s^3)$$

operations. If $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse, this can be bounded by

$$2C_{\mathrm{sp}}\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} k_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_s^3\Big).$$

If in addition K, L, K_C and L_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I} + n_\mathcal{J}))$.

Proof. Similar to the proof of Lemma 7.3. □

Remark 7.6 (Optimality). If the row cluster basis $V_C = (V_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ and the column cluster basis $W_C = (W_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ chosen for the matrix C are orthogonal, our construction implies that the matrix $\widetilde{C}$ defined in (7.5) is the best approximation of $V_t S_b W_s^*$ in $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$. □

7.3 Matrix addition

Let us now consider the computation of $M := C + A$ for $A, C, M \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

According to Remark 3.37, $\mathcal{H}^2$-matrix spaces are subspaces of $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, i.e., the sum of two $\mathcal{H}^2$-matrices based on the same block cluster tree and the same row and column cluster bases will again be an $\mathcal{H}^2$-matrix with the same structure, and its representation can be computed in linear complexity by adding the near- and farfield matrices. We consider generalizations of this result: the matrices C and A can be based on different block cluster trees and different cluster bases.

Let $V_A = (V_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $V_C = (V_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ be nested cluster bases for $\mathcal{T}_\mathcal{I}$ with rank distributions $K_A = (K_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $K_C = (K_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ and families of transfer matrices $E_A = (E_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $E_C = (E_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$.

Let $W_A = (W_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $W_C = (W_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ be nested cluster bases for $\mathcal{T}_\mathcal{J}$ with rank distributions $L_A = (L_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $L_C = (L_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ and families of transfer matrices $F_A = (F_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $F_C = (F_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$.

Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ be admissible block cluster trees for the row cluster tree $\mathcal{T}_{\mathcal{I}}$ and the column cluster tree $\mathcal{T}_{\mathcal{J}}$.

Let $A \in \mathcal{H}^2(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, V_A, W_A)$ and $C \in \mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$. We assume that the $\mathcal{H}^2$-matrices A and C are given in the usual $\mathcal{H}^2$-matrix representations

$$A = \sum_{b=(t,s)\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}} V_{A,t} S_{A,b} W^*_{A,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}} \chi_t A \chi_s, \tag{7.10a}$$

$$C = \sum_{b=(t,s)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}} V_{C,t} S_{C,b} W^*_{C,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}} \chi_t C \chi_s \tag{7.10b}$$

for coupling matrices $(S_{A,b})_{b\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}}$ and $(S_{C,b})_{b\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}}$ and now introduce two algorithms for computing M.

7.4 Projected matrix-matrix addition

The sum $M := C + A$ is, in general, not an element of the $\mathcal{H}^2$-matrix spaces $\mathcal{H}^2(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, V_A, W_A)$ or $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$. We assume that M can be *approximated* reasonably well in the space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$ and try to find its best approximation in this matrix space.

In order to be able to use the techniques introduced in Chapter 5, we require the cluster bases V_C and W_C to be orthogonal, which can be easily ensured by using the Algorithms 16 or 19.

According to Lemma 5.5, the best approximation (with respect to the Frobenius norm) of the matrix M in the space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$ is given by

$$\begin{aligned}\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}},V_C,W_C} M &= \sum_{b=(t,s)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}} V_{C,t} V^*_{C,t} M W_{C,s} W^*_{C,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}} \chi_t M \chi_s \\ &= \sum_{b=(t,s)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}} V_{C,t} S_{M,b} W^*_{C,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}} \chi_t M \chi_s\end{aligned}$$

for the matrices $(S_{M,b})_{b\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}}$ defined by

$$S_{M,b} = V^*_{C,t} M W_{C,s}$$

for all $b = (t,s) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$. Computing the desired result $\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}},V_C,W_C} M$ is equivalent to computing $S_{M,b}$ for all admissible leaves $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$ and computing $\chi_t M \chi_s$ for all inadmissible leaves $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$.

Case 1: Admissible leaves $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$

Let $b = (t, s) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$. For $M = C + A$, the equation above takes the form

$$S_{M,b} = V_{C,t}^* C W_{C,s} + V_{C,t}^* A W_{C,s}.$$

Since V_C and W_C are orthogonal, combining Lemma 5.4 and (7.10b) yields

$$S_{M,b} = S_{C,b} + V_{C,t}^* A W_{C,s}, \tag{7.11}$$

i.e., computing $S_{M,b}$ is a simple task once we have prepared the matrix $V_{C,t}^* A W_{C,s}$ in an efficient way.

If $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ holds, we can apply the matrix forward transformation to the matrix A and the cluster bases V_C and W_C in order to compute

$$\widehat{S}_{A,b} = V_{C,t}^* A W_{C,s}.$$

We can see that this is exactly the matrix we need, and that (7.11) takes the form

$$S_{M,b} = S_{C,b} + \widehat{S}_{A,b}.$$

If $b = (t, s) \notin \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ holds, we can use the following result in order to find a predecessor $b^* \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ of b:

Lemma 7.7. *Let $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ with $b \notin \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$. Then we can find an admissible leaf $b^* \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ which is a predecessor of b, i.e., which satisfies $b \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b^*)$.*

Proof. We consider the set

$$\mathcal{A} := \{b^+ \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \cap \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}} \ : \ b \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b^+)\}.$$

Since $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ share the same root, $\mathcal{A}$ is not empty. Let $b^* = (t^*, s^*) \in \mathcal{A}$ be the block with maximal level in $\mathcal{A}$. If it had sons in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, Definition 3.12 would imply that the sons in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ would coincide with those in $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, and one of the latter would have to be a predecessor of b.

This predecessor would therefore be included in $\mathcal{A}$, and its level would be higher than that of b^*, which contradicts the assumption. Therefore b^* has to be a leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$.

If b^* was an inadmissible leaf, both t^* and s^* would have to be leaves, according to Definition 3.18, therefore b^* would also have to be a leaf in $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$. Since $b \neq b^*$, this is impossible, so b^* has to be an admissible leaf. □

According to this result, there is a $b^* = (t^*, s^*) \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ with

$$S_{M,b} = S_{C,b} + V_{C,t}^* V_{A,t^*} S_{A,b^*} W_{A,s^*}^* W_{C,s},$$

and due to Lemma 3.15, b^* is unique. Applying the matrix backward transformation to the block b^* yields a matrix $\widehat{S}_{C,b}$ with

$$S_{M,b} = S_{C,b} + V_{C,t}^* V_{A,t} \widehat{S}_{C,b} W_{A,s}^* W_{C,s},$$

and using the cluster operators $P_V := (P_{V,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $P_W := (P_{W,s})_{s\in\mathcal{T}_\mathcal{J}}$ defined by

$$P_{V,t} := V_{C,t}^* V_{A,t}, \quad P_{W,s} := W_{A,s}^* W_{C,s}$$

for all $t \in \mathcal{T}_\mathcal{I}$, $s \in \mathcal{T}_\mathcal{J}$ gives us

$$S_{M,b} = S_{C,b} + P_{V,t} \widehat{S}_{C,b} P_{W,s}.$$

We conclude that admissible leaves b of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ can be handled by the matrix forward transformation if they are contained in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and by the matrix backward transformation if they are not.

Case 2: Inadmissible leaves $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$

Let $b = (t,s) \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$. If $b \notin \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, Lemma 7.7 yields the existence of an admissible leaf $b^* \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ with $b \in \operatorname{sons}^*(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b^*)$, and we can use the matrix backward transformation.

If $b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, Definition 3.12 implies that the block has to be a leaf in $\mathcal{L}_{A,\mathcal{I}\times\mathcal{J}}$, too. If it is admissible, we can compute its explicit representation by the matrix backward transformation. Otherwise, we can simply add $\chi_t A \chi_s$ to $\chi_t C \chi_s$.

Algorithm 39 uses the matrix forward and backward transformation in order to compute the projection $\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}},V_C,W_C}(A + C)$ of the sum of A and C. The far- and nearfield matrices of C are overwritten with the result.

Lemma 7.8 (Complexity). *Let V_A and V_C be nested cluster bases for $\mathcal{T}_\mathcal{I}$ with rank distributions K_A and K_C, and let $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(k_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ be defined as in (3.16). Let W_A and W_C be nested cluster bases for $\mathcal{T}_\mathcal{J}$ with rank distributions L_A and L_C, and let $(l_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $(l_{C,s})_{s\in\mathcal{T}_\mathcal{J}}$ be defined as in (3.18). Let*

$$\hat{k}_t := \max\{k_{A,t}, k_{C,t}\}, \quad \hat{l}_s := \max\{l_{A,s}, l_{C,s}\}$$

for all $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$. Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse admissible block cluster trees. Algorithm 39 requires not more than

$$4\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} \hat{l}_s^3\Big) + \frac{C_{\mathrm{sp}}}{2}\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} \hat{k}_t^2 + \sum_{s\in\mathcal{T}_\mathcal{J}} l_s^2\Big)$$

operations. If K_A, K_C, L_A and L_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I} + n_\mathcal{J}))$.

Algorithm 39. Projected matrix addition.

procedure ProjectedAdd(A, **var** C);
ClusterBasisProduct(root($\mathcal{T}_\mathcal{I}$), V_C, V_A, P_V); {Algorithm 13}
ClusterBasisProduct(root($\mathcal{T}_\mathcal{J}$), W_A, W_C, P_W); {Algorithm 13}
MatrixForward(root($\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$), V_C, W_C, P_V, P_W, S_A, $\widehat{S}_A$); {Algorithm 36}
for $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ **do**
  $\widehat{S}_{C,b} \leftarrow 0$
end for;
for $b = (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \cap \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ **do**
  if $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$ **then**
    $S_{C,b} \leftarrow S_{C,b} + \widehat{S}_{A,b}$
  else if $b \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ **then**
    $\widehat{S}_{C,b} \leftarrow S_{A,b}$
  else if $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ **then**
    $\chi_t C \chi_s \leftarrow \chi_t C \chi_s + \chi_t A \chi_s$ {$b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ implies $b \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$}
  end if
end for;
MatrixBackward(root($\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$), V_A, W_A, P_V, P_W, $\widehat{S}_{C,b}$, S_C) {Algorithm 38}

Proof. According to Lemma 5.13, we can compute P_V and P_W using Algorithm 13 in

$$2\sum_{t\in\mathcal{T}_\mathcal{I}}(k_{A,t}^3 + k_{C,t}^3) + 2\sum_{s\in\mathcal{T}_\mathcal{J}}(l_{A,s}^3 + l_{C,s}^3) \leq 4\Big(\sum_{t\in\mathcal{T}_\mathcal{I}}\hat{k}_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}}\hat{l}_s^3\Big)$$

operations.

Handling a block $b = (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \cap \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ in the main loop of Algorithm 39 requires not more than $\hat{k}_t\hat{l}_s$ operations, and we get an upper bound of

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\cap\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}} \hat{k}_t\hat{l}_s &\leq \frac{1}{2}\sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\cap\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}} \hat{k}_t^2 + \hat{l}_s^2 \\
&\leq \frac{1}{2}\sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} \hat{k}_t^2 + \hat{l}_s^2 \\
&= \frac{1}{2}\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{s\in\operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}},t)} \hat{k}_t^2 + \frac{1}{2}\sum_{s\in\mathcal{T}_\mathcal{J}}\sum_{t\in\operatorname{col}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}},s)} \hat{l}_s^2 \\
&\leq \frac{C_{\mathrm{sp}}}{2}\Big(\sum_{t\in\mathcal{T}_\mathcal{I}}\hat{k}_t^2 + \sum_{s\in\mathcal{T}_\mathcal{J}}\hat{l}_s^2\Big).
\end{aligned} \tag{7.12}$$

Adding the bound for the number of operation required for the computation of the cluster basis products yields the desired estimate.

If K_A, K_C, L_A and L_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded, we can use Lemma 5.12 in combination with the Lemmas 3.45 and 3.48 to complete the proof. □

Remark 7.9. If $V_A = V_C$, $W_A = W_C$ and $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} = \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ hold, we do not have to prepare the cluster operators P_V and P_W, and since we also do not have to use the matrix forward and backward transformation, we can reduce Algorithm 39 to its main loop, which requires not more than

$$\frac{C_{\mathrm{sp}}}{2}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^2\Big)$$

operations. If the rank distributions K_A and L_A are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, we find that the number of operations is in $\mathcal{O}((\alpha+\beta)^r(n_{\mathcal{I}}+n_{\mathcal{J}}))$. □

Remark 7.10 (Auxiliary storage). Algorithm 39 requires temporary storage for the families $\hat{S}_A = (\hat{S}_{A,b})_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}$ and $\hat{S}_C = (\hat{S}_{C,b})_{b\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}}$ used in the matrix forward and backward transformation. As in Lemma 3.38, we can prove that not more than

$$\frac{C_{\mathrm{sp}}}{2}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^2 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^2\Big)$$

units of auxiliary storage are required.

In the special case of Algorithm 39, we can avoid storing all matrices in $\hat{S}_A$ and $\hat{S}_C$ by relying on the fact that the computation of, e.g., $\hat{S}_{C,b}$ for $b \in \mathcal{L}_{C,\mathcal{I}\times\mathcal{J}}$ can be accomplished by keeping only the predecessors of b in storage. Using this approach means that we only have to store a small number of matrices *per level* of the block cluster tree instead of per block, i.e., it is far more storage-efficient as long as the block cluster tree is not degenerate. □

7.5 Exact matrix-matrix addition

We have seen that we can compute the best approximation of the sum of two $\mathcal{H}^2$-matrices in a prescribed $\mathcal{H}^2$-matrix space by the efficient Algorithm 39. Estimates for the error of this best approximation depend on the nature of the matrices A and C, i.e., on the underlying problem.

Our goal is now to construct an $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$ that can represent any matrix $M = A + C$ with $A \in \mathcal{H}^2(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, V_A, W_A)$ and $C \in \mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$, i.e., the error of the best approximation in this space will be zero.

Based on the exact representation of the sum in $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$, we can construct more efficient approximations by truncating the cluster bases V_M and W_M using Algorithm 19 or (preferably) by applying the compression Algorithm 30.

We construct the block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ by merging the trees $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$: we require that $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ has the same root as $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, namely $(\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}))$, and that the set of blocks of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is the union of the blocks of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$.

Definition 7.11 (Induced block cluster tree). Let $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ be the minimal block cluster tree (i.e., the block cluster tree with minimal number of blocks) for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ satisfying the following conditions:

- A block $b = (t, s) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is subdivided if it is subdivided in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, i.e., the sons of b in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ are given by

$$\operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b) = \begin{cases} \operatorname{sons}^+(t) \times \operatorname{sons}^+(s) & \text{if } b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{A,\mathcal{I}\times\mathcal{J}} \\ & \text{or } b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{C,\mathcal{I}\times\mathcal{J}}, \\ \emptyset & \text{otherwise.} \end{cases} \tag{7.13}$$

- A block $b = (t, s) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is admissible if and only if it is a descendant of admissible blocks in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, i.e.,

$$\begin{aligned} b \in \mathcal{L}^+_{M,\mathcal{I}\times\mathcal{J}} \iff & \text{ there are } b^*_A \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}} \text{ and } b^*_C \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}} \text{ with} \\ & b \in \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b^*_A) \cap \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b^*_C). \end{aligned} \tag{7.14}$$

Then $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is called the *induced block cluster tree* for the addition.

Lemma 7.12. *The induced block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is admissible. For each $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, we have $b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$. For each $b \in \mathcal{L}^-_{M,\mathcal{I}\times\mathcal{J}}$, we have $b \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$. If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ are C_{sp}-sparse, the tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is $2C_{\mathrm{sp}}$-sparse.*

Proof. We start by proving that each $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is an element of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ by induction on $\operatorname{level}(b)$. Let $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = 0$. Then we have $b = \operatorname{root}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}) = \operatorname{root}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$.

Let now $n \in \mathbb{N}$ be such that $b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ holds for all $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = n$, and let $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = n + 1$. Then we can find a father $b^+ \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ with $b \in \operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b^+)$ and $\operatorname{level}(b^+) = n$. According to the induction assumption, we have $b^+ \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $b^+ \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, and (7.13) yields $b \in \operatorname{sons}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b^+) \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \operatorname{sons}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, b^+) \subseteq \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, respectively, which completes the induction.

Let now $b \in \mathcal{L}^-_{M,\mathcal{I}\times\mathcal{J}}$. We prove that $b \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ holds. We have already seen that b has to be a block in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$. Without loss of generality, let us assume $b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$. Since b is a leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, (7.13) implies that it has to be a leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$. If it is an inadmissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, we have already proven $b \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$.

If b is an admissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, we can let $b_A^* := b$ and find $b = b_A^* \in \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b_A^*)$. If $b \in \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b_C^*)$ would hold for a $b_C^* \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$, (7.14) would imply that b is admissible in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, which is not the case. Therefore $b \notin \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b_C^*)$ has to hold for all $b_C^* \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$.

If $b \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ would hold, Lemma 7.7 would give us an admissible leaf $b_C^* \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$ with $b \in \operatorname{sons}^*(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b_C^*) \subseteq \operatorname{sons}^*(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, b_C^*)$, and we have already seen that this is impossible.

Therefore we have $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, and since b is a leaf in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, it has to be a leaf in $\mathcal{L}_{C,\mathcal{I}\times\mathcal{J}}$, as well. Since b is inadmissible in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ and admissible in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, (7.14) implies that it has to be inadmissible in $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$, i.e., $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$.

In order to prove that $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is an admissible block cluster tree, we only have to verify that $\operatorname{sons}(t) = \emptyset = \operatorname{sons}(s)$ holds for all inadmissible leaves $b = (t,s) \in \mathcal{L}^-_{M,\mathcal{I}\times\mathcal{J}}$. Let $b \in \mathcal{L}^-_{M,\mathcal{I}\times\mathcal{J}}$. Since we know that $b \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ holds and that $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ are admissible block cluster trees, this is obvious.

Let us now consider the sparsity of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$. Let $t \in \mathcal{T}_\mathcal{I}$, and let $s \in \operatorname{row}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, t)$. This means $b = (t,s) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, and we have already seen that this implies $b \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or $b \in \mathcal{T}_{B,\mathcal{I}\times\mathcal{J}}$, so we find

$$\#\operatorname{row}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, t) \leq \#\operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t) + \#\operatorname{row}(\mathcal{T}_{B,\mathcal{I}\times\mathcal{J}}, t) \leq 2C_{\mathrm{sp}},$$

which concludes the proof. □

We can see that the block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is "finer" than $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ and that each admissible leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ is a descendant of admissible leaves in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$. In order to be able to express both A and C in the space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$, we therefore only have to ensure that the descendants of admissible leaves in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ can be expressed by V_M and W_M.

In order to simplify the presentation, let us assume that the index sets are disjoint, i.e., that

$$K_{A,t} \cap K_{C,t} = \emptyset, \quad L_{A,s} \cap L_{C,s} = \emptyset \quad \text{holds for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}.$$

Due to this assumption, we can construct V_M or W_M by simply merging V_A and V_C or W_A and W_C, respectively.

Definition 7.13 (Induced cluster bases). We define the *induced row cluster basis* $V_M := (V_{M,t})_{t\in\mathcal{T}_\mathcal{I}}$ with corresponding rank distribution $K_M := (K_{M,t})_{t\in\mathcal{T}_\mathcal{I}}$ by

$$K_{M,t} := K_{A,t} \mathbin{\dot\cup} K_{C,t}, \quad V_{M,t} := \begin{pmatrix} V_{A,t} & V_{C,t} \end{pmatrix} \in \mathbb{R}_t^{\mathcal{I}\times K_{M,t}} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

and the *induced column cluster basis* $W_M := (W_{M,s})_{s\in\mathcal{T}_\mathcal{J}}$ with corresponding rank distribution $L_M := (L_{M,s})_{s\in\mathcal{T}_\mathcal{J}}$ by

$$L_{M,s} := L_{A,s} \mathbin{\dot\cup} L_{C,s}, \quad W_{M,s} := \begin{pmatrix} W_{A,s} & W_{C,s} \end{pmatrix} \in \mathbb{R}_s^{\mathcal{J}\times L_{M,s}} \quad \text{for all } s \in \mathcal{T}_\mathcal{J}.$$

Lemma 7.14. *The families $V_M = (V_{M,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $W_M = (W_{M,s})_{s\in\mathcal{T}_\mathcal{J}}$ are nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively. The transfer matrices $E_M := (E_{M,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $F_M := (F_{M,s})_{s\in\mathcal{T}_\mathcal{J}}$ are given by*

$$E_{M,t} := \begin{pmatrix} E_{A,t} & \\ & E_{C,t} \end{pmatrix}, \quad F_{M,s} := \begin{pmatrix} F_{A,s} & \\ & F_{C,s} \end{pmatrix} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}.$$

We have

$$\begin{aligned} V_{A,t} &= V_{M,t} P_{VA,t}, & V_{C,t} &= V_{M,t} P_{VC,t} && \text{for all } t \in \mathcal{T}_\mathcal{I}, \\ W_{A,s} &= W_{M,s} P_{WA,s}, & W_{C,s} &= W_{M,s} P_{WC,s} && \text{for all } s \in \mathcal{T}_\mathcal{J}, \end{aligned}$$

where the cluster operators $P_{VA} := (P_{VA,t})_{t\in\mathcal{T}_\mathcal{I}}$, $P_{VC} := (P_{VC,t})_{t\in\mathcal{T}_\mathcal{I}}$, $P_{WA} := (P_{WA,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $P_{WC} := (P_{WC,s})_{s\in\mathcal{T}_\mathcal{J}}$ are defined by

$$\begin{aligned} P_{VA,t} &:= \begin{pmatrix} I \\ 0 \end{pmatrix} \in \mathbb{R}^{K_{M,t}\times K_{A,t}}, & P_{VC,t} &:= \begin{pmatrix} 0 \\ I \end{pmatrix} \in \mathbb{R}^{K_{M,t}\times K_{C,t}} && \text{for all } t \in \mathcal{T}_\mathcal{I}, \\ P_{WA,s} &:= \begin{pmatrix} I \\ 0 \end{pmatrix} \in \mathbb{R}^{L_{M,s}\times L_{A,s}}, & P_{WC,s} &:= \begin{pmatrix} 0 \\ I \end{pmatrix} \in \mathbb{R}^{L_{M,s}\times L_{C,s}} && \text{for all } s \in \mathcal{T}_\mathcal{J}, \end{aligned}$$

i.e., the cluster bases V_M or W_M can be used to express anything which can be expressed by V_A and V_C or W_A and W_C, respectively.

Proof. This is trivial. □

Theorem 7.15 (Exact matrix addition). *Let $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$ be the induced block cluster tree from Definition 7.11. Let $V_M = (V_{M,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $W_M = (W_{M,s})_{s\in\mathcal{T}_\mathcal{J}}$ be the induced row and column cluster bases from Definition 7.13. Then we have $M = C + A \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$.*

Proof. Due to Remark 3.37, it suffices to prove that $A \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$ and $C \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$ hold.

Without loss of generality, we consider only the representation of the matrix A in $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$. Let $b = (t, s) \in \mathcal{L}_{A,\mathcal{I}\times\mathcal{J}}$. Lemma 7.12 implies $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$.

If b is inadmissible, it has to be a leaf in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$, and we can use the corresponding nearfield matrix directly. If b is admissible, we observe that

$$V_{M,t} \begin{pmatrix} S_{A,b} & 0 \\ 0 & 0 \end{pmatrix} W_{M,s}^* = \begin{pmatrix} V_{A,t} & V_{C,t} \end{pmatrix} \begin{pmatrix} S_{A,b} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} W_{A,s}^* \\ W_{C,s}^* \end{pmatrix} = V_{A,t} S_{A,b} W_{A,s}^*$$

holds. □

The representation of $M = C + A$ can be computed efficiently by using the matrix backward transformation Algorithm 38: we introduce $\widehat{S}_M := (\widehat{S}_{M,b})_{b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}}$ by

$$\widehat{S}_{M,b} := \begin{cases} \begin{pmatrix} S_{A,b} & 0 \\ 0 & S_{C,b} \end{pmatrix} & \text{if } b \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}} \cap \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}, \\ \begin{pmatrix} S_{A,b} & 0 \\ 0 & 0 \end{pmatrix} & \text{if } b \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}, \\ \begin{pmatrix} 0 & 0 \\ 0 & S_{C,b} \end{pmatrix} & \text{if } b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}$$

and use Algorithm 38 with trivial P_V and P_W in order to construct the representation of $M = A + C$ in the space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$.

7.6 Matrix multiplication

Let us now turn our attention towards the computation of $M := C + AB$ for $A \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, $B \in \mathbb{R}^{\mathcal{J}\times\mathcal{K}}$ and $C, M \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}$.

Let $V_A = (V_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $V_C = (V_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ be nested cluster bases for the cluster tree $\mathcal{T}_\mathcal{I}$ with rank distributions $K_A = (K_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $K_C = (K_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ and families $E_A = (E_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $E_C = (E_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$ of transfer matrices

Let $W_A = (W_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $V_B = (V_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ be nested cluster bases for the cluster tree $\mathcal{T}_\mathcal{J}$ with rank distributions $L_A = (L_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $K_B = (K_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ and families $F_A = (F_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $E_B = (E_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ of transfer matrices.

Let $W_B = (W_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ and $W_C = (W_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$ be nested cluster bases for the cluster tree $\mathcal{T}_\mathcal{K}$ with rank distributions $L_B = (L_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$, $L_C = (L_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$ and families $F_B = (F_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$, $F_C = (F_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$ of transfer matrices.

Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ be admissible block cluster trees for the cluster trees $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ and $\mathcal{T}_\mathcal{K}$, respectively.

We assume that A, B and C are $\mathcal{H}^2$-matrices given in the usual $\mathcal{H}^2$-matrix representation:

$$A = \sum_{b=(t,s)\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}} V_{A,t} S_{A,b} W^*_{A,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}} \chi_t A \chi_s,$$

$$B = \sum_{b=(s,r)\in\mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}} V_{B,s} S_{B,b} W^*_{B,r} + \sum_{b=(s,r)\in\mathcal{L}^-_{B,\mathcal{J}\times\mathcal{K}}} \chi_s B \chi_r,$$

$$C = \sum_{b=(t,r)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}} V_{C,t} S_{C,b} W^*_{C,r} + \sum_{b=(t,r)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}} \chi_t C \chi_r$$

for families $S_A = (S_{A,b})_{b \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}}$, $S_B = (S_{B,b})_{b \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}}$ and $S_C = (S_{C,b})_{b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}}$ of coupling matrices.

7.7 Projected matrix-matrix multiplication

As in the case of the addition, we first consider the projection of the result $M := C + AB$ into the space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$. We once again use the results of Chapter 5 to compute the best approximation of M, and to keep the presentation simple we assume that the cluster bases V_C and W_C are orthogonal.

According to Lemma 5.5, the best approximation of the matrix M in the $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$ is given by the projection

$$\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},V_C,W_C} M = \sum_{b=(t,r)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}} V_{C,t} S_{M,b} W_{C,r}^* + \sum_{b=(t,r)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}} \chi_t M \chi_r$$

for the matrices $(S_{M,b})_{b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}}$ defined by

$$S_{M,b} := V_{C,t}^* M W_{C,r} \quad \text{for all } b = (t,r) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}.$$

We have to compute $S_{M,b}$ for all admissible leaves $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ and $\chi_t M \chi_r$ for all inadmissible leaves $b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}$.

The main difficulty of the matrix-matrix multiplication is due to the fact that the computation of one block

$$\chi_t M \chi_r = \chi_t (C + AB) \chi_r = \chi_t C \chi_r + \chi_t AB \chi_r$$

requires us to consider the interaction of *all* indices of $\mathcal{J}$ in the evaluation of AB, i.e., it is a non-local operation.

In order to handle the non-locality efficiently, we split the computation of $\chi_t AB \chi_r$ into a sum

$$\chi_t AB \chi_r = \sum_{s \in \mathcal{S}_{t,r}} \chi_t A \chi_s B \chi_r$$

over a suitably-chosen subset $\mathcal{S}_{t,r}$ of clusters $s \in \mathcal{T}_{\mathcal{J}}$ and have to compute

$$\chi_t M \chi_r = \chi_t C \chi_r + \sum_{s \in \mathcal{S}_{t,r}} \chi_t A \chi_s B \chi_r$$

in the case of an inadmissible leaf $b = (t,r) \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{J}}$ and

$$V_{C,t}^* M W_{C,r} = S_{C,b} + \sum_{s \in \mathcal{S}_{t,r}} V_{C,t}^* A \chi_s B W_{C,r}$$

in the case of an admissible leaf $b = (t, r) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{J}}$.

We can split both computations into a number of elementary steps

$$\chi_t M \chi_r \leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r, \quad S_{M,b} \leftarrow S_{M,b} + V^*_{C,t} A \chi_s B W_{C,r}, \tag{7.15}$$

and we can manage these elementary steps by a recursion.

Let $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$. We let $b_A := (t, s)$, $b_B := (s, r)$ and $b_C := (t, r)$. Each of these three blocks can fall into one of three categories: it can be admissible, it can be an inadmissible leaf, and it can be a non-leaf block. In the first case, we can use the factorized representation of the corresponding submatrix, in the second case, we can assume that the submatrix is small, while the third case requires a proper handling of the block's descendents.

Definition 7.16 (Subsets of blocks). Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block cluster tree. We define the *set of subdivided blocks* by

$$\mathcal{S}_{\mathcal{I}\times\mathcal{J}} := \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$$

and the *set of inadmissible blocks* by

$$\mathcal{I}_{\mathcal{I}\times\mathcal{J}} := \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

The following observation simplifies the treatment of recursions for subdivided blocks:

Lemma 7.17. *For all admissible block cluster trees $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we have*

$$\mathcal{S}_{\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{I}_{\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{T}_{\mathcal{I}\times\mathcal{J}}.$$

If $b = (t, s) \in \mathcal{I}_{\mathcal{I}\times\mathcal{J}}$, we have $b' := (t', s') \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ for all $t' \in \operatorname{sons}^+(t)$ and all $s' \in \operatorname{sons}^+(s)$.

Proof. Since $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, the first inclusion is a direct consequence of Definition 7.16.

Let now $b = (t, s) \in \mathcal{I}_{\mathcal{I}\times\mathcal{J}}$, $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ and $b' := (t', s')$.

If $\operatorname{sons}(t) = \emptyset = \operatorname{sons}(s)$, we have $t' = t$ and $s' = s$, i.e., $b' = b \in \mathcal{I}_{\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

Otherwise, b cannot be an inadmissible leaf, since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, and b cannot be an admissible leaf, since admissible leaves are excluded from $\mathcal{I}_{\mathcal{I}\times\mathcal{J}}$. Therefore b is not a leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and Definition 3.12 yields $\operatorname{sons}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, b) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$, i.e., $(t', s') \in \operatorname{sons}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, b) \subseteq \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. □

Depending on the nature of b_A, b_B and b_C, i.e., depending on whether they are admissible leaves, inadmissible leaves or subdivided blocks, we have to handle different combinations of block types differently.

We do not discuss situations involving inadmissible leaf blocks in detail. Since inadmissible leaves of an admissible block cluster tree correspond to leaves of the cluster tree, which we can assume to correspond to small sets of indices, we can handle most of these situations in the standard way.

Case 1: b_C is subdivided

Let $b_C \in \mathcal{S}_{C,\mathcal{I}\times\mathcal{K}}$ be a subdivided block. We have to perform the elementary step

$$\chi_t M \chi_r \leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r.$$

Case 1a: b_A and b_B are subdivided

Let $b_A \in \mathcal{S}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B \in \mathcal{S}_{B,\mathcal{J}\times\mathcal{K}}$ be subdivided blocks in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, respectively.

Since Definition 3.12 implies

$$\operatorname{sons}(b_A) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s), \quad \operatorname{sons}(b_B) = \operatorname{sons}^+(s) \times \operatorname{sons}^+(r),$$
$$\operatorname{sons}(b_C) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(r),$$

the elementary step (7.15) is equivalent to

$$\begin{aligned}
\chi_t M \chi_r &\leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r \\
&= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} \chi_{t'} M \chi_{r'} + \chi_{t'} A \chi_s B \chi_{r'} \\
&= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} M \chi_{r'} + \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} A \chi_{s'} B \chi_{r'},
\end{aligned}$$

and we can split it up into a sequence of elementary steps

$$\chi_{t'} M \chi_{r'} \leftarrow \chi_{t'} M \chi_{r'} + \chi_{t'} A \chi_{s'} B \chi_{r'},$$

where $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ and $r' \in \operatorname{sons}^+(r)$. Each of these steps can be handled by recursion.

Case 1b: b_A is admissible, b_B is subdivided

Let b_A be admissible and let $b_B \in \mathcal{S}_{B,\mathcal{J}\times\mathcal{K}}$ be a subdivided block in $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$.

Under these conditions, the step (7.15) is given by

$$\begin{aligned}
\chi_t M \chi_r &\leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r \\
&= \sum_{b'_C = (t',r') \in \operatorname{sons}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, b_C)} (\chi_{t'} M \chi_{r'} + \chi_{t'} A \chi_s B \chi_{r'}) \\
&= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} (\chi_{t'} M \chi_{r'} + \chi_{t'} A \chi_s B \chi_{r'}) \\
&= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} \Big(\chi_{t'} M \chi_{r'} + \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} A \chi_{s'} B \chi_{r'} \Big).
\end{aligned}$$

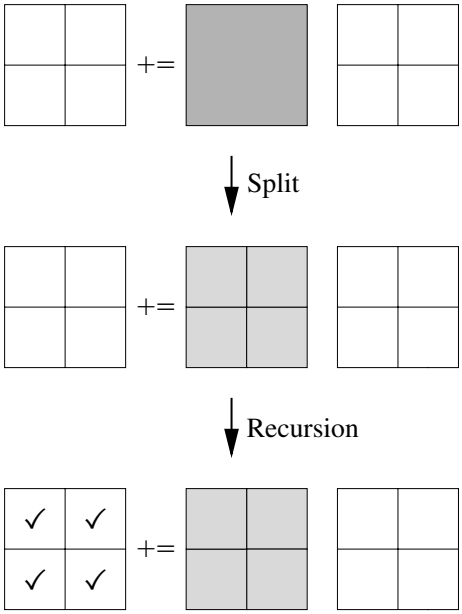


Figure 7.4. Case 1b of the multiplication: b_A is admissible, b_B and b_C are subdivided.

Since b_A is admissible, and since the cluster bases are nested, we can use

$$\chi_{t'} A \chi_{s'} = \chi_{t'} V_{A,t} S_{A,b_A} W_{A,s}^* \chi_{s'} = V_{A,t'} E_{A,t',t} S_{A,b_A} F_{A,s',s}^* W_{A,s'}^* = V_{A,t'} S_{A,b'_A} W_{A,s'}^*$$

with the auxiliary matrix

$$S_{A,b'_A} := E_{A,t',t} S_{A,b_A} F_{A,s',s}^*$$

for all $b'_A := (t', s') \in \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$ in order to get

$$\begin{aligned}\chi_t M \chi_r &\leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r \\ &= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} \Big(\chi_{t'} M \chi_{r'} + \sum_{s' \in \operatorname{sons}^+(s)} V_{A,t'} S_{A,(t',s')} W_{A,s'}^* \chi_{s'} B \chi_{r'} \Big),\end{aligned}$$

and this operation can be split up into a sequence of elementary steps of the form

$$\chi_{t'} M \chi_{r'} \leftarrow \chi_{t'} M \chi_{r'} + V_{A,t'} S_{A,b'_A} W_{A,s'}^* \chi_{s'} B \chi_{r'},$$

for all $b'_A = (t', s') \in \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$ and $r' \in \operatorname{sons}^+(r)$, where the original matrix $\chi_t A \chi_s$ is now replaced by $\chi_{t'} A \chi_{s'} = V_{A,t'} S_{A,b'_A} W_{A,s'}$. The same recursion as in Case 1a can be applied.

The auxiliary matrices S_{A,b'_A} can be computed by the same approach as in the matrix backward transformation, i.e., by using the splitting Algorithm 37. Once these matrices have been constructed, we can proceed to evaluate the sum over t', s' and r' by recursion as if (t', s') was an admissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$.

The case that b_A is subdivided and b_B is admissible can be handled by a similar procedure.

Case 1c: b_A and b_B are admissible

Let b_A and b_B be admissible (either admissible leaves of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{I}\times\mathcal{J}}$, respectively, or auxiliary blocks introduced in Case 1b). We have

$$\chi_t A \chi_s = V_{A,t} S_{A,b_A} W_{A,s}^*, \quad \chi_s B \chi_r = V_{B,s} S_{B,b_B} W_{B,r}^*$$

and find that the step (7.15) takes the form

$$\begin{aligned}\chi_t M \chi_r &\leftarrow \chi_t M \chi_r + \chi_t A \chi_s B \chi_r \\ &= \chi_t M \chi_r + V_{A,t} S_{A,b_A} W_{A,s}^* V_{B,s} S_{B,b_B} W_{B,r}^* \\ &= \chi_t M \chi_r + V_{A,t} S_{A,b_A} P_{AB,s} S_{B,b_B} W_{B,r}^*,\end{aligned}$$

where the cluster operator $P_{AB} := (P_{AB,s})_{s\in\mathcal{T}_{\mathcal{J}}}$ is defined by

$$P_{AB,s} := W_{A,s}^* V_{B,s}$$

for all $s \in \mathcal{T}_{\mathcal{J}}$. We can introduce

$$\widehat{S}_{C,b_C} := S_{A,b_A} P_{AB,s} S_{B,b_B}$$

in order to get

$$\chi_t M \chi_r \leftarrow \chi_t M \chi_r + V_{A,t} \widehat{S}_{C,b_C} W_{B,r}^*.$$

Now we only have to add an admissible block to a subdivided block matrix, and this task we can manage by means of the matrix backward transformation Algorithm 38.

The computation of the coupling matrix $\widehat{S}_{C,b_C}$ involves only matrices of dimensions $\#K_{A,t}$, $\#L_{A,s}$, $\#K_{B,s}$ and $\#L_{B,r}$, so it can be handled efficiently.

The cluster operator P_{AB} can be prepared in advance by using the cluster basis product Algorithm 13.

Case 2: b_C is admissible

Let b_C be admissible (either an admissible leaf of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}$ or an auxiliary block introduced in Case 2a). In this case, we have to perform the elementary step

$$S_{M,b_C} \leftarrow S_{M,b_C} + V_{C,t}^* A \chi_s B W_{C,r}.$$

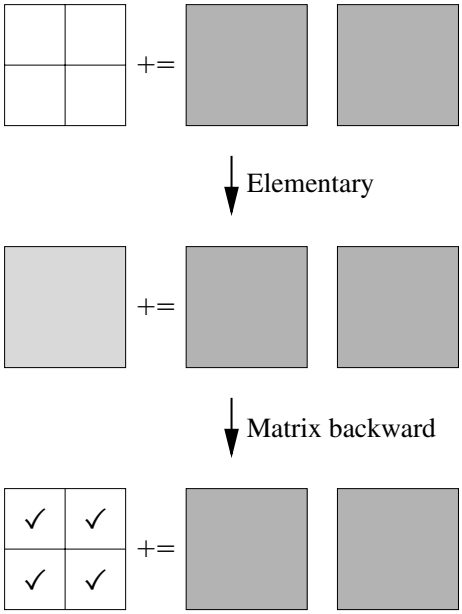


Figure 7.5. Case 1c of the multiplication: b_A and b_B are admissible, b_C is subdivided.

Case 2a: b_A and b_B are subdivided

Let $b_A \in \mathcal{S}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B \in \mathcal{S}_{B,\mathcal{J}\times\mathcal{K}}$ be subdivided blocks in $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, respectively.

Definition 3.12 implies

$$\operatorname{sons}(b_A) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s), \quad \operatorname{sons}(b_B) = \operatorname{sons}^+(s) \times \operatorname{sons}^+(r),$$

and using

$$V_{C,t} = \sum_{t' \in \operatorname{sons}^+(t)} V_{C,t'} E_{C,t',t}, \quad W_{C,r} = \sum_{r' \in \operatorname{sons}^+(r)} W_{C,r'} F_{C,r',r}$$

allows us to translate the elementary step (7.15) into

$$\begin{aligned} S_{M,b_C} &\leftarrow S_{M,b_C} + \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} E^*_{C,t',t} V^*_{C,t'} A \chi_s B W_{C,r'} F_{C,r',r} \\ &= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} S_{M,b} + E^*_{C,t',t} \Big(\sum_{s' \in \operatorname{sons}^+(s)} V^*_{C,t'} A \chi_{s'} B W_{C,r'} \Big) F_{C,r',r} \\ &= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{r' \in \operatorname{sons}^+(r)} S_{M,b} + E^*_{C,t',t} S_{M,b'_C} F_{C,r',r} \end{aligned} \tag{7.16}$$

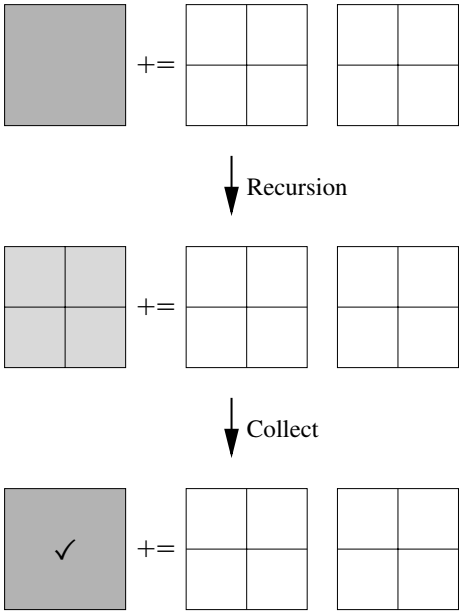


Figure 7.6. Case 2a of the multiplication: b_A and b_B are subdivided, b_C is admissible.

for $b'_C := (t', r')$ and the auxiliary matrices

$$S_{M,b'_C} := \sum_{s' \in \mathrm{sons}^+(s)} V^*_{C,t'} A \chi_{s'} B W_{C,r'},$$

which can be computed by a sequence of elementary steps

$$S_{M,b'_C} \leftarrow S_{M,b'_C} + V^*_{C,t'} A \chi_{s'} B W_{C,r'}$$

for all $b'_C = (t', r') \in \mathrm{sons}^+(t) \times \mathrm{sons}^+(r)$ and $s' \in \mathrm{sons}^+(s)$. As in the Cases 1a and 1b, these elementary steps are handled by a recursion. Once all auxiliary matrices S_{M,b'_C} have been computed, we can use the collecting Algorithm 35 to evaluate (7.16) and update $S_{M,b}$.

Case 2b: b_A is admissible, b_B is subdivided

Let b_A be admissible (either an admissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or an auxiliary block introduced in Case 1b) and let $b_B \in \mathcal{S}_{B,\mathcal{J}\times\mathcal{K}}$ be a subdivided block in $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$.

Since b_A is admissible, we have

$$\chi_t A \chi_s = V_{A,t} S_{A,b_A} W^*_{A,s},$$

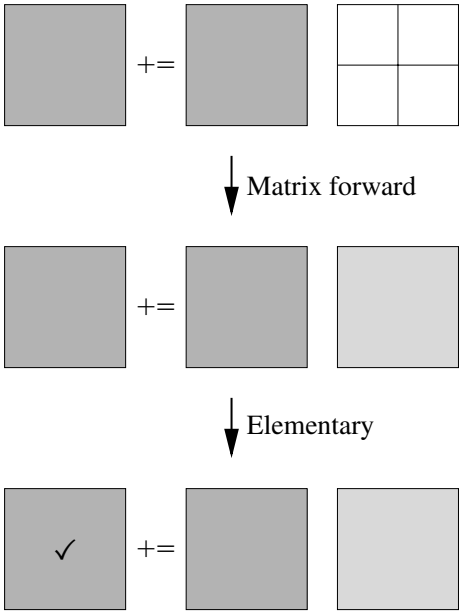


Figure 7.7. Case 2b of the multiplication: b_A and b_C are admissible, b_B is subdivided.

and the step (7.15) is equivalent to

$$\begin{aligned} S_{M,b_C} &\leftarrow S_{M,b_C} + V_{C,t}^* A \chi_s B W_{C,r} \\ &= S_{M,b_C} + V_{C,t}^* V_{A,t} S_{A,b_A} W_{A,s}^* B W_{C,r} \\ &= S_{M,b_C} + P_{CA,t} S_{A,b_A} W_{A,s}^* B W_{C,r} \\ &= S_{M,b_C} + P_{CA,t} S_{A,b_A} \widehat{S}_{B,b_B} \end{aligned} \tag{7.17}$$

for the cluster operator $P_{CA} = (P_{CA,t})_{t \in \mathcal{T}_\mathcal{I}}$ given by

$$P_{CA,t} = V_{C,t}^* V_{A,t}$$

for all $t \in \mathcal{T}_\mathcal{I}$ and for the matrices $\widehat{S}_B := (\widehat{S}_{B,b})_{b \in \mathcal{T}_{B,\mathcal{J} \times \mathcal{K}}}$ defined by

$$\widehat{S}_{B,b} := W_{A,s}^* B W_{C,r}$$

for all $b = (t, s) \in \mathcal{T}_{B,\mathcal{J} \times \mathcal{K}}$. The matrices $\widehat{S}_B = (\widehat{S}_{B,b})_{b \in \mathcal{T}_{B,\mathcal{J} \times \mathcal{K}}}$ can be prepared in advance by using the matrix forward transformation Algorithm 36, and once they are available, the step (7.17) can be performed efficiently.

The case that b_A is subdivided and b_B is admissible can be handled similarly and requires the matrices $\widehat{S}_A = (\widehat{S}_{A,b})_{b \in \mathcal{T}_{A,\mathcal{I} \times \mathcal{J}}}$ given by

$$\widehat{S}_{A,b} := V_{C,t}^* A V_{B,s}$$

for all $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ which can also be prepared by the matrix forward transformation.

Case 2c: b_A and b_B are admissible

Let b_A and b_B be admissible blocks (either admissible leaves of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, respectively, or auxiliary blocks introduced in Case 1b). We have

$$\chi_t A \chi_s = V_{A,t} S_{A,b_A} W_{A,s}^*, \quad \chi_s B \chi_r = V_{B,s} S_{B,b_B} W_{B,r}^*,$$

and the step (7.15) is given by

$$\begin{aligned} S_{M,b_C} &\leftarrow S_{M,b_C} + V_{C,t}^* A \chi_s B W_{C,r} \\ &= S_{M,b_C} + V_{C,t}^* V_{A,t} S_{A,b_A} W_{A,s}^* V_{B,s} S_{B,b_B} W_{B,r}^* W_{C,r} \\ &= S_{M,b_C} + P_{CA,t} S_{A,b_A} P_{AB,s} S_{B,b_B} P_{BC,r}, \end{aligned}$$

where the cluster operator P_{AB} is defined as in Case 1c, the cluster operator P_{CA} is defined as in Case 2b, and the cluster operator $P_{BC} = (P_{BC,r})_{r\in\mathcal{T}_\mathcal{K}}$ is given by

$$P_{BC,r} := W_{B,r}^* W_{C,r}$$

for all $r \in \mathcal{T}_\mathcal{K}$. This update involves only matrices of dimensions $\#K_{C,t}, \#K_{A,t}, \#K_{B,s}$, $\#L_{A,s}, \#L_{B,r}$ and $\#L_{C,r}$, so it can be handled efficiently. The cluster operators P_{CA}, P_{AB} and P_{BC} can be prepared efficiently by the cluster basis product Algorithm 13.

Overview of the multiplication algorithm

Let us summarize the algorithm. Its core is the update step (7.15) for blocks $b_A = (t, s)$, $b_B = (s, r)$ and $b_C = (t, r)$. Each of these blocks can be inadmissible or admissible.

If all blocks are admissible (cf. Case 2c), we can rely on cluster operators in order to perform the update efficiently. The necessary cluster operators P_{CA}, P_{AB} and P_{BC} can be prepared in advance.

If two blocks are admissible (cf. Cases 1c and 2b), we use either the matrix forward transformation to find a suitable projection of the remaining inadmissible block, or the matrix backward transformation to translate an admissible matrix representation into the one required by the inadmissible block.

If one block is admissible (cf. Cases 1b and 2a), we split this block into auxiliary subblocks and proceed by recursion. The auxiliary blocks either correspond to a splitting of the admissible block, which ensures compatibility with the subsequent recursion steps, or collect the results of the recursion steps, which can then be accumulated to update the admissible block.

If no block is admissible (cf. Case 1a), we proceed by recursion if subdivided blocks are present or compute the result directly if all blocks are leaves.

In order to implement the projected matrix-matrix multiplication efficiently, we have to handle all of these cases in the optimal way. Each of the blocks b_A, b_B and b_C can be admissible (i.e., an admissible leaf of the corresponding block cluster tree or an auxiliary block created in the Cases 1b and 2a), an inadmissible leaf of the corresponding block cluster tree, or a non-leaf element of this tree. This leads to $27 = 3^3$ different situations, which all require special handling, so the resulting algorithm is rather lengthy. In order to make it more readable, we split the algorithm into a number of parts: we have Algorithms 41, 42 and 43 for handling the special cases that b_A, b_B and b_C are admissible, respectively, Algorithm 44 for the case that none of the blocks is admissible, and a central "dispatch" Algorithm 40 which decides which one of the other algorithms is appropriate.

Algorithm 40. Projected matrix multiplication, recursion.

procedure ProjectedMulRec(t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B, **var** C);
if $(t,s) \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ **or** $(t,s) \notin \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ **then**
  ProjectedMulAdmA(t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B, C) {Algorithm 41}
else if $(s,r) \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}$ **or** $(s,r) \notin \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ **then**
  ProjectedMulAdmB(t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B, C) {Algorithm 42}
else if $(t,r) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ **or** $(t,r) \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  ProjectedMulAdmC(t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B, C) {Algorithm 43}
else
  ProjectedMulInadm(t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B, C) {Algorithm 44}
end if

Algorithm 41 handles the case that the block b_A is admissible, i.e., that it is an admissible leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ or an auxiliary block, i.e., not an element of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$. In both cases, we have $\chi_t A \chi_s = V_{A,t} S_{A,b_A} W^*_{A,s}$.

If b_B is also admissible, we can compute the product by using the cluster basis product P_{AB} and use the matrix backward transformation to translate the result into a format which is suitable for the block $\chi_t C \chi_r$ corresponding to b_C.

If b_B is not admissible, but b_C is, we can apply the matrix forward transformation to b_B in order to construct a projection of the matrix block $\chi_s B \chi_r$.

If b_B and b_C are inadmissible leaves, we can use Definition 3.18 to infer that t, s and r have to be leaves of the respective cluster trees, so we can afford to expand $\chi_t A \chi_s$ and compute the product directly.

Otherwise, i.e., if b_B and b_C are not admissible and at least one of them is further subdivided, we split b_A into auxiliary matrices and proceed by recursion.

Algorithm 42 is responsible for the case that b_B is admissible, i.e., that b_B is an admissible leaf of $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ or an auxiliary block, i.e., not an element of $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$. In this situation, we have $\chi_s B \chi_r = V_{B,s} S_{B,b_B} W^*_{B,r}$.

If b_A is admissible, the multiplication is handled by Algorithm 41.

Algorithm 41. Projected matrix multiplication, b_A is admissible.

procedure ProjectedMulAdmA($t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B$, **var** C);
$b_A \leftarrow (t,s)$; $b_B \leftarrow (s,r)$; $b_C \leftarrow (t,r)$;
if $b_B \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}$ **or** $b_B \notin \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ **then**
  $\widehat{S}_{C,b_C} \leftarrow \widehat{S}_{C,b_C} + S_{A,b_A} P_{AB,s} S_{B,b_B}$ {b_B is admissible}
else if $b_C \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ **or** $b_C \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  $S_{C,b_C} \leftarrow S_{C,b_C} + P_{CA,t} S_{A,b_A} \widehat{S}_{B,b_B}$ {b_C is admissible}
else if $b_B \in \mathcal{L}^-_{B,\mathcal{J}\times\mathcal{K}}$ **and** $b_C \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  $\chi_t C \chi_s \leftarrow \chi_t C \chi_s + V_{A,t} S_{A,b_A} W^*_{A,s} B \chi_r$ {b_B and b_C are inadmissible leaves}
else
  for $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ **do**
    $b'_A \leftarrow (t', s')$; $S_{A,b'_A} \leftarrow 0$ {Set up auxiliary blocks b'_A}
  end for;
  Split(b_A, V_A, W_A, S_A); {Algorithm 37}
  for $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$, $r' \in \operatorname{sons}^+(r)$ **do**
    ProjectedMulRec($t', s', r', P_{CA}, P_{AB}, P_{BC}, A, B, C$) {Algorithm 40}
  end for
end if

Algorithm 42. Projected matrix multiplication, b_B is admissible.

procedure ProjectedMulAdmB($t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B$, **var** C);
$b_A \leftarrow (t,s)$; $b_B \leftarrow (s,r)$; $b_C \leftarrow (t,r)$;
if $b_C \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ **or** $b_C \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  $S_{C,b_C} \leftarrow S_{C,b_C} + \widehat{S}_{A,b_A} S_{B,b_B} P_{BC,r}$ {b_C is admissible}
else if $b_A \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$ **and** $b_C \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  $\chi_t C \chi_s \leftarrow \chi_t C \chi_s + \chi_t A V_{B,s} S_{B,b_B} W^*_{B,r}$ {b_A and b_C are inadmissible leaves}
else
  for $s' \in \operatorname{sons}^+(s)$, $r' \in \operatorname{sons}^+(r)$ **do**
    $b'_B \leftarrow (s', r')$; $S_{B,b'_B} \leftarrow 0$ {Set up auxiliary blocks b'_B}
  end for;
  Split(b_B, V_B, W_B, S_B); {Algorithm 37}
  for $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$, $r' \in \operatorname{sons}^+(r)$ **do**
    ProjectedMulRec($t', s', r', P_{CA}, P_{AB}, P_{BC}, A, B, C$) {Algorithm 40}
  end for
end if

If b_C is admissible, we can use the matrix forward transformation to get a "coarse" representation of the block of $\chi_t A \chi_s$ corresponding to b_A and compute the product directly.

If b_A and b_C are inadmissible leaves, Definition 3.18 yields that t, s and r are leaves of the respective cluster trees, so we expand $\chi_s B \chi_r$ and compute the product explicitly.

Otherwise, i.e., if b_A and b_C are not admissible and at least one of them is subdivided, we split b_B into auxiliary blocks, initialize them by using Algorithm 37 and proceed by using Algorithm 40 recursively.

Algorithm 43 takes care of the case that b_C is admissible, i.e., that b_C is an admissible leaf of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or an auxiliary block, i.e., not an element of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$. In this case, we are interested in adding $V_{C,t}^* A \chi_s B W_{C,r}$ to the coupling matrix S_{C,b_C}.

Algorithm 43. Projected matrix multiplication, b_C is admissible.

```
procedure ProjectedMulAdmC(t, s, r, P_CA, P_AB, P_BC, A, B, var C);
b_A ← (t, s); b_B ← (s, r); b_C ← (t, r);
if b_A ∈ L⁻_{A,I×J} and b_B ∈ L⁻_{B,J×K} then
  S_{C,b_C} ← S_{C,b_C} + V*_{C,t} A χ_s B W_{C,r}      {b_A and b_B are inadmissible leaves}
else
  for t' ∈ sons⁺(t), r' ∈ sons⁺(r) do
    b'_C ← (t', r'); S_{C,b'_C} ← 0                     {Set up auxiliary matrices blocks b'_C}
  end for;
  for t' ∈ sons⁺(t), s' ∈ sons⁺(s), r' ∈ sons⁺(r) do
    ProjectedMulRec(t', s', r', P_CA, P_AB, P_BC, A, B, C)      {Algorithm 40}
  end for;
  Collect(b_C, V_C, W_C, S_C)                           {Algorithm 35}
end if
```

If b_A or b_B are admissible, Algorithm 41 or 42 are responsible.

If b_A and b_B are inadmissible leaves, Definition 3.18 implies that t, s and r are leaf clusters, and we compute the product directly.

Otherwise, i.e., if b_A and b_B are not admissible and at least one of them is subdivided, we split b_C into auxiliary blocks, which are initialized by zero. We use Algorithm 40 to compute all subblocks and then apply Algorithm 35 to combine them and perform the update of S_{C,b_C}.

Finally, we consider the case that neither b_A, b_B or b_C are admissible. This means that each of these blocks can either be an inadmissible leaf or further subdivided. If all blocks are inadmissible leaves, we compute the update of $\chi_t C \chi_r$ directly. Otherwise, at least one of the blocks has to be subdivided, and we apply Algorithm 40 recursively to handle all of the subblocks.

The final Algorithm 45 performs the computation $C \leftarrow \Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},V_C,W_C}(C + AB)$. It uses the cluster basis product Algorithm 13 to prepare the cluster operators $P_{CA} = (P_{CA,t})_{t\in\mathcal{T}_\mathcal{I}}$, $P_{AB} = (P_{AB,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $P_{BC} = (P_{BC,r})_{r\in\mathcal{T}_\mathcal{K}}$ and the matrix forward transformation Algorithm 36 to prepare the families $\widehat{S}_A = (\widehat{S}_{A,b})_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}$ and $\widehat{S}_B = (\widehat{S}_{B,b})_{b\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}}$. Then it uses the recursive multiplication Algorithm 40 to perform the central part of the operation. Afterwards, the intermediate results stored

Algorithm 44. Projected matrix multiplication, b_A, b_B and b_C are not admissible.

procedure ProjectedMulAdmC($t, s, r, P_{CA}, P_{AB}, P_{BC}, A, B$, **var** C);
$b_A \leftarrow (t, s)$; $b_B \leftarrow (s, r)$; $b_C \leftarrow (t, r)$;
if $b_A \in \mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$ **and** $b_B \in \mathcal{L}^-_{B,\mathcal{J}\times\mathcal{K}}$ **and** $b_C \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}$ **then**
  $\chi_t C \chi_r \leftarrow \chi_t C \chi_r + \chi_t A \chi_s B \chi_r$ {b_A, b_B and b_C are inadmissible leaves}
else
  for $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$, $r' \in \operatorname{sons}^+(r)$ **do**
    ProjectedMulRec($t', s', r', P_{CA}, P_{AB}, P_{BC}, A, B, C$) {Algorithm 40}
  end for
end if

Algorithm 45. Projected matrix-matrix multiplication, computes the best approximation of $C \leftarrow C + AB$ in the space $\mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{J}}, V_C, W_C)$.

procedure ProjectedMul(A, B, **var** C);
$r_{\mathcal{I}} \leftarrow \operatorname{root}(\mathcal{T}_{\mathcal{I}})$; $r_{\mathcal{J}} \leftarrow \operatorname{root}(\mathcal{T}_{\mathcal{J}})$; $r_{\mathcal{K}} \leftarrow \operatorname{root}(\mathcal{T}_{\mathcal{K}})$;
ClusterBasisProduct($r_{\mathcal{I}}, V_C, V_A, P_{CA}$); {Algorithm 13}
ClusterBasisProduct($r_{\mathcal{J}}, W_A, V_B, P_{AB}$); {Algorithm 13}
ClusterBasisProduct($r_{\mathcal{K}}, W_B, W_C, P_{BC}$); {Algorithm 13}
MatrixForward(root($\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$), $V_C, V_B, P_{CA}, P_{AB}, A, \widehat{S}_A$); {Algorithm 36}
MatrixForward(root($\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$), $W_A, W_C, P_{AB}, P_{BC}, B, \widehat{S}_B$); {Algorithm 36}
for $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ **do**
  $\widehat{S}_{C,b} \leftarrow 0$ {Initialize $\widehat{S}_C$}
end for
ProjectedMulRec($r_{\mathcal{I}}, r_{\mathcal{J}}, r_{\mathcal{K}}, P_{CA}, P_{AB}, P_{BC}, A, B, C$); {Algorithm 40}
MatrixBackward(root($\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$), $V_A, W_B, P_{CA}, P_{BC}, \widehat{S}_C, C$) {Algorithm 38}

in the family $\widehat{S}_C = (\widehat{S}_{C,b})_{b\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}}$ have to be added to C in order to complete the computation. This is done by the matrix backward transformation Algorithm 38.

Complexity analysis

We base the analysis of the complexity of the projected matrix-matrix multiplication on the triples (t, s, r) of clusters passed as parameters to the recursive Algorithm 40. Since it uses Algorithms 41, 42, 43 and 44 to accomplish the computation, and since each of these algorithms only uses sums and products of small matrices, we can bound the total complexity once we have bounded the number of cluster triples (t, s, r) passed to Algorithm 40.

Let us consider the situations in which this algorithm is called. The first call occurs in Algorithm 45 with the triple $(t, s, r) = (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}), \operatorname{root}(\mathcal{T}_{\mathcal{K}}))$ corresponding to the roots of the cluster trees $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{K}}$. Taking a close look at the Algorithms 41,

42, 43 and 44 reveals that a recursive call to the dispatch Algorithm 40 takes place for the triples (t', s', r') with $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ and $r' \in \operatorname{sons}^+(r)$ if, and only if, at least two of the blocks (t, s), (s, r) and (t, r) are inadmissible and at least one of them is subdivided.

We define the subsets

$$\mathcal{S}_A := \mathcal{S}_{A,\mathcal{I}\times\mathcal{J}} = \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{A,\mathcal{I}\times\mathcal{J}}, \tag{7.18a}$$
$$\mathcal{S}_B := \mathcal{S}_{B,\mathcal{J}\times\mathcal{K}} = \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}} \setminus \mathcal{L}_{B,\mathcal{J}\times\mathcal{K}}, \tag{7.18b}$$
$$\mathcal{S}_C := \mathcal{S}_{C,\mathcal{I}\times\mathcal{K}} = \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}_{C,\mathcal{I}\times\mathcal{K}}, \tag{7.18c}$$
$$\mathcal{I}_A := \mathcal{I}_{A,\mathcal{I}\times\mathcal{J}} = \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}, \tag{7.18d}$$
$$\mathcal{I}_B := \mathcal{I}_{B,\mathcal{J}\times\mathcal{K}} = \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}} \setminus \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}, \tag{7.18e}$$
$$\mathcal{I}_C := \mathcal{I}_{C,\mathcal{I}\times\mathcal{K}} = \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}, \tag{7.18f}$$

of subdivided and inadmissible blocks. The case that (t, s) is not a leaf and (s, r) is inadmissible, e.g., can now be characterized by $(t, s) \in \mathcal{S}_A$ and $(s, r) \in \mathcal{I}_B$.

We organize the triples (t, s, r) for which Algorithm 40 is called in a tree structure, the *call tree* $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$. It is uniquely defined as the minimal tree with root

$$\operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}) := (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}), \operatorname{root}(\mathcal{T}_{\mathcal{K}}))$$

and a father-son relation given by

$$\operatorname{sons}(t, s, r) := \begin{cases} \operatorname{sons}^+(t) \times \operatorname{sons}^+(s) \times \operatorname{sons}^+(r) & \text{if } (t, s) \in \mathcal{S}_A, (s, r) \in \mathcal{I}_B, \\ & \text{or } (t, s) \in \mathcal{I}_A, (s, r) \in \mathcal{S}_B, \\ & \text{or } (s, r) \in \mathcal{S}_B, (t, r) \in \mathcal{I}_C, \\ & \text{or } (s, r) \in \mathcal{I}_B, (t, r) \in \mathcal{S}_C, \\ & \text{or } (t, r) \in \mathcal{I}_C, (t, s) \in \mathcal{S}_A, \\ & \text{or } (t, r) \in \mathcal{S}_C, (t, s) \in \mathcal{I}_A, \\ \emptyset & \text{otherwise,} \end{cases} \tag{7.19}$$

for all triples $(t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ it contains. Since the concept of *sparsity* has served us well in the analysis of block cluster trees, we aim to establish a similar property for the call tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$.

Lemma 7.18 (Sparsity of the call tree). *Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ be C_{sp}-sparse admissible block cluster trees, and let*

$$C_{\mathcal{I}}(t) := \{(s, r) \in \mathcal{T}_{\mathcal{J}} \times \mathcal{T}_{\mathcal{K}} \ : \ (t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$
$$C_{\mathcal{J}}(s) := \{(t, r) \in \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{K}} \ : \ (t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} \quad \text{for all } s \in \mathcal{T}_{\mathcal{J}},$$
$$C_{\mathcal{K}}(r) := \{(t, s) \in \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{J}} \ : \ (t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} \quad \text{for all } r \in \mathcal{T}_{\mathcal{K}}.$$

Then we have

$$\#C_{\mathcal{I}}(t), \ \#C_{\mathcal{J}}(s), \ C_{\mathcal{K}}(r) \leq 3C_{\mathrm{sp}}^2, \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}}, s \in \mathcal{T}_{\mathcal{J}}, r \in \mathcal{T}_{\mathcal{K}}.$$

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. We now prove

$$\begin{aligned} C_{\mathcal{I}}(t) \subseteq C'_{\mathcal{I}}(t) := \{&(s,r) \,:\, (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}\} \\ &\cup \{(s,r) \,:\, (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, (t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}\} \\ &\cup \{(s,r) \,:\, (t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\}. \end{aligned}$$

Let $(s,r) \in C_{\mathcal{I}}(t)$. If $(t,s,r) = \operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}})$, we have $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$, $s = \operatorname{root}(\mathcal{T}_{\mathcal{J}})$ and $r \in \operatorname{root}(\mathcal{T}_{\mathcal{K}})$, which implies $(t,s) = \operatorname{root}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s,r) = \operatorname{root}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$.

Otherwise, i.e., if $(t,s,r) \neq \operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}})$, the definition of the call tree implies that we can find a father triple $(t^+,s^+,r^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $(t,s,r) \in \operatorname{sons}(t^+,s^+,r^+)$. Due to

$$\mathcal{S}_A, \mathcal{I}_A \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, \quad \mathcal{S}_B, \mathcal{I}_B \subseteq \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, \quad \mathcal{S}_C, \mathcal{I}_C \subseteq \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},$$

the definition (7.19) yields either $(t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ or $(s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $(t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or $(t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ and $(t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, i.e., (s,r) is contained in $C'_{\mathcal{I}}(t)$.

Since $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ are C_{sp}-sparse, we find

$$\begin{aligned} &\#\{(s,r) \,:\, (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}\} \\ &\quad = \#\{(s,r) \,:\, s \in \operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t), r \in \operatorname{row}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, s)\} \\ &\quad \leq \sum_{s\in\operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}},t)} \#\operatorname{row}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, s) \leq C_{\mathrm{sp}}^2, \\ &\#\{(s,r) \,:\, (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, (t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}\} \\ &\quad = \#\{(s,r) \,:\, r \in \operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, t), s \in \operatorname{col}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r)\} \\ &\quad \leq \sum_{r\in\operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},t)} \#\operatorname{col}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r) \leq C_{\mathrm{sp}}^2, \\ &\#\{(s,r) \,:\, (t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\} \\ &\quad = \#\{(s,r) \,:\, r \in \operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, t), s \in \operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)\} \\ &\quad \leq \sum_{r\in\operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},t)} \#\operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t) \leq C_{\mathrm{sp}}^2, \end{aligned}$$

and can conclude

$$\#C_{\mathcal{I}}(t) \leq \#C'_{\mathcal{I}}(t) \leq 3C_{\mathrm{sp}}^2.$$

The same arguments can be applied to prove bounds for $C_{\mathcal{J}}$ and $C_{\mathcal{K}}$. □

In order to prove a bound for the complexity, we only have to bound the number of operations required for one call to the dispatch Algorithm 40, i.e., for one triple $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$.

Theorem 7.19 (Complexity). *Let K_A, K_B, K_C, L_A, L_B and L_C be the rank distributions of V_A, V_B, V_C, W_A, W_B and W_C, respectively, and let $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(k_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$, $(k_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(l_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $(l_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ and $(l_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$ be defined as in (3.16) and (3.18). Let*

$$\hat{k}_t := \max\{k_{C,t}, k_{A,t}\} \quad \text{for all } t \in \mathcal{T}_\mathcal{I}, \tag{7.20}$$

$$\hat{l}_s := \max\{l_{A,s}, k_{B,s}\} \quad \text{for all } s \in \mathcal{T}_\mathcal{J}, \tag{7.21}$$

$$\hat{m}_r := \max\{l_{B,r}, l_{C,r}\} \quad \text{for all } r \in \mathcal{T}_\mathcal{K}. \tag{7.22}$$

Algorithm 45 requires not more than

$$\begin{aligned}
&4\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} \hat{l}_s^3 + \sum_{r\in\mathcal{T}_\mathcal{K}} \hat{m}_r^3\Big)\\
&\quad + 2\Big(\sum_{(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} (\hat{k}_t^3 + \hat{l}_s^3) + \sum_{(s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}} (\hat{l}_s^3 + \hat{m}_r^3) + \sum_{(t,r)\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}} (\hat{k}_t^3 + \hat{m}_r^3)\Big)\\
&\quad + 3 \sum_{(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}} (\hat{k}_t^3 + \hat{l}_s^3 + \hat{m}_r^3).
\end{aligned}$$

operations. If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ are C_{sp}-sparse, this can be bounded by

$$(4 + 4C_{\mathrm{sp}} + 9C_{\mathrm{sp}}^2)\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} \hat{l}_s^3 + \sum_{r\in\mathcal{T}_\mathcal{K}} \hat{m}_r^3\Big).$$

If K_A, K_B, K_C, L_A, L_B and L_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ and $\mathcal{T}_\mathcal{K}$ are $(C_{\mathrm{rc}}, \alpha, \beta, r, \xi)$-regular, the number of operations is in $\mathcal{O}((\alpha+\beta)^{2r}(n_\mathcal{I} + n_\mathcal{J} + n_\mathcal{K}))$.

Proof. The preparation of the cluster operators P_{CA}, P_{AB} and P_{BC} by Algorithm 13 requires not more than

$$4\Big(\sum_{t\in\mathcal{T}_\mathcal{I}} \hat{k}_t^3 + \sum_{s\in\mathcal{T}_\mathcal{J}} \hat{l}_s^3 + \sum_{r\in\mathcal{T}_\mathcal{K}} \hat{m}_r^3\Big)$$

operations due to Lemma 5.13.

The computation of $\widehat{S}_A$ and $\widehat{S}_B$ by Algorithm 36 and the handling of $\widehat{S}_C$ by Algorithm 38 requires not more than

$$2\Big(\sum_{(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} (\hat{k}_t^3 + \hat{l}_s^3) + \sum_{(s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}} (\hat{l}_s^3 + \hat{m}_r^3) + \sum_{(t,r)\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}} (\hat{k}_t^3 + \hat{m}_r^3)\Big)$$

operations due to the Lemmas 7.3 and 7.5.

This leaves only Algorithm 40 to be analyzed. In order to bound the total complexity, we have to find bounds for the number of operations required for each triple $(t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ of the call tree.

Let $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$. Algorithm 40 uses one of the Algorithms 41, 42, 43 and 44 to handle this triple.

Let us consider Algorithm 41. If b_B is admissible, it requires

$$2(\#K_{A,t})(\#L_{A,s})(\#K_{B,s}) + 2(\#K_{A,t})(\#K_{B,s})(\#L_{B,r}) \leq 2\hat{k}_t\hat{l}_s^2 + 2\hat{k}_t\hat{l}_s\hat{m}_r$$

operations to update $\widehat{S}_{C,b_C}$. Otherwise, if b_C is admissible, it uses

$$2(\#K_{C,t})(\#K_{A,t})(\#L_{A,s}) + 2(\#K_{C,t})(\#L_{A,s})(\#L_{C,r}) \leq 2\hat{k}_t^2\hat{l}_s + 2\hat{k}_t\hat{l}_s\hat{m}_r$$

operations to update S_{C,b_C} directly. Otherwise, if b_B and b_C are inadmissible leaves, the algorithm needs

$$2(\#\hat{t})(\#K_{A,t})(\#L_{A,s}) + 2(\#\hat{t})(\#L_{A,s})(\#\hat{s}) + 2(\#\hat{t})(\#\hat{s})(\#\hat{r}) \leq 2\hat{k}_t^2\hat{l}_s + 2\hat{k}_t\hat{l}_s^2 + 2\hat{k}_t\hat{l}_s\hat{m}_r$$

operations to update $\chi_t C \chi_s$. Otherwise, i.e., if b_B and b_C are not admissible and one of these blocks is subdivided, auxiliary blocks are created by using Algorithm 37, which requires not more than $2\hat{k}_t^2\hat{l}_s + 2\hat{k}_t\hat{l}_s^2$ operations due to Lemma 7.4, before proceeding to the sons in the call tree by recursion. We can conclude that the number of operations performed for one triple $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ by Algorithm 41 is bounded by

$$2\hat{k}_t^2\hat{l}_s + 2\hat{k}_t\hat{l}_s^2 + 2\hat{k}_t\hat{l}_s\hat{m}_r.$$

Using (5.9) and the elementary inequality

$$\begin{aligned} xyz &= \frac{1}{6}\left((2xy)z + (2xz)y + (2yz)x\right) \\ &\leq \frac{1}{6}(x^2z + y^2z + x^2y + z^2y + y^2x + z^2x) \\ &\leq \frac{1}{6}(x^3 + z^3 + y^3 + z^3 + x^3 + y^3) = \frac{1}{3}(x^3 + y^3 + z^3) \end{aligned} \tag{7.23}$$

for all $x, y, z \in \mathbb{R}_{\geq 0}$, we get the bound

$$2\hat{k}_t^2\hat{l}_s + 2\hat{k}_t\hat{l}_s^2 + 2\hat{k}_t\hat{l}_s\hat{m}_r \leq \frac{8}{3}\hat{k}_t^3 + \frac{8}{3}\hat{l}_s^3 + \frac{2}{3}\hat{m}_r^3 < 3(\hat{k}_t^3 + \hat{l}_s^3 + \hat{m}_r^3).$$

By similar arguments, we can prove that the remaining Algorithms 42, 43 and 44 also require not more than $3(\hat{k}_t^3 + \hat{l}_s^3 + \hat{m}_r^3)$ operations per triple.

Let now $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ be C_{sp}-sparse. Due to Lemma 7.18, we can bound the total number of operations for Algorithm 40 by

$$\begin{aligned} &\sum_{(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}} 3(\hat{k}_t^3 + \hat{l}_s^3 + \hat{m}_r^3) \\ &\quad\leq 3\sum_{t\in\mathcal{T}_{\mathcal{I}}} \hat{k}_t^3 \#C_{\mathcal{I}}(t) + 3\sum_{s\in\mathcal{T}_{\mathcal{J}}} \hat{l}_s^3 \#C_{\mathcal{J}}(s) + 3\sum_{r\in\mathcal{T}_{\mathcal{K}}} \hat{m}_r^3 \#C_{\mathcal{K}}(r) \end{aligned}$$

$$\leq 9C_{\mathrm{sp}}^2\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}}\hat{k}_t^3+\sum_{s\in\mathcal{T}_{\mathcal{J}}}\hat{l}_s^3+\sum_{r\in\mathcal{T}_{\mathcal{K}}}\hat{m}_r^3\Big),$$

while Lemmas 7.3 and 7.5 yield the bound

$$\begin{aligned}2C_{\mathrm{sp}}\Big(&\sum_{t\in\mathcal{T}_{\mathcal{I}}}\hat{k}_t^3+\sum_{s\in\mathcal{T}_{\mathcal{J}}}\hat{l}_s^3+\sum_{s\in\mathcal{T}_{\mathcal{J}}}\hat{l}_s^3+\sum_{r\in\mathcal{T}_{\mathcal{K}}}\hat{m}_r^3+\sum_{t\in\mathcal{T}_{\mathcal{I}}}\hat{k}_t^3+\sum_{r\in\mathcal{T}_{\mathcal{K}}}\hat{m}_r^3\Big)\\ &\leq 4C_{\mathrm{sp}}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}}\hat{k}_t^3+\sum_{s\in\mathcal{T}_{\mathcal{J}}}\hat{l}_s^3+\sum_{r\in\mathcal{T}_{\mathcal{K}}}\hat{m}_r^3\Big).\end{aligned}$$

We can complete the proof using the Lemmas 5.12, 3.45 and 3.48. □

This is a quite surprising result: although the matrix-matrix multiplication involves non-trivial interactions between all levels of the block cluster trees, it can be carried out in *linear* complexity.

Remark 7.20 (Auxiliary storage). Algorithm 45 requires auxiliary storage for the matrix families $\widehat{S}_A$, $\widehat{S}_B$ and $\widehat{S}_C$. As in Lemma 3.38, we can bound the storage requirements by

$$\sum_{(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}(\hat{k}_t^2+\hat{l}_s^2)+\sum_{(s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}}(\hat{l}_s^2+\hat{m}_r^2)+\sum_{(t,r)\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}}(\hat{k}_t^2+\hat{m}_r^2).$$

If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ are C_{sp}-sparse, this can be bounded by

$$2C_{\mathrm{sp}}\Big(\sum_{t\in\mathcal{T}_{\mathcal{I}}}\hat{k}_t^2+\sum_{s\in\mathcal{T}_{\mathcal{J}}}\hat{l}_s^2+\sum_{r\in\mathcal{T}_{\mathcal{K}}}\hat{m}_r^2\Big),$$

and if the rank distributions are $(C_{\mathrm{bn}},\alpha,\beta,r,\xi)$-bounded and if $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{K}}$ are $(C_{\mathrm{rc}},\alpha,\beta,r,\xi)$-regular, the storage requirements are in $\mathcal{O}((\alpha+\beta)^r(n_{\mathcal{I}}+n_{\mathcal{J}}+n_{\mathcal{K}}))$.

By arranging the recursion in a manner similar to that suggested in Remark 7.10, we can significantly reduce the storage requirements for $\widehat{S}_A$ and $\widehat{S}_B$: in a first phase we perform all operations for admissible blocks b_A. By using an outer loop over $\operatorname{sons}(b_A)$ and an inner loop over $\operatorname{sons}^+(r)$, we can simultaneously compute the matrix forward transformation for A. In a second phase we perform the remaining operations. By using an outer loop over $\operatorname{sons}(b_B)$ and an inner loop over $\operatorname{sons}^+(t)$, we can simultaneously compute the matrix forward transformation for B. □

7.8 Exact matrix-matrix multiplication

The projection $\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},V_C,W_C}M=C+\Pi_{\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},V_C,W_C}(AB)$ of the matrix $M=C+AB$ can be computed efficiently by the recursive Algorithm 45. The quality of

the approximation depends on how well the block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ and the cluster bases V_C and W_C match the inherent structure of the product AB.

We now investigate a different approach: we construct an admissible block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ and nested cluster bases V_M and W_M such that $M = C + AB \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$ holds, i.e., that the product can be represented *exactly* as an $\mathcal{H}^2$-matrix.

In order to reach a meaningful result, we require that the block cluster trees $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ are admissible and C_{sp}-sparse.

Let $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$, and let $b_A := (t,s)$, $b_B := (s,r)$ and $b_C := (t,r)$. We are looking for a block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ and cluster bases V_M and W_M which ensure that all intermediate products of the form $\chi_t M \chi_r = \chi_t A \chi_s B \chi_r$ can be expressed exactly in the space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$, since this implies that also the sum of all intermediate products, i.e., the product AB, can be expressed in this space.

We assume that we have chosen a block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ for M and that b_C is one of its admissible leaves, i.e., that $b_C \in \mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}$ holds (inadmissible leaves of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ are not interesting, since they are not limited by V_M and W_M).

In order to gain some insight into the proper choices for $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, V_M and W_M, we investigate three special cases.

Case 1: b_B is admissible

If b_B is admissible, we find

$$\chi_s B \chi_r = V_{B,s} S_{B,b_B} W^*_{B,r}$$

and have to ensure that V_M and W_M can represent the block

$$\chi_t A \chi_s B \chi_r = \chi_t A \chi_s V_{B,s} S_{B,b_B} W^*_{B,r}.$$

If b_A is admissible, this is not a problem, since we find

$$\chi_t A \chi_s B \chi_r = V_{A,t} S_{A,b_A} W^*_{A,s} V_{B,s} S_{B,b_B} W^*_{B,r} = V_{A,t} S_{A,b_A} P_{AB,s} S_{B,b_B} W^*_{B,r}$$

and observe that the cluster bases V_A and W_B are sufficient.

If b_A is inadmissible, we have to be able to express terms of the form $\chi_t A \chi_s V_{B,s} = \chi_t A V_{B,s}$ in the basis V_M, i.e., we require that there is a matrix $R_{V,t,s}$ with

$$\chi_t A V_{B,s} = V_{M,t} R_{V,t,s}, \tag{7.24}$$

since this implies

$$\chi_t A \chi_s B \chi_r = \chi_t A \chi_s V_{B,s} S_{B,b_B} W^*_{B,r} = V_{M,t} R_{V,t,s} S_{B,b_B} W^*_{B,r},$$

i.e., the product can be expressed by V_M and W_B. Since b_A is not admissible, we can infer $b_A \in \mathcal{I}_{A,\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, i.e., $s \in \mathrm{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)$. If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse, we therefore have to ensure (7.24) for not more than C_{sp} clusters s.

In order to determine the induced row cluster basis $V_M = (V_{M,t})_{t\in\mathcal{T}_I}$, we introduce the set

$$\begin{aligned}\operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t) &:= \{s \in \mathcal{T}_\mathcal{J} \,:\, (t,s) \in \mathcal{T}_{A,I\times\mathcal{J}} \setminus \mathcal{L}^+_{A,I\times\mathcal{J}}\}\\ &= \operatorname{row}(\mathcal{T}_{A,I\times\mathcal{J}}, t) \setminus \operatorname{row}^+(\mathcal{T}_{A,I\times\mathcal{J}}, t)\end{aligned}$$

(compare Definition 3.29 and 3.40) and can define a row cluster basis satisfying our requirements:

Definition 7.21 (Induced row basis). Let all index sets in the rank distributions $K_A = (K_{A,t})_{t\in\mathcal{T}_I}$, $K_B = (K_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $K_C = (K_{C,t})_{t\in\mathcal{T}_I}$ be disjoint. We define the *induced row cluster basis* for the multiplication by $V_M := (V_{M,t})_{t\in\mathcal{T}_I}$ by

$$V_{M,t} := \begin{pmatrix} V_{C,t} & V_{A,t} & \chi_t A V_{B,s_1} & \dots & \chi_t A V_{B,s_\sigma}\end{pmatrix} \tag{7.25}$$

for all $t \in \mathcal{T}_I$, where $\sigma := \#\operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t)$ is the number of inadmissible blocks connected to t and $\{s_1,\dots,s_\sigma\} = \operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t)$ are the corresponding column clusters. The rank distribution for V_M is given by $K_M := (K_{M,t})_{t\in\mathcal{T}_I}$ with

$$K_{M,t} := K_{C,t} \mathbin{\dot\cup} K_{A,t} \mathbin{\dot\cup} K_{B,s_1} \mathbin{\dot\cup} \cdots \mathbin{\dot\cup} K_{B,s_\sigma}.$$

The definition of $\mathcal{H}^2$-matrices requires the cluster bases to be nested. Fortunately the induced row cluster basis inherits this property from the nested structure of the cluster bases V_A and V_C and from the structure of the block cluster tree $\mathcal{T}_{A,I\times\mathcal{J}}$:

Lemma 7.22 (Nested cluster basis). *The induced row cluster basis V_M is nested.*

Proof. Let $t \in \mathcal{T}_I$ with $\operatorname{sons}(t) \neq \emptyset$. Proving

$$V_{M,t} = \sum_{t'\in\operatorname{sons}(t)} V_{M,t'} E_{M,t'}$$

for suitably chosen transfer matrices $(E_{M,t'})_{t'\in\mathcal{T}_I}$ is equivalent to proving

$$\chi_{t'} V_{M,t} = V_{M,t'} E_{M,t'} \tag{7.26}$$

for all $t' \in \operatorname{sons}(t)$.

We fix $t' \in \operatorname{sons}(t)$ and have to find a matrix $E_{M,t'}$ satisfying equation (7.26).

Let $\sigma := \#\operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t)$ and $\sigma' := \#\operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t')$. For $\{s_1,\dots,s_\sigma\} = \operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t)$ and $\{s'_1,\dots,s'_{\sigma'}\} = \operatorname{row}^-(\mathcal{T}_{A,I\times\mathcal{J}}, t')$, the matrices $V_{M,t'}$ and $\chi_{t'} V_{M,t}$ take the form

$$\begin{aligned}V_{M,t'} &= \begin{pmatrix} V_{C,t'} & V_{A,t'} & \chi_{t'} A V_{B,s'_1} & \dots & \chi_{t'} A V_{B,s'_{\sigma'}}\end{pmatrix},\\ \chi_{t'} V_{M,t} &= \begin{pmatrix} \chi_{t'} V_{C,t} & \chi_{t'} V_{A,t} & \chi_{t'} A V_{B,s_1} & \dots \chi_{t'} A V_{B,s_\sigma}\end{pmatrix}.\end{aligned}$$

Since V_C and V_A are nested, we have

$$\chi_{t'} V_{C,t} = V_{C,t'} E_{C,t'} = V_{M,t'} \begin{pmatrix} E_{C,t'} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \chi_{t'} V_{A,t} = V_{A,t'} E_{A,t'} = V_{M,t'} \begin{pmatrix} 0 \\ E_{A,t'} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{7.27}$$

this defines the first two columns of the transfer matrix $E_{M,t'}$.

Let us now consider $s \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)$. According to our definition, $b_A := (t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ holds. Since $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ is an admissible block cluster tree and since t is not a leaf cluster, Definition 3.18 implies that b_A cannot be an inadmissible leaf, therefore it has to be a subdivided block, i.e., $b_A \in \mathcal{S}_{A,\mathcal{I}\times\mathcal{J}}$ holds.

We have

$$\chi_{t'} \chi_t A V_{B,s} = \chi_{t'} A \chi_s V_{B,s} = \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} A \chi_{s'} V_{B,s} = \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} A V_{B,s'} E_{B,s',s} \tag{7.28}$$

for the long-range transfer matrices $E_{B,s',s}$ from (7.3) and Definition 5.27.

Let $s' \in \operatorname{sons}^+(s)$. Due to $(t,s) \in \mathcal{S}_{A,\mathcal{I}\times\mathcal{J}}$, Lemma 7.17 yields $b'_A := (t',s') \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$.

If b'_A is inadmissible, we have $s' \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t')$, therefore we can find an index $i \in \{1, \ldots, \sigma'\}$ with $s' = s'_i$ and conclude

$$\chi_{t'} A V_{B,s'} E_{B,s',s} = (\chi_{t'} A V_{B,s'_i}) E_{B,s'_i,s} = V_{M,t'} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ E_{B,s_i,s} \\ 0 \\ \vdots \\ 0 \end{pmatrix}; \tag{7.29}$$

the entry of the right matrix is in the row corresponding to $s'_i = s' \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t')$. If b'_A is admissible, we have $s' \in \operatorname{row}^+(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t')$ and can use the factorized representation of $\chi_{t'} A \chi_{s'}$ in order to get

$$\chi_{t'} A V_{B,s'} E_{B,s',s} = V_{A,t'} S_{A,b'_A} W^*_{A,s'} V_{B,s'} E_{B,s',s} = V_{M,t'} \begin{pmatrix} 0 \\ S_{A,b'_A} P_{AB,s'} E_{B,s',s} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{7.30}$$

Accumulating the matrices (7.29) and (7.30) according to (7.28) yields the column of $E_{M,t'}$ corresponding to s. Combining the columns yields $E_{M,t'}$ and thus proves that V_M is nested. □

Our complexity estimates rely on bounds for the rank distributions of the cluster bases. Using standard assumptions we can establish these bounds also for the induced cluster basis V_M:

Lemma 7.23 (Bounded cluster basis). *Let K_A, K_B and K_C be $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded, let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse, and let $C_{\mathrm{cr}} \in \mathbb{R}$ be a constant satisfying $c_{\mathcal{J}} \leq C_{\mathrm{cr}} c_{\mathcal{I}}$ (as usual with $c_{\mathcal{I}} = \#\mathcal{T}_{\mathcal{I}}$ and $c_{\mathcal{J}} = \#\mathcal{T}_{\mathcal{J}}$). Then the rank distribution K_M corresponding to the induced row cluster basis is $(\hat{C}_{\mathrm{bn}}, \hat{\alpha}, \hat{\beta}, r, \xi)$-bounded with*

$$\hat{C}_{\mathrm{bn}} := C_{\mathrm{bn}}(C_{\mathrm{cr}}C_{\mathrm{sp}} + 2), \quad \hat{\alpha} := \alpha \sqrt[r]{C_{\mathrm{sp}} + 2}, \quad \hat{\beta} := \beta \sqrt[r]{C_{\mathrm{sp}} + 2}.$$

Proof. We introduce the sets

$$\begin{aligned}
\mathcal{D}_{A,\ell} &:= \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{A,t} > (\alpha + \beta(\ell-1))^r\},\\
\mathcal{D}_{B,\ell} &:= \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{B,s} > (\alpha + \beta(\ell-1))^r \text{ for } s \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)\},\\
\mathcal{D}_{C,\ell} &:= \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{C,t} > (\alpha + \beta(\ell-1))^r\},\\
\mathcal{D}_{M,\ell} &:= \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{M,t} > (\hat{\alpha} + \hat{\beta}(\ell-1))^r\}
\end{aligned}$$

for all $\ell \in \mathbb{N}$. Let $\ell \in \mathbb{N}$, and let $t \in \mathcal{D}_{M,\ell}$. According to the definition of $\hat{\alpha}$ and $\hat{\beta}$, this implies

$$(C_{\mathrm{sp}} + 2)(\alpha + \beta\ell)^r < \#K_{M,t} = \#K_{C,t} + \#K_{A,t} + \sum_{s\in\operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}},t)} \#K_{B,s}. \quad (7.31)$$

Due to $\operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t) \subseteq \operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)$, we have $\#\operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t) \leq C_{\mathrm{sp}}$ and find that (7.31) can only hold if at least one of the following holds:

Case 1: $\#K_{C,t} > (\alpha + \beta(\ell-1))^r$. This implies $t \in \mathcal{D}_{C,\ell}$.

Case 2: $\#K_{A,t} > (\alpha + \beta(\ell-1))^r$. This implies $t \in \mathcal{D}_{A,\ell}$.

Case 3: There exists a cluster $s \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)$ with $\#K_{B,s} > (\alpha + \beta(\ell-1))^r$. This implies $t \in \mathcal{D}_{B,\ell}$.

We conclude that

$$\mathcal{D}_{M,\ell} \subseteq \mathcal{D}_{C,\ell} \cup \mathcal{D}_{A,\ell} \cup \mathcal{D}_{B,\ell} \quad (7.32)$$

holds. Since K_A and K_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded, we have

$$\#\mathcal{D}_{C,\ell} \leq C_{\mathrm{bn}}\xi^{-\ell}c_{\mathcal{I}}, \quad \#\mathcal{D}_{A,\ell} \leq C_{\mathrm{bn}}\xi^{-\ell}c_{\mathcal{I}}.$$

For the set $\mathcal{D}_{B,\ell}$, we have

$$\begin{aligned}
\mathcal{D}_{B,\ell} &= \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{B,s} > (\alpha + \beta(\ell-1))^r \text{ for an } s \in \operatorname{row}^-(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)\}\\
&\subseteq \{t \in \mathcal{T}_{\mathcal{I}} : \#K_{B,s} > (\alpha + \beta(\ell-1))^r \text{ for an } s \in \operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, t)\}
\end{aligned}$$

$$
\begin{aligned}
&= \{t \in \mathcal{T}_{\mathcal{I}} \,:\, \#K_{B,s} > (\alpha + \beta(\ell-1))^r \text{ for an } s \in \mathcal{T}_{\mathcal{J}} \text{ with } t \in \operatorname{col}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, s)\} \\
&= \bigcup_{\substack{s\in\mathcal{T}_{\mathcal{J}} \\ \#K_{B,s}>(\alpha+\beta(\ell-1))^r}} \operatorname{col}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, s)
\end{aligned}
$$

and can use the sparsity of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ to find

$$
\begin{aligned}
\#\mathcal{D}_{B,\ell} &\leq C_{\mathrm{sp}} \#\{s \in \mathcal{T}_{\mathcal{J}} \,:\, \#K_{B,s} > (\alpha + \beta(\ell-1))^r\} \\
&\leq C_{\mathrm{sp}} C_{\mathrm{bn}} \xi^{-\ell} c_{\mathcal{J}} \\
&\leq C_{\mathrm{sp}} C_{\mathrm{bn}} C_{\mathrm{cr}} \xi^{-\ell} c_{\mathcal{I}}.
\end{aligned}
$$

Combining the estimates for the cardinalities of $\mathcal{D}_{C,\ell}$, $\mathcal{D}_{A,\ell}$ and $\mathcal{D}_{B,\ell}$ with the inclusion (7.32) yields the required bound for $\#\mathcal{D}_{M,\ell}$. □

Case 2: b_A is admissible

If b_A is admissible, we have

$$
\chi_t A \chi_s = V_{A,t} S_{A,b_A} W_{A,s}^*
$$

and need to ensure that the cluster bases V_M and W_M are chosen in such a way that

$$
\chi_t A \chi_s B \chi_r = V_{A,t} S_{A,b_A} W_{A,s}^* \chi_s B \chi_r
$$

can be expressed. Case 1 already covers the situation that b_B is admissible.

If b_B is not admissible, we have to ensure that the cluster basis W_M is able to express $\chi_r B^* \chi_s W_{A,s}$, i.e., that there is a matrix $R_{W,r,s}$ with

$$
\chi_r B^* \chi_s W_{A,s} = W_{M,r} R_{W,r,s}, \tag{7.33}
$$

since then the product takes the form

$$
\chi_t A \chi_s B \chi_r = V_{A,t} S_{A,b_A} W_{A,s}^* \chi_s B \chi_r = V_{A,t} S_{A,b_A} R_{W,r,s}^* W_{M,r}^*,
$$

i.e., it can be expressed in V_A and W_M.

Since b_B is not admissible, we have $b_B \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, i.e., $s \in \operatorname{col}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r)$, and since $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ is C_{sp}-sparse, we only have to ensure (7.33) for up to C_{sp} clusters s.

We introduce the set

$$
\begin{aligned}
\operatorname{col}^-(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r) &:= \{s \in \mathcal{T}_{\mathcal{J}} \,:\, (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}} \setminus \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}\} \\
&= \operatorname{col}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r) \setminus \operatorname{col}^+(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r)
\end{aligned}
$$

and can now define a cluster basis satisfying our requirements.

Definition 7.24 (Induced column basis). Let all index sets in the rank distributions $L_A = (L_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $L_B = (L_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ and $L_C = (L_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$ be disjoint. We define the *induced column cluster basis* for the multiplication by $W_M := (W_{M,r})_{r\in\mathcal{T}_\mathcal{K}}$ with

$$W_{M,r} := \begin{pmatrix} W_{C,r} & W_{B,r} & \chi_r B^* W_{A,s_1} & \dots & \chi_r B^* W_{A,s_\sigma} \end{pmatrix} \tag{7.34}$$

for all $r \in \mathcal{T}_\mathcal{K}$, where $\sigma := \#\operatorname{col}^-(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r)$ is the number of inadmissible blocks connected to r and $\{s_1, \dots, s_\sigma\} = \operatorname{col}^-(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, r)$ are the corresponding row clusters. The rank distribution for W_M is given by $L_M := (L_{M,r})_{r\in\mathcal{T}_\mathcal{K}}$ with

$$L_{M,r} := L_{C,r} \mathbin{\dot\cup} L_{B,r} \mathbin{\dot\cup} L_{A,s_1} \mathbin{\dot\cup} \dots \mathbin{\dot\cup} L_{A,s_\sigma}.$$

Lemma 7.25. *The induced column cluster basis W_M is nested.*

If the rank distributions L_A, L_B and L_C are $(C_{\mathrm{bn}}, \alpha, \beta, r, \xi)$-bounded and if $c_\mathcal{J} \leq C_{\mathrm{cr}} c_\mathcal{K}$ holds, the rank distribution L_M of the induced column cluster basis is $(\hat{C}_{\mathrm{bn}}, \hat\alpha, \hat\beta, r, \xi)$-bounded with

$$\hat{C}_{\mathrm{bn}} := C_{\mathrm{bn}}(C_{\mathrm{cr}} C_{\mathrm{sp}} + 2), \quad \hat\alpha := \alpha \sqrt[r]{C_{\mathrm{sp}} + 2}, \quad \hat\beta := \beta \sqrt[r]{C_{\mathrm{sp}} + 2}.$$

Proof. Similar to the proofs of the Lemmas 7.22 and 7.23. □

Case 3: b_A and b_B are inadmissible

We cannot handle the case that b_A and b_B are both inadmissible while b_C is admissible, since it leaves us with no information whatsoever about the structure of the product $\chi_t A \chi_s B \chi_r$.

Therefore we have to ensure that we never encounter the situation that b_C is an admissible leaf, but b_A and b_B are inadmissible.

This can be done by choosing the block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ appropriately: we construct $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ based on $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ and subdivide its leaves (t, r) as long as there is a cluster $s \in \mathcal{T}_\mathcal{J}$ such that (t, s) and (s, r) are both inadmissible and at least one of them can be subdivided further. A block in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is considered admissible if it can be expressed by V_M and W_M, i.e., if for each $s \in \mathcal{T}_\mathcal{J}$ with $(t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ at least one of the blocks is admissible.

Definition 7.26 (Induced block cluster tree). Let $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ be the minimal block cluster tree for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{K}$ satisfying the following conditions:

- A block $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is subdivided if it is subdivided in $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or if there exists a cluster $s \in \mathcal{T}_\mathcal{J}$ with $(t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ such that both are inadmissible and at least one of them is subdivided, i.e., the sons of b

in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ are given by

$$\operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}},b) = \begin{cases} \operatorname{sons}^+(t)\times\operatorname{sons}^+(r) & \text{if } b\in\mathcal{S}_C \\ & \text{or there exists } s\in\mathcal{T}_{\mathcal{J}} \\ & \text{with } (t,s)\in\mathcal{S}_A, (s,r)\in\mathcal{I}_B \\ & \text{or } (t,s)\in\mathcal{I}_A, (s,r)\in\mathcal{S}_B, \\ \emptyset & \text{otherwise.} \end{cases} \tag{7.35}$$

- A block $b = (t,r)\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is admissible if and only if for all $s\in\mathcal{T}_{\mathcal{J}}$ with $(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ at least one of these blocks is admissible, i.e.,

$$b=(t,r)\in\mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}} \iff (\forall s\in\mathcal{T}_{\mathcal{J}} : (t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} \wedge (s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}} \Rightarrow (t,s)\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}} \vee (s,r)\in\mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}). \tag{7.36}$$

Then $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is called the *induced block cluster tree* for the multiplication.

Lemma 7.27 (Admissibility). *The induced block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is admissible.*

Proof. In order to prove that $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is admissible, we have to ensure that each inadmissible leaf $b=(t,r)\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ corresponds to leaf clusters t and r.

Let $b=(t,r)\in\mathcal{L}^-_{M,\mathcal{I}\times\mathcal{K}}$ be an inadmissible leaf. According to (7.36), this means that there exists a cluster $s\in\mathcal{T}_{\mathcal{J}}$ satisfying $(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ with $(t,s)\notin\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ and $(s,r)\notin\mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}$, i.e., $(t,s)\in\mathcal{I}_A$ and $(s,r)\in\mathcal{I}_B$.

If $\operatorname{sons}(t)\neq\emptyset$ would hold, we would have $(t,s)\notin\mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}$, and $(t,s)\in\mathcal{I}_A = \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\setminus\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ would imply $(t,s)\in\mathcal{S}_A=\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}\setminus\mathcal{L}_{A,\mathcal{I}\times\mathcal{J}}$. By (7.35), we would get $\operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}},b)\neq\emptyset$, which is impossible, since b is a leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. Therefore we can conclude $\operatorname{sons}(t)=\emptyset$.

We can apply the same reasoning to r in order to prove $\operatorname{sons}(r)=\emptyset$, which implies that $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is indeed an admissible block cluster tree. □

Lemma 7.28 (Sparsity). *If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ are C_{sp}-sparse, the tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ is $C_{\mathrm{sp}}(C_{\mathrm{sp}}+1)$-sparse.*

Proof. Let $t\in\mathcal{T}_{\mathcal{I}}$. We prove

$$\begin{aligned}\operatorname{row}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}},t) \subseteq{}& \operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}},t) \\ &\cup\{r\in\mathcal{T}_{\mathcal{I}} : \text{ there exists } s\in\mathcal{T}_{\mathcal{J}} \text{ with } (t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, (s,r)\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}\}.\end{aligned} \tag{7.37}$$

Let $r\in\operatorname{row}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}},t)$. If $b:=(t,r)=\operatorname{root}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}})$, we have $t=\operatorname{root}(\mathcal{T}_{\mathcal{I}})$ and $r=\operatorname{root}(\mathcal{T}_{\mathcal{K}})$, i.e., $(t,r)=\operatorname{root}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}})\in\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$.

Otherwise, b has to have a father, i.e., a block $b^+ = (t^+, r^+) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ with $b \in \operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b^+)$. If $b^+ \in \mathcal{S}_C$, we have $b \in \operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b^+) = \operatorname{sons}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, b^+)$ and therefore $r \in \operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, t)$.

Otherwise, we can find a cluster $s^+ \in \mathcal{T}_{\mathcal{J}}$ with $b_A^+ := (t^+, s^+) \in \mathcal{S}_A \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B^+ := (s^+, r^+) \in \mathcal{I}_B \subseteq \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ or with $b_A^+ = (t^+, s^+) \in \mathcal{I}_A \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B^+ = (s^+, r^+) \in \mathcal{S}_B \subseteq \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$.

We let $s \in \operatorname{sons}^+(s^+)$ and $b_A := (t, s)$ and $b_B := (s, r)$. Due to $t \in \operatorname{sons}^+(t^+)$ and $r \in \operatorname{sons}^+(r^+)$, we can apply Lemma 7.17 in order to find $b_A \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, which proves (7.37).

This inclusion yields

$$\begin{aligned}\#\operatorname{row}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, t) &\leq \#\operatorname{row}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, t) + \sum_{s\in\operatorname{row}(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}},t)} \#\operatorname{row}(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, s)\\ &\leq C_{\mathrm{sp}} + C_{\mathrm{sp}}^2 = C_{\mathrm{sp}}(C_{\mathrm{sp}} + 1).\end{aligned}$$

A similar argument can be applied to demonstrate

$$\#\operatorname{col}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, r) \leq C_{\mathrm{sp}}(C_{\mathrm{sp}} + 1)$$

and conclude this proof. □

Theorem 7.29 (Exact matrix multiplication). *Let $V_M = (V_{M,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ be the induced row cluster basis from Definition 7.21 and let $W_M = (W_{M,r})_{r\in\mathcal{T}_{\mathcal{K}}}$ be the induced column cluster basis from Definition 7.24. Let $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ be the induced block cluster tree from Definition 7.26.*

Then we have $M = C + AB \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$.

Proof. Due to the construction of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, V_M and W_M, we can directly see that $C \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$ holds. Since $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$ is a matrix space (cf. Remark 3.37), it is sufficient to prove $AB \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$.

We prove the more general statement

$$\chi_t A \chi_s B \chi_r \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M) \tag{7.38}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$ with $(t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $(s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $(t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. For $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$, $s = \operatorname{root}(\mathcal{T}_{\mathcal{J}})$ and $r = \operatorname{root}(\mathcal{T}_{\mathcal{K}})$, this implies the desired property.

Case 1: Let us first consider the case that $b_A := (t, s) \in \mathcal{L}_{A,\mathcal{I}\times\mathcal{J}}^+$ holds, i.e., that b_A is an admissible leaf. We have seen in Case 1 that

$$\chi_t A \chi_s B \chi_r = V_{A,t}(S_{A,b_A} R_{W,r,s}^*) W_{M,r}^* \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M),$$

holds, i.e., the product can be expressed by the induced row and column cluster bases.

Case 2: Let us now consider the case that $b_B := (s, r) \in \mathcal{L}_{B,\mathcal{J}\times\mathcal{K}}^+$ holds, i.e., that b_B is an admissible leaf. According to Case 2, we have

$$\chi_t A \chi_s B \chi_r = V_{M,t}(R_{V,t,s} S_{B,b_B}) W_{B,r}^* \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M),$$

i.e., the product can once more be expressed in the induced cluster bases.

Induction: We prove (7.38) by an induction on the number of descendants of b_A and b_B, i.e., on $\#\operatorname{sons}^*(b_A) + \#\operatorname{sons}^*(b_B) \in \mathbb{N}_{\geq 2}$. Let us first consider $b_A = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $b_B = (s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ with $\#\operatorname{sons}^*(b_A) + \#\operatorname{sons}^*(b_B) = 2$. This implies $\operatorname{sons}^*(b_A) = \{b_A\}$ and $\operatorname{sons}^*(b_B) = \{b_B\}$, i.e., b_A and b_B are leaves. If at least one of them is admissible, we have already established that (7.38) holds.

Otherwise, i.e., if b_A and b_B are inadmissible leaves, (7.36) implies that $b_C := (t, r)$ is also an inadmissible leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, i.e., (7.38) holds trivially.

Let now $n \in \mathbb{N}_{\geq 2}$, and assume that (7.38) holds for all $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$ with $b_A = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $b_B = (s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $b_C = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ satisfying $\#\operatorname{sons}^*(b_A) + \#\operatorname{sons}^*(b_B) \leq n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$ with $b_A = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $b_B = (s, r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $b_C = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ and $\#\operatorname{sons}^*(b_A) + \#\operatorname{sons}^*(b_B) = n + 1$.

If b_A or b_B are admissible leaves, i.e., if $b_A \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ or $b_B \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}$ holds, we can again use Case 1 or Case 2 to conclude that (7.38) holds.

Otherwise, i.e., if b_A and b_B are inadmissible, they cannot both be leaves, since this would imply $\#\operatorname{sons}^*(\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}, b_A) + \#\operatorname{sons}^*(\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}, b_B) = 2 < n + 1$.

Therefore at least one of b_A and b_B has to be subdivided. Definition 7.26 yields $\operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b_C) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(r)$, and we observe

$$\chi_t A \chi_s B \chi_r = \sum_{t' \in \operatorname{sons}^+(t)} \sum_{s' \in \operatorname{sons}^+(s)} \sum_{r' \in \operatorname{sons}^+(r)} \chi_{t'} A \chi_{s'} B \chi_{r'}.$$

For all $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ and $r' \in \operatorname{sons}^+(r)$, Lemma 7.17 yields $(t', s') \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $(s', r') \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$. Since at least one of b_A and b_B is subdivided, we can apply the induction assumption to get

$$\chi_{t'} A \chi_{s'} B \chi_{r'} \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M)$$

for all $t' \in \operatorname{sons}^+(t)$, $s' \in \operatorname{sons}^+(s)$ and $r' \in \operatorname{sons}^+(r)$. According to Remark 3.37, we can conclude

$$\chi_t A \chi_s B \chi_r = \sum_{t' \in \operatorname{sons}^+(t)} \sum_{s' \in \operatorname{sons}^+(s)} \sum_{r' \in \operatorname{sons}^+(r)} \chi_{t'} A \chi_{s'} B \chi_{r'} \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_M, W_M).$$

□

7.9 Numerical experiments

On a smooth surface, the matrices V and K introduced in Section 4.9 correspond to discretizations of asymptotically smooth kernel functions. The products VV, VK, KV and KK of the matrices correspond, up to discretization errors, to the convolutions of these kernel functions, and we expect (cf. [63], Satz E.3.7) that these convolutions

will again be asymptotically smooth. Due to the results of Chapter 4, this means that they can be approximated by polynomials and that the approximation will converge exponentially if the order is increased.

Up to errors introduced by the discretization process, the matrix products VV, KV, VK and KK can be expected to share the properties of their continuous counterparts, i.e., it should be possible to approximate the products in the $\mathcal{H}^2$-matrix space defined by polynomials of a certain order m, and the approximation should converge exponentially as the order increases.

In the first example, we consider a sequence of polygonal approximations of the unit sphere $\Gamma_S = \{x \in \mathbb{R}^3 : \|x\| = 1\}$ consisting of $n \in \{512, 2048, \ldots, 131072\}$ plane triangles.

On each of these grids, we construct orthogonal nested cluster bases by applying Algorithm 16 to the polynomial bases introduced in Sections 4.4 and 4.5. Together with a standard block cluster tree, these cluster bases define $\mathcal{H}^2$-matrix spaces, and we use Algorithm 45 to compute the best approximation of the products VV, KV, VK and KK in these spaces. The results are collected in Table 7.1 (which has been taken from [11]). For each grid, each product and each interpolation order, it contains the

Table 7.1. Multiplying double and single layer potential on the unit sphere.

Oper.	n	$m=2$		$m=3$		$m=4$	
VV	512	2.2	1.5_{-4}	1.0	8.5_{-7}	0.4	conv.
	2048	13.0	2.6_{-4}	32.3	9.6_{-6}	40.8	4.6_{-7}
	8192	66.5	4.6_{-4}	184.2	3.6_{-5}	355.2	1.6_{-6}
	32768	283.9	5.4_{-4}	897.0	4.9_{-5}	1919.2	2.3_{-6}
	131072	1196.8	5.6_{-4}	3625.6	4.7_{-5}	7918.5	3.6_{-6}
KV	512	2.3	2.6_{-3}	1.0	4.9_{-5}	0.4	conv.
	2048	13.8	5.4_{-3}	34.4	1.7_{-4}	41.7	5.9_{-6}
	8192	71.1	1.0_{-2}	196.7	8.5_{-4}	374.8	5.3_{-5}
	32768	304.3	1.6_{-2}	959.9	2.3_{-3}	1982.2	1.4_{-4}
	131072	1257.4	2.3_{-2}	3876.3	4.0_{-3}	8361.8	2.8_{-4}
VK	512	2.3	5.0_{-3}	1.0	7.8_{-6}	0.4	conv.
	2048	14.0	2.1_{-2}	35.6	4.2_{-4}	41.7	1.9_{-5}
	8192	72.7	4.2_{-2}	204.3	2.0_{-3}	395.3	1.3_{-4}
	32768	313.1	7.0_{-2}	1003.3	4.1_{-3}	2098.5	2.6_{-4}
	131072	1323.6	1.1_{-1}	4101.6	7.1_{-3}	8744.8	5.1_{-4}
KK	512	2.2	6.9_{-4}	1.0	9.4_{-6}	0.4	conv.
	2048	12.9	2.9_{-3}	32.4	5.2_{-5}	40.6	4.8_{-6}
	8192	66.4	6.1_{-3}	184.0	2.6_{-4}	354.4	1.8_{-5}
	32768	283.9	1.1_{-2}	894.0	5.5_{-4}	1881.5	3.5_{-5}
	131072	1169.1	1.6_{-2}	3602.0	9.6_{-4}	7839.9	6.8_{-5}

time[1] in seconds for computing the product by Algorithm 45 and the resulting approximation error in the relative operator norm $\|X - AB\|_2 / \|AB\|_2$, which was estimated using a power iteration.

We can see that increasing the interpolation order m will typically reduce the approximation error by a factor of 10, i.e., we observe the expected exponential convergence.

Since we are working with interpolation in three space dimensions, the rank of the cluster bases is bounded by $k = m^3$, i.e., we expect a behaviour like $\mathcal{O}(nk^2) = \mathcal{O}(nm^6)$ in the computation time. Especially for higher interpolation orders and higher problem dimensions, this behaviour can indeed be observed.

In Table 7.2, we investigate the behaviour of the polynomial approximation on the surface $\Gamma_C = \partial[-1,1]^3$ of the cube. Since it is not a smooth manifold, we expect

Table 7.2. Multiplying double and single layer potential on the unit cube.

Oper.	n	$m=2$		$m=3$		$m=4$	
VV	768	2.0	2.9_{-3}	3.8	3.5_{-4}	1.2	conv.
	3072	10.9	3.6_{-3}	32.0	6.3_{-4}	133.7	2.0_{-4}
	12288	49.7	3.8_{-3}	176.7	7.9_{-4}	455.6	2.8_{-4}
	49152	208.6	4.0_{-3}	850.7	8.5_{-4}	1867.4	3.4_{-4}
	196608	833.0	4.0_{-3}	3692.2	8.6_{-4}	7542.0	3.6_{-4}
KV	768	2.2	6.3_{-2}	3.9	5.5_{-3}	1.2	conv.
	3072	11.7	7.0_{-2}	34.9	1.4_{-2}	134.6	3.2_{-3}
	12288	53.4	8.5_{-2}	200.4	1.9_{-2}	470.0	7.2_{-3}
	49152	222.8	8.6_{-2}	978.0	2.1_{-2}	1922.1	8.6_{-3}
	196608	869.8	8.2_{-2}	4245.9	2.1_{-2}	7862.8	9.1_{-3}
VK	768	2.4	1.9_{-1}	3.8	4.4_{-2}	1.2	conv.
	3072	11.8	2.7_{-1}	37.7	1.1_{-1}	134.4	3.9_{-2}
	12288	55.0	3.4_{-1}	214.5	1.6_{-1}	484.0	8.3_{-2}
	49152	232.3	3.7_{-1}	1059.7	2.0_{-1}	2045.5	1.2_{-1}
	196608	930.5	3.7_{-1}	4614.4	2.3_{-1}	8454.1	1.4_{-1}
KK	768	2.0	2.7_{-2}	3.8	5.1_{-3}	1.2	conv.
	3072	11.0	3.5_{-2}	32.6	1.2_{-2}	132.6	5.1_{-3}
	12288	49.7	4.7_{-2}	185.3	1.9_{-2}	454.3	9.4_{-3}
	49152	206.5	5.7_{-2}	903.1	2.4_{-2}	1823.6	1.4_{-2}
	196608	804.8	6.3_{-2}	3945.2	2.7_{-2}	7374.3	1.8_{-2}

that the smoothness of the result of the multiplication, and therefore the speed of convergence, will be reduced. This is indeed visible in the numerical experiments: we no longer observe a convergence like 10^{-m}, but only 2^{-m} for VV and KV and even

[1] On a single 900 MHz UltraSPARC IIIcu processor of a SunFire 6800 computer.

slower convergence for VK and KK. The latter effect might be a consequence of the fact that the column cluster bases used for K involve the normal derivative of the kernel function, which reduces the order of smoothness even further.

Chapter 8
A posteriori matrix arithmetic

In the previous chapter, we have considered two techniques for performing arithmetic operations like matrix addition and multiplication with $\mathcal{H}^2$-matrices.

The first approach computes the best approximation of sums and products in a *prescribed* $\mathcal{H}^2$-matrix space, and this can be accomplished in optimal complexity. Since the a priori construction of suitable cluster bases and block partitions can be very complicated in practical applications, the applicability of this technique is limited.

The second approach overcomes these limitations by constructing $\mathcal{H}^2$-matrix spaces that can represent the *exact* sums and products without any approximation error. Unfortunately, the resulting matrix spaces lead to a very high computational complexity, and this renders them unattractive as building blocks for higher-level operations like the LU factorization or the matrix inversion.

In short, the first approach reaches the *optimal complexity*, while the second approach reaches the *optimal accuracy*. In this chapter, we introduce a third approach that combines the advantages of both: it allows us to construct adaptive cluster bases a posteriori which ensure a prescribed accuracy and it does not lead to excessively high storage requirements.

The new algorithm consists of three parts: first the exact matrix-matrix product is computed in an intermediate representation that is closely related to the left and right semi-uniform matrices introduced in Section 6.1. Since the block cluster tree for the intermediate representation would lead to prohibitively high storage requirements, we do not construct this representation explicitly, but instead approximate it on the fly by a suitable $\mathcal{H}$-matrix format with a coarse block structure. In the last step, this $\mathcal{H}$-matrix is converted into an $\mathcal{H}^2$-matrix by Algorithm 33. The resulting procedure for approximating the matrix-matrix product is sketched in Figure 8.1. In practical applications, it is advisable to combine the second and third step in order to avoid the necessity of storing the entire intermediate $\mathcal{H}$-matrix (cf. Algorithm 33), but for our theoretical investigations, it is better to treat them separately.

The accuracy of these two approximation steps can be controlled, therefore the product can be computed with any given precision. Since the product is computed block by block, the new procedure can be used as a part of inversion and factorization algorithms. The price for its flexibility is an increased algorithmic complexity: the new algorithm does not scale linearly with the number of degrees of freedom, but involves an additional logarithmic factor.

The chapter is organized as follows:

- Section 8.1 introduces the structure of *semi-uniform matrices*, a generalization of the left and right semi-uniform matrices defined in Section 6.1.

- Section 8.2 contains an algorithm for computing the matrix $M_0 := C + AB$ exactly and representing it as a semi-uniform matrix.
- Section 8.3 introduces an algorithm for coarsening the block structure to compute an $\mathcal{H}$-matrix $\tilde{M}$ that approximates M_0.
- Section 8.4 is devoted to the construction of adaptive cluster bases for this $\mathcal{H}$-matrix and the conversion of $\tilde{M}$ into an $\mathcal{H}^2$-matrix approximation of $C + AB$.
- Section 8.5 presents numerical experiments that demonstrate that the new multiplication algorithm ensures the desired accuracy and produces data-sparse approximations of the exact result.

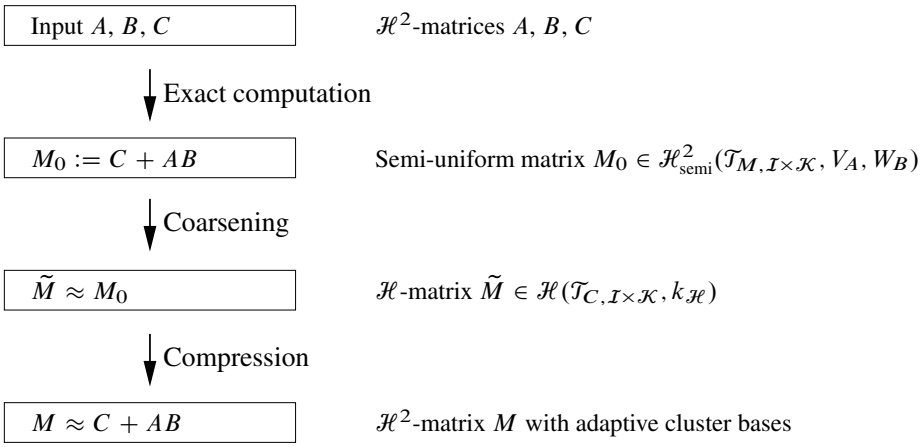


Figure 8.1. A posteriori multiplication.

Assumptions in this chapter: We assume that cluster trees $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{K}}$ for the finite index sets $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$, respectively, are given. Let $n_{\mathcal{I}} := \#\mathcal{I}$, $n_{\mathcal{J}} := \#\mathcal{J}$ and $n_{\mathcal{K}} := \#\mathcal{K}$ denote the number of indices in each of the sets, and let $c_{\mathcal{I}} := \#\mathcal{T}_{\mathcal{I}}$, $c_{\mathcal{J}} := \#\mathcal{T}_{\mathcal{J}}$ and $c_{\mathcal{K}} := \#\mathcal{K}$ denote the number of clusters in each of the cluster trees. Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ be admissible block cluster trees for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{K}}$, and $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{K}}$, respectively.

Let V_A and V_C be nested cluster bases for $\mathcal{T}_{\mathcal{I}}$ with rank distributions K_A and K_C, let W_A and V_B be nested cluster bases for $\mathcal{T}_{\mathcal{J}}$ with rank distributions L_A and K_B, and let W_B and W_C be nested cluster bases for $\mathcal{T}_{\mathcal{K}}$ with rank distributions L_B and L_C.

8.1 Semi-uniform matrices

The new algorithm for approximating the matrix-matrix product can be split into two phases: first an exact representation of the product is computed, then this representation is converted into an $\mathcal{H}^2$-matrix with adaptively-chosen cluster bases.

In Section 7.8, we have seen that the exact product is an $\mathcal{H}^2$-matrix for a modified block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ and modified cluster bases V_M and W_M. In practice, these modified cluster bases V_M and W_M will have very high ranks, i.e., the representation of the exact product as an $\mathcal{H}^2$-matrix will require large amounts of storage (cf. Lemma 7.23 taking into account that $C_{\mathrm{sp}} \geq 100$ holds in important applications).

In order to reduce the storage requirements, we replace the $\mathcal{H}^2$-matrix space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_M, W_M)$ used in Section 7.8 by a new kind of matrix subspace:

Definition 8.1 (Semi-uniform hierarchical matrices). Let $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be matrix. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree, and let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ be nested cluster bases with rank distributions $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $L = (L_s)_{s\in\mathcal{T}_\mathcal{J}}$. X is a *semi-uniform matrix* for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, V and W if there are families $A = (A_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ and $B = (B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ of matrices satisfying $A_b \in \mathbb{R}^{\mathcal{I}\times L_s}_{\hat{t}}$ and $B_b \in \mathbb{R}^{\mathcal{J}\times K_t}_{\hat{s}}$ for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and

$$X = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} (A_b W_s^* + V_t B_b^*) + \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} \chi_t X \chi_s. \tag{8.1}$$

The matrices $A = (A_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ and $B = (B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ are called *left* and *right coefficient matrices*.

Obviously, a matrix X is semi-uniform if and only if it can be split into a sum of a left and a right semi-uniform matrix, i.e., the set of semi-uniform matrices is the sum of the spaces of left and right semi-uniform matrices.

Remark 8.2 (Matrix subspaces). Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, V and W be as in Definition 8.1. The set

$$\mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W) := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} : X \text{ is a semi-uniform matrix for } \mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V \text{ and } W\}$$

is a subspace of $\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

Proof. Since we have to compute sums of matrices in $\mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ during the course of the new multiplication algorithm, we prove the claim constructively instead of simply relying on the fact that $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, *, W)$ and $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, *)$ are subspaces.

Let $X, Y \in \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$, and let $\alpha \in \mathbb{R}$. Let $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ be an admissible leaf. Due to Definition 8.1 and Lemma 5.4, there are matrices $A_{X,b}, A_{Y,b} \in$

$\mathbb{R}^{\mathcal{I}\times L_s}_{\hat{t}}$ and $B_{X,b}, B_{Y,b} \in \mathbb{R}^{\mathcal{J}\times K_t}_{\hat{s}}$ satisfying

$$\begin{aligned}\chi_t(X+\alpha Y)\chi_s &= \chi_t X\chi_s + \alpha\chi_t Y\chi_s = A_{X,b}W_s^* + V_t B_{X,b}^* + \alpha(A_{Y,b}W_s^* + V_t B_{Y,b}^*)\\ &= (A_{X,b}+\alpha A_{Y,b})W_s^* + V_t(B_{X,b}+\alpha B_{Y,b})^*.\end{aligned}$$

Applying this equation to all admissible leaves $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ yields the desired result and proves that we can compute the representation of linear combinations efficiently by forming linear combinations of the left and right coefficient matrices. □

We construct semi-uniform matrices by accumulating blockwise updates, and since these updates can appear for arbitrary blocks of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, not only for leaf blocks, we have to ensure that the result still matches Definition 8.1.

Lemma 8.3 (Restriction). *Let $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$, $A \in \mathbb{R}^{\mathcal{I}\times L_s}_{\hat{t}}$ and $B \in \mathbb{R}^{\mathcal{J}\times K_t}_{\hat{s}}$. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, V and W be as in Definition 8.1. For all $t^* \in \operatorname{sons}^*(t)$ and $s^* \in \operatorname{sons}^*(s)$, we have*

$$\chi_{t^*}(AW_s^* + V_t B^*)\chi_{s^*} = (\chi_{t^*}AF^*_{s^*,s})W^*_{s^*} + V_{t^*}(\chi_{s^*}BE^*_{t^*,t})^*.$$

If $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ holds, this equation implies

$$AW_s^* + V_t B^* \in \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W).$$

Proof. Let $t^* \in \operatorname{sons}^*(t)$ and $s^* \in \operatorname{sons}^*(s)$. Due to Lemma 6.13, we have

$$\begin{aligned}\chi_{t^*}(AW_s^*)\chi_{s^*} &= \chi_{t^*}A(\chi_{s^*}W_s)^* = \chi_{t^*}A(W_{s^*}F_{s^*,s})^* = \chi_{t^*}AF^*_{s^*,s}W^*_{s^*},\\ \chi_{t^*}(V_tB^*)\chi_{s^*} &= \chi_{t^*}V_tB^*\chi_{s^*} = V_{t^*}E_{t^*,t}B^*\chi_{s^*} = V_{t^*}(\chi_{s^*}BE^*_{t^*,t})^*,\end{aligned}$$

and adding both equations proves our first claim.

Now let $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. We have to prove that $AW_s^* + V_tB^*$ is an element of the space $\mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. Let $b := (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and let $\mathcal{T}_b$ be the subtree of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with root b (cf. Definition 6.30). Due to Corollary 3.15, we find that

$$\mathcal{L}_b = \{b^* = (t^*,s^*) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \,:\, b^* \in \operatorname{sons}^*(b)\}$$

describes a disjoint partition

$$\{\hat{b}^* = \hat{t}^* \times \hat{s}^* \,:\, b^* \in \mathcal{L}_b\}$$

of $\hat{b}$, and this implies

$$\begin{aligned}AW_s^* + V_tB^* &= \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b} \chi_{t^*}(AW_s^* + V_tB^*)\chi_{s^*}\\ &= \sum_{b^*=(t^*,s^*)\in\mathcal{L}_b} (\chi_{t^*}AF^*_{s^*,s})W^*_{s^*} + V_{t^*}(\chi_{s^*}BE^*_{t^*,t})^*.\end{aligned}$$

Since all blocks b^* appearing in this sum are leaves of $\mathcal{T}_b$, each of the terms in the sum is an element of $\mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ due to Definition 8.1. According to Remark 8.2, $\mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ is a linear space, so the sum itself is also contained and we conclude $AW_s^* + V_tB^* \in \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$. □

8.2 Intermediate representation

We now consider the computation of the matrix $M_0 := C + AB$ for $A \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$, $B \in \mathbb{R}^{\mathcal{J}\times\mathcal{K}}$ and $C \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}$. As in Section 7.6, we assume that A, B and C are $\mathcal{H}^2$-matrices given in the standard $\mathcal{H}^2$-matrix representation

$$\begin{aligned}
A &= \sum_{b=(t,s)\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}} V_{A,t} S_{A,b} W^*_{A,s} + \sum_{b=(t,s)\in\mathcal{L}^-_{A,\mathcal{I}\times\mathcal{J}}} \chi_t A \chi_s,\\
B &= \sum_{b=(s,r)\in\mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}} V_{B,s} S_{B,b} W^*_{B,r} + \sum_{b=(s,r)\in\mathcal{L}^-_{B,\mathcal{J}\times\mathcal{K}}} \chi_s B \chi_r,\\
C &= \sum_{b=(t,r)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}} V_{C,t} S_{C,b} W^*_{C,r} + \sum_{b=(t,r)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}} \chi_t C \chi_r
\end{aligned}$$

for families $S_A = (S_{A,b})_{b\in\mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}}$, $S_B = (S_{B,b})_{b\in\mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}}$ and $S_C = (S_{C,b})_{b\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}}$ of coupling matrices.

We denote the sets of subdivided blocks of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ by $\mathcal{S}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{S}_{\mathcal{J}\times\mathcal{K}}$ and $\mathcal{S}_{\mathcal{I}\times\mathcal{K}}$ and the sets of inadmissible blocks by $\mathcal{I}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{I}_{\mathcal{J}\times\mathcal{K}}$ and $\mathcal{I}_{\mathcal{I}\times\mathcal{K}}$ (cf. Definition 7.16).

The first step of the new multiplication algorithm consists of representing the *exact* result $M_0 := C + AB$ as a semi-uniform matrix.

As in Section 7.7, we express the operation

$$M_0 \leftarrow C + AB$$

as a sequence

$$\begin{aligned}
M_0 &\leftarrow C,\\
M_0 &\leftarrow M_0 + \chi_{t_1} A \chi_{s_1} B \chi_{r_1},\\
&\vdots\\
M_0 &\leftarrow M_0 + \chi_{t_m} A \chi_{s_m} B \chi_{r_m}
\end{aligned}$$

of updates involving subblocks $\chi_{t_i} A \chi_{s_i}$ of A and $\chi_{s_i} B \chi_{r_i}$ of B. If we can find a good bound for the number of updates and perform each update efficiently, the resulting algorithm is also efficient.

Let us consider one of the updates. We fix $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$, and let $b_A := (t,s)$, $b_B := (s,r)$ and $b_C := (t,r)$. Our goal is to perform the update

$$\chi_t M_0 \chi_r \leftarrow \chi_t M_0 \chi_r + \chi_t A \chi_s B \chi_r \tag{8.2}$$

efficiently. As in Section 7.8, we now investigate three special cases.

Case 1: b_B is admissible

If b_B is admissible, we have

$$\chi_s B \chi_r = V_{B,s} S_{B,b_B} W_{B,r}^*,$$

since B is an $\mathcal{H}^2$-matrix. This means that the product on the right-hand side of (8.2) has the form

$$\chi_t A \chi_s B \chi_r = \chi_t A V_{B,s} S_{B,b_B} W_{B,r}^* = (\chi_t A V_{B,s} S_{B,b_B}) W_{B,r}^*. \tag{8.3}$$

We recall Definition 6.1 and see that the structure of the product is that of a *right semi-uniform matrix*. We can split the computation of the product

$$\chi_t A V_{B,s} S_{B,b_B}$$

into two parts: the challenging part is finding

$$\widehat{A}_{b_A} := \chi_t A V_{B,s} \tag{8.4}$$

efficiently, the remaining multiplication by S_{B,b_B} can be handled directly. Since the matrices $\widehat{A}_{b_A}$ resemble the matrices computed by the matrix forward transformation Algorithm 36, we can adapt its recursive approach: we have

$$\chi_t = \sum_{t' \in \operatorname{sons}^+(t)} \chi_{t'}, \quad V_{B,s} = \sum_{s' \in \operatorname{sons}^+(s)} V_{B,s'} E_{B,s',s}.$$

If $\operatorname{sons}(b_A) \neq \emptyset$, Definition 3.12 implies $\operatorname{sons}(b_A) = \operatorname{sons}^+(t) \times \operatorname{sons}^+(s)$ and we find

$$\begin{aligned} \widehat{A}_{b_A} = \chi_t A V_{B,s} &= \sum_{t' \in \operatorname{sons}^+(t)} \sum_{s' \in \operatorname{sons}^+(s)} \chi_{t'} A V_{B,s'} E_{B,s',s} \\ &= \sum_{b'_A = (t',s') \in \operatorname{sons}(b_A)} \widehat{A}_{b'_A} E_{B,s',s}. \end{aligned} \tag{8.5}$$

Otherwise, i.e., if b_A is a leaf of $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, it can be either an inadmissible leaf, in which case we can compute $\widehat{A}_{b_A}$ directly by using its definition (8.4), or it is an admissible leaf, and we get

$$\widehat{A}_{b_A} = \chi_t A V_{B,s} = \chi_t A \chi_s V_{B,s} = V_{A,t} S_{A,b_A} W_{A,s}^* V_{B,s}.$$

The computation of this product can be split into three steps: the cluster basis product $P_{AB} = (P_{AB,s})_{s \in \mathcal{T}_\mathcal{J}}$ given by $P_{AB,s} := W_{A,s}^* V_{B,s}$ for all $s \in \mathcal{T}_\mathcal{J}$ can be computed efficiently by Algorithm 13, multiplying $P_{AB,s}$ by S_{A,b_A} is straightforward, so we only require a method for multiplying the intermediate matrix $\widehat{Y}_t := S_{A,b_A} W_{A,s}^* V_{B,s} = S_{A,b_A} P_{AB,s}$ with $V_{A,t}$. We could use a block variant of the backward transformation

Algorithm 7 to compute $V_{A,t}\widehat{Y}_t$, but for our application it is simpler and in the general case even more efficient to prepare and store the matrices $(V_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ explicitly instead of using the transfer matrices. Algorithm 46 can be used to compute these matrices efficiently.

Combining Algorithm 13 for the computation of cluster basis products with the block backward transformation and the recursion (8.5) yields Algorithm 47, the *semi-uniform matrix forward transformation*.

Algorithm 46. Expansion of a nested cluster basis.

procedure ExpandBasis(t, **var** V);
if sons(t) $\neq \emptyset$ **then**
  $V_t \leftarrow 0 \in \mathbb{R}^{\hat{t}\times K_t}$;
  for $t' \in$ sons(t) **do**
    ExpandBasis(t', V);
    $V_t \leftarrow V_t + V_{t'}E_{t'}$
  end for
end if

Algorithm 47. Semi-uniform matrix forward transformation for the matrix A.

procedure SemiMatrixForward(b, V_A, V_B, P_{AB}, S_A, A, **var** $\widehat{A}$);
$(t,s) \leftarrow b$;
if sons(b) $\neq \emptyset$ **then**
  $\widehat{A}_b \leftarrow 0$;
  for $b' \in$ sons(b) **do**
    $(t',s') \leftarrow b'$;
    SemiMatrixForward(b', V_A, V_B, P_{AB}, S_A, A, $\widehat{A}$);
    $\widehat{A}_b \leftarrow \widehat{A}_b + \widehat{A}_{b'}E_{B,s',s}$
  end for
else if b is admissible **then**
  $\widehat{Y}_t \leftarrow S_{A,b}P_{AB,s}$;
  $\widehat{A}_b \leftarrow V_{A,t}\widehat{Y}_t$
else
  $\widehat{A}_b \leftarrow \chi_t A V_{B,s}$
end if

This procedure is a variant of the matrix forward transformation Algorithm 36 used in Chapter 7: instead of applying cluster basis matrices from the left and the right and efficiently computing a projection of a matrix block into a given $\mathcal{H}^2$-matrix space, we apply only one cluster basis matrix from the right and end up with a matrix in a semi-uniform matrix space.

Remark 8.4 (Block backward transformation). The computation of the product $V_{A,t}\,\widehat{Y}_t$ is closely related to the computation of the vector $V_{A,t}\,\hat{y}_t$ performed by the backward transformation Algorithm 7 that is introduced in Section 3.7, and generalizing this procedure yields the *block backward transformation* Algorithm 48, the counterpart of the block forward transformation Algorithm 10 introduced in Section 5.2.

Instead of using Algorithm 46 to compute all matrices $V_{A,t}$ and $W_{B,r}$ explicitly, we therefore can also apply the block backward transformation Algorithm 48 to $\widehat{Y}_t$ and $\widehat{Y}_r$ and reduce the amount of auxiliary storage. This will increase the number of operations, but should not hurt the asymptotic complexity. □

Algorithm 48. Block backward transformation.

```
procedure BlockBackwardTransformation(t, V, var Y, Ŷ);
if sons(t) = ∅ then
  Y ← Y + V_t Ŷ_t
else
  for t' ∈ sons(t) do
    Ŷ_t' ← E_t' Ŷ_t;
    BlockBackwardTransformation(t', V, Y, Ŷ)
  end for
end if
```

Case 2: b_A is admissible

If b_A is admissible, the fact that A is an $\mathcal{H}^2$-matrix implies

$$\chi_t A \chi_s = V_{A,t} S_{A,b_A} W_{A,s}^*$$

and we conclude that the product on the right-hand side of (8.2) now satisfies

$$\chi_t A \chi_s B \chi_r = V_{A,t} S_{A,b_A} W_{A,s}^* B \chi_r = V_{A,t} (\chi_r B^* W_{A,s} S_{A,b_A}^*)^*, \tag{8.6}$$

i.e., it is a block in a *left semi-uniform matrix*. As in Case 1, we can split the computation of the product

$$\chi_r B^* W_{A,s} S_{A,b_A}^*$$

into two parts: first we compute

$$\widehat{B}_{b_B} := \chi_r B^* W_{A,s} \tag{8.7}$$

for all $b_B = (s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$, then the remaining multiplication with S_{A,b_A}^* can be handled directly. The structure of (8.7) closely resembles the structure of (8.4): if

$\operatorname{sons}(b_B) \neq \emptyset$, we can use the recursion

$$\widehat{B}_{b_B} = \sum_{b'=(s',r')\in\operatorname{sons}(b_B)} \chi_{r'} B^* W_{A,s'} F_{A,s',s} = \sum_{b'=(s',r')\in\operatorname{sons}(b_B)} \widehat{B}_{b_B} F_{A,s',s}, \quad (8.8)$$

otherwise, we can either rely on the definition (8.7) if b_B is an inadmissible leaf or on

$$\widehat{B}_{b_B} = \chi_r B^* W_{A,s} = W_{B,r} S^*_{B,b_B} V^*_{B,s} W_{A,s} = W_{B,r} S^*_{B,b_B} P^*_{AB,s}$$

if it is an admissible leaf. The resulting recursive procedure is given in Algorithm 49.

Algorithm 49. Semi-uniform matrix forward transformation for the matrix B^*.

```
procedure SemiTransMatrixForward(b, W_B, W_A, P_AB, S_B, B, var B̂);
(s, r) ← b;
if sons(b) ≠ ∅ then
  B̂_b ← 0;
  for b' ∈ sons(b) do
    (s', r') ← b';
    SemiTransMatrixForward(b', W_B, W_A, P_AB, S_B, B, B̂);
    B̂_b ← B̂ + B̂_b' F_A,s',s
  end for
else if b is admissible then
  Ŷ_r ← S*_B,b P*_AB,s;
  B̂_b ← W_B,r Ŷ_t
else
  B̂_b ← χ_r B* W_A,s
end if
```

Case 3: b_A and b_B are inadmissible

As in Section 7.8, we do not have any information about the product of two inadmissible blocks and cannot hope to be able to express it in a simple way, e.g., as a semi-uniform matrix. Fortunately, we can use the induced block cluster tree introduced in Definition 7.26 to ensure that this situation only appears if b_C is also inadmissible. If b_C is a leaf, we can handle it directly. Otherwise, we can proceed to the sons of b_A, b_B and b_C by recursion.

Lemma 8.5 (Exact matrix product). *Let $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ be the induced block cluster tree of Definition 7.26. Then we have $AB \in \mathcal{H}^2_{\text{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B)$.*

Proof. Similar to the proof of Theorem 7.29: we prove the more general statement

$$\chi_t A \chi_s B \chi_r \in \mathcal{H}^2_{\text{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B)$$

for all $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$ with $(t,s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$, $(s,r) \in \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ and $(t,r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ by induction.

If $b_A = (t,s) \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}$ holds, our investigation of Case 1 yields

$$\chi_t A \chi_s B \chi_r = V_{A,t}(\chi_r B W_{A,s} S^*_{A,b_A})^* \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, *) \subseteq \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B).$$

If $b_B = (s,r) \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}$ holds, the results of Case 2 imply

$$\chi_t A \chi_s B \chi_r = (\chi_t A V_{B,s} S_{B,b_B}) W^*_{B,s} \in \mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, *, W_B) \subseteq \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B).$$

We can continue as in the proof of Theorem 7.29. □

Our algorithm requires an explicit representation

$$M_{AB} := AB = \sum_{b=(t,r)\in\mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}} (X_b W^*_{B,r} + V_{A,t} Y^*_b) + \sum_{b=(t,r)\in\mathcal{L}^-_{M,\mathcal{I}\times\mathcal{K}}} \chi_t AB \chi_r \tag{8.9}$$

of the product AB as a semi-uniform matrix, i.e., we have to compute the left and right coefficient matrices $(X_b)_{b\in\mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}}$ and $(Y_b)_{b\in\mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}}$.

We accomplish this task in two steps: first we compute an intermediate representation

$$M_{AB} = AB = \sum_{b=(t,r)\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}} (X_b W^*_{B,r} + V_{A,t} Y^*_b) + \sum_{b=(t,r)\in\mathcal{L}^-_{M,\mathcal{I}\times\mathcal{K}}} \chi_t AB \chi_r$$

of the product with coefficient matrices in *all* blocks, and then we apply a procedure similar to the matrix backward transformation Algorithm 38 in order to translate it into the desired form (8.9). This second step is part of the coarsening Algorithm 54.

For the first part of the construction, the structure of the proof of Theorem 7.29 suggests a recursive approach: we start with $t = \mathrm{root}(\mathcal{T}_{\mathcal{I}})$, $s = \mathrm{root}(\mathcal{T}_{\mathcal{J}})$ and $r = \mathrm{root}(\mathcal{T}_{\mathcal{K}})$. If $b_A := (t,s)$ or $b_B := (s,r)$ are admissible leaves, we update X_{b_C} or Y_{b_C} for $b_C := (t,r)$ using the equations (8.3) and (8.6). Otherwise, all blocks are inadmissible, and we can either handle an inadmissible leaf directly or proceed by recursion.

Since we are looking for the matrix $M_0 = C + AB = C + M_{AB}$, we also have to handle the matrix C. We solve this task by a very simple procedure: we construct a representation of $C \in \mathcal{H}^2(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$ in the matrix space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$ corresponding to the induced block partition $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. Due to Definition 7.26, each block of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ is also present in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, and we can use the matrix backward transformation Algorithm 38 to compute the representation of C in $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$ efficiently.

Algorithm 50 computes the exact product $M_{AB} = AB$ of two $\mathcal{H}^2$-matrices A and B, represented as a semi-uniform matrix $AB \in \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{J}}, V_A, W_B)$. Algorithm 38 computes an $\mathcal{H}^2$-matrix representation of C in $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C) \subseteq \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$. Each admissible block $b = (t,r)$ of the matrix $M = C + AB$ is therefore represented in the form $V_{C,t}\widehat{S}_{C,b}W^*_{C,r} + X_b W^*_{B,r} + V_{A,t}Y^*_b$, i.e., its rank is bounded by $(\#L_{C,r}) + (\#L_{B,r}) + (\#K_{A,t})$. This fairly low rank is a major advantage of the new approach compared to the $\mathcal{H}^2$-matrix representation introduced in Section 7.8: the latter is based on induced cluster bases, and the rank of these cluster bases can become very large (depending on the size of the sparsity constant C_{sp}, cf. Lemma 7.23).

Algorithm 50. Semi-uniform representation of the matrix product.

procedure MatrixProductSemi(t, s, r, A, B, **var** M_{AB}, X, Y);
$b_A \leftarrow (t,s)$; $b_B \leftarrow (s,r)$; $b_C \leftarrow (t,r)$;
if b_B is admissible **then**
  $X_{b_C} \leftarrow X_{b_C} + \widehat{A}_{b_A} S_{B,b_B}$ $\{b_B \in \mathcal{L}^+_{B,\mathcal{J}\times\mathcal{K}}\}$
else if b_A is admissible **then**
  $Y_{b_C} \leftarrow Y_{b_C} + \widehat{B}_{b_B} S^*_{A,b_A}$ $\{b_A \in \mathcal{L}^+_{A,\mathcal{I}\times\mathcal{J}}\}$
else if b_A and b_B are leaves **then**
  $M_{AB} \leftarrow M_{AB} + \chi_t A \chi_s B \chi_r$ $\{b_C \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}\}$
else
  for $t' \in \mathrm{sons}^+(t), s' \in \mathrm{sons}^+(s), r' \in \mathrm{sons}^+(r)$ **do**
    MatrixProductSemi($t', s', r', A, B, M_{AB}, X, Y$) $\{b_A \in \mathcal{S}_A \text{ or } b_B \in \mathcal{S}_B\}$
  end for
end if

As already mentioned this advantage comes at a price: the storage requirements of semi-uniform matrices typically do not grow linearly with the matrix dimension, as for $\mathcal{H}$-matrices the depth of the block cluster tree is also a factor in the complexity estimates.

Complexity estimates

Let us now investigate the complexity of Algorithm 50. Since it is based on the matrices $(\widehat{A}_b)_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}}$ and $(\widehat{B}_b)_{b\in\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}}$ prepared by Algorithm 47 and 49, we first have to analyze the complexity of these preliminary steps.

Lemma 8.6. *Let V be a nested cluster basis with rank distribution K, let $(k_t)_{t\in\mathcal{T}_\mathcal{I}}$ be defined as in (3.16), and let*

$$\hat{k} := \max\{k_t \ : \ t \in \mathcal{T}_\mathcal{I}\}.$$

The computation of all matrices $(V_t)_{t\in\mathcal{T}_\mathcal{I}}$ *by Algorithm 46 requires not more than*

$$2(p_\mathcal{I}+1)\hat{k}^2 n_\mathcal{I}$$

operations.

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. If t is not a leaf, the algorithm computes the products of $V_{t'} \in \mathbb{R}^{\mathcal{I}\times K_{t'}}_{\hat{t}'}$ and $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$ for all $t' \in \operatorname{sons}(t)$. This can be accomplished in

$$\sum_{t'\in\operatorname{sons}(t)} 2(\#\hat{t}')(\#K_{t'})(\#K_t) \leq \sum_{t'\in\operatorname{sons}(t)} 2(\#\hat{t}')k_t^2 = 2(\#\hat{t})k_t^2 \leq 2(\#\hat{t})\hat{k}^2$$

operations, since Definition 3.4 implies

$$\sum_{t'\in\operatorname{sons}(t)} \#\hat{t}' = \# \bigcup_{t'\in\operatorname{sons}(t)} \hat{t}' = \#\hat{t}.$$

Adding these bounds for all $t \in \mathcal{T}_\mathcal{I}$ and applying Corollary 3.10 yields the bound

$$\begin{aligned}\sum_{t\in\mathcal{T}_\mathcal{I}} 2(\#\hat{t})\hat{k}^2 &= 2\hat{k}^2 \sum_{t\in\mathcal{T}_\mathcal{I}} \#\hat{t} = 2\hat{k}^2 \sum_{\ell=0}^{p_\mathcal{I}} \sum_{\substack{t\in\mathcal{T}_\mathcal{I}\\ \operatorname{level}(t)=\ell}} \#\hat{t}\\ &\leq 2\hat{k}^2 \sum_{\ell=0}^{p_\mathcal{I}} n_\mathcal{I} = 2\hat{k}^2(p_\mathcal{I}+1)n_\mathcal{I}.\end{aligned}$$

□

Once the cluster bases have been prepared, the complexity of the semi-uniform matrix forward transformation can be investigated.

Lemma 8.7. *Let* K_A, L_A, K_B *and* L_B *be the rank distributions of* V_A, W_A, V_B *and* W_B. *Let* $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(l_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $(k_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ *and* $(l_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ *be defined as in* (3.16) *and* (3.18), *and let*

$$\hat{k} := \max\{k_{A,t}, l_{A,s}, k_{B,s}, l_{B,r} \ : \ t\in\mathcal{T}_\mathcal{I},\ s\in\mathcal{T}_\mathcal{J},\ r\in\mathcal{T}_\mathcal{K}\}. \tag{8.10}$$

If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ *is* C_{sp}*-sparse, Algorithm 47 requires not more than*

$$2C_{\mathrm{sp}}(\hat{k}^3 c_\mathcal{I} + \hat{k}^2(p_\mathcal{I}+1)n_\mathcal{I})$$

operations. If $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ *is* C_{sp}*-sparse, Algorithm 49 requires not more than*

$$2C_{\mathrm{sp}}(\hat{k}^3 c_\mathcal{K} + \hat{k}^2(p_\mathcal{K}+1)n_\mathcal{K})$$

operations. If $\mathcal{T}_\mathcal{I}$ *and* $\mathcal{T}_\mathcal{K}$ *are* $(C_{\mathrm{rc}}, \hat{k}, 0, 1)$*-regular, the number of operations is in* $\mathcal{O}(\hat{k}^2(p_\mathcal{I}+1)n_\mathcal{I})$ *and* $\mathcal{O}(\hat{k}^2(p_\mathcal{K}+1)n_\mathcal{K})$, *respectively.*

Proof. Since Algorithms 47 and 49 are very similar, we only consider the first one.

Let $b = (t, s) \in \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$. If $\operatorname{sons}(b) \neq \emptyset$, the algorithm computes $\widehat{A}_{b'} E_{B,s',s}$ for all sons b' of b and adds the result to $\widehat{A}_b$. This can be accomplished in

$$\begin{aligned}
&\sum_{b'=(t',s')\in\operatorname{sons}(b)} 2(\#\hat{t}')(\#K_{B,s'})(\#K_{B,s}) \\
&\quad = 2\Big(\sum_{t'\in\operatorname{sons}^+(t)} \#\hat{t}'\Big)\Big(\sum_{s'\in\operatorname{sons}^+(s)} \#K_{B,s'}\Big)(\#K_{B,s}) \\
&\quad \leq 2(\#\hat{t})k_{B,s}^2 \leq 2(\#\hat{t})\hat{k}^2.
\end{aligned}$$

Otherwise, i.e., if b is a leaf, it can be either inadmissible or admissible. If it is inadmissible, the clusters t and s are leaves of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, respectively, so we have $\#\hat{s} \leq k_{B,s}$ and find that we can compute the product in

$$2(\#\hat{t})(\#\hat{s})(\#K_{B,s}) \leq 2(\#\hat{t})k_{B,s}(\#K_{B,s}) \leq 2(\#\hat{t})\hat{k}^2$$

operations. If we are dealing with an admissible leaf, the computation of $\widehat{Y}_t$ can be performed in

$$2(\#K_{A,t})(\#L_{A,s})(\#K_{B,s}) \leq 2k_{A,t}l_{A,s}k_{B,s} \leq 2\hat{k}^3$$

operations, and the multiplication by $V_{A,t}$ takes not more than

$$2(\#\hat{t})(\#K_{A,t})(\#K_{B,s}) \leq 2(\#\hat{t})k_{A,t}k_{B,s} \leq 2(\#\hat{t})\hat{k}^2$$

operations. We can summarize that we need not more than

$$2(\hat{k} + \#\hat{t})\hat{k}^2$$

operations for the block b. Adding the bounds for all blocks yields

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} 2(\hat{k} + \#\hat{t})\hat{k}^2 &= 2\sum_{b\in\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}} \hat{k}^3 + 2\sum_{t\in\mathcal{T}_{\mathcal{I}}}\sum_{s\in\operatorname{row}(t)} (\#\hat{t})\hat{k}^2 \\
&\leq 2\hat{k}^3\#\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} + 2C_{\mathrm{sp}}\hat{k}^2\sum_{t\in\mathcal{T}_{\mathcal{I}}} \#\hat{t} \\
&= 2\hat{k}^3\#\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} + 2C_{\mathrm{sp}}\hat{k}^2\sum_{\ell=0}^{p_{\mathcal{I}}}\sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)=\ell}} \#\hat{t} \\
&\leq 2\hat{k}^3\#\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} + 2C_{\mathrm{sp}}\hat{k}^2\sum_{\ell=0}^{p_{\mathcal{I}}} n_{\mathcal{I}} \\
&= 2\hat{k}^3\#\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} + 2C_{\mathrm{sp}}\hat{k}^2(p_{\mathcal{I}} + 1)n_{\mathcal{I}}.
\end{aligned}$$

Using the estimate

$$\#\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}} = \sum_{t\in\mathcal{T}_{\mathcal{I}}} \#\operatorname{row}(t) \leq C_{\mathrm{sp}}\#\mathcal{T}_{\mathcal{I}} = C_{\mathrm{sp}}c_{\mathcal{I}}$$

we get the desired result, and if $\mathcal{T}_{\mathcal{I}}$ is $(C_{\mathrm{rc}}, \hat{k}, 0, 1)$-regular, we can find a constant C independent of $\hat{k}$ and $n_{\mathcal{I}}$ satisfying $c_{\mathcal{I}} \leq Cn_{\mathcal{I}}/\hat{k}$ and complete the proof. □

In order to analyze the complexity of Algorithm 50, we follow the approach used in Section 7.7: we observe that the triples (t, s, r) for which the algorithm is called are organized in a tree structure. The corresponding *call tree* $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ is the minimal tree with root

$$\operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}) := (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}), \operatorname{root}(\mathcal{T}_{\mathcal{K}}))$$

and a father-son relation given by

$$\operatorname{sons}(t,s,r) := \begin{cases} \operatorname{sons}^+(t)\times\operatorname{sons}^+(s)\times\operatorname{sons}^+(r) & \text{if } (t,s)\in\mathcal{S}_A, (s,r)\in\mathcal{I}_B, \\ & \text{or } (t,s)\in\mathcal{I}_A, (s,r)\in\mathcal{S}_B, \\ \emptyset & \text{otherwise,} \end{cases} \tag{8.11}$$

for all triples $(t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ it contains. Here $\mathcal{S}_A \subseteq \mathcal{I}_A \subseteq \mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{S}_B \subseteq \mathcal{I}_B \subseteq \mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ denote the sets of subdivided and inadmissible blocks of A and B, respectively (cf. Definition 7.16 and (7.18)).

As in Section 7.7, the complexity estimate depends on a *sparsity* estimate for the call tree.

Lemma 8.8 (Sparsity of the call tree). *Let $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ be C_{sp}-sparse admissible block cluster trees, and let*

$$\begin{aligned} C_{\mathcal{I}}(t) &:= \{(s,r)\in\mathcal{T}_{\mathcal{J}}\times\mathcal{T}_{\mathcal{K}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} && \text{for all } t\in\mathcal{T}_{\mathcal{I}}, \\ C_{\mathcal{J}}(s) &:= \{(t,r)\in\mathcal{T}_{\mathcal{I}}\times\mathcal{T}_{\mathcal{K}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} && \text{for all } s\in\mathcal{T}_{\mathcal{J}}, \\ C_{\mathcal{K}}(r) &:= \{(t,s)\in\mathcal{T}_{\mathcal{I}}\times\mathcal{T}_{\mathcal{J}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\} && \text{for all } r\in\mathcal{T}_{\mathcal{K}}. \end{aligned}$$

Then we have

$$\#C_{\mathcal{I}}(t), \#C_{\mathcal{J}}(s), \#C_{\mathcal{K}}(r) \leq 3C_{\mathrm{sp}}^2 \quad \text{for all } t\in\mathcal{T}_{\mathcal{I}}, s\in\mathcal{T}_{\mathcal{J}}, r\in\mathcal{T}_{\mathcal{K}}.$$

Proof. As in the proof of Lemma 7.18: comparing (8.11) and (7.19) reveals that each node of the call tree of Algorithm 50 is also a node of the call tree of Algorithm 45 (with a minimal $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ consisting only of the root). □

Since we are working with semi-uniform matrices instead of $\mathcal{H}^2$-matrices, the number of operations for a block $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ depends on the cardinalities of $\hat{t}$ and $\hat{r}$, which means that the weak assumptions used in the previous chapters do not allow us to derive meaningful complexity estimates. Instead, we have to require that the strict conditions introduced in Section 3.8 are met, i.e., that the rank distribution is k-bounded and that the cluster trees are (C_{rc}, k)-bounded.

Lemma 8.9 (Complexity). *Let K_A, L_A, K_B and L_B be the rank distributions of V_A, W_A, V_B and W_B. Let $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(l_{A,s})_{s\in\mathcal{T}_\mathcal{J}}$, $(k_{B,s})_{s\in\mathcal{T}_\mathcal{J}}$ and $(l_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ be defined as in (3.16) and (3.18), and let $\hat{k}$ be defined as in (8.10). If $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ are C_{sp}-sparse, Algorithm 50 requires not more than*

$$6C_{\mathrm{sp}}^2\hat{k}^2((p_\mathcal{I}+1)n_\mathcal{I}+(p_\mathcal{K}+1)n_\mathcal{K})$$

operations to compute the semi-uniform representation $M_{AB}=AB$ of the product of the $\mathcal{H}^2$-matrices A and B.

Proof. Let $(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$, and let $b_A:=(t,s)$, $b_B:=(s,r)$ and $b_C:=(t,r)$.

If b_B is admissible, we have to compute the matrix $\widehat{A}_{b_A}S_{B,b_B}$ and add it to X_{b_C}, and this requires not more than

$$2(\#\hat{t})(\#K_{B,s})(\#L_{B,r})\leq 2(\#\hat{t})\hat{k}^2$$

operations. If b_A is admissible, we have to compute the matrix $\widehat{B}_{b_B}S^*_{A,b_A}$ and add it to Y_{b_C}, and this requires not more than

$$2(\#\hat{r})(\#L_{A,s})(\#K_{A,t})\leq 2(\#\hat{r})\hat{k}^2$$

operations. If b_A and b_B are inadmissible leaves, the fact that $\mathcal{T}_{A,\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{B,\mathcal{J}\times\mathcal{K}}$ are admissible block cluster trees implies that the clusters t, s and r are leaves of $\mathcal{T}_\mathcal{I}$, $\mathcal{T}_\mathcal{J}$ and $\mathcal{T}_\mathcal{K}$, and the update of $\chi_t M_{AB}\chi_r$ requires not more than

$$2(\#\hat{t})(\#\hat{s})(\#\hat{r})\leq(\#\hat{t})\hat{k}^2$$

operations, and we conclude that the number of operations for each triple $(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ is bounded by

$$2\hat{k}^2(\#\hat{t}+\#\hat{r}).$$

Combining Corollary 3.10 and Lemma 8.8 we can find and estimate for the number of operations of the entire algorithm: we have

$$\begin{aligned}\sum_{(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}}\#\hat{t}&=\sum_{t\in\mathcal{T}_\mathcal{I}}(\#\hat{t})\#C_\mathcal{I}(t)\leq 3C_{\mathrm{sp}}^2\sum_{t\in\mathcal{T}_\mathcal{I}}\#\hat{t}\\&=3C_{\mathrm{sp}}^2\sum_{\ell=0}^{p_\mathcal{I}}\sum_{\substack{t\in\mathcal{T}_\mathcal{I}\\ \mathrm{level}(t)=\ell}}\#\hat{t}\leq 3C_{\mathrm{sp}}^2\sum_{\ell=0}^{p_\mathcal{I}}n_\mathcal{I}=3C_{\mathrm{sp}}^2(p_\mathcal{I}+1)n_\mathcal{I},\\\sum_{(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}}\#\hat{r}&=\sum_{r\in\mathcal{T}_\mathcal{K}}(\#\hat{r})\#C_\mathcal{K}(r)\leq\cdots\leq 3C_{\mathrm{sp}}^2(p_\mathcal{K}+1)n_\mathcal{K},\end{aligned}$$

and this implies

$$\sum_{(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}}2\hat{k}^2(\#\hat{t}+\#\hat{r})\leq 6C_{\mathrm{sp}}^2\hat{k}^2((p_\mathcal{I}+1)n_\mathcal{I}+(p_\mathcal{K}+1)n_\mathcal{K}).\qquad\square$$

Remark 8.10 (Computation of $M_0 = C + AB$). Algorithm 50 computes the representation of $M_{AB} = AB$ as a semiuniform matrix in the block cluster tree $\widetilde{\mathcal{T}}_{M,\mathcal{I}\times\mathcal{K}}$ defined by the call tree through

$$(t,r) \in \widetilde{\mathcal{T}}_{M,\mathcal{I}\times\mathcal{K}} \iff \text{there is an } s \in \mathcal{T}_{\mathcal{J}} \text{ with } (t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}.$$

In general, $\widetilde{\mathcal{T}}_{M,\mathcal{I}\times\mathcal{K}}$ is only a subtree of the induced block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, but we can use Lemma 8.3 to compute a semi-uniform representation of M_{AB} using $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$.

Similarly we can use the matrix backward transformation Algorithm 38 to compute an $\mathcal{H}^2$-matrix representation of C in the space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C)$.

The result is a representation of the *exact* result $C + AB = C + M_{AB}$ in the matrix space $\mathcal{H}^2(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_C, W_C) + \mathcal{H}^2_{\mathrm{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B)$. □

8.3 Coarsening

In most practical applications, the induced block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ used in the exact representation of $M_0 = C + AB$ provided by Lemma 8.5 will not coincide with the block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ we have prescribed for the result of the multiplication: although Definition 7.26 ensures that each block of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ will also appear in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, leaves of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ may correspond to subdivided blocks in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, and admissible blocks in $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ may be inadmissible in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ due to the modified admissibility condition used in the construction of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$.

The second phase of the a posteriori matrix-matrix multiplication algorithm addresses this problem: for all admissible leaves $b = (t,r) \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$, we check whether they are also admissible leaves in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. If they are not, we construct low-rank approximations of the corresponding submatrices by using the *hierarchical approximation* approach described in [52]. The result of this procedure is an approximation $\widetilde{M}$ of $M_0 = C + AB$ by a hierarchical matrix based on the *prescribed* block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ instead of the induced block cluster tree $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. In the third and final phase of the multiplication algorithm, we convert this intermediate approximation into the desired $\mathcal{H}^2$-matrix based on the block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$.

Since we aim to approximate the matrix $M = C + AB$ by an $\mathcal{H}$-matrix, we can compute all approximations block by block and do not have to take interactions within rows or columns into account. This computation can be carried out by a recursive procedure working from the leaves of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ upwards.

Inadmissible leaves of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$

Let $b \in \mathcal{L}^-_{M,\mathcal{I}\times\mathcal{K}}$ be an inadmissible leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. If b is also an inadmissible leaf of $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$, we are done. Otherwise, we can use a singular value decomposition to turn the corresponding submatrix into a low-rank matrix:

Lemma 8.11 (Singular value decomposition). *Let $\mathcal{I}' \subseteq \mathcal{I}$ and $\mathcal{J}' \subseteq \mathcal{J}$. Let $M \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}',\mathcal{J}'}$. Let $p := \operatorname{rank}(M) \leq \min\{\#\mathcal{I}', \#\mathcal{J}'\}$ with $p > 0$. Let $\sigma_1 \geq \cdots \geq \sigma_p > 0$ be the non-zero singular values of M. For each $l \in \{1, \ldots, p\}$, we can find an index set $\mathcal{K}$ with $n_{\mathcal{K}} = l$, orthogonal matrices $V \in \mathbb{R}^{\mathcal{I}\times\mathcal{K}}_{\mathcal{I}'}$ and $W \in \mathbb{R}^{\mathcal{J}\times\mathcal{K}}_{\mathcal{J}'}$ and a diagonal matrix $S \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ which satisfy*

$$\|M - VSW^*\|_2 = \begin{cases} \sigma_{l+1} & \text{if } l < p, \\ 0 & \text{otherwise}, \end{cases} \tag{8.12}$$

$$\|M - VSW^*\|_F = \begin{cases} \left(\sum_{i=l+1}^{p} \sigma_i^2\right)^{1/2} & \text{if } l < p, \\ 0 & \text{otherwise}, \end{cases} \tag{8.13}$$

Proof. As in Lemma 5.19. □

For a given error tolerance $\epsilon_b \in \mathbb{R}_{>0}$, this result allows us to find an approximation with sufficiently small error and the lowest possible rank: for an inadmissible leaf $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$, we can find an index set K_b, orthogonal matrices $V_b \in \mathbb{R}^{\mathcal{I}\times K_b}_{\hat{t}}$, $W_b \in \mathbb{R}^{\mathcal{K}\times K_b}_{\hat{r}}$ and a diagonal matrix $S_b \in \mathbb{R}^{K_b\times K_b}$ such that

$$\|\chi_t M \chi_r - V_b S_b W_b^*\|_2 \leq \epsilon_b \quad \text{or} \quad \|\chi_t M \chi_r - V_b S_b W_b^*\|_F \leq \epsilon_b \tag{8.14}$$

holds. Low-rank representations of this type are closely related to the ones used in the context of $\mathcal{H}$-matrices: by setting $A_b := V_b S_b$ and $B_b := W_b$, we immediately get the desired factorized representation of a low-rank approximation of the submatrix corresponding to the block b.

Remark 8.12 (Complexity). According to Lemma 5.17, the Householder factorization in Algorithm 51 requires not more than

$$C_{\mathrm{qr}} m n q$$

operations, and according to Remark 5.20 the singular value decomposition can be computed in not more than

$$C_{\mathrm{svd}} q^3$$

operations. If $m > n$, the computation of $\hat{V}$ can be carried out in $2mn^2 = 2mnq$ operations, otherwise the computation of $\hat{W}$ can be performed in $2m^2 n = 2mnq$ operations.

The entire Algorithm 51 therefore takes not more than

$$C_{\mathrm{svdf}}(\#\mathcal{I}')(\#\mathcal{J}')\min\{\#\mathcal{I}', \#\mathcal{J}'\}$$

operations with $C_{\mathrm{svdf}} = C_{\mathrm{qr}} + C_{\mathrm{svd}} + 2$ to complete. □

Algorithm 51. Compute a low-rank approximation VSW^* of a matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}_{\mathcal{I}',\mathcal{J}'}$.

procedure LowrankFactor(X, $\mathcal{I}'$, $\mathcal{J}'$, ϵ, **var** V, S, W, $\widehat{K}$);
$m \leftarrow \#\mathcal{I}'$; $n \leftarrow \#\mathcal{J}'$; $q \leftarrow \min\{m, n\}$;
Fix arbitrary isomorphisms $\iota_m \colon \{1, \ldots, m\} \to \mathcal{I}'$ and $\iota_n \colon \{1, \ldots, n\} \to \mathcal{J}'$;
$\widehat{X} \leftarrow 0 \in \mathbb{R}^{m\times n}$
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ **do**
  $\widehat{X}_{ij} \leftarrow X_{\iota_m(i),\iota_n(j)}$
end for;
if $m > n$ **then**
  Compute a Householder factorization $\widehat{X} = \widehat{Q}\widehat{R}$ of $\widehat{X}$;
  $\widehat{Y} \leftarrow \widehat{R} \in \mathbb{R}^{q\times q}$
else
  Compute a Householder factorization $\widehat{X}^* = \widehat{Q}\widehat{R}$ of $\widehat{X}^*$;
  $\widehat{Y} \leftarrow \widehat{R}^* \in \mathbb{R}^{q\times q}$
end if
Compute a singular value decomposition $\widehat{Y} = \widehat{V}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_q)\widehat{W}^*$ of $\widehat{Y}$;
if $m > n$ **then**
  $\widehat{V} \leftarrow \widehat{Q}\widehat{V}$
else
  $\widehat{W} \leftarrow \widehat{Q}\widehat{W}$
end if
FindRank(ϵ, q, $(\sigma_i)_{i=1}^q$, l); {Algorithm 17}
$\widehat{K} \leftarrow \iota_m(\{1, \ldots, l\})$;
$V \leftarrow 0 \in \mathbb{R}^{\mathcal{I}\times\widehat{K}}_{\mathcal{I}'}$; $W \leftarrow 0 \in \mathbb{R}^{\mathcal{J}\times\widehat{K}}_{\mathcal{J}'}$; $S \leftarrow 0 \in \mathbb{R}^{\widehat{K}\times\widehat{K}}$;
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, l\}$ **do**
  $V_{\iota_m(i),\iota_m(j)} \leftarrow \widehat{V}_{ij}$
end for;
for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, l\}$ **do**
  $W_{\iota_n(i),\iota_m(j)} \leftarrow \widehat{W}_{ij}$
end for;
for $i \in \{1, \ldots, l\}$ **do**
  $S_{\iota_m(i),\iota_m(i)} \leftarrow \sigma_i$
end for

Admissible leaves of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$

We can apply the same approach to admissible leaves: let $b = (t, r) \in \mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}$. Assuming that the index sets $L_{C,r}$, $L_{B,r}$ and $K_{A,t}$ are pairwise disjoint, the fact that

M is semi-uniform implies

$$\begin{aligned}\chi_t M \chi_r &= V_{C,t} \widehat{S}_{C,b} W_{C,r} + X_b W_{B,r}^* + V_{A,t} Y_b^* \\ &= \begin{pmatrix} V_{C,t} \widehat{S}_{C,b} & X_b & V_{A,t} \end{pmatrix} \begin{pmatrix} W_{C,r}^* \\ W_{B,r}^* \\ Y_b^* \end{pmatrix}, \end{aligned} \tag{8.15}$$

i.e., the submatrix corresponding to b is already given in a factorized representation of rank not higher than $(\#L_{C,r}) + (\#L_{B,r}) + (\#K_{A,t})$. If we want to keep the storage requirements low, we can try to reduce the rank by applying Lemma 8.11 to $\chi_t M \chi_r$. Since the direct computation of the singular value decomposition of the submatrix corresponding to b could become computationally expensive if the index sets $\hat{t}$ and $\hat{r}$ become large, we have to exploit the factorized structure (8.15): we apply Lemma 5.16 to the left factor in (8.15) in order to find an index set $\widehat{K}_b$, an orthogonal matrix $Q_b \in \mathbb{R}^{\mathcal{I} \times \widehat{K}_b}_{\hat{t}}$ and a matrix $\widehat{R}_b \in \mathbb{R}^{\widehat{K}_b \times (L_{C,r} \cup L_{B,r} \cup K_{A,t})}$ with

$$\begin{pmatrix} V_{C,t} \widehat{S}_{C,b} & X_b & V_{A,t} \end{pmatrix} = Q_b \widehat{R}_b.$$

Combining this factorization with (8.15) yields

$$\begin{aligned}\chi_t M \chi_r &= \begin{pmatrix} V_{C,t} \widehat{S}_{C,b} & X_b & V_{A,t} \end{pmatrix} \begin{pmatrix} W_{C,r}^* \\ W_{B,r}^* \\ Y_b^* \end{pmatrix} \\ &= Q_b \begin{pmatrix} \widehat{R}_b|_{\widehat{K}_b \times L_{C,r}} & \widehat{R}_b|_{\widehat{K}_b \times L_{B,r}} & \widehat{R}_b|_{\widehat{K}_b \times K_{A,t}} \end{pmatrix} \begin{pmatrix} W_{C,r}^* \\ W_{B,r}^* \\ Y_b^* \end{pmatrix} \\ &= Q_b \left(\widehat{R}_b|_{\widehat{K}_b \times L_{C,r}} W_{C,r}^* + \widehat{R}_b|_{\widehat{K}_b \times L_{B,r}} W_{B,r}^* + \widehat{R}_b|_{\widehat{K}_b \times K_{A,t}} Y_b^* \right) = Q_b \widehat{M}_b^* \end{aligned}$$

for the auxiliary matrix

$$\widehat{M}_b := W_{C,r} \widehat{R}_b|^*_{\widehat{K}_b \times L_{C,r}} + W_{B,r} \widehat{R}_b|^*_{\widehat{K}_b \times L_{B,r}} + Y_b \widehat{R}_b|^*_{\widehat{K}_b \times K_{A,t}} \in \mathbb{R}^{\mathcal{K} \times \widehat{K}_b}_{\hat{s}}.$$

Since $\widehat{M}_b$ has only $\#\widehat{K}_b \leq (\#L_{C,r}) + (\#L_{B,r}) + (\#K_{A,t})$ rows, we can compute its singular value decomposition efficiently, and this gives us an index set K_b, orthogonal matrices $\widehat{V}_b \in \mathbb{R}^{\widehat{K}_b \times K_b}$, $W_b \in \mathbb{R}^{\mathcal{J} \times K_b}_{\hat{s}}$, and a diagonal matrix $S_b \in \mathbb{R}^{K_b \times K_b}$ such that

$$\|\widehat{M}_b - W_b S_b \widehat{V}_b^*\|_2 \leq \epsilon_b \quad \text{or} \qquad \|\widehat{M}_b - W_b S_b \widehat{V}_b^*\|_F \leq \epsilon_b$$

holds. Since Q_b is an orthogonal matrix, the same holds for $V_b := Q_b \widehat{V}_b$, and we find

$$\chi_t M \chi_r - V_b S_b W_b^* = Q_b (\widehat{M}_b^* - \widehat{V}_b S_b W_b^*),$$

so (8.14) is guaranteed (since S_b is diagonal, we have $S_b = S_b^*$). The resulting procedure for finding low-rank approximations of leaves of $\mathcal{T}_{M,\mathcal{I} \times \mathcal{K}}$ is given in Algorithm 52.

Algorithm 52. Low-rank approximation of leaf blocks.

procedure CoarsenLeaf(b, M, ϵ, **var** V_b, S_b, W_b);
$(t,r) \leftarrow b$;
if b is admissible in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ **then**
  $X_{1,b} \leftarrow V_{C,t}\widehat{S}_{C,b}$;
  $X_{2,b} \leftarrow X_b$;
  $X_{3,b} \leftarrow V_{A,t}$;
  $Z_b \leftarrow 0 \in \mathbb{R}^{\hat{t}\times(L_{C,r}\cup L_{B,r}\cup K_{A,t})}$;
  $Z_b|_{\hat{t}\times L_{C,r}} \leftarrow X_{1,b}$; $Z_b|_{\hat{t}\times L_{B,r}} \leftarrow X_{2,b}$; $Z_b|_{\hat{t}\times K_{A,t}} \leftarrow X_{3,b}$;
  Householder(Z_b, $\hat{t}$, Q_b, $\widehat{R}_b$, $\widehat{K}_b$); {Algorithm 15}
  $\widehat{R}_{1,b} \leftarrow \widehat{R}|_{\widehat{K}_b\times L_{C,r}}$; $\widehat{R}_{2,b} \leftarrow \widehat{R}|_{\widehat{K}_b\times L_{B,r}}$; $\widehat{R}_{3,b} \leftarrow \widehat{R}|_{\widehat{K}_b\times K_{A,t}}$;
  $\widehat{M}_{1,b} \leftarrow W_{C,r}\widehat{R}_{1,b}^*$;
  $\widehat{M}_{2,b} \leftarrow W_{B,r}\widehat{R}_{2,b}^*$;
  $\widehat{M}_{3,b} \leftarrow Y_b\widehat{R}_{3,b}^*$;
  $\widehat{M}_b \leftarrow \widehat{M}_{1,b} + \widehat{M}_{2,b} + \widehat{M}_{3,b} \in \mathbb{R}_{\hat{s}}^{\hat{\jmath}\times\widehat{K}_b}$;
  LowrankFactor($\widehat{M}_b$, $\hat{s}$, $\widehat{K}_b$, ϵ_b, W_b, S_b, $\widehat{V}_b$, K_b); {Algorithm 51}
  $V_b \leftarrow Q_b\widehat{V}_b$
else
  LowrankFactor($\chi_t M\chi_r$, $\hat{t}$, $\hat{r}$, ϵ_b, V_b, S_b, W_b, K_b) {Algorithm 51}
end if

Subdivided blocks in $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$

Now let us consider blocks $b = (t,r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ which are not leaves. We assume that we have already found low-rank approximations $V_{b'}S_{b'}W_{b'}^*$ for all sons $b' \in \operatorname{sons}(b)$, and we have to find a way of combining these approximations into a low-rank approximation of the entire block.

This means that we have to approximate the matrix

$$\sum_{b'\in\operatorname{sons}(b)} V_{b'}S_{b'}W_{b'}^* = \sum_{t'\in\operatorname{sons}^+(t)}\sum_{r'\in\operatorname{sons}^+(r)} V_{t',r'}S_{t',r'}W_{t',r'}^*. \tag{8.16}$$

Let $\tau := \#\operatorname{sons}^+(t)$ and $\{t_1,\ldots,t_\tau\} := \operatorname{sons}^+(t)$, and let $\rho := \#\operatorname{sons}^+(r)$ and $\{r_1,\ldots,r_\rho\} := \operatorname{sons}^+(r)$. The inner sum can be expressed in the form

$$\sum_{r'\in\operatorname{sons}^+(r)} V_{t',r'}S_{t',r'}W_{t',r'}^* = \begin{pmatrix} V_{t',r_1}S_{t',r_1} & \ldots & V_{t',r_\rho}S_{t',r_\rho}\end{pmatrix}\begin{pmatrix} W_{t',r_1}^* \\ \vdots \\ W_{t',r_\rho}^*\end{pmatrix}. \tag{8.17}$$

We once more use Lemma 5.16 in order to find an index set $\hat{K}_{t'}$, an orthogonal matrix $Q_{t'} \in \mathbb{R}^{\mathcal{I}\times\hat{K}_{t'}}_{\hat{t}'}$ and a matrix $\hat{R}_{t'} \in \mathbb{R}^{\hat{K}_{t'}\times(K_{t',r_1}\cup\cdots\cup K_{t',r_\rho})}$ with

$$\begin{pmatrix} V_{t',r_1}S_{t',r_1} & \dots & V_{t',r_\rho}S_{t',r_\rho}\end{pmatrix} = Q_{t'}\hat{R}_{t'}.$$

We combine this equation with (8.17) and get

$$\begin{aligned}\sum_{r'\in\operatorname{sons}^+(r)} V_{t',r'}S_{t',r'}W^*_{t',r'} &= \begin{pmatrix} V_{t',r_1}S_{t',r_1} & \dots & V_{t',r_\sigma}S_{t',r_\rho}\end{pmatrix}\begin{pmatrix} W^*_{t',r_1}\\ \vdots\\ W^*_{t',r_\rho}\end{pmatrix}\\ &= Q_{t'}\begin{pmatrix} \hat{R}_{t'}|_{\hat{K}_{t'}\times K_{t',r_1}} & \dots & \hat{R}_{t'}|_{\hat{K}_{t'}\times K_{t',r_\rho}}\end{pmatrix}\begin{pmatrix} W^*_{t',r_1}\\ \vdots\\ W^*_{t',r_\rho}\end{pmatrix}\\ &= Q_{t'}\Bigl(\sum_{r'\in\operatorname{sons}^+(r)} \hat{R}_{t'}|_{\hat{K}_{t'}\times K_{t',r'}}W^*_{t',r'}\Bigr) = Q_{t'}\hat{M}^*_{t'}\end{aligned}$$

for the auxiliary matrix

$$\hat{M}_{t'} := \sum_{r'\in\operatorname{sons}^+(r)} W_{t',r'}\hat{R}_{t'}|^*_{\hat{K}_{t'}\times K_{t',r'}} \in \mathbb{R}^{\mathcal{K}\times\hat{K}_{t'}}_{\hat{r}}.$$

We assume that the index sets $\hat{K}_{t_1},\dots,\hat{K}_{t_\tau}$ are pairwise disjoint and combine the above equation with (8.16) in order to get

$$\sum_{b'\in\operatorname{sons}(b)} V_{b'}S_{b'}W^*_{b'} = \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau}\end{pmatrix}\begin{pmatrix} \hat{M}^*_{t_1}\\ \vdots\\ \hat{M}^*_{t_\tau}\end{pmatrix} = Q_b\hat{M}^*_b$$

for the auxiliary matrices

$$Q_b := \begin{pmatrix} Q_{t_1} & \dots & Q_{t_\tau}\end{pmatrix} \in \mathbb{R}^{\mathcal{I}\times\hat{K}_b}_{\hat{t}}, \quad \hat{M}_b := \begin{pmatrix} \hat{M}_{t_1} & \dots & \hat{M}_{t_\tau}\end{pmatrix} \in \mathbb{R}^{\mathcal{K}\times\hat{K}_b}_{\hat{r}}$$

with $\hat{K}_b := \hat{K}_{t_1}\mathbin{\dot\cup}\cdots\mathbin{\dot\cup}\hat{K}_{t_\tau}$. We have $Q_{t'} = \chi_{t'}Q_{t'}$ for all $t' \in \operatorname{sons}^+(t)$, and since the index sets corresponding to different sons of t are disjoint, we can conclude that the orthogonality of the matrices $Q_{t'}$ implies that also Q_b is orthogonal. Therefore we can proceed as in the case of leaf blocks: we have to find a low-rank approximation of $Q_b\hat{M}^*_b$. The resulting procedure for finding a low-rank approximation of a subdivided matrix is given in Algorithm 53.

Remark 8.13 (Intermediate approximation). In practice, it can be more efficient to replace the exact QR factorization of the matrices $Z_{t'}$ by truncations which reduce the rank of the intermediate matrices $Q_{t'}\hat{R}_{t'}$ at an early stage.

Algorithm 53. Low-rank approximation of subdivided blocks.

procedure CoarsenSubdivided(b, M, ϵ, **var** V_b, S_b, W_b);
$(t, r) \leftarrow b$;
$\hat{K}_b \leftarrow \emptyset$;
for $t' \in \operatorname{sons}^+(t)$ **do**
  $K_{t',*} \leftarrow \emptyset$;
  for $r' \in \operatorname{sons}^+(r)$ **do**
    $b' \leftarrow (t', r')$;
    $K_{t',*} \leftarrow K_{t',*} \cup K_{b'}$
  end for;
  $Z_{t'} \leftarrow 0 \in \mathbb{R}^{\mathcal{I} \times K_{t',*}}_{\hat{t}}$;
  for $r' \in \operatorname{sons}^+(r)$ **do**
    $b' \leftarrow (t', r')$;
    $Z_{t'}|_{\mathcal{I} \times K_{b'}} \leftarrow V_{b'} S_{b'}$
  end for;
  Householder($Z_{t'}$, $\hat{t}'$, $Q_{t'}$, $\hat{R}_{t'}$, $\hat{K}_{t'}$); {Algorithm 15}
  $\hat{M}_{t'} \leftarrow 0 \in \mathbb{R}^{\mathcal{J} \times \hat{K}_{t'}}$;
  for $r' \in \operatorname{sons}^+(r)$ **do**
    $b' \leftarrow (t', r')$;
    $\hat{M}_{t'} \leftarrow \hat{M}_{t'} + W_{b'} \hat{R}_{t'}|^*_{\hat{K}_{t'} \times K_{b'}}$
  end for;
  $\hat{K}_b \leftarrow \hat{K}_b \cup \hat{K}_{t'}$
end for;
$Q_b \leftarrow 0 \in \mathbb{R}^{\mathcal{I} \times \hat{K}_b}_{\hat{t}}$; $\hat{M}_b \leftarrow 0 \in \mathbb{R}^{\mathcal{J} \times \hat{K}_b}_{\hat{r}}$;
for $t' \in \operatorname{sons}^+(t)$ **do**
  $Q_b|_{\mathcal{I} \times K_{t'}} \leftarrow Q_{t'}$; $\hat{M}_b|_{\mathcal{J} \times K_{t'}} \leftarrow \hat{M}_{t'}$
end for;
LowrankFactor($\hat{M}_b$, $\hat{r}$, $\hat{K}_b$, ϵ_b, W_b, S_b, $\hat{V}_b$, K_b); {Algorithm 51}
$V_b \leftarrow Q_b \hat{V}_b$

This approach also makes the implementation of the algorithm simpler: it is only necessary to implement the approximation of matrices of the form

$$X = \begin{pmatrix} X_1 & \dots & X_\sigma \end{pmatrix}, \quad X = \begin{pmatrix} X_1 \\ \vdots \\ X_\tau \end{pmatrix}$$

and handle the case of general block matrices by first compressing all rows, thus creating an intermediate block matrix with only one block column, and then compressing this intermediate matrix in order to get the final result. □

Using the Algorithm 52 for leaf blocks and the Algorithm 53 for subdivided blocks recursively leads to the *coarsening* Algorithm 54 which approximates the semi-uniform matrix $M \in \mathcal{H}^2_{\text{semi}}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, V_A, W_B)$ by an $\mathcal{H}$-matrix $\tilde{M} \in \mathcal{H}(\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}, k)$ given by

$$\tilde{M} = \sum_{b=(t,r)\in\mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}} V_b S_b W_b^* + \sum_{b=(t,r)\in\mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}} \chi_t M \chi_r. \tag{8.18}$$

Since we use the matrices X_b and Y_b of the semi-uniform representation only in leaf blocks, we also have to embed a variant of the matrix backward transformation into Algorithm 54 (cf. Lemma 8.3).

Algorithm 54. Coarsening of the block structure.

procedure Coarsen(b, M, ϵ, **var** V_b, S_b, W_b);
$(t, r) \leftarrow b$;
if sons($\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b$) $= \emptyset$ **then**
  if $b \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ **then**
    CoarsenLeaf(b, M, ϵ_b, V_b, S_b, W_b); {Algorithm 52}
  end if
else
  for $b' = (t', r') \in$ sons($\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b$) **do**
    $X_{b'} \leftarrow X_{b'} + \chi_{t'} X_b F^*_{B,r',r}$; $Y_{b'} \leftarrow Y_{b'} + \chi_{r'} Y_b E^*_{A,t',t}$;
    Coarsen(b', M, ϵ, $V_{b'}$, $S_{b'}$, $W_{b'}$)
  end for;
  if $b \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ **then**
    CoarsenSubdivided(b, M, ϵ_b, V_b, S_b, W_b) {Algorithm 53}
  end if
end if

Lemma 8.14 (Error estimate). *Let $\tilde{M}$ be the matrix (8.18) constructed by Algorithm 54 with the error tolerances $(\epsilon_b)_{b\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}}$. We have*

$$\|M - \tilde{M}\|_2 \leq \sum_{b\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}} \epsilon_b,$$

if the matrices are approximated with respect to the spectral norm, and

$$\|M - \tilde{M}\|_F \leq \sum_{b\in\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}} \epsilon_b,$$

if the Frobenius norm is used instead.

Proof. We introduce the intermediate matrices

$$\tilde{M}_b := \begin{cases} \chi_t M \chi_r & \text{if } b \in \mathcal{L}^-_{C,\mathcal{I}\times\mathcal{K}}, \\ \sum_{b' \in \operatorname{sons}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}, b)} \tilde{M}_{b'} & \text{if } b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}_{C,\mathcal{I}\times\mathcal{K}}, \\ V_b S_b W_b^* & \text{otherwise} \end{cases} \quad \text{for } b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$$

and observe $\tilde{M} = \tilde{M}_{\operatorname{root}(\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}})}$. A closer look at the definition of $\tilde{M}_b$ yields that $\tilde{M}_b$ is the approximation of the submatrix $\chi_t M \chi_r$ computed by Algorithm 54 for a block $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$.

We assume that matrices are approximated with respect to the spectral norm and now prove

$$\|\chi_t M \chi_r - \tilde{M}_b\|_2 \leq \sum_{b' \in \operatorname{sons}^*(b)} \epsilon_{b'} \tag{8.19}$$

by induction on $\#\operatorname{sons}^*(b) \in \mathbb{N}$. We start by considering blocks $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ with $\#\operatorname{sons}^*(b) = 1$. In this case, we have $\operatorname{sons}(b) = \emptyset$, i.e., b is a leaf. If it is inadmissible, we change nothing an get an error of zero. Otherwise we apply Algorithm 52 which guarantees (8.14) and therefore (8.19) directly.

Let now $n \in \mathbb{N}$, and assume that the induction assumption holds for all $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ with $\#\operatorname{sons}^*(b) \leq n$. Let $b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ with $\#\operatorname{sons}^*(b) = n + 1$. Due to $\#\operatorname{sons}^*(b) = n + 1 > 1$, we find that b is not a leaf, so Algorithm 54 computes approximations for all blocks $b' \in \operatorname{sons}(b)$. Due to $\operatorname{sons}^*(b') \subseteq \operatorname{sons}^*(b) \setminus \{b\}$, we have $\#\operatorname{sons}^*(b') \leq n$ and can apply the induction assumption to all of these submatrices:

$$\begin{aligned} \|\chi_t M \chi_r - \tilde{M}_b\|_2 &= \left\| \sum_{b'=(t',r') \in \operatorname{sons}(b)} (\chi_{t'} M \chi_{r'} - \tilde{M}_{b'}) + \sum_{b' \in \operatorname{sons}(b)} \tilde{M}_{b'} - \tilde{M}_b \right\|_2 \\ &\leq \sum_{b'=(t',r') \in \operatorname{sons}(b)} \|\chi_{t'} M \chi_{r'} - \tilde{M}_{b'}\|_2 + \left\| \sum_{b' \in \operatorname{sons}(b)} \tilde{M}_{b'} - \tilde{M}_b \right\|_2 \\ &\leq \sum_{b'=(t',r') \in \operatorname{sons}(b)} \sum_{b^* \in \operatorname{sons}^*(b')} \epsilon_{b^*} + \left\| \sum_{b' \in \operatorname{sons}(b)} \tilde{M}_{b'} - \tilde{M}_b \right\|_2. \end{aligned}$$

If $b \notin \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ or $b \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$, $\tilde{M}_b$ is constructed from the submatrices $\tilde{M}_{b'}$ by Algorithm 53 and the norm on the right is bounded by ϵ_b. Otherwise, no further approximation takes place and this norm is equal to zero. In both cases, we can conclude

$$\|\chi_t M \chi_r - \tilde{M}_b\|_2 \leq \epsilon_b + \sum_{b' \in \operatorname{sons}(b)} \sum_{b^* \in \operatorname{sons}^*(b')} \epsilon_{b^*} = \sum_{b^* \in \operatorname{sons}^*(b)} \epsilon_{b^*}$$

and have completed the induction.

For the Frobenius norm, we can apply exactly the same argument. □

Complexity estimates

We have seen that Algorithm 54 can provide us with a good $\mathcal{H}$-matrix approximation $\widetilde{M}$ of the matrix $M = C + AB$. Now we have to investigate the number of operations required to reach this goal.

Lemma 8.15. *Let $(k_{C,t})_{t\in\mathcal{T}_\mathcal{I}}$, $(l_{C,r})_{r\in\mathcal{T}_\mathcal{K}}$, $(k_{A,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(l_{B,r})_{r\in\mathcal{T}_\mathcal{K}}$ be defined as in (3.16) and (3.18). Let*

$$\hat{k} := \max\{k_{C,t}, l_{C,r}, k_{A,t}, l_{B,r} \ : \ t \in \mathcal{T}_\mathcal{I},\ r \in \mathcal{T}_\mathcal{K}\}.$$

Computing low-rank approximations $V_b S_b W_b^*$ for all leaves $b \in \mathcal{L}_{M,\mathcal{I}\times\mathcal{K}}$ of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ by applying Algorithm 52 requires not more than a total of

$$C_{\mathrm{lfc}} C_{\mathrm{sp}}^2 \hat{k}^2((p_\mathcal{I}+1)n_\mathcal{I} + (p_\mathcal{K}+1)n_\mathcal{K})$$

operations for a constant $C_{\mathrm{lfc}} \in \mathbb{R}_{>0}$ depending only on C_{qr} and C_{svdf} (cf. Lemma 5.17 and Remark 8.12).

Proof. Let $b = (t,r) \in \mathcal{L}_{M,\mathcal{I}\times\mathcal{K}}$ be a leaf of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. If b is inadmissible, Algorithm 52 uses the singular value decomposition to compute a low-rank approximation. Since $b = (t,r)$ is an inadmissible leaf, both t and r have to be leaves of $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{K}$, respectively, and since the cluster trees are bounded, we have $\#\hat{r} \leq k_{C,r} \leq \hat{k}$. According to Remark 8.12, the singular value decomposition can be found in not more than

$$C_{\mathrm{svdf}}(\#\hat{t})(\#\hat{r})\min\{\#\hat{t},\#\hat{r}\} \leq C_{\mathrm{svdf}}(\#\hat{t})(\#\hat{r})^2 \leq C_{\mathrm{svdf}}\hat{k}^2\#\hat{t}$$

operations.

If b is admissible, Algorithm 52 computes the matrix $X_{1,b} = V_{C,t}\widehat{S}_{C,b}$, and this can be done in $2(\#\hat{t})(\#K_{C,t})(\#L_{C,r}) \leq 2\hat{k}^2\#\hat{t}$ operations.

Next, Algorithm 15 is used to compute the QR factorization of Z_b. Since Z_b has $\#\hat{t}$ rows and

$$(\#L_{C,r}) + (\#L_{B,r}) + (\#K_{A,t}) \leq 3\hat{k}$$

columns, this computation requires not more than

$$C_{\mathrm{qr}}(\#\hat{t})(3\hat{k})^2 \leq 9C_{\mathrm{qr}}\hat{k}^2\#\hat{t}$$

operations.

The matrix $\widehat{M}_{1,b}$ is the product of $W_{C,r} \in \mathbb{R}_{\hat{r}}^{\mathcal{K}\times L_{C,r}}$ and $\widehat{R}_{1,b}^* \in \mathbb{R}^{L_{C,r}\times\hat{K}_b}$, adding it to $\widehat{M}_b$ takes not more than $2(\#\hat{r})(\#L_{C,r})(\#\hat{K}_b) \leq 6\hat{k}^2\#\hat{r}$ operations since $\#\hat{K}_b \leq 3\hat{k}$. The matrix $\widehat{M}_{2,b}$ is the product of $W_{B,r} \in \mathbb{R}_{\hat{r}}^{\mathcal{K}\times L_{B,r}}$ and $\widehat{R}_{2,b}^* \in \mathbb{R}^{L_{B,r}\times\hat{K}_b}$ and can be added in $2(\#\hat{r})(\#L_{B,r})(\#\hat{K}_b) \leq 6\hat{k}^2\#\hat{r}$ operations. The matrix $\widehat{M}_{3,b}$ is the product

of $Y_b \in \mathbb{R}_{\hat{r}}^{\mathcal{K}\times K_{A,t}}$ and $\widehat{R}_{3,b}^* \in \mathbb{R}^{K_{A,t}\times\hat{K}_b}$ and can be added in $2(\#\hat{r})(\#K_{A,t})(\#\hat{K}_b) \leq 6\hat{k}^2\#\hat{r}$ operations. We conclude that $\widehat{M}_b$ can be constructed in

$$18\hat{k}^2\#\hat{r}$$

operations.

Once $\widehat{M}_b$ is available, Algorithm 51 is applied, and according to Remark 8.12 it requires not more than

$$C_{\mathrm{svdf}}(\#\hat{r})(\#K_b)^2 \leq C_{\mathrm{svdf}}(\#\hat{r})(3\hat{k})^2 = 9C_{\mathrm{svdf}}\hat{k}^2\#\hat{r}$$

operations. The matrix V_b is the product of Q_b and $\widehat{V}_b$, and this product can be computed in

$$2(\#\hat{t})(\#\hat{K}_b)(\#K_b) \leq 2(\#\hat{t})(3\hat{k})^2 = 18\hat{k}^2\#\hat{t}$$

operations. Adding the bounds yields that the algorithm requires not more than

$$(20 + 9C_{\mathrm{qr}})\hat{k}^2\#\hat{t} + (18 + 9C_{\mathrm{svdf}})\hat{k}^2\#\hat{r}$$

operations for one block $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$.

In order to get a bound for the entire algorithm, we add the bounds for all blocks. Combining the structure of the proof of Lemma 3.31 with the estimate of Lemma 8.8, we find

$$\sum_{b=(t,r)\in\mathcal{L}^+_{M,\mathcal{I}\times\mathcal{K}}} \hat{k}^2(\#\hat{t} + \#\hat{r}) \leq 3C_{\mathrm{sp}}^2\hat{k}^2((p_{\mathcal{I}} + 1)n_{\mathcal{I}} + (p_{\mathcal{K}} + 1)n_{\mathcal{K}}) \tag{8.20}$$

and setting $C_{\mathrm{lfc}} := 3\max\{20 + 9C_{\mathrm{qr}}, 18 + 9C_{\mathrm{svdf}}\}$ completes the proof. □

Lemma 8.16 (Complexity). *Let $(k_{C,t})_{t\in\mathcal{T}_{\mathcal{I}}}$, $(l_{C,r})_{r\in\mathcal{T}_{\mathcal{K}}}$, $(k_{A,t})_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(l_{B,r})_{r\in\mathcal{T}_{\mathcal{K}}}$ be defined as in (3.16) and (3.18). Let*

$$\hat{k} := \max\{k_{C,t}, l_{C,r}, k_{A,t}, l_{B,r}, k_b \ : \ t \in \mathcal{T}_{\mathcal{I}},\ r \in \mathcal{T}_{\mathcal{K}}, b \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}\},$$
$$C_{\mathrm{bc}} := \max\{\#\operatorname{sons}(t), \#\operatorname{sons}(r) \ : \ t \in \mathcal{T}_{\mathcal{I}},\ r \in \mathcal{T}_{\mathcal{K}}\}.$$

Computing an $\mathcal{H}$-matrix approximation of $M = C + AB$ with the block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ using Algorithm 54 requires not more than

$$(C_{\mathrm{lfc}} + C_{\mathrm{sbc}}C_{\mathrm{bc}}^4)C_{\mathrm{sp}}^2\hat{k}^2((p_{\mathcal{I}} + 1)n_{\mathcal{I}} + (p_{\mathcal{K}} + 1)n_{\mathcal{K}})$$

operations for a constant $C_{\mathrm{sbc}} \in \mathbb{R}_{>0}$ and the constant C_{lfc} defined in Lemma 8.15. Both constants depend only on C_{qr} and C_{svdf}.

Proof. Due to Lemma 8.15, the compression of all leaf blocks of $\mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$ can be handled in

$$C_{\mathrm{lfc}} C_{\mathrm{sp}}^2 \hat{k}^2 ((p_{\mathcal{I}}+1)n_{\mathcal{I}} + (p_{\mathcal{K}}+1)n_{\mathcal{K}})$$

operations, so we only have to consider non-leaf blocks treated by Algorithm 53.

Let $b = (t,r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. If $b \in \mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$, no arithmetic operations are needed. Otherwise Algorithm 53 constructs the matrix $Z_{t'}$ using not more than

$$\sum_{r'\in\mathrm{sons}^+(r)} 2(\#\hat{t}')(\#K_{t',r'})^2 \leq 2C_{\mathrm{bc}}(\#\hat{t}')\hat{k}^2$$

operations. Due to

$$\#K_{t',*} = \# \bigcup_{r'\in\mathrm{sons}^+(r)} K_{t',r'} \leq \sum_{r'\in\mathrm{sons}^+(r)} \#K_{t',r'} \leq C_{\mathrm{bc}}\hat{k},$$

the matrix $Z_{t'}$ has not more than $C_{\mathrm{bc}}\hat{k}$ columns, and Lemma 5.17 yields that Algorithm 15 requires not more than

$$C_{\mathrm{qr}}(\#\hat{t}')(C_{\mathrm{bc}}\hat{k})^2 = C_{\mathrm{qr}}C_{\mathrm{bc}}^2(\#\hat{t}')\hat{k}^2$$

operations to compute $Q_{t'}$ and $\hat{R}_{t'}$. The matrix $\hat{M}_{t'}$ can be assembled in

$$\sum_{r'\in\mathrm{sons}^+(r)} 2(\#\hat{r}')(\#K_{t',r'})(\#\hat{K}_{t'}) \leq 2 \sum_{r'\in\mathrm{sons}^+(r)} (\#\hat{r}')\hat{k}(C_{\mathrm{bc}}\hat{k}) = 2C_{\mathrm{bc}}(\#\hat{r})\hat{k}^2$$

operations. This procedure is repeated for all $t' \in \mathrm{sons}(t)$, and adding the estimates yields the bound

$$\sum_{t'\in\mathrm{sons}(t)} (2C_{\mathrm{bc}} + C_{\mathrm{qr}}C_{\mathrm{bc}}^2)(\#\hat{t}')\hat{k}^2 + 2C_{\mathrm{bc}}(\#\hat{r})\hat{k}^2 \leq (2C_{\mathrm{bc}} + C_{\mathrm{qr}}C_{\mathrm{bc}}^2)(\#\hat{t})\hat{k}^2 + 2C_{\mathrm{bc}}^2(\#\hat{r})\hat{k}^2$$

for the construction of the matrices Q_b and $\hat{M}_b$. Due to Remark 8.12, the matrices W_b, S_b and $\hat{V}_b$ can be computed by Algorithm 51 in

$$C_{\mathrm{svdf}}(\#\hat{r})(\#\hat{K}_b)^2 \leq C_{\mathrm{svdf}}(\#\hat{r})(C_{\mathrm{bc}}^2\hat{k})^2 = C_{\mathrm{svdf}}C_{\mathrm{bc}}^4(\#\hat{r})\hat{k}^2$$

operations due to

$$\#\hat{K}_b = \# \bigcup_{t'\in\mathrm{sons}^+(t)} \hat{K}_{t'} = \sum_{t'\in\mathrm{sons}^+(t)} \#\hat{K}_{t'} \leq \sum_{t'\in\mathrm{sons}^+(t)} C_{\mathrm{bc}}\hat{k} \leq C_{\mathrm{bc}}^2\hat{k}.$$

Finally the algorithm constructs V_b in not more than

$$2(\#\hat{t})(\#\hat{K}_b)(\#K_b) \leq 2(\#\hat{t})(C_{\mathrm{bc}}^2\hat{k})(C_{\mathrm{bc}}^2\hat{k}) = 2C_{\mathrm{bc}}^4(\#\hat{t})\hat{k}^2$$

operations. Adding the estimates gives us the bound

$$\begin{aligned}(2C_{\mathrm{bc}} + C_{\mathrm{qr}}C_{\mathrm{bc}}^2)(\#\hat{t})\hat{k}^2 + 2C_{\mathrm{bc}}^2(\#\hat{r}) + C_{\mathrm{svdf}}C_{\mathrm{bc}}^4(\#\hat{r})\hat{k}^2 + 2C_{\mathrm{bc}}^4(\#\hat{t})\hat{k}^2 \\ \leq \max\{2C_{\mathrm{bc}} + C_{\mathrm{qr}}C_{\mathrm{bc}}^2 + 2C_{\mathrm{bc}}^4, 2C_{\mathrm{bc}}^2 + C_{\mathrm{svdf}}C_{\mathrm{bc}}^4\}(\#\hat{t} + \#\hat{r})\hat{k}^2 \\ \leq \max\{4 + C_{\mathrm{qr}}, 2 + C_{\mathrm{svdf}}\}C_{\mathrm{bc}}^4(\#\hat{t} + \#\hat{r})\hat{k}^2\end{aligned}$$

for the number of operations used by Algorithm 53 applied to a block $b = (t, r) \in \mathcal{T}_{M,\mathcal{I}\times\mathcal{K}}$. Using (8.20) yields the bound

$$C_{\mathrm{sbc}}C_{\mathrm{bc}}^4C_{\mathrm{sp}}^2\hat{k}^2((p_{\mathcal{I}} + 1)n_{\mathcal{I}} + (p_{\mathcal{K}} + 1)n_{\mathcal{K}})$$

for the number of operations required by all calls to Algorithm 53, where

$$C_{\mathrm{sbc}} := 3\max\{4 + C_{\mathrm{qr}}, 2 + C_{\mathrm{svdf}}\}.$$

Adding this estimate to the one given by Lemma 8.15 completes the proof. □

8.4 Construction of adaptive cluster bases

In the first step of the adaptive multiplication algorithm, we have computed the exact product by Algorithm 50 using $\mathcal{O}(\hat{k}^2((p_{\mathcal{I}} + 1)n_{\mathcal{I}} + n_{\mathcal{J}} + (p_{\mathcal{K}} + 1)n_{\mathcal{K}}))$ operations.

In the second step, we have approximated the exact product by an $\mathcal{H}$-matrix $\widetilde{M}$ of local rank $\hat{k}$ with the prescribed block cluster tree $\mathcal{T}_{C,\mathcal{I}\times\mathcal{K}}$ using $\mathcal{O}(\hat{k}^2((p_{\mathcal{I}} + 1)n_{\mathcal{I}} + (p_{\mathcal{K}} + 1)n_{\mathcal{K}}))$ operations.

Now we have to approximate this intermediate $\mathcal{H}$-matrix by an $\mathcal{H}^2$-matrix, i.e., we have to find suitable cluster bases and compute the $\mathcal{H}^2$-matrix best approximation of $\widetilde{M}$ in the corresponding $\mathcal{H}^2$-matrix space.

Fortunately, we have already solved both problems: assuming that $\widetilde{M}$ can be approximated by an $\mathcal{H}^2$-matrix with $\hat{k}$-bounded cluster bases, Algorithm 27 computes adaptive cluster bases in $\mathcal{O}(\hat{k}^2(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_{\mathcal{I}} + n_{\mathcal{K}}))$ operations (cf. Lemma 6.25), and Algorithm 12 computes the best approximation of $\widetilde{M}$ in the corresponding space in $\mathcal{O}(\hat{k}^2(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_{\mathcal{I}} + n_{\mathcal{K}}))$ operations (cf. Lemma 5.9).

We can conclude that computing the adaptive $\mathcal{H}^2$-matrix approximation of the product $M = C + AB$ can be computed in $\mathcal{O}(\hat{k}^2(p_{\mathcal{I}} + p_{\mathcal{I}\times\mathcal{J}} + p_{\mathcal{K}} + 1)(n_{\mathcal{I}} + n_{\mathcal{K}}) + \hat{k}^2 n_{\mathcal{J}})$ operations.

Due to the structure of Algorithm 27, the computation of a cluster basis for a cluster $t \in \mathcal{T}_{\mathcal{I}}$ requires information on *all* admissible blocks $b = (t^+, r) \in \mathcal{L}^+_{C,\mathcal{I}\times\mathcal{K}}$ connected to ancestors $t^+ \in \mathrm{pred}(t)$, i.e., we essentially have to store the $\mathcal{H}$-matrix representation of $\widetilde{M}$ explicitly, i.e., $\mathcal{O}(\hat{k}(p_{\mathcal{I}\times\mathcal{J}} + 1)(n_{\mathcal{I}} + n_{\mathcal{K}}))$ units of auxiliary storage are required.

In order to avoid the need for this large amount of temporary storage, the blockwise compression Algorithm 33 can be used. Combining this procedure with the coarsening

and multiplication routines allows us to avoid storing both the intermediate semi-uniform matrix M_0 and the $\mathcal{H}$-matrix approximation $\widetilde{M}$, since we can convert each admissible block into an $\mathcal{H}^2$-matrix as soon as it becomes available. According to Theorem 6.32, this requires $\mathcal{O}(\hat{k}^2(p_{\mathcal{I}\times\mathcal{J}}+1)(n_{\mathcal{I}}+n_{\mathcal{K}}))$ operations, and the order of complexity is not changed if we use the blockwise compression Algorithm 33 instead of the direct Algorithm 27.

8.5 Numerical experiments

We test the a posteriori matrix-matrix multiplication algorithm by applying it to compressed approximations of the single and double layer potential matrices V and K on the sphere and the cube. The results for the approximation of $X := V^2$ are given in Table 8.1: $\hat{\epsilon} \approx n^{-1/2}$ is the error tolerance for the compression algorithm, the column "A post." gives the time in seconds for the matrix-matrix multiplication with adaptive cluster bases, the columns "Mem" and "Mem/n" the total storage requirements in MB and the requirements per degree of freedom in KB, and the column ϵ gives the relative spectral error $\|X - V^2\|_2/\|V^2\|_2$ of the product.

Table 8.1. A posteriori and a priori multiplication for the single layer potential on the unit sphere (top half) and the unit cube (bottom half.)

n	$\hat{\epsilon}$	A post.	A prio.	Mem	Mem/n	ϵ
512	2.0_{-4}	0.4	0.3	1.9	3.7	2.8_{-5}
2048	1.0_{-4}	14.6	9.2	9.8	4.9	4.0_{-5}
8192	5.0_{-5}	113.3	53.3	42.6	5.3	2.4_{-5}
32768	2.0_{-5}	685.0	249.0	183.7	5.7	9.0_{-6}
131072	1.0_{-5}	3626.3	1030.9	753.7	5.9	5.5_{-6}
768	2.0_{-4}	1.0	0.7	3.6	4.8	4.8_{-5}
3072	1.0_{-4}	20.8	8.7	12.4	4.1	5.7_{-5}
12288	5.0_{-5}	166.3	81.2	96.6	8.1	2.9_{-5}
49152	2.0_{-5}	1041.7	356.5	415.4	8.7	9.9_{-6}
196608	1.0_{-5}	5661.1	1316.3	1599.9	8.3	4.6_{-6}

We can combine the new multiplication algorithm with the approach discussed in Chapter 7: according to Lemma 6.23, we can assume that the cluster bases chosen by the adaptive algorithm will be almost optimal, therefore computing the best approximation of the exact product in the corresponding $\mathcal{H}^2$-matrix space should yield good results.

According to Theorem 7.19, we can expect the a priori multiplication Algorithm 45 to compute this best approximation in optimal complexity. The column "A prio." in Table 8.1 gives the time in seconds required by Algorithm 45 to compute the best

approximation of the product with respect to the quasi-optimal cluster bases constructed by the adaptive multiplication algorithm.

We can see that the algorithms work as expected: the time for the adaptive algorithm grows like $\mathcal{O}(n \log(n))$, the time for the a priori algorithm grows like $\mathcal{O}(n)$. The storage requirements seem also to grow like $\mathcal{O}(n)$. The measured error ϵ is always below the prescribed tolerance $\hat{\epsilon}$, and this indicates that the error control works reliably.

Now we perform the same experiment with the double layer potential matrix K instead of V. The results given in Table 8.2 show that the adaptive algorithm works as expected: the time required for the multiplication is almost linear, and the measured error ϵ is always bounded by the prescribed tolerance $\hat{\epsilon}$. Using the cluster bases chosen by the adaptive algorithm, the a priori multiplication Algorithm 45 can compute approximations of the product that are far better than the ones given in Table 7.2 for the non-adaptive approach.

Table 8.2. A posteriori and a priori multiplication for the double layer potential on the unit sphere (top half) and the unit cube (bottom half).

n	$\hat{\epsilon}$	A post.	A prio.	Mem	Mem/n	ϵ
512	2.0_{-4}	0.4	0.3	1.9	3.8	8.9_{-6}
2048	1.0_{-4}	16.6	11.9	11.0	5.5	1.1_{-5}
8192	5.0_{-5}	140.2	78.5	49.4	6.2	1.1_{-5}
32768	2.0_{-5}	895.2	384.6	216.4	6.8	7.4_{-6}
131072	1.0_{-5}	5245.4	1709.7	916.1	7.2	5.5_{-6}
768	2.0_{-4}	1.3	0.9	3.8	5.0	1.3_{-5}
3072	1.0_{-4}	29.0	14.7	16.6	5.5	1.3_{-5}
12288	5.0_{-5}	202.3	96.6	107.3	8.9	1.2_{-5}
49152	2.0_{-5}	1184.7	383.8	445.0	9.3	8.9_{-6}
196608	1.0_{-5}	5896.7	1268.8	1664.4	8.7	6.4_{-6}

In a final experiment, we compare the adaptive multiplication algorithm of Chapter 8, the projected multiplication algorithm of Chapter 7 and the $\mathcal{H}$-matrix multiplication algorithm [49], [52]. Table 8.3 lists computing times and accuracies for the adaptive algorithm ("Adaptive"), the a priori algorithm combined with the cluster bases provided by the adaptive algorithm ("A priori/new") and the $\mathcal{H}$-matrix algorithm. We can see that the $\mathcal{H}$-matrix algorithm yields accuracies that are slightly better than those of the other algorithms, but also that it takes very long to complete. The adaptive algorithm finds $\mathcal{H}^2$-matrix approximations that are almost as good as the ones of the $\mathcal{H}$-matrix algorithm, but it is faster and exhibits a better scaling behaviour. The a priori algorithm is easily the fastest, but it is also very inaccurate if the wrong kind of cluster basis is used. In combination with the cluster bases provided by the adaptive algorithm, the a priori algorithm is very fast and yields very good approximations.

Table 8.3. Adaptive and non-adaptive multiplication algorithms for the single and double layer potential on the cube.

Oper.	n	Adaptive		A priori/new		$\mathcal{H}$-arithmetic	
VV	768	0.7	conv.	0.7	conv.	0.7	conv.
	3072	35.8	1.8_{-5}	32.6	1.1_{-5}	153.0	1.7_{-5}
	12288	276.6	3.1_{-5}	72.3	3.0_{-5}	824.8	2.5_{-5}
	49152	1343.7	4.1_{-5}	240.5	4.0_{-5}	6591.3	2.5_{-5}
	196608	6513.3	4.7_{-5}	805.5	4.6_{-5}	29741.6	2.5_{-5}
KV	768	0.7	conv.	0.7	conv.	0.7	conv.
	3072	37.6	2.7_{-5}	37.2	2.6_{-5}	152.9	5.5_{-5}
	12288	270.3	3.5_{-5}	87.6	3.4_{-5}	755.5	5.4_{-5}
	49152	1267.0	4.3_{-5}	291.6	4.2_{-5}	5688.3	5.6_{-5}
	196608	5933.5	9.0_{-5}	989.9	9.0_{-5}	26404.6	5.5_{-5}
VK	768	0.7	conv.	0.7	conv.	0.7	conv.
	3072	37.8	3.2_{-5}	40.2	3.1_{-5}	150.2	3.3_{-5}
	12288	267.1	3.3_{-5}	94.4	3.3_{-5}	737.4	3.7_{-5}
	49152	1219.5	4.5_{-5}	310.4	4.4_{-5}	5886.5	3.4_{-5}
	196608	5734.5	8.0_{-5}	1067.0	7.8_{-5}	27303.6	3.6_{-5}
KK	768	0.7	conv.	0.7	conv.	0.7	conv.
	3072	39.8	6.8_{-6}	44.0	6.6_{-6}	151.4	5.3_{-6}
	12288	272.4	1.6_{-5}	100.3	1.6_{-5}	658.2	6.4_{-6}
	49152	1327.9	3.2_{-5}	324.5	3.2_{-5}	5066.8	1.4_{-5}
	196608	5791.6	4.5_{-5}	1080.8	4.5_{-5}	24862.1	1.5_{-5}

Chapter 9
Application to elliptic partial differential operators

According to Theorem 6.21 and Corollary 6.22, we can approximate a matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ by an efficient $\mathcal{H}^2$-matrix if the total cluster bases of X and X^* can be approximated by low rank. We have already seen (e.g., in Chapter 4 and Lemma 6.39) that these assumptions hold for integral operators.

Now we turn our attention to elliptic partial differential operators. The discretization of a partial differential operator $\mathcal{L}$ by a standard finite element scheme always leads to a matrix L in which for all blocks $b = (t, s)$ satisfying even the fairly weak admissibility condition

$$\operatorname{dist}(\Omega_t, \Omega_s) > 0$$

the equation $\chi_t X \chi_s = 0$ (cf. Definition 3.20) holds, i.e., each admissible block can be "approximated" by rank zero without any loss. Therefore the matrix L is an $\mathcal{H}^2$-matrix with trivial cluster bases.

The inverse matrix L^{-1} is more interesting: in typical situations, it corresponds to the non-local inverse of the partial differential operator, and we can expect most of its entries to be non-zero. In order to be able to handle L^{-1} efficiently, we need a data-sparse representation, and our goal in this chapter is to prove that an $\mathcal{H}^2$-matrix can be used to approximate L^{-1} up to a certain accuracy proportional to the discretization error. This limitation is due to the structure of the proof, it is not experienced in numerical experiments.

The proof is based on an approximation result [6] for the solution operator $\mathcal{L}^{-1}$ corresponding to the equation: if τ and σ are subdomains satisfying a suitable admissibility condition and if the support of a functional f is contained in σ, the restriction of $\mathcal{L}^{-1} f$ to τ can be approximated efficiently in a low-dimensional space.

The operator $\mathcal{L}^{-1}$ and the matrix L^{-1} are connected by the Galerkin projection: applying this projection and its adjoint from the left and right to $\mathcal{L}^{-1}$, respectively, directly yields L^{-1} due to Galerkin orthogonality. Unfortunately, the Galerkin projection is typically a non-local mapping, therefore local approximation properties of $\mathcal{L}^{-1}$ would be lost by this procedure.

In order to fix this problem, we replace the Galerkin projection by a different mapping into the discrete space. In [6], the L^2-orthogonal projection is used, which is at least quasi-local (i.e., exhibits exponential decay as the distance to the support grows) and leads to an error estimate for the approximation of L^{-1} by an $\mathcal{H}$-matrix. Since the L^2-projection is only quasi-local, the construction of the blockwise error estimates needed by $\mathcal{H}^2$-matrix approximation theory is complicated.

In [15] a different approach is presented: the Galerkin projection is replaced by a Clément-type interpolation operator, and we get a new matrix S approximating the

inverse L^{-1}. Since the interpolation operators are local, they can be used to easily derive the blockwise error estimates we need. Since the operators are also L^2-stable, they provide approximations that are almost as good as those of the L^2-orthogonal projection.

This chapter is organized as follows:

- Section 9.1 introduces a model problem for an elliptic partial differential equations with non-smooth coefficients.
- Section 9.2 describes a construction for a low-rank approximation of the solution operator of the partial differential equation.
- Section 9.3 uses this result to find low-rank approximations of admissible submatrices of the discrete solution operator S.
- Section 9.4 applies the error estimates of Theorem 6.16 and Corollary 6.17 to prove that the discrete solution operator S can be approximated by an efficient $\mathcal{H}^2$-matrix.
- Numerical experiments are described together with the approximative inversion algorithm in Section 10.5.

Assumptions in this chapter: We assume that a cluster tree $\mathcal{T}_\mathcal{I}$ for the finite index $\mathcal{I}$ is given. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$. Let $n_\mathcal{I} := \#\mathcal{I}$ and $c_\mathcal{I} := \#\mathcal{T}_\mathcal{I}$ denote the number of indices and clusters of $\mathcal{I}$ and $\mathcal{T}_\mathcal{I}$. Let $p_\mathcal{I}$ be the depth of $\mathcal{T}_\mathcal{I}$.

9.1 Model problem

We fix a domain $\Omega \subseteq \mathbb{R}^d$ and a coefficient function $C : \Omega \to \mathbb{R}^{d\times d}$ satisfying

$$C(x) = C(x)^*, \quad \sigma(C(x)) \subseteq [\alpha, \beta] \quad \text{for all } x \in \Omega.$$

We are interested in the partial differential operator

$$\mathcal{L}u := -\sum_{i,j=1}^{d} \partial_i C_{ij} \partial_j u$$

mapping the Sobolev space $H_0^1(\Omega)$ into $H^{-1}(\Omega)$. For $f \in H^{-1}(\Omega)$, the partial differential equation

$$\mathcal{L}u = f \tag{9.1}$$

is equivalent to the variational equation

$$a(v,u) := \int_\Omega \langle \nabla v(x), C(x)\nabla u(x)\rangle_2 \, dx = f(v) \quad \text{for all } v \in H_0^1(\Omega). \tag{9.2}$$

The bounds for the spectrum of C imply

$$\alpha\|w\|_2^2 \leq \langle C(x)w, w\rangle_2 = \|C(x)^{1/2}w\|_2^2 \leq \beta\|w\|_2^2, \quad \text{for all } w \in \mathbb{R}^d.$$

Combining this inequality with the Cauchy–Schwarz inequality provides us with the upper bound

$$\begin{aligned}\langle \nabla v(x), C(x)\nabla u(x)\rangle_2 &= \langle C(x)^{1/2}\nabla v(x), C(x)^{1/2}\nabla u(x)\rangle_2 \\ &\leq \|C(x)^{1/2}\nabla v(x)\|_2 \|C(x)^{1/2}\nabla u(x)\|_2 \\ &\leq \beta\|\nabla v(x)\|_2\|\nabla u(x)\|_2,\end{aligned}$$

and the definition of the Sobolev space $H^1(\Omega)$ yields

$$|a(v,u)| \leq \beta\|v\|_{H^1(\Omega)}\|u\|_{H^1(\Omega)} \quad \text{for all } u, v \in H^1(\Omega).$$

This means that the bilinear form a is bounded, i.e., continuous. Due to

$$\langle \nabla u(x), C(x)\nabla u(x)\rangle_2 \geq \alpha\|\nabla u(x)\|_2^2,$$

Friedrichs' inequality implies the existence of a constant $C_\Omega \in \mathbb{R}_{>0}$ depending only on the domain Ω such that

$$a(u,u) \geq \alpha\|\nabla u\|_{L^2(\Omega)}^2 \geq C_\Omega\alpha\|u\|_{H^1(\Omega)}^2 \quad \text{for all } u \in H_0^1(\Omega),$$

i.e., a is a coercive bilinear form, therefore (9.2) and the equivalent (9.1) possess unique solutions [34].

Usually, strongly elliptic partial differential equations of the type (9.1) are treated numerically by a finite element method: a mesh Ω_h for the domain Ω is constructed, and basis functions $(\varphi_i)_{i\in\mathcal{I}}$ are used to define a finite-dimensional space

$$V_n := \operatorname{span}\{\varphi_i \,:\, i \in \mathcal{I}\} \subseteq H_0^1(\Omega),$$

where $\mathcal{I}$ is a finite index set and $n := \#\mathcal{I}$ is the dimension of the discrete space V_n.

Using the standard Galerkin approach, an approximation $u_n \in V_n$ of u is represented in the form

$$u_n = \sum_{i\in\mathcal{I}} x_i\varphi_i$$

for the solution vector $x \in \mathbb{R}^{\mathcal{I}}$ of the linear system

$$Lx = b \tag{9.3}$$

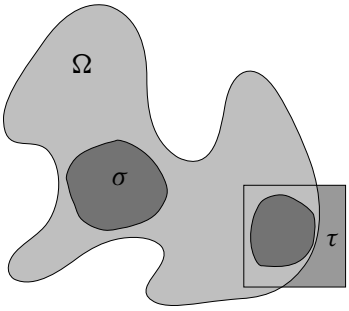
given by the stiffness matrix $L \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and the load vector $b \in \mathbb{R}^{\mathcal{I}}$ defined by

$$L_{ij} = a(\varphi_i, \varphi_j), \quad b_i = f(\varphi_i) \quad \text{for all } i, j \in \mathcal{I}. \tag{9.4}$$

The system (9.3) can be solved by several techniques, e.g., by fast direct solvers [93], multigrid iterations [61] or $\mathcal{H}$- and $\mathcal{H}^2$-matrix methods [62], [52], [18], [15]. $\mathcal{H}$- and $\mathcal{H}^2$-matrix techniques offer the advantage that they can handle jumps and anisotropies in the coefficient matrix C better than multigrid techniques and that they are more efficient than direct solvers for large problems.

9.2 Approximation of the solution operator

Let $\tau \subseteq \mathbb{R}^d$ be a convex set with $\tau \cap \Omega \neq \emptyset$. Let $\sigma \subseteq \Omega$ be a subset with $\operatorname{dist}(\tau, \sigma) > 0$.



Let $\epsilon \in \mathbb{R}_{>0}$. We are looking for a low-dimensional space $V \subseteq H^1(\tau)$ such that for each right-hand side $f \in H^{-1}(\Omega)$ with $\operatorname{supp} f \subseteq \sigma$ the corresponding solution $u \in H_0^1(\Omega)$ of the variational equation (9.2) can be approximated in V, i.e., such that there exists a $v \in V$ with

$$\|u - v\|_{H^1(\tau)} \leq \epsilon \|f\|_{H^{-1}(\Omega)}.$$

Since V is required to be independent of f, this property implies that the interaction between the domains τ and σ can be described by a low-rank operator.

If the coefficient function C and the boundary of Ω were sufficiently smooth, interior regularity estimates would yield an estimate of the form

$$\|u|_\tau\|_{H^m(\tau)} \leq C \left(\frac{c}{\operatorname{dist}(\tau, \sigma)}\right)^m m! \|f\|_{H^{-1}(\Omega)} \quad \text{for all } m \in \mathbb{N}_0$$

and we could simply approximate $u|_\tau$ by a polynomial $\tilde{u}$ of order m. In this setting, the space V would have a dimension $\lesssim m^d$ and the approximation $\tilde{u}$ would converge

exponentially with respect to the order m if an admissibility condition of the type (4.11) or (4.37) holds.

In the general case, we have to use a refined approach first presented in [6]: since $u \in H^1(\Omega)$ holds, we can approximate the solution by a piecewise constant function, but the convergence rate will not be exponential. Projecting this function into a local space of L-harmonic functions (cf. Definition 9.1 below) yields an approximation v_1. We can apply a weak interior regularity argument to show that $v_1|_{\tau_1}$ is contained in $H^1(\tau_1)$ for a subset $\tau_1 \subseteq \Omega$, therefore the error $u_1 := u|_{\tau_1} - v_1|_{\tau_1}$ is also an L-harmonic function in $H^1(\tau_1)$, and the argument can be repeated until a sufficiently accurate approximation $v := v_1 + \cdots + v_p$ has been found.

The key element of the proof is the space of locally L-harmonic functions:

Definition 9.1 (Locally L-harmonic functions). Let $\omega \subseteq \mathbb{R}^d$ be a domain (that may be unrelated to Ω). A function $u \in L^2(\omega)$ is called *locally L-harmonic* on ω if for all $\tilde{\omega} \subseteq \omega$ with $\operatorname{dist}(\tilde{\omega}, \partial\omega) > 0$ the following conditions hold:

$$u|_{\tilde{\omega}} \in H^1(\tilde{\omega}), \tag{9.5a}$$

$$a(v, u|_\Omega) = 0 \qquad \text{for all } v \in H_0^1(\Omega) \text{ with } \operatorname{supp} v \subseteq \tilde{\omega}, \tag{9.5b}$$

$$u|_{\omega\setminus\Omega} = 0. \tag{9.5c}$$

The space of all locally L-harmonic functions on ω is denoted by $\mathcal{H}(\omega)$.

For functions in $\mathcal{H}(\omega)$, the following weak interior regularity estimate holds (cf. Lemma 2.4 in [6]):

Lemma 9.2 (Cacciopoli inequality). *Let $u \in \mathcal{H}(\omega)$, and let $\tilde{\omega} \subseteq \omega$ be a domain with $\operatorname{dist}(\tilde{\omega}, \partial\omega) > 0$. Then we have $u|_{\tilde{\omega}} \in H^1(\tilde{\omega})$ and*

$$\|\nabla u\|_{L^2(\tilde{\omega})} \leq \frac{c_{\mathrm{reg}}}{\operatorname{dist}(\tilde{\omega}, \partial\omega)} \|u\|_{L^2(\omega)}, \quad c_{\mathrm{reg}} := 4\sqrt{\beta/\alpha} \geq 4.$$

Proof. Let $\delta := \operatorname{dist}(\tilde{\omega}, \partial\omega)$. Let $\eta \in C^1(\Omega \cup \omega)$ be a function satisfying

$$0 \leq \eta \leq 1, \quad \eta|_{\tilde{\omega}} \equiv 1, \quad \|\nabla\eta\|_2 \leq 2/\delta,$$
$$\eta(x) = 0 \quad \text{for all } x \in \omega \text{ with } \operatorname{dist}(x, \partial\omega) < \delta/4.$$

Such a function exists since the distance between the subdomain $\tilde{\omega}$ and the boundary of ω is $\delta > 0$.

For the domain

$$\hat{\omega} := \{x \in \omega \ : \ \operatorname{dist}(x, \partial\omega) > \delta/8\},$$

we have $\operatorname{dist}(\hat{\omega}, \partial\omega) \geq \delta/8 > 0$, so (9.5a) implies $u|_{\hat{\omega}} \in H^1(\hat{\omega})$, and since η is continuously differentiable and bounded, we have $v := \eta^2 u \in H_0^1(\omega)$ and can extend this function by zero to $H_0^1(\Omega \cup \omega)$. Due to (9.5c), we get $v|_\Omega \in H_0^1(\Omega)$ with $\operatorname{supp} v \subseteq \hat{\omega}$ and $u|_\Omega \in H_0^1(\Omega)$, therefore we can use (9.5b) in order to prove

$$0 = a(v|_\Omega, u|_\Omega) = \int_{\hat{\omega}\cap\Omega} \langle C(x)\nabla(\eta^2 u)(x), \nabla u(x)\rangle_2 \, dx$$

$$= \int_{\hat{\omega}\cap\Omega} \langle C(x)(2\eta u\nabla\eta + \eta^2\nabla u)(x), \nabla u\rangle_2 \, dx.$$

Moving the second term of this sum to the left side of the equation yields

$$\begin{aligned}
\int_{\hat{\omega}\cap\Omega} \eta(x)^2 \|C(x)^{1/2}\nabla u(x)\|_2^2 \, dx &= \int_{\hat{\omega}\cap\Omega} \eta(x)^2 \langle C(x)\nabla u(x), \nabla u(x)\rangle_2 \, dx \\
&= -2\int_{\hat{\omega}\cap\Omega} \eta(x)u(x)\langle C(x)\nabla\eta(x), \nabla u(x)\, dx \\
&\le 2\int_{\hat{\omega}\cap\Omega} \eta(x)|u(x)| \, \|C(x)^{1/2}\nabla\eta(x)\|_2 \|C(x)^{1/2}\nabla u(x)\|_2 \, dx \\
&\le 2\beta^{1/2}\int_{\hat{\omega}\cap\Omega} \eta(x)|u(x)| \, \|\nabla\eta(x)\|_2 \|C(x)^{1/2}\nabla u(x)\|_2 \, dx \\
&\le 4\frac{\beta^{1/2}}{\delta}\int_{\hat{\omega}\cap\Omega} \eta(x)|u(x)| \, \|C(x)^{1/2}\nabla u(x)\|_2 \, dx \\
&\le 4\frac{\beta^{1/2}}{\delta}\|u|_{\hat{\omega}\cap\Omega}\|_{L^2(\hat{\omega}\cap\Omega)} \left(\int_{\hat{\omega}\cap\Omega} \eta(x)^2 \|C(x)^{1/2}\nabla u(x)\|_2^2 \, dx\right)^{1/2}
\end{aligned}$$

Dividing both sides by the rightmost factor (if it is not zero) yields

$$\left(\int_{\hat{\omega}\cap\Omega} \eta(x)^2 \|C(x)^{1/2}\nabla u(x)\|_2^2 \, dx\right)^{1/2} \le 4\frac{\beta^{1/2}}{\delta}\|u|_{\hat{\omega}\cap\Omega}\|_{L^2(\hat{\omega}\cap\Omega)}.$$

Due to $\eta|_{\tilde{\omega}} \equiv 1$ and (9.5c), we get

$$\begin{aligned}
\|\nabla u|_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} = \|\nabla u|_{\tilde{\omega}\cap\Omega}\|_{L^2(\tilde{\omega}\cap\Omega)} &= \left(\int_{\tilde{\omega}\cap\Omega} \eta(x)^2 \|\nabla u(x)\|_2^2 \, dx\right)^{1/2} \\
&\le \alpha^{-1/2}\left(\int_{\tilde{\omega}\cap\Omega} \eta(x)^2 \|C(x)^{1/2}\nabla u(x)\|_2^2 \, dx\right)^{1/2} \\
&\le \alpha^{-1/2}\left(\int_{\hat{\omega}\cap\Omega} \eta(x)^2 \|C(x)^{1/2}\nabla u(x)\|_2^2 \, dx\right)^{1/2} \\
&\le \frac{4}{\delta}\left(\frac{\beta}{\alpha}\right)^{1/2} \|u|_{\hat{\omega}\cap\Omega}\|_{L^2(\hat{\omega}\cap\Omega)} \le \frac{4\sqrt{\beta/\alpha}}{\delta}\|u\|_{L^2(\omega)}.
\end{aligned}$$

This is the required estimate. □

As mentioned before, we use orthogonal projections to map functions from $L^2(\omega)$ into $\mathcal{H}(\omega)$. The construction of these projections is straightforward if $\mathcal{H}(\omega)$ is a complete set, i.e., closed in $L^2(\omega)$. Using Lemma 9.2, this property can be proven (the proof is a variant of that of Lemma 2.2 in [6] which requires only elementary tools):

Lemma 9.3. *The space $\mathcal{H}(\omega)$ is a closed subspace of $L^2(\omega)$.*

Proof. Let $(u_n)_{n\in\mathbb{N}}$ be a Cauchy sequence in $\mathcal{H}(\omega)$ with respect to the $L^2(\omega)$-norm. Since $L^2(\omega)$ is complete, we can find a function $u \in L^2(\omega)$ with

$$\lim_{n\to\infty} \|u_n - u\|_{L^2(\omega)} = 0. \tag{9.6}$$

Let $\tilde{\omega} \subseteq \omega$ be a domain with $\operatorname{dist}(\tilde{\omega}, \partial\omega) > 0$. Lemma 9.2 implies

$$\begin{aligned} \|v|_{\tilde{\omega}}\|_{H^1(\tilde{\omega})} &= \left(\|v|_{\tilde{\omega}}\|^2_{L^2(\tilde{\omega})} + \|\nabla v|_{\tilde{\omega}}\|^2_{L^2(\tilde{\omega})}\right)^{1/2} \\ &\leq C\,\|v\|_{L^2(\omega)} \quad \text{for all } v \in \mathcal{H}(\omega) \end{aligned} \tag{9.7}$$

with the constant

$$C := \left(1 + \frac{16\beta/\alpha}{\operatorname{dist}(\tilde{\omega}, \partial\omega)^2}\right)^{1/2},$$

therefore we have

$$\|u_n|_{\tilde{\omega}} - u_m|_{\tilde{\omega}}\|_{H^1(\tilde{\omega})} \leq C\,\|u_n - u_m\|_{L^2(\omega)} \quad \text{for all } n, m \in \mathbb{N}$$

and conclude that $(u_n|_{\tilde{\omega}})_{n\in\mathbb{N}}$ is a Cauchy sequence with respect to the $H^1(\tilde{\omega})$-norm. Since $H^1(\tilde{\omega})$ is complete, we can find a function $u_{\tilde{\omega}} \in H^1(\tilde{\omega})$ with

$$\lim_{n\to\infty} \|u_n|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{H^1(\tilde{\omega})} = 0. \tag{9.8}$$

Since the restriction to $\tilde{\omega}$ is a continuous mapping from $L^2(\omega)$ to $L^2(\tilde{\omega})$, (9.6) implies

$$\lim_{n\to\infty} \|u_n|_{\tilde{\omega}} - u|_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} = 0.$$

Combining this property with (9.8) and the estimate

$$\begin{aligned} \|u|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} &= \|u|_{\tilde{\omega}} - u_n|_{\tilde{\omega}} + u_n|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} \\ &\leq \|u|_{\tilde{\omega}} - u_n|_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} + \|u_n|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} \\ &\leq \|u|_{\tilde{\omega}} - u_n|_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} + \|u_n|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{H^1(\tilde{\omega})} \quad \text{for all } n \in \mathbb{N} \end{aligned}$$

yields $\|u|_{\tilde{\omega}} - u_{\tilde{\omega}}\|_{L^2(\tilde{\omega})} = 0$, i.e., $u|_{\tilde{\omega}} = u_{\tilde{\omega}} \in H^1(\tilde{\omega})$.

Let now $v \in H^1_0(\Omega)$ with $\operatorname{supp} v \subseteq \tilde{\omega}$. We have just proven that $u|_{\tilde{\omega}} \in H^1(\tilde{\omega})$ is the limit of $u_n|_{\tilde{\omega}}$ with respect to the H^1-norm, therefore (9.5b) and the continuity of a imply

$$a(v, u|_\Omega) = \lim_{n\to\infty} a(v, u_n|_\Omega) = 0.$$

The continuity of the restriction from $L^2(\omega)$ to $L^2(\omega \setminus \Omega)$ yields

$$u|_{\omega\setminus\Omega} = \lim_{n\to\infty} u_n|_{\omega\setminus\Omega} = 0,$$

and we can conclude that $u \in \mathcal{H}(\omega)$ holds, therefore $\mathcal{H}(\omega)$ is closed. □

We introduce the maximum-norm diameter

$$\begin{aligned}\operatorname{diam}_\infty(\omega) &:= \sup\{\|x-y\|_\infty \ : \ x,y\in\omega\}\\ &= \sup\{|x_i-y_i| \ : \ x,y\in\omega, i\in\{1,\dots,d\}\end{aligned}$$

and can now state the basic approximation result (the proof is a slight modification of the proof of Lemma 2.6 in [6]):

Lemma 9.4 (Finite-dimensional approximation). *Let $\omega\subseteq\mathbb{R}^d$ be a convex domain. Let $\ell\in\mathbb{N}$. Let Z be a closed subspace of $L^2(\omega)$. There is a space $V\subseteq Z$ with $\dim(V)\leq\ell^d$ such that for all $u\in Z\cap H^1(\omega)$ a function $v\in V$ can be found with*

$$\|u-v\|_{L^2(\omega)}\leq c_{\mathrm{apx}}\frac{\operatorname{diam}_\infty(\omega)}{\ell}\|\nabla u\|_{L^2(\omega)},\quad c_{\mathrm{apx}}:=\frac{2\sqrt{d}}{\pi}.$$

Proof. We let $\delta:=\operatorname{diam}_\infty(\omega)$ and introduce

$$a_i:=\inf\{x_i \ : \ x\in\omega\}\quad\text{for all } i\in\{1,\dots,d\}.$$

By definition, we have

$$x_i-a_i=|x_i-a_i|\leq\delta\quad\text{for all } i\in\{1,\dots,d\}, x\in\omega,$$

i.e., the d-dimensional hypercube $Q:=a+[0,\delta]^d$ satisfies $\omega\subseteq Q$.

We let $S_Q:=\{1,\dots,\ell\}^d$ and define

$$Q_\nu:=a+\bigotimes_{i=1}^d\left(\frac{\nu_i-1}{\ell}\delta,\frac{\nu_i}{\ell}\delta\right)\quad\text{for all } \nu\in S_Q$$

and observe that $(Q_\nu)_{\nu\in S_Q}$ is a family of ℓ^d disjoint open hypercubes satisfying $\operatorname{diam}_\infty(Q_\nu)=\delta/\ell$ and $\operatorname{diam}(Q_\nu)=\sqrt{d}\delta/\ell$ that covers Q up to a null set.

For each $\nu\in S_Q$, we let $\omega_\nu:=\omega\cap Q_\nu$. Defining

$$S_\omega:=\{\nu\in S_Q \ : \ |\omega_\nu|>0\},$$

we have found that $(\omega_\nu)_{\nu\in S_\omega}$ is a family of not more than ℓ^d convex sets that covers ω up to a null set.

We construct an intermediate approximation by piecewise constant functions defined on $(\omega_\nu)_{\nu\in S_\omega}$, i.e., by functions in the space

$$W:=\{w\in L^2(\omega) \ : \ w|_{\omega_\nu}\text{ is constant almost everywhere for all } \nu\in S_\omega\}\subseteq L^2(\omega).$$

For a function $u\in L^2(\omega)$, we let

$$w_\nu:=\frac{1}{|\omega_\nu|}\int_{\omega_\nu}u(x)\,dx\quad\text{for all } \nu\in S_\omega$$

and introduce the piecewise constant approximation $w \in W$ by

$$w(x) := w_\nu \quad \text{for all } \nu \in S_\omega, x \in \omega_\nu.$$

Due to $u \in H^1(\omega)$, the Poincaré inequality yields

$$\begin{aligned}\int_{\omega_\nu} |u(x) - w_\nu|^2 &\leq \frac{\operatorname{diam}(\omega_\nu)^2}{\pi^2} \int_{\omega_\nu} \|\nabla u(x)\|_2^2 \, dx \\ &\leq \frac{d\delta^2}{\pi^2 \ell^2} \int_{\omega_\nu} \|\nabla u(x)\|_2^2 \, dx \quad \text{for all } \nu \in S_\omega\end{aligned}$$

and summing over all $\nu \in S_\omega$ yields

$$\|u - w\|_{L^2(\omega)} \leq \frac{\sqrt{d}\delta}{\pi \ell} \|\nabla u\|_{L^2(\omega)}.$$

This is already the desired estimate, but w is not necessarily contained in the space Z.

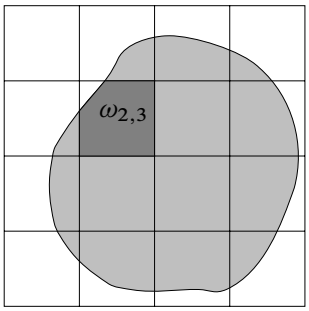


Figure 9.1. Domains for the piecewise constant approximation in Lemma 9.4.

Since Z is a closed subspace of $L^2(\omega)$, the orthogonal projection $\Pi_Z : L^2(\omega) \to Z$ defined by

$$\langle \Pi_Z f, g \rangle_{L^2(\omega)} = \langle f, g \rangle_{L^2(\omega)} \quad \text{for all } f \in L^2(\omega), g \in Z$$

exists and satisfies $\|\Pi_Z\|_{L^2(\omega) \leftarrow L^2(\omega)} \leq 1$ and $\Pi_Z g = g$ for all $g \in Z$.

We let $V := \Pi_Z W$ and $v := \Pi_Z w \in V$ and conclude

$$\|u - v\|_{L^2(\omega)} = \|\Pi_Z u - \Pi_Z w\|_{L^2(\omega)} \leq \|u - w\|_{L^2(\omega)} \leq \frac{\sqrt{d}\delta}{\pi \ell} \|\nabla u\|_{L^2(\omega)},$$

i.e., V is a subspace of Z with a dimension not larger than ℓ^d satisfying the desired estimate. □

Combining the approximation result of Lemma 9.4 with the regularity result of Lemma 9.2 allows us to find finite-dimensional spaces approximating the solutions of the variational equation (9.2):

Theorem 9.5 (Low-rank approximation). *Let $\eta \in \mathbb{R}_{>0}$ and $q \in (0,1)$. There are constants $C_{\mathrm{apx}}, C_{\dim} \in \mathbb{R}_{>0}$ such that for all convex open domains $\tau \subseteq \mathbb{R}^d$ and all $p \in \mathbb{N}_{\geq 2}$, we can find a space $V \subseteq \mathcal{H}(\tau)$ satisfying*

$$\dim V \leq C_{\dim} p^{d+1}. \tag{9.9}$$

For all domains $\sigma \subseteq \Omega$ with

$$\operatorname{diam}_\infty(\tau) \leq 2\eta \operatorname{dist}(\tau, \sigma) \tag{9.10}$$

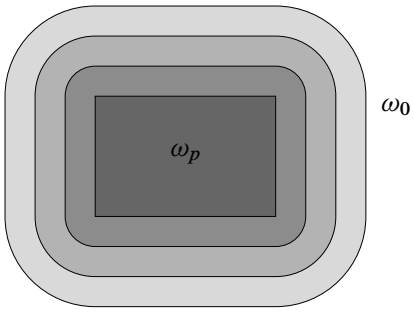
and all right-hand sides $f \in H^{-1}(\Omega)$ with $\operatorname{supp} f \subseteq \sigma$, the corresponding solution $u \in H^1_0(\Omega)$ of the variational equation (9.2) can be approximated by a function $v \in V$ with

$$\|\nabla u|_\tau - \nabla v|_\tau\|_{L^2(\tau)} \leq C_{\mathrm{apx}} q^p \|f\|_{H^{-1}(\Omega)}, \tag{9.11}$$

$$\|u|_\tau - v|_\tau\|_{H^1(\tau)} \leq C_{\mathrm{apx}} (\operatorname{dist}(\tau,\sigma)/8 + 1) q^p \|f\|_{H^{-1}(\Omega)}. \tag{9.12}$$

Proof. Let $\tau \subseteq \mathbb{R}^d$ be a convex open domain, let $\sigma \subseteq \Omega$ be a domain satisfying (9.10), let $\delta := \operatorname{diam}(\tau)/(2\eta)$, and let $p, \ell \in \mathbb{N}$. We introduce the domains

$$\omega_i := \left\{ x \in \mathbb{R}^d \;:\; \operatorname{dist}(x,\tau) < \frac{(p-i)\delta}{p} \right\} \quad \text{for all } i \in \{0,\dots,p\}.$$



By construction we have $\tau \subseteq \omega_p$, $\omega_i \subseteq \omega_{i-1}$ for all $i \in \{1,\dots,p\}$ and $\omega_0 \cap \sigma = \emptyset$. In order to apply Lemma 9.2, we need an estimate for the distance between the boundaries of these subdomains. Let $i \in \{1,\dots,p\}$, $x \in \omega_i$ and $y \in \partial\omega_{i-1}$. Due to $\operatorname{dist}(x,\tau) < (p-i)\delta/p$, we can find $z \in \tau$ with

$$\|x - z\|_2 \leq \frac{(p-i)\delta}{p}$$

Due to $\operatorname{dist}(y,\tau) = (p-i+1)\delta/p$ and $z \in \tau$, we have

$$\|y-z\|_2 \geq \frac{(p-i+1)\delta}{p}$$

and conclude

$$\|x-y\|_2 \geq \|y-z\|_2 - \|z-x\|_2 \geq \frac{(p-i+1)\delta}{p} - \frac{(p-i)\delta}{p} = \frac{\delta}{p},$$

i.e., $\operatorname{dist}(\omega_i, \partial\omega_{i-1}) \geq \delta/p$.

Let $f \in H^{-1}(\Omega)$ with $\operatorname{supp} f \subseteq \sigma$. Let $u \in H^1_0(\Omega)$ be the corresponding solution of the variational equation (9.2). We define $u_0 \in L^2(\omega_0)$ by letting

$$u_0|_{\omega_0\cap\Omega} := u|_{\omega_0}, \quad u_0|_{\omega_0\setminus\Omega} := 0.$$

Since u equals zero on $\partial\Omega$, we get $u_0 \in H^1(\omega_0)$. For all $v \in H^1_0(\Omega)$ with $\operatorname{supp} v \subseteq \tilde{\omega}$, we have $\operatorname{supp} v \cap \operatorname{supp} f = \emptyset$ and therefore

$$a(v, u_0|_\Omega) = a(v,u) = f(v) = 0,$$

so we can conclude $u_0 \in \mathcal{H}(\omega_0)$.

We now construct $u_i \in \mathcal{H}(\omega_i)$ and $v_i \in V_i \subseteq \mathcal{H}(\omega_{i-1})$ for all $i \in \{1,\dots,p\}$ such that $u_i = (u_{i-1} - v_i)|_{\omega_i}$ and

$$\begin{aligned} \|u_i\|_{L^2(\omega_i)} &\leq \frac{2c_{\mathrm{apx}}(\eta+1)\delta}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})}, \\ \|\nabla u_i\|_{L^2(\omega_i)} &\leq c\frac{p}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})} \end{aligned} \tag{9.13}$$

hold for the constant $c := 2c_{\mathrm{reg}}c_{\mathrm{apx}}(\eta+1)$.

Let $i \in \{1,\dots,p\}$ and assume that $u_{i-1} \in \mathcal{H}(\omega_{i-1})$ is given. We apply Lemma 9.4 to find a space $V_i \subseteq \mathcal{H}(\omega_{i-1})$ with $\dim V_i \leq \ell^d$ and a function $v_i \in V_i$ with

$$\begin{aligned} \|u_{i-1}-v_i\|_{L^2(\omega_{i-1})} &\leq c_{\mathrm{apx}}\frac{\operatorname{diam}_\infty(\omega_{i-1})}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})} \\ &\leq c_{\mathrm{apx}}\frac{\operatorname{diam}_\infty(\tau)+2\delta}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})}. \end{aligned}$$

By definition, we have $\operatorname{diam}_\infty(\tau) \leq 2\eta\delta$, and the estimate becomes

$$\|u_{i-1}-v_i\|_{L^2(\omega_{i-1})} \leq c_{\mathrm{apx}}\frac{2(\eta+1)\delta}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})}. \tag{9.14}$$

According to Lemma 9.2, the restriction $u_i := (u_{i-1}-v_i)|_{\omega_i}$ of the error $u_{i-1}-v_i$ is contained in $\mathcal{H}(\omega_i)$ and the interior regularity estimate

$$\begin{aligned} \|\nabla u_i\|_{L^2(\omega_i)} &\leq \frac{c_{\mathrm{reg}}}{\operatorname{dist}(\omega_i,\partial\omega_{i-1})}\|u_{i-1}-v_i\|_{L^2(\omega_{i-1})} \\ &\leq c_{\mathrm{reg}}\frac{p}{\delta}c_{\mathrm{apx}}\frac{2(\eta+1)\delta}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})} = c\frac{p}{\ell}\|\nabla u_{i-1}\|_{L^2(\omega_{i-1})}. \end{aligned}$$

holds. Restricting the left side of (9.14) to the subdomain $\omega_i \subseteq \omega_{i-1}$ completes the induction and we have proven (9.13) for all $i \in \{1, \dots, p\}$.

Iterating the second estimate of (9.13) yields

$$\|\nabla u_i\|_{L^2(\omega_i)} \leq \left(c\frac{p}{\ell}\right)^i \|\nabla u_0\|_{L^2(\omega_0)} \quad \text{for all } i \in \{1, \dots, p\}.$$

By construction, we have

$$u_p|_\tau = u_{p-1}|_\tau - v_p|_\tau = u_{p-2}|_\tau - v_{p-1}|_\tau - v_p|_\tau = \cdots = u_0|_\tau - v_1|_\tau - \cdots - v_p|_\tau,$$

so $v := v_1|_\tau + \cdots + v_p|_\tau$ is an approximation of $u_0|_\tau = u|_\tau$. We define the space

$$V := V_1|_\tau + \cdots + V_p|_\tau,$$

where the restriction of the space is interpreted as the space spanned by the restriction of its elements, and get $v \in V$ with $\dim V \leq p\ell^d$. The estimates (9.13) imply

$$\begin{aligned}
\|u|_\tau - v\|_{L^2(\tau)} = \|u_p|_\tau\|_{L^2(\tau)} &\leq \frac{2c_{\mathrm{apx}}(\eta+1)\delta}{\ell}\|\nabla u_{p-1}\|_{L^2(\omega_{i-1})} \\
&\leq \frac{2c_{\mathrm{apx}}(\eta+1)\delta}{\ell}\left(c\frac{p}{\ell}\right)^{p-1}\|\nabla u_0\|_{L^2(\omega_0)} \\
&= \frac{\delta}{c_{\mathrm{reg}}p}\left(c\frac{p}{\ell}\right)^{p}\|\nabla u_0\|_{L^2(\omega_0)}, \\
\|\nabla(u|_\tau - v)\|_{L^2(\tau)} = \|\nabla u_p|_\tau\|_{L^2(\tau)} &\leq \left(c\frac{p}{\ell}\right)^{p}\|\nabla u_0\|_{L^2(\omega_0)}.
\end{aligned}$$

Combining both estimates allows us to bound the full H^1-norm by

$$\begin{aligned}
\|u|_\tau - v\|_{H^1(\tau)} &= \left(\|u|_\tau - v\|^2_{L^2(\tau)} + \|\nabla(u|_\tau - v)|_\tau\|^2_{L^2(\tau)}\right)^{1/2} \\
&\leq \left(\frac{\delta^2}{c_{\mathrm{reg}}^2 p^2} + 1\right)^{1/2}\left(c\frac{p}{\ell}\right)^{p}\|\nabla u_0\|_{L^2(\omega_0)} \\
&\leq \left(\frac{\delta^2}{4^2 2^2} + 1\right)^{1/2}\left(c\frac{p}{\ell}\right)^{p}\|\nabla u_0\|_{L^2(\omega_0)} \\
&\leq (\delta/8 + 1)\left(c\frac{p}{\ell}\right)^{p}\|\nabla u_0\|_{L^2(\omega_0)}.
\end{aligned}$$

Since u is the solution of (9.2), we have

$$\|u\|^2_{H^1(\Omega)} \leq \frac{1}{C_\Omega \alpha} a(u,u) = \frac{1}{C_\Omega \alpha} f(u) \leq \frac{1}{C_\Omega \alpha}\|f\|_{H^{-1}(\Omega)}\|u\|_{H^1(\Omega)},$$

and this implies

$$\|\nabla u_0\|_{L^2(\omega_0)} \leq \|\nabla u\|_{L^2(\Omega)} \leq \|u\|_{H^1(\Omega)} \leq \frac{1}{C_\Omega \alpha}\|f\|_{H^{-1}(\Omega)}.$$

Combining this estimate with the error bounds for $u - v$ yields

$$\|\nabla(u|_\tau - v)\|_{L^2(\tau)} \leq \frac{1}{C_\Omega \alpha} \left(c\frac{p}{\ell}\right)^p \|f\|_{H^{-1}(\Omega)},$$
$$\|u|_\tau - v\|_{H^1(\tau)} \leq \frac{1}{C_\Omega \alpha}(\delta/8 + 1)\left(c\frac{p}{\ell}\right)^p \|f\|_{H^{-1}(\Omega)}.$$

In order to get the estimates (9.9), (9.11) and (9.12), we have to choose ℓ appropriately. A simple approach is to let

$$\ell := \left\lceil \frac{cp}{q} \right\rceil,$$

since this yields

$$c\frac{p}{\ell} \leq c\frac{pq}{cp} = q, \quad \left(c\frac{p}{\ell}\right)^p \leq q^p,$$

and the dimension of V can be bounded by

$$\ell \leq \frac{cp}{q} + 1 \leq \frac{c}{q}p + \frac{1}{2}p = \left(\frac{c}{q} + \frac{1}{2}\right)p$$

due to $p \geq 2$, so setting

$$C_{\mathrm{dim}} := \left(\frac{c}{q} + \frac{1}{2}\right)^d, \quad C_{\mathrm{apx}} := \frac{1}{C_\Omega \alpha}$$

yields $\dim V \leq p\ell^d \leq C_{\mathrm{dim}} p^{d+1}$ and

$$\|\nabla(u|_\tau - v)\|_{L^2(\tau)} \leq C_{\mathrm{apx}} q^p \|f\|_{H^{-1}(\Omega)},$$
$$\|u|_\tau - v\|_{H^1(\tau)} \leq C_{\mathrm{apx}} q^p (\delta/8 + 1)\|f\|_{H^{-1}(\Omega)},$$

therefore the proof is complete. □

This result is closely related to Theorem 2.8 in [6], but it yields an H^1-norm estimate for the solution of the variational equation (9.2) using the H^{-1}-norm of the right-hand side functional instead of an L^2-norm estimate of Green's function. The main difference between both proofs is that the one given here exploits the fact that the original solution u already is L-harmonic in τ, therefore we can perform the approximation by Lemma 9.4 first, and follow it by the regularity estimate of Lemma 9.2 in order to get an H^1-estimate for the error. The proof of [6], Theorem 2.8, on the other hand, deals with Green's function, and this function is not globally in H^1, therefore the first step has to be the regularity estimate and the resulting error bound is given only for the L^2-norm.

Remark 9.6 (Direct L^2-norm estimate). Of course, we can use the same ordering of regularity estimates and approximation steps in Theorem 9.5 in order to get an estimate of the form

$$\|u|_\tau - v\|_{L^2(\tau)} \leq \left(c\frac{p}{\ell}\right)^p \|u_0\|_{L^2(\omega_0)} \leq C_{\mathrm{apx}} q^p \|f\|_{H^{-1}(\Omega)}$$

instead of (9.11). Since the space V constructed in this way would differ from the one used in Theorem 9.5, we cannot simply combine both estimates in order to get an estimate for the full H^1-norm and have to rely on results of the type (9.12) instead. □

Remark 9.7 (Influence of α and β). The bounds α and β for the spectra of the coefficient matrices influence the constants C_{dim} and C_{apx}. Due to

$$c = 2c_{\mathrm{reg}}c_{\mathrm{apx}}(\eta+1) = 8\sqrt{\beta/\alpha}\frac{2\sqrt{d}}{\pi}(\eta+1),$$

we have

$$C_{\mathrm{dim}} = \left(16\sqrt{\beta/\alpha}\frac{2\sqrt{d}}{q\pi}(\eta+1)+\frac{1}{2}\right)^d$$

and can expect $C_{\mathrm{dim}} \lesssim (\beta/\alpha)^{d/2}$ for $\beta \gg \alpha$.

The constant α also appears in the definition of C_{apx}, and we get $C_{\mathrm{apx}} \lesssim 1/\alpha$. □

9.3 Approximation of matrix blocks

The problem of proving that the $\mathcal{H}^2$-matrix arithmetic algorithms yield a sufficiently accurate approximation of the inverse can be reduced to an existence result: since the adaptive arithmetic operations (cf. Section 8 and [49]) have a best-approximation property, we only have to show that an approximation of L^{-1} by an $\mathcal{H}^2$-matrix exists, because this already implies that the computed approximation $\widetilde{S}$ will be at least as good as this approximation.

This proof of existence can be accomplished using our main result stated in Theorem 9.5: a block $\chi_t L^{-1}\chi_s$ describes the mapping from a right-hand side vector b with support in s to the restriction of the corresponding discrete solution to t. In order to apply our approximation result, we have to exploit the relationship between the inverse matrix L^{-1} and the inverse operator $\mathcal{L}^{-1}$.

In [6], this problem is solved by applying L^2-orthogonal projections. Since these projections are non-local operators, additional approximation steps are required, which increase the rank, lead to sub-optimal error estimates, and make the overall proof quite complicated.

We follow the approach presented in [15]: instead of a non-local L^2-projection, a Clément-type interpolation operator [35] can be used to map continuous functions into the discrete space. These operators are "sufficiently local" to provide us with improved error estimates and guarantee that the rank of the approximation will not deteriorate.

In order to keep the presentation simple, we assume that the finite element mesh is shape-regular in the sense of [38], Definition 2.2. It is not required to be quasi-uniform.

Let us recall the basic definitions and properties of Clément interpolation operators: we fix a family $(\lambda_i)_{i\in\mathcal{I}}$ of functionals mapping $L^2(\Omega)$ into $\mathbb{R}$ such that

$$\operatorname{supp}\lambda_i \subseteq \operatorname{supp}\varphi_i \quad \text{for all } i \in \mathcal{I}, \tag{9.15}$$

holds, that the local projection property

$$\lambda_i(\varphi_j) = \delta_{ij} \quad \text{for all } i, j \in \mathcal{I} \tag{9.16}$$

is satisfied and also that the local stability property

$$\|\lambda_i(u)\varphi_i\|_{L^2(\Omega)} \leq C_{\mathrm{cs}}\|u\|_{L^2(\operatorname{supp}\varphi_i)} \quad \text{for all } i \in \mathcal{I}, u \in L^2(\Omega) \tag{9.17}$$

holds for a constant $C_{\mathrm{cs}} \in \mathbb{R}_{>0}$ depending only on the shape-regularity of the mesh. Constructions of this kind can be found in [95], [8].

The interpolation operator is defined by

$$\mathfrak{I}\colon L^2(\Omega) \to V_n, \quad u \mapsto \sum_{i\in\mathcal{I}} \lambda_i(u)\varphi_i. \tag{9.18}$$

The local projection property (9.16) implies its global counterpart

$$\mathfrak{I}v_n = v_n \quad \text{for all } v_n \in V_n. \tag{9.19}$$

Since the matrices we are dealing with are given with respect to the space $\mathbb{R}^{\mathcal{I}}$, not V_n, we need a way of switching between both spaces. This is handled by the standard basis isomorphism

$$\Phi\colon \mathbb{R}^{\mathcal{I}} \to V_n \subseteq H_0^1(\Omega), \quad x \mapsto \sum_{i\in\mathcal{I}} x_i\varphi_i.$$

The interpolation operator $\mathfrak{I}$ can be expressed by

$$\mathfrak{I} = \Phi\Lambda$$

if we define $\Lambda\colon L^2(\Omega) \to \mathbb{R}^{\mathcal{I}}$ by

$$(\Lambda v)_i := \lambda_i(v) \quad \text{for all } i \in \mathcal{I}, v \in L^2(\Omega).$$

In order to construct the approximation of L^{-1} by using $\mathcal{L}^{-1}$, we turn a vector $b \in \mathbb{R}^{\mathcal{I}}$ into a functional, apply $\mathcal{L}^{-1}$, and approximate the result again in V_h. The first step can be accomplished by using the adjoint of Λ: we define

$$\Lambda^*\colon \mathbb{R}^{\mathcal{I}} \to (L^2(\Omega))', \quad b \mapsto \sum_{i\in\mathcal{I}} b_i\lambda_i.$$

This operator turns each vector in $\mathbb{R}^{\mathcal{I}}$ into a functional on $L^2(\Omega)$, and due to

$$\langle \Lambda^* b, v\rangle = \sum_{i\in\mathcal{I}} b_i\langle\lambda_i, v\rangle = \sum_{i\in\mathcal{I}} b_i\lambda_i(v) = \sum_{i\in\mathcal{I}} b_i(\Lambda v)_i = \langle b, \Lambda v\rangle_2,$$

it is indeed the adjoint of Λ.

If the vector b is given by (9.4) for a right-hand side functional $f \in H^{-1}(\Omega)$, the projection property (9.16) implies

$$(\Lambda^* b)(\varphi_j) = \sum_{i \in \mathcal{I}} b_i \lambda_i(\varphi_j) = b_j = f(\varphi_j) \quad \text{for all } j \in \mathcal{I}, b \in \mathbb{R}^{\mathcal{I}},$$

therefore the functional $\Lambda^* b$ and the original right-hand side f of (9.1) yield the same Galerkin approximation u_n.

We have to prove that Λ^* is a bounded mapping with respect to the correct norms. In order to do so, we first have to prove that $\mathfrak{I}$ is L^2-stable.

The shape-regularity of the mesh implies that there is a constant $C_{\mathrm{sr}} \in \mathbb{N}$ such that

$$\#\{j \in \mathcal{I} : \operatorname{supp}\varphi_i \cap \operatorname{supp}\varphi_j \neq \emptyset\} \leq C_{\mathrm{sr}} \quad \text{for all } i \in \mathcal{I}$$

holds. Since the local finite element spaces are finite-dimensional, there is also a constant $C_{\mathrm{ov}} \in \mathbb{N}$ such that

$$\#\{j \in \mathcal{I} : \varphi_j(x) \neq 0\} \leq C_{\mathrm{ov}} \quad \text{for all } x \in \Omega.$$

Using these estimates, we can not only prove the well-established *global* L^2-stability of the interpolation operator $\mathfrak{I}$, but also localized counterparts on subdomains corresponding to clusters:

Lemma 9.8 (Stability). *Let* $C_{\mathrm{cl}} := C_{\mathrm{cs}}\sqrt{C_{\mathrm{sr}} C_{\mathrm{ov}}}$. *For all* $\hat{t} \subseteq \mathcal{I}$ *and all* $\tau \subseteq \Omega$ *with*

$$\operatorname{supp}\varphi_i \subseteq \tau \quad \textit{for all } i \in \hat{t}$$

we define the local interpolation operator

$$\mathfrak{I}_t : L^2(\tau) \to V_n, \quad v \mapsto \sum_{i \in \hat{t}} \lambda_i(v)\varphi_i.$$

Then we have

$$\|\mathfrak{I}_t v\|_{L^2(\Omega)} \leq C_{\mathrm{cl}} \|v\|_{L^2(\tau)} \quad \textit{for all } v \in L^2(\tau). \tag{9.20}$$

In particular, the interpolation operator $\mathfrak{I}$ *is* L^2*-stable, i.e., satisfies*

$$\|\mathfrak{I} v\|_{L^2(\Omega)} \leq C_{\mathrm{cl}} \|v\|_{L^2(\Omega)} \quad \textit{for all } v \in L^2(\Omega). \tag{9.21}$$

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$ and $v \in L^2(\Omega_t)$. We define the functions χ_i and χ_{ij} by

$$\chi_i : \Omega \to \mathbb{N}, \quad x \mapsto \begin{cases} 1 & \text{if } \varphi_i(x) \neq 0, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for all } i \in \mathcal{I},$$

$$\chi_{ij} : \Omega \to \mathbb{N}, \quad x \mapsto \begin{cases} 1 & \text{if } \varphi_i(x) \neq 0, \varphi_j(x) \neq 0, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i, j \in \mathcal{I}$$

and observe

$$\sum_{i\in\mathcal{I}} \chi_i(x) \leq C_{\mathrm{ov}}, \quad \sum_{i\in\mathcal{I}} \chi_{ij}(x) = \sum_{i\in\mathcal{I}} \chi_{ji}(x) \leq C_{\mathrm{sr}} \quad \text{for all } j \in \mathcal{I}, x \in \Omega.$$

Combining these estimates with Cauchy's inequality yields

$$\begin{aligned}
\|\mathfrak{I}_t v\|_{L^2(\Omega)}^2 &= \Big\| \sum_{i\in\hat{t}} \lambda_i(v)\varphi_i \Big\|_{L^2(\Omega)}^2 = \int_\Omega \Big(\sum_{i\in\hat{t}} \lambda_i(v)\varphi_i(x) \Big)^2 \, dx \\
&= \int_\Omega \sum_{i\in\hat{t}} \sum_{j\in\hat{t}} \lambda_i(v)\lambda_j(v)\varphi_i(x)\varphi_j(x)\, dx \\
&= \int_\Omega \sum_{i\in\hat{t}} \sum_{j\in\hat{t}} \chi_{ij}(x)\lambda_i(v)\lambda_j(v)\varphi_i(x)\varphi_j(x)\, dx \\
&\leq \int_\Omega \Big(\sum_{i\in\hat{t}} \sum_{j\in\hat{t}} \chi_{ij}(x)\lambda_i(v)^2\varphi_i(x)^2 \Big)^{1/2} \\
&\qquad \Big(\sum_{i\in\hat{t}} \sum_{j\in\hat{t}} \chi_{ij}(x)\lambda_j(v)^2\varphi_j(x)^2 \Big)^{1/2} dx \\
&= \int_\Omega \sum_{i\in\hat{t}} \sum_{j\in\hat{t}} \chi_{ij}(x)\lambda_i(v)^2\varphi_i(x)^2\, dx \leq C_{\mathrm{sr}} \sum_{i\in\hat{t}} \int_\Omega \lambda_i(v)^2\varphi_i(x)^2\, dx \\
&= C_{\mathrm{sr}} \sum_{i\in\hat{t}} \|\lambda_i(v)\varphi_i\|_{L^2(\Omega)}^2.
\end{aligned}$$

Now we apply the local stability estimate (9.17) to get

$$\begin{aligned}
\|\mathfrak{I}_t v\|_{L^2(\Omega)}^2 &\leq C_{\mathrm{sr}} C_{\mathrm{cs}}^2 \sum_{i\in\hat{t}} \|v\|_{L^2(\operatorname{supp}\varphi_i)}^2 = C_{\mathrm{sr}} C_{\mathrm{cs}}^2 \sum_{i\in\hat{t}} \int_{\operatorname{supp}\varphi_i} v(x)^2\, dx \\
&= C_{\mathrm{sr}} C_{\mathrm{cs}}^2 \sum_{i\in\hat{t}} \int_\tau \chi_i(x) v(x)^2\, dx \leq C_{\mathrm{sr}} C_{\mathrm{ov}} C_{\mathrm{cs}}^2 \int_\tau v(x)^2\, dx \\
&= C_{\mathrm{sr}} C_{\mathrm{ov}} C_{\mathrm{cs}}^2 \|v\|_{L^2(\tau)}^2
\end{aligned}$$

This is equivalent to (9.20).

For $\hat{t} := \mathcal{I}$ and $\tau := \Omega$, this estimate implies (9.21). □

We are interested in bounding $\Lambda^* b$ by a norm of the vector b, so we need a connection between coefficient vectors and the corresponding elements of V_n. Since the finite element mesh is shape-regular, a simple application of Proposition 3.1 in [38] yields that there is a positive definite diagonal matrix $H \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ satisfying

$$C_{\mathrm{b1}} \|H^{d/2} x\|_2 \leq \|\Phi x\|_{L^2(\Omega)} \leq C_{\mathrm{b2}} \|H^{d/2} x\|_2 \quad \text{for all } x \in \mathbb{R}^{\mathcal{I}},$$

where $C_{\mathrm{b1}}, C_{\mathrm{b2}} \in \mathbb{R}_{>0}$ are constants depending only on the shape-regularity of the mesh. Using this inequality, we can prove the necessary properties of Λ^*:

Lemma 9.9 (Λ^* bounded and local). *Let $\varrho \in [0,1]$ and $b \in \mathbb{R}^{\mathcal{I}}$. We have*

$$\|\Lambda^* b\|_{H^{-1+\varrho}(\Omega)} \leq \frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}} \|H^{-d/2} b\|_2, \tag{9.22}$$

i.e., Λ^* is a continuous mapping from $\mathbb{R}^{\mathcal{I}}$ to $H^{-1+\varrho}(\Omega)$.

The mapping preserves locality, i.e., it satisfies

$$\operatorname{supp}(\Lambda^* b) \subseteq \bigcup \{\operatorname{supp} \varphi_i \ : \ i \in \mathcal{I}, b_i \neq 0\}. \tag{9.23}$$

Proof. Let $v \in H_0^{1-\varrho}(\Omega)$. Let $y := \Lambda v$ and $v_n := \Phi y = \mathfrak{I} v$. By the definition of Λ^* we get

$$\begin{aligned}
(\Lambda^* b)(v) &= \langle b, \Lambda v \rangle_2 = \langle b, y \rangle_2 = \langle b, H^{-d/2} H^{d/2} y \rangle_2 \\
&= \langle H^{-d/2} b, H^{d/2} y \rangle_2 \leq \|H^{-d/2} b\|_2 \|H^{d/2} y\|_2 \\
&\leq \frac{\|H^{-d/2} b\|_2}{C_{\mathrm{b1}}} \|\Phi y\|_2 = \frac{\|H^{-d/2} b\|_2}{C_{\mathrm{b1}}} \|v_n\|_{L^2(\Omega)} \\
&= \frac{\|H^{-d/2} b\|_2}{C_{\mathrm{b1}}} \|\mathfrak{I} v\|_{L^2(\Omega)} \leq \frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}} \|H^{-d/2} b\|_2 \|v\|_{L^2(\Omega)},
\end{aligned}$$

and this implies (9.22) due to $\|v\|_{L^2(\Omega)} \leq \|v\|_{H^{1-\varrho}(\Omega)}$.

Due to (9.15) and the definition of Λ^*, we have

$$\operatorname{supp}(\Lambda^* b) = \operatorname{supp}\Big(\sum_{i \in \mathcal{I}} b_i \lambda_i\Big) = \operatorname{supp}\Big(\sum_{\substack{i \in \mathcal{I} \\ b_i \neq 0}} b_i \lambda_i\Big) \subseteq \bigcup_{\substack{i \in \mathcal{I} \\ b_i \neq 0}} \operatorname{supp} \lambda_i \subseteq \bigcup_{\substack{i \in \mathcal{I} \\ b_i \neq 0}} \operatorname{supp} \varphi_i.$$

This is the desired inclusion. □

This result allows us to switch from the vector b corresponding to the discrete setting to the functional f of the variational setting. In the variational setting, we can apply Theorem 9.5 to construct the desired approximation of the solution, then we have to switch back to the discrete setting. Unfortunately, we cannot use the Galerkin projection to perform this last step, which would be the natural choice considering that we want to approximate u_n, since it is a global operator and the approximation result only holds for a subdomain. Therefore we have to rely on the Clément-type interpolation operator again, which has the desired locality property.

Using interpolation instead of the Galerkin projection leads to a second discrete approximation of $\mathcal{L}^{-1}$, given by the matrix

$$S = \Lambda \mathcal{L}^{-1} \Lambda^* \in \mathbb{R}^{\mathcal{I} \times \mathcal{I}}.$$

If we let $b \in \mathbb{R}^{\mathcal{I}}$, $f := \Lambda^* b$, $u := \mathcal{L}^{-1} f$ and $\tilde{u}_n := \mathfrak{I} u$, we observe $\Phi S b = \tilde{u}_n$, i.e., S provides us with the coefficients of the Clément-type approximation of the solution operator. Since $\mathfrak{I}$ is L^2-stable (cf. (9.21)) and a projection (cf. (9.19)), we have the estimate

$$\begin{aligned}
\|\tilde{u}_n - u\|_{L^2(\Omega)} &= \|\mathfrak{I} u - u\|_{L^2(\Omega)} \\
&= \|\mathfrak{I} u - \mathfrak{I} v_n + v_n - u\|_{L^2(\Omega)} \\
&= \|\mathfrak{I}(u - v_n) + (v_n - u)\|_{L^2(\Omega)} \\
&\leq \|\mathfrak{I}(v_n - u)\|_{L^2(\Omega)} + \|v_n - u\|_{L^2(\Omega)} \\
&\leq (C_{\mathrm{cl}} + 1)\|v_n - u\|_{L^2(\Omega)}
\end{aligned} \tag{9.24}$$

for all $v_n \in V_n$, i.e., $\tilde{u}_n$ is close to the best possible approximation of u with respect to the L^2-norm.

In particular we can apply (9.24) to the Galerkin solution u_n and get

$$\|\tilde{u}_n - u\|_{L^2(\Omega)} \leq (C_{\mathrm{cl}} + 1)\|u_n - u\|_{L^2(\Omega)}, \tag{9.25}$$

therefore $\tilde{u}_n$ will converge to the same limit as u_n, and its rate of convergence will be at least as good.

In fact, (9.24) even implies that $\tilde{u}_n$ may converge faster than u_n in situations of low regularity: due to $u \in H^1(\Omega)$, we can always expect $\tilde{u}_n$ to converge at least like $\mathcal{O}(h)$ with respect to the L^2-norm, where h is the maximal meshwidth.

If we assume that the equation (9.1) is $H^{1-\varrho}(\Omega)$-regular, it is possible to derive a refined error estimate for the matrices S and L^{-1}:

Lemma 9.10 (Clément vs. Galerkin). *Let $\varrho \in [0, 1]$. We assume that for all functionals $f \in H^{-1+\varrho}(\Omega)$, the solution $u := \mathcal{L}^{-1} f$ satisfies $u \in H_0^{1+\varrho}(\Omega)$ and*

$$\|u\|_{H^{1+\varrho}(\Omega)} \leq C_{\mathrm{rg}} \|f\|_{H^{-1+\varrho}(\Omega)}. \tag{9.26}$$

Then there is a constant $C_{\mathrm{cg}} \in \mathbb{R}_{>0}$ depending only on C_{rg}, C_{cl}, C_{b1}, and the shape-regularity of the mesh with

$$\|H^{d/2}(S - L^{-1})b\|_2 \leq C_{\mathrm{cg}} h^{2\alpha} \|H^{-d/2} b\|_2 \quad \text{for all } b \in \mathbb{R}^{\mathcal{I}},$$

where $h \in \mathbb{R}_{>0}$ is the maximal meshwidth.

Proof. Let $b \in \mathbb{R}^{\mathcal{I}}$, let $f := \Lambda^* b$ and $u := \mathcal{L}^{-1} f$. Let $x := L^{-1} b$ and $u_n := \Phi x$. Let $\tilde{x} := S b$ and $\tilde{u}_n := \Phi \tilde{x}$. By definition, we have $\tilde{u}_n = \mathfrak{I} u$.

Using the standard Aubin–Nitsche lemma yields

$$\|u - u_n\|_{L^2(\Omega)} \leq C_{\mathrm{an}} h^{2\alpha} \|f\|_{H^{-1+\varrho}(\Omega)},$$

with a constant C_{an} depending only on C_{rg} and the shape-regularity parameters of the mesh. Combining this estimate with (9.25) gives us

$$\begin{aligned}
\|u_n - \tilde{u}_n\|_{L^2(\Omega)} &\leq \|u_n - u\|_{L^2(\Omega)} + \|u - \tilde{u}_n\|_{L^2(\Omega)} \\
&\leq C_{\mathrm{an}}(C_{\mathrm{cl}} + 2) h^{2\alpha} \|f\|_{H^{-1+\alpha}(\Omega)}.
\end{aligned}$$

We observe

$$\begin{aligned}\|H^{d/2}(S-L^{-1})b\|_2 &\leq \frac{1}{C_{\mathrm{b1}}}\|\Phi Sb-\Phi L^{-1}b\|_{L^2(\Omega)}\\ &= \frac{1}{C_{\mathrm{b1}}}\|\tilde{u}_n-u_n\|_{L^2(\Omega)} \leq \frac{C_{\mathrm{an}}(C_{\mathrm{cl}}+2)}{C_{\mathrm{b1}}}h^{2\alpha}\|f\|_{H^{-1+\alpha}(\Omega)}\\ &\leq \frac{C_{\mathrm{an}}(C_{\mathrm{cl}}+2)}{C_{\mathrm{b1}}}h^{2\alpha}\frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}}\|H^{-d/2}b\|_2\end{aligned}$$

and complete the proof be setting $C_{\mathrm{cg}} := C_{\mathrm{an}}C_{\mathrm{cl}}(C_{\mathrm{cl}}+2)/C_{\mathrm{b1}}^2$. □

Due to this result, a good approximation of S on a sufficiently fine mesh is also a good approximation of L^{-1}, and a good approximation of S can be constructed by Theorem 9.5: we pick subsets $\hat{t},\hat{s}\subseteq\mathcal{I}$ and subsets $\tau,\sigma\subseteq\mathbb{R}^d$ with

$$\operatorname{supp}\varphi_i\subseteq\tau,\quad \operatorname{supp}\varphi_j\subseteq\sigma,\quad \tau \text{ is convex}\quad \text{for all } i\in\hat{t},\ j\in\hat{s}. \tag{9.27}$$

Theorem 9.5 yields the following result:

Theorem 9.11 (Blockwise low-rank approximation). *Let $\eta\in\mathbb{R}_{>0}$, $q\in(0,1)$. There are constants $C_{\mathrm{blk}},C_{\mathrm{dim}}\in\mathbb{R}_{>0}$ depending only on η, q, Ω and the shape regularity of the mesh such that for all $\hat{t},\hat{s}\subseteq\mathcal{I}$ and $\tau,\sigma\subseteq\mathbb{R}^d$ satisfying (9.27) and the admissibility condition*

$$\operatorname{diam}(\tau)\leq 2\eta\operatorname{dist}(\tau,\sigma) \tag{9.28}$$

and all $p\in\mathbb{N}_{\geq 2}$ we can find a rank $k\in\mathbb{N}$ with $k\leq C_{\mathrm{dim}}p^{d+1}$ and matrices $X_{t,s}\in\mathbb{R}^{\hat{t}\times k}$, $Y_{t,s}\in\mathbb{R}^{\hat{s}\times k}$ with

$$\|H_t^{d/2}(S|_{\hat{t}\times\hat{s}}-X_{t,s}Y_{t,s}^*)b\|_2\leq C_{\mathrm{blk}}q^p\|H_s^{-d/2}b\|_2\quad \text{for all } b\in\mathbb{R}^{\hat{s}}$$

for $H_t := H|_{\hat{t}\times\hat{t}}$, $H_s := H|_{\hat{s}\times\hat{s}}$, i.e., the submatrix of S corresponding to the block $\hat{t}\times\hat{s}$ can be approximated by a matrix of rank k.

Proof. If $\tau\cap\Omega=\emptyset$ or $\sigma\cap\Omega=\emptyset$, we have $u\equiv 0$ and the error estimates holds for the trivial space $V=\{0\}$. Therefore we can restrict our attention to the case $\tau\cap\Omega\neq\emptyset$, $\sigma\cap\Omega\neq\emptyset$.

Due to Theorem 9.5, there is a space $V\subseteq L^2(\tau)$ with $\dim V\leq C_{\mathrm{dim}}p^{d+1}$ such that for all $f\in H^{-1}(\Omega)$ with $\operatorname{supp}f\subseteq\sigma$ we can find a function $v\in V$ satisfying

$$\|u|_\tau-v\|_{H^1(\tau)}\leq C_{\mathrm{apx}}(\operatorname{dist}(\tau,\sigma)/8+1)q^p\|f\|_{H^{-1}(\Omega)} \tag{9.29}$$

with $u := \mathcal{L}^{-1}f$.

Let $b\in\mathbb{R}^{\hat{s}}$. We extend b to a vector $\hat{b}\in\mathbb{R}^{\mathcal{I}}$ by

$$\hat{b}_j := \begin{cases} b_j & \text{if } j\in\hat{s},\\ 0 & \text{otherwise}\end{cases}\qquad \text{for all } j\in\mathcal{I}.$$

Due to Lemma 9.9, the functional $f := \Lambda^* \hat{b}$ satisfies

$$\operatorname{supp} f \subseteq \sigma, \quad \|f\|_{H^{-1}(\Omega)} \leq \frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}} \|H^{-d/2}\hat{b}\|_2 = \frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}} \|H_s^{-d/2} b\|_2. \tag{9.30}$$

Let $u := \mathcal{L}^{-1} f$, and let $v \in V$ be the local approximation introduced in (9.29). Since v approximates u only locally, we need local variants of Λ and Φ:

$$\Lambda_t \colon L^2(\tau) \to \mathbb{R}^{\hat{t}}, \quad v \mapsto (\lambda_i(v))_{i \in \hat{t}},$$
$$\Phi_t \colon \mathbb{R}^{\hat{t}} \to V_h, \quad y \mapsto \sum_{i \in \hat{t}} y_i \varphi_i.$$

We let $\tilde{x} := \Lambda_t u$. According to the definition of S, we have $\tilde{x} = (S\hat{b})|_{\hat{t}} = S|_{\hat{t}\times\hat{s}} b$.

Let us now turn our attention to the local approximation of u. Due to $\tau \cap \Omega \neq \emptyset$ and $\sigma \cap \Omega \neq \emptyset$, we have $\operatorname{dist}(\tau, \sigma) \leq \operatorname{diam}(\Omega)$, and we have already seen that we can find a function $v \in V$ with

$$\begin{aligned} \|u|_\tau - v\|_{H^1(\tau)} &\leq C_{\mathrm{apx}}(\operatorname{dist}(\tau,\sigma)/8 + 1) q^p \|f\|_{H^{-1}(\Omega)} \\ &\leq C_{\mathrm{apx}}(\operatorname{diam}(\Omega)/8 + 1) q^p \|f\|_{H^{-1}(\Omega)}. \end{aligned} \tag{9.31}$$

We let $\tilde{y} := \Lambda_t v$ and observe that Lemma 9.8 implies

$$\begin{aligned} \|H_t^{d/2}(\tilde{x} - \tilde{y})\|_{L^2(\tau)} &\leq \frac{1}{C_{\mathrm{b1}}} \|\Phi_t(\tilde{x} - \tilde{y})\|_{L^2(\tau)} = \frac{1}{C_{\mathrm{b1}}} \|\Phi_t \Lambda_t (u|_\tau - v)\|_{L^2(\tau)} \\ &= \frac{1}{C_{\mathrm{b1}}} \|\mathfrak{I}_t (u|_\tau - v)\|_{L^2(\tau)} \leq \frac{C_{\mathrm{cl}}}{C_{\mathrm{b1}}} \|u|_\tau - v\|_{L^2(\tau)}. \end{aligned} \tag{9.32}$$

Now we can define

$$C_{\mathrm{blk}} := C_{\mathrm{apx}}(\operatorname{diam}(\Omega)/8 + 1) \frac{C_{\mathrm{cl}}^2}{C_{\mathrm{b1}}^2}$$

and combining (9.30), (9.31) and (9.32) yields

$$\|H_t^{d/2}(\tilde{x} - \tilde{y})\|_{L^2(\tau)} \leq C_{\mathrm{blk}} q^p \|H_s^{-d/2} b\|_2.$$

Using this result, we can now derive the low-rank approximation $X_{t,s} Y_{t,s}^*$ of $S|_{\hat{t}\times\hat{s}}$: we introduce the space

$$Z_h := \{H_t^{d/2} \Lambda_t w \ : \ w \in V\}$$

and observe $k := \dim Z_h \leq \dim V \leq C_{\dim} p^{d+1}$ for the dimension of Z_h and $H_t^{d/2}\tilde{y} = H_t^{d/2}\Lambda_t v \in Z_h$. We fix an orthogonal basis of Z_h, i.e., a matrix $Q \in \mathbb{R}^{\hat{t}\times k}$ with orthogonal columns and range $Q = Z_h$.

We define $\tilde{z} := H_t^{-1/2} Q Q^* H_t^{d/2} \tilde{x}$. Since Q is orthogonal, QQ^* is the orthogonal projection onto Z_h and we get

$$\langle H_t^{d/2}(\tilde{x} - \tilde{z}), w\rangle_2 = \langle H_t^{d/2}\tilde{x} - QQ^* H_t^{d/2}\tilde{x}, QQ^* w\rangle_2 = 0$$

for all $w \in Z_h$, , i.e., $H_t^{d/2}\tilde{z}$ is the best approximation of $H_t^{d/2}\tilde{x}$ in the space Z_h. In particular, $H_t^{d/2}\tilde{z}$ is at least as good as $H_t^{d/2}\tilde{y}$, and we get

$$\|H_t^{d/2}(\tilde{x}-\tilde{z})\|_2 \leq \|H_t^{d/2}(\tilde{x}-\tilde{y})\|_2 \leq C_{\mathrm{blk}} q^p \|H_s^{-1/2} b\|_2.$$

We let

$$X_{t,s} := H_t^{-1/2} Q \in \mathbb{R}^{\hat{t}\times k}, \quad Y_{t,s} := S|_{\hat{t}\times\hat{s}}^* H_t^{d/2} Q \in \mathbb{R}^{\hat{s}\times k}$$

and conclude

$$\tilde{z} = H_t^{-1/2} Q Q^* H_t^{d/2} \tilde{x} = (H_t^{-1/2} Q)(Q^* H_t^{d/2} S|_{\hat{t}\times\hat{s}}) b = X_{t,s} Y_{t,s}^* b,$$

which completes the proof. □

9.4 Compression of the discrete solution operator

In order to apply Theorem 9.11, we have to use an admissibility condition of the type (9.10). The straightforward choice for the convex set τ is the bounding box (cf. (3.10)), and with this choice, the admissibility condition takes the form

$$\max\{\operatorname{diam}(\mathcal{Q}_t), \operatorname{diam}(\mathcal{Q}_s)\} \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)$$

we have already encountered in (4.49).

According to Corollary 6.17, finding low-rank approximations for the total cluster basis $(S_t)_{t\in\mathcal{T}_\mathcal{I}}$ (cf. Definition 6.11) corresponding to the matrix S is equivalent to finding an orthogonal nested cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ such that the orthogonal projection $\Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{I}},V,*} S$ into the space of left semi-uniform matrices defined by $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ and V is a good approximation of S.

Lemma 9.12 (Approximation of S_t). *Let $\eta \in \mathbb{R}_{>0}$ and $q \in (0,1)$. There are constants $C_{\mathrm{blk}}, C_{\dim} \in \mathbb{R}_{>0}$ depending only on η, q, Ω and the shape regularity of the mesh such that for all $t \in \mathcal{T}_\mathcal{I}$ and all $p \in \mathbb{N}_{\geq 2}$ we can find a rank $k \in \mathbb{N}$ with $k \leq C_{\dim} p^{d+1}$ and matrices $X_t \in \mathbb{R}^{\hat{t}\times k}$, $Y_t \in \mathbb{R}^{\mathcal{I}\times k}$ with*

$$\|H^{d/2}(S_t - X_t Y_t^*) b\|_2 \leq C_{\mathrm{blk}} q^p \|H^{-d/2} b\| \quad \text{for all } b \in \mathbb{R}^\mathcal{I}.$$

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. The matrix S_t of the total cluster basis of S is given by

$$S_t = \sum_{s\in\operatorname{row}^*(t)} \chi_t S \chi_s,$$

i.e., it corresponds to the restriction of S to the rows in $\hat{t}$ and the columns in

$$N_t := \bigcup_{s\in\operatorname{row}^*(t)} \hat{s}.$$

Due to definition (cf. Lemma 5.7) we can find a $t^+ \in \operatorname{pred}(t)$ for each $s \in \operatorname{row}^*(t)$ such that (t^+, s) is an admissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, i.e., $(t^+, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}$. Since we are using the admissibility condition (4.49), this implies

$$\operatorname{diam}(\mathcal{Q}_t) \leq \operatorname{diam}(\mathcal{Q}_{t^+}) \leq 2\eta \operatorname{dist}(\mathcal{Q}_{t^+}, \mathcal{Q}_s) \leq 2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s).$$

We let

$$\tau := \mathcal{Q}_t, \quad \sigma := \bigcup_{s\in\operatorname{row}^*(t)} \mathcal{Q}_s$$

and conclude that τ is convex, that

$$\operatorname{supp}\varphi_i \subseteq \tau, \quad \operatorname{supp}\varphi_j \subseteq \sigma \quad \text{hold for all } i \in \hat{t},\ j \in N_t,$$

and that

$$\operatorname{diam}(\tau) = \operatorname{diam}(\mathcal{Q}_t) \leq \min\{2\eta \operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s) \ :\ s \in \operatorname{row}^*(t)\} = 2\eta \operatorname{dist}(\tau, \sigma)$$

holds, i.e., τ and σ satisfy the requirements of Theorem 9.11.

Applying the theorem and extending the resulting rank-k-approximation of $S|_{\hat{t}\times N_t}$ by zero completes the proof. □

Theorem 9.13 (Approximation of S). *Let $\eta \in \mathbb{R}_{>0}$ and $q \in (0,1)$. There are constants $C_{\mathrm{blk}}, C_{\mathrm{dim}} \in \mathbb{R}_{>0}$ depending only on η, q, Ω and the shape regularity of the mesh such that for all $p \in \mathbb{N}_{\geq 2}$ we can find a nested cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ with a rank distribution $(L_t)_{t\in\mathcal{T}_\mathcal{I}}$ satisfying $\#L_t \leq C_{\mathrm{dim}} p^{d+1}$ and*

$$\|H^{d/2}(S - \widetilde{S})b\|_2 \leq \widehat{C}_{\mathrm{blk}}(p_\mathcal{I} + 1)\sqrt{c_\mathcal{I}} q^p \|H^{-d/2} b\|_2 \quad \text{for all } b \in \mathbb{R}^\mathcal{I},$$

where $\widetilde{S} \in \mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}, V, V)$ is an $\mathcal{H}^2$-matrix and $p_\mathcal{I}$ is the depth of $\mathcal{T}_\mathcal{I}$.

Proof. Let C_{blk} and C_{dim} be defined as in Lemma 9.12. We let $\widehat{S} := H^{d/2} S H^{d/2}$ and denote its total cluster basis by $(\widehat{S}_t)_{t\in\mathcal{T}_\mathcal{I}}$. Replacing b by $H^{1/2}\hat{b}$ in this lemma yields

$$\|H^{d/2}(S_t - X_{t,s} Y_{t,s}^*) H^{d/2} \hat{b}\|_2 \leq C_{\mathrm{blk}} q^p \|\hat{b}\|_2 \quad \text{for all } \hat{b} \in \mathbb{R}^\mathcal{I},$$

i.e., the matrices $\widehat{S}_t$ can be approximated by rank $\leq C_{\mathrm{dim}} p^{d+1}$ up to an accuracy of $C_{\mathrm{blk}} q^p$.

We apply Corollary 6.18 to prove that there exists a nested orthogonal cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ with rank distribution $L = (L_t)_{t\in\mathcal{T}_\mathcal{I}}$ satisfying $\#L_t \leq C_{\mathrm{dim}} p^{d+1}$ and

$$\|\widehat{S} - \Pi_{\mathcal{T}_{\mathcal{I}\times\mathcal{I}}, Q, *}[\widehat{S}]\|_2^2 \leq \sum_{t\in\mathcal{T}_\mathcal{I}} C_{\mathrm{blk}}^2 q^{2p} \leq C_{\mathrm{blk}}^2 (\#\mathcal{T}_\mathcal{I}) q^{2p}. \tag{9.33}$$

This provides us with an estimate for the row cluster basis.

Since $\mathcal{L}$ is self-adjoint, we have $\mathcal{L} = \mathcal{L}^*$ and $\mathcal{L}^{-1} = (\mathcal{L}^{-1})^*$ and conclude

$$S^* = (\Lambda \mathcal{L}^{-1} \Lambda^*)^* = (\Lambda^*)^* (\mathcal{L}^{-1})^* \Lambda^* = \Lambda \mathcal{L}^{-1} \Lambda^* = S,$$

therefore we can use Q also as a column cluster basis.

Restricting (9.33) to admissible blocks $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ yields

$$\|\chi_t \widehat{S} \chi_s - Q_t Q_t^* \widehat{S} \chi_s\|_2 \leq C_{\mathrm{blk}} \sqrt{\#\mathcal{T}_{\mathcal{I}}} q^p \quad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}.$$

We introduce the cluster basis $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ by

$$V_t := H^{-d/2} Q_t \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

define the approximation of S by

$$\widetilde{S} := \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}} \chi_t S \chi_s + \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}} V_t Q_t^* \widehat{S} Q_s V_s^*$$

and get

$$\begin{aligned}
\|H^{d/2}(\chi_t(S-\widetilde{S})\chi_s)H^{d/2}\|_2^2 &= \|H^{d/2}(\chi_t S \chi_s - V_t Q_t^* \widehat{S} Q_s V_s^*) H^{d/2}\|_2^2 \\
&= \|\chi_t \widehat{S} \chi_s - Q_t Q_t^* \widehat{S} Q_s Q_s^*\|_2^2 \\
&= \|\chi_t \widehat{S} \chi_s - Q_t Q_t^* \widehat{S} \chi_s\|_2^2 + \|Q_t Q_t^* (\widehat{S}\chi_s - \widehat{S} Q_s Q_s^*)\|_2^2 \\
&\leq \|\chi_t \widehat{S} \chi_s - Q_t Q_t^* \widehat{S} \chi_s\|_2^2 + \|\chi_s \widehat{S}^* \chi_t - Q_s Q_s^* \widehat{S}^* \chi_t\|_2^2 \\
&\leq 2 C_{\mathrm{blk}}^2 (\#\mathcal{T}_{\mathcal{I}}) q^{2p} \quad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}.
\end{aligned}$$

We use Theorem 4.47 to get

$$\|H^{d/2}(S-\widetilde{S})H^{d/2}\|_2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{p_{\mathcal{I}}} C_{\mathrm{blk}} \sqrt{2\#\mathcal{T}_{\mathcal{I}}} q^p \leq C_{\mathrm{blk}} C_{\mathrm{sp}} (p_{\mathcal{I}} + 1) \sqrt{2\#\mathcal{T}_{\mathcal{I}}} q^p,$$

and setting $\widehat{C}_{\mathrm{blk}} := C_{\mathrm{sp}} C_{\mathrm{blk}} \sqrt{2}$ and substituting b for $\hat{b}$ completes the proof. □

Chapter 10
Applications

We have considered methods for approximating integral operators in the Chapters 2 and 4, we have discussed methods for reducing the storage requirements of these approximations in the Chapters 5 and 6, and we have investigated techniques for performing algebraic operations with $\mathcal{H}^2$-matrices in the Chapters 7 and 8. Using the tools provided by these chapters, we can now consider practical applications of $\mathcal{H}^2$-matrices.

Due to their special structure, $\mathcal{H}^2$-matrix techniques are very well suited for problems related to elliptic partial differential equations, and we restrict our attention to applications of this kind:

- In Section 10.1, we consider a simple boundary integral equation related to Poisson's equation on a bounded or unbounded domain.
- In Section 10.2, we approximate the operator mapping Dirichlet to Neumann boundary values of Poisson's equation.
- Section 10.3 demonstrates that the approximative arithmetic operations for $\mathcal{H}^2$-matrices can be used to construct efficient preconditioners for boundary integral equations.
- Section 10.4 shows that our techniques also work for non-academic examples.
- Section 10.5 is devoted to the construction of solution operators for elliptic partial differential equations. These operators are non-local and share many properties of the integral operators considered here, but they cannot be approximated by systems of polynomials.

Our goal in this chapter is not to prove theoretical estimates for the accuracy or the complexity, but to demonstrate that the $\mathcal{H}^2$-matrix techniques work in practice and to provide "rules of thumb" for choosing the parameters involved in setting up the necessary algorithms.

Assumptions in this chapter: We assume that cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ for the finite index sets $\mathcal{I}$ and $\mathcal{J}$ are given. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ be an admissible block cluster tree for $\mathcal{T}_\mathcal{I}$. Let $p_\mathcal{I}$ and $p_\mathcal{J}$ be the depths of $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$.

10.1 Indirect boundary integral equation

Let $\Omega \subseteq \mathbb{R}^3$ be a bounded Lipschitz domain. We consider Laplace's equation

$$-\Delta u := -\sum_{i=1}^{3} \partial_i^2 u = 0 \tag{10.1a}$$

with Dirichlet-type boundary conditions

$$u|_\Gamma = u_D \tag{10.1b}$$

for the solution $u \in H^1(\Omega)$ and the Dirichlet boundary values $u_D \in H^{1/2}(\Gamma)$.

Boundary integral equation

According to [92], Subsection 4.1.1, we can solve the problem (10.1) by finding a *density function* $f \in H^{-1/2}(\Gamma)$ satisfying Symm's integral equation

$$\int_\Gamma \frac{f(y)}{4\pi \|x-y\|_2}\, dy = u_D(x) \quad \text{for almost all } x \in \Gamma. \tag{10.2}$$

Using this f, the solution $u \in H^1(\Omega)$ is given by

$$u(x) = \int_\Gamma \frac{f(y)}{4\pi \|x-y\|_2}\, dy \quad \text{for almost all } x \in \Omega.$$

We introduce the single layer potential operator

$$\mathcal{V}\colon H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma), \quad f \mapsto \left(x \mapsto \int_\Gamma \frac{f(y)}{4\pi \|x-y\|_2}\, dy \right),$$

and note that (10.2) takes the form

$$\mathcal{V} f = u_D. \tag{10.3}$$

We multiply both sides of the equation by test functions $v \in H^{-1/2}(\Gamma)$, integrate, and see that (10.3) is equivalent to the variational equation

$$a_{\mathcal{V}}(v, f) = \langle v, u_D \rangle_\Gamma \quad \text{for all } v \in H^{-1/2}(\Gamma) \tag{10.4}$$

with the bilinear form

$$a_{\mathcal{V}}\colon H^{-1/2}(\Gamma) \times H^{-1/2}(\Gamma) \to \mathbb{R}, \quad (v, f) \mapsto \int_\Gamma v(x) \int_\Gamma \frac{f(y)}{4\pi \|x-y\|_2}\, dy\, dx.$$

Due to [92], Satz 3.5.3, $a_{\mathcal{V}}$ is $H^{-1/2}(\Gamma)$-coercive, therefore (10.4) has a unique solution.

Discretization

In order to discretize (10.4) by Galerkin's method, we assume that Γ can be represented by a conforming triangulation $\mathcal{G} := \{\tau_i : i \in \mathcal{I}\}$ (cf. Definition 4.1.2 in [92]) and introduce the family $(\varphi_i)_{i\in\mathcal{I}}$ of piecewise constant basis functions given by

$$\varphi_i(x) := \begin{cases} 1 & \text{if } x \in \tau_i, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{I}, x \in \Gamma.$$

Applying Galerkin's method to the bilinear variational equation (10.4) in the finite-dimensional space

$$S_0 := \operatorname{span}\{\varphi_i : i \in \mathcal{I}\} \subseteq H^{-1/2}(\Gamma)$$

leads to the system of linear equations

$$Vx = b \tag{10.5}$$

for the matrix $V \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and the right-hand side vector $b \in \mathbb{R}^{\mathcal{I}}$ defined by

$$\begin{aligned} V_{ij} &:= \int_\Gamma \varphi_i(x) \int_\Gamma \frac{\varphi_j(y)}{4\pi\|x-y\|_2}\,dy\,dx \\ &= \int_{\tau_i}\int_{\tau_j} \frac{1}{4\pi\|x-y\|_2}\,dy\,dx && \text{for all } i,j \in \mathcal{I}, \\ b_i &:= \int_{\tau_i} u_D(x)\,dx && \text{for all } i \in \mathcal{I}. \end{aligned} \tag{10.6}$$

The solution vector $x \in \mathbb{R}^{\mathcal{I}}$ of (10.5) defines the approximation

$$f_h := \sum_{i\in\mathcal{I}} x_i \varphi_i$$

of the solution $f \in H^{-1/2}(\Gamma)$.

Approximation of the stiffness matrix

We can see that $V_{ij} > 0$ holds for all $i, j \in \mathcal{I}$, therefore the matrix V is densely populated and cannot be handled efficiently by standard techniques. Fortunately, the matrix $V \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ fits perfectly into the framework presented in Chapter 4, i.e., we can construct an $\mathcal{H}^2$-matrix approximation $\widetilde{V} \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and solve the perturbed system

$$\widetilde{V}\tilde{x} = b. \tag{10.7}$$

The solution of the perturbed system is given by

$$\tilde{f}_h := \sum_{i \in \mathcal{I}} \tilde{x}_i \varphi_i$$

and can be characterized by the variational equation

$$\tilde{a}_{\mathcal{V}}(v_h, \tilde{f}_h) = \langle v_h, u_D \rangle_\Gamma \quad \text{for all } v_h \in S_0,$$

where the perturbed bilinear form $\tilde{a}_{\mathcal{V}}(\cdot,\cdot)$ is defined by

$$\tilde{a}_{\mathcal{V}}(\varphi_i, \varphi_j) := \widetilde{V}_{ij} \quad \text{for all } i, j \in \mathcal{I}. \tag{10.8}$$

We have to ensure that approximating V by $\widetilde{V}$, i.e., approximating $a(\cdot,\cdot)$ by $\tilde{a}(\cdot,\cdot)$, does not reduce the speed of convergence of the discretization scheme.

According to [92], Satz 4.1.32, we can expect

$$\|f - f_h\|_{-1/2,\Gamma} \lesssim h^{3/2}$$

in the case of maximal regularity, i.e., if $f \in H^1(\Gamma)$ holds, therefore our goal is to choose the accuracy of the approximation $\widetilde{V}$ in such a way that

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim h^{3/2}$$

holds. Strang's first lemma (e.g., Theorem 4.1.1 in [34] or Satz III.1.1 in [26]) yields the estimate

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim \inf_{v_h \in S_0} \left(\|f - v_h\|_{-1/2,\Gamma} + \sup_{w_h \in S_0} \frac{|a_{\mathcal{V}}(v_h, w_h) - \tilde{a}_{\mathcal{V}}(v_h, w_h)|}{\|w_h\|_{-1/2,\Gamma}} \right)$$

if $\tilde{a}_{\mathcal{V}}$ is $H^{-1/2}(\Gamma)$-elliptic, i.e., if $\tilde{a}_{\mathcal{V}}$ is "close enough" to the elliptic bilinear form $a_{\mathcal{V}}$. We choose $v_h = f_h$ in order to bound the first term on the right-hand side and now have to prove that the second term is also bounded. The latter term is related to the approximation error introduced by using $\widetilde{V}$ instead of V.

In order to bound it, we assume that the triangulation $\mathcal{G}$ is shape-regular and quasi-uniform with grid parameter $h \in \mathbb{R}_{>0}$. Due to the inverse estimate [38], Theorem 4.6, we have

$$h^{1/2} \|w_h\|_{0,\Gamma} \lesssim \|w_h\|_{-1/2,\Gamma} \quad \text{for all } w_h \in S_0$$

and conclude

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim \inf_{v_h \in S_0} \left(\|f - v_h\|_{-1/2,\Gamma} + h^{-1/2} \sup_{w_h \in S_0} \frac{|a_{\mathcal{V}}(v_h, w_h) - \tilde{a}_{\mathcal{V}}(v_h, w_h)|}{\|w_h\|_{0,\Gamma}} \right).$$

We can bound the right-hand side of this estimate by using $v_h = f_h$ and the following estimate of the approximation error:

Lemma 10.1 (Accuracy of $\tilde{a}_{\mathcal{V}}(\cdot,\cdot)$). *We have*

$$|a_{\mathcal{V}}(v_h, w_h) - \tilde{a}_{\mathcal{V}}(v_h, w_h)| \lesssim h^{-d_\Gamma} \|V - \widetilde{V}\|_2 \|v_h\|_{0,\Gamma} \|w_h\|_{0,\Gamma} \quad \text{for all } v_h, w_h \in S_0.$$

Proof. Let $x, y \in \mathbb{R}^{\mathcal{I}}$ be the coefficient vectors of v_h and w_h satisfying

$$v_h = \sum_{i\in\mathcal{I}} x_i \varphi_i, \quad w_h = \sum_{j\in\mathcal{I}} y_j \varphi_j.$$

We have

$$\begin{aligned} |a_{\mathcal{V}}(v_h, w_h) - \tilde{a}_{\mathcal{V}}(v_h, w_h)| &= \left| \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_i y_j (a_{\mathcal{V}}(\varphi_i, \varphi_j) - \tilde{a}_{\mathcal{V}}(\varphi_i, \varphi_j)) \right| \\ &= \left| \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_i y_j (V_{ij} - \widetilde{V}_{ij}) \right| \\ &= |\langle x, (V - \widetilde{V}) y \rangle_2| \leq \|V - \widetilde{V}\|_2 \|x\|_2 \|y\|_2. \end{aligned}$$

Due to [38], Proposition 3.1, we find

$$h^{d_\Gamma/2} \|x\|_2 \lesssim \|v_h\|_{0,\Gamma}, \quad h^{d_\Gamma/2} \|y\|_2 \lesssim \|w_h\|_{0,\Gamma},$$

and combining these estimates completes the proof. □

Applying this result to the estimate from Strang's lemma yields

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim \|f - f_h\|_{-1/2,\Gamma} + h^{-d_\Gamma - 1/2} \|V - \widetilde{V}\|_2 \|f_h\|_{0,\Gamma}$$

if $\|V - \widetilde{V}\|_2$ is small enough to ensure that $\widetilde{V}$ is still positive definite. Since f_h converges to f if the mesh size decreases, we can assume that $\|f_h\|_{0,\Gamma}$ is bounded by a constant and get

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim \|f - f_h\|_{-1/2,\Gamma} + h^{-d_\Gamma - 1/2} \|V - \widetilde{V}\|_2.$$

If we can ensure

$$\|V - \widetilde{V}\|_2 \lesssim h^{d_\Gamma + 2}, \tag{10.9}$$

we get

$$\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim h^{3/2},$$

which yields the desired optimal order of convergence.

Combining the error estimate provided by Theorem 4.49 with the simplified interpolation error estimate of Remark 4.23 yields

$$\|V - \widetilde{V}\|_2 \lesssim C_{\mathrm{ov}} C_{\mathcal{I}}^2 q^m \sum_{\ell=0}^{p_{\mathcal{I}}} \max \left\{ \frac{|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} \ : \ t \in \mathcal{T}_{\mathcal{I}}^{(\ell)} \right\}. \tag{10.10}$$

The piecewise constant basis functions satisfy $C_{\mathrm{ov}} = 1$ and $C_{\mathcal{I}} \lesssim h^{d_\Gamma/2}$. The kernel function defining V has a singularity of order $\sigma = 1$, therefore we can assume

$$\frac{|\Omega_t|}{\operatorname{diam}(\mathcal{Q}_t)^\sigma} \lesssim \operatorname{diam}(\mathcal{Q}_t)^{d_\Gamma-\sigma} = \operatorname{diam}(\mathcal{Q}_t).$$

This means that the sum in estimate (10.10) can be bounded by a geometric sum dominated by the large clusters (cf. Remark 4.50), and these clusters can be bounded independently of h. We conclude

$$\|V - \tilde{V}\|_2 \lesssim h^{d_\Gamma} q^m.$$

In order to guarantee (10.9), we have to choose the order $m \in \mathbb{N}$ large enough to yield $q^m \lesssim h^2$, i.e., $\log(h)/\log(q) \lesssim m$.

Experiments

According to our experiments (cf. Table 4.2), increasing the interpolation order by one if the mesh width h is halved should be sufficient to ensure $q^m \lesssim h^2$ and therefore $\|f - \tilde{f}_h\|_{-1/2,\Gamma} \lesssim h^{3/2}$.

We test the method for the approximations of the unit sphere Γ_S used in the previous chapters and the harmonic functions

$$\begin{aligned} u_1 &: \mathbb{R}^3 \to \mathbb{R}, \quad x \mapsto x_1 + x_2 + x_3, \\ u_2 &: \mathbb{R}^3 \to \mathbb{R}, \quad x \mapsto x_1^2 - x_3^2, \\ u_3 &: \mathbb{R}^3 \to \mathbb{R}, \quad x \mapsto \frac{1}{\|x - x_0\|_2} \quad \text{for } x_0 = (6/5, 6/5, 6/5) \end{aligned}$$

by computing the corresponding densities $f_{1,h}$, $f_{2,h}$ and $f_{3,h}$ and comparing

$$u_{i,h}(x) := \int_\Gamma \frac{f_{i,h}(y)}{4\pi\|x - y\|_2}\, dy \quad \text{for all } i \in \{1, 2, 3\}$$

to the real value of the harmonic function at the test point $\hat{x} := (1/2, 1/2, 1/2)$.

Due to the results in Chapter 12 of [97], we expect the error to behave like

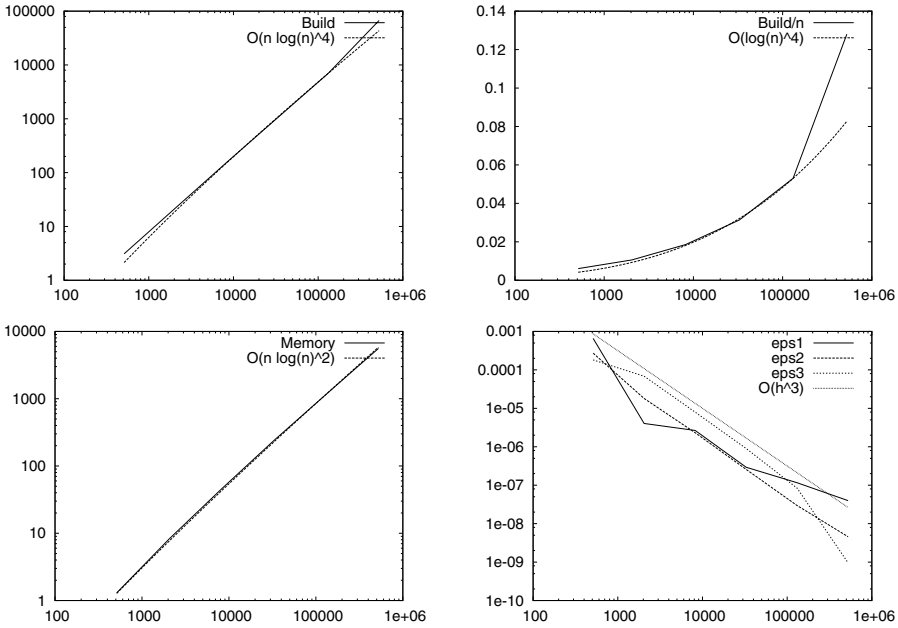
$$\epsilon_i := |u_i(\hat{x}) - u_{i,h}(\hat{x})| \lesssim h^3 \quad \text{for all } i \in \{1, 2, 3\}.$$

In a first series of experiments, we use interpolation (cf. Section 4.4) to create an initial approximation of V, and then apply Algorithm 30 with blockwise control of the relative spectral error (cf. Section 6.8) for an error tolerance $\hat{\epsilon} \sim h^2$. In order to reduce the complexity, we use an incomplete cross approximation [104] of the coupling matrices S_b, which can be computed efficiently by the ACA algorithm [4]. The nearfield matrices are constructed using the black-box quadrature scheme described in [89], [43]. The

resulting system of linear equations is solved approximately by the conjugate gradient method [99] with a suitable preconditioner (cf. [50]). The parameters used for the interpolation and the recompression are given in Table 10.1: as usual, n gives the number of degrees of freedom, in this case the number of triangles of the surface approximation, m is the order of the initial interpolation and $\hat{\epsilon}$ is the tolerance used by the recompression algorithm.

Table 10.1. Solving Poisson's equation using an indirect boundary element formulation. The matrix is constructed by applying the recompression Algorithm 30 to an initial $\mathcal{H}^2$-matrix approximation constructed by interpolation.

n	m	$\hat{\epsilon}$	Build	Mem	Mem/n	ϵ_1	ϵ_2	ϵ_3
512	3	2_{-3}	3.1	1.3	2.7	6.6_{-4}	2.7_{-4}	1.8_{-4}
2048	4	5_{-4}	21.7	8.2	4.1	1.8_{-4}	2.3_{-5}	1.3_{-5}
8192	5	1_{-4}	152.1	45.3	5.7	2.7_{-6}	2.3_{-6}	8.0_{-6}
32768	6	2_{-5}	1027.8	238.1	7.4	2.9_{-7}	2.6_{-7}	9.0_{-7}
131072	7	5_{-6}	6950.0	1160.9	9.1	1.2_{-7}	3.0_{-8}	8.4_{-8}
524288	8	1_{-6}	67119.0	5583.8	10.9	4.0_{-8}	4.6_{-9}	9.6_{-10}



We know that a local rank of $k \sim \log^2(n)$ is sufficient to ensure (10.9), since multipole expansions require this rank. Due to the best-approximation property of Algorithm 25, and therefore Algorithm 30, we expect the compressed $\mathcal{H}^2$-matrix to

have the same rank, and this expectation is indeed justified by Table 10.1: the time required for the matrix construction is proportional to $n \log^4(n)$, i.e., nk^2 for $k \sim \log^2(n)$, and the storage requirements are proportional to $n \log^2(n)$, i.e., nk for our estimate of k.

The surprisingly long computing time for the last experiment ($n = 524288$) may be attributed to the NUMA architecture of the computer: although the resulting matrix requires less than 6 GB, Algorithm 30 needs almost 40 GB for weight matrices of the original cluster bases. Since this amount of storage is not available on a single processor board of the computer, some memory accesses have to use the backplane, and the bandwidth of the backplane is significantly lower than that of the local storage.

The high temporary storage requirements are caused by the high rank of the initial approximation constructed by interpolation: for $m = 8$, the initial rank is $k = m^3 = 512$, and using two 512×512-matrices for each of the 23855 clusters requires a very large amount of temporary storage.

In practical applications, it can even happen that the auxiliary matrices require far more storage than the resulting compact $\mathcal{H}^2$-matrix approximation (it is not uncommon to see that the matrices $(R_{X,s})_{s \in \mathcal{T}_\mathcal{I}}$ require more than two times the storage of the final $\mathcal{H}^2$-matrix, and the ratio grows as the order m becomes larger).

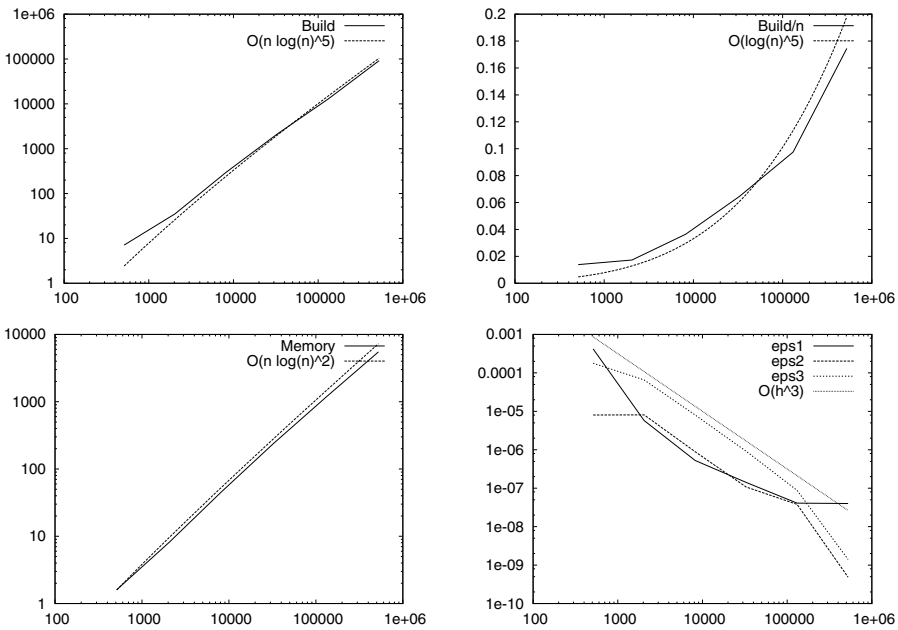
Therefore we look for an alternative approach providing a lower rank for the initial approximation. A good choice is the hybrid cross approximation technique [17], which constructs blockwise low-rank approximations of the matrix, thus providing an initial $\mathcal{H}$-matrix approximation. We can apply the recompression Algorithm 33 in order to convert the low-rank blocks into $\mathcal{H}^2$-matrices as soon as they become available. This allows us to take advantage of *both* the fast convergence of the HCA approximation and the low storage requirements of the resulting $\mathcal{H}^2$-matrix.

The results of this approach are listed in Table 10.2. We use a modified HCA implementation by Lars Grasedyck that is able to increase the order m according to an internal heuristic depending on the prescribed accuracy $\hat{\epsilon}$. The resulting orders are given in the column "m_{ad}". We can see that this approach yields results similar to those provided by Algorithm 30, but that the reduction of the amount of temporary storage seems to lead to an execution time proportional to $n \log^5(n)$ instead of the time proportional to $n \log^4(n)$ that we could observe for Algorithm 30.

Remark 10.2 (Symmetry). In the case investigated here, the matrix V is symmetric and can therefore be approximated by a symmetric $\mathcal{H}^2$-matrix $\widetilde{V}$. Instead of storing the entire matrix, it would be sufficient to store the lower or upper half, thus reducing the storage and time requirements by almost 50 percent. □

Table 10.2. Solving Poisson's equation using an indirect boundary element formulation. The matrix is constructed by applying the recompression Algorithm 33 to an initial $\mathcal{H}$-matrix approximation constructed by the HCA method.

n	$\hat{\epsilon}$	m_{ad}	Build	Mem	Mem/n	ϵ_1	ϵ_2	ϵ_3
512	2_{-3}	4	7.1	1.6	3.2	4.2_{-4}	8.0_{-6}	1.7_{-4}
2048	5_{-4}	5	35.4	8.1	4.0	5.7_{-6}	8.1_{-6}	6.6_{-5}
8192	1_{-4}	7	298.3	45.6	5.7	5.3_{-7}	8.9_{-7}	8.0_{-6}
32768	2_{-5}	9	2114.5	240.9	7.5	1.4_{-7}	1.1_{-7}	9.2_{-7}
131072	5_{-6}	11	12774.5	1170.0	9.1	4.1_{-8}	3.8_{-8}	8.8_{-8}
524288	1_{-6}	14	91501.2	5507.3	10.8	4.0_{-8}	4.8_{-10}	1.4_{-9}



10.2 Direct boundary integral equation

Now we consider a refined boundary integral approach for solving Laplace's equation (10.1) with Dirichlet boundary values.

Instead of working with the density function f, we now want to compute the Neumann values

$$u_N : \Gamma \to \mathbb{R}, \quad x \mapsto \frac{\partial u}{\partial n}(x),$$

corresponding to the Dirichlet values $u_D = u|_\Gamma$ of the harmonic function u directly.

According to [97], Kapitel 12, both functions are connected by the variational equation

$$a_{\mathcal{V}}(v,u_N) = a_{\mathcal{K}}(v,u_D) + \frac{1}{2}\langle v,u_D\rangle_{0,\Gamma} \quad \text{for all } v \in H^{-1/2}(\Gamma), \tag{10.11}$$

where the bilinear form $a_{\mathcal{K}}$ corresponding to the double layer potential is given by

$$a_{\mathcal{K}}\colon H^{-1/2}(\Gamma)\times H^{1/2}(\Gamma)\to\mathbb{R}, \quad (v,u)\mapsto \int_\Gamma v(x)\int_\Gamma \frac{\langle x-y,n(y)\rangle u(y)}{4\pi\|x-y\|_2^3}\,dy\,dx.$$

As in the previous example, existence and uniqueness of $u_N \in H^{-1/2}(\Gamma)$ is guaranteed since $a_{\mathcal{V}}$ is $H^{-1/2}(\Gamma)$-coercive.

Discretization

Usually we cannot compute the right-hand side of (10.11) explicitly, instead we rely on a discrete approximation: we introduce the family $(x_j)_{j\in\mathcal{J}}$ of vertices of the triangulation $\mathcal{G}$ and the family $(\psi_j)_{j\in\mathcal{J}}$ of corresponding continuous piecewise linear basis functions given by

$$\psi_j(x_k) = \begin{cases} 1 & \text{if } i=k,\\ 0 & \text{otherwise}\end{cases} \quad \text{for all } j,k\in\mathcal{J},$$
$$\psi_j|_{\tau_i} \text{ is affine} \quad \text{for all } i\in\mathcal{I}, j\in\mathcal{J}.$$

We approximate u_D by its L^2-orthogonal projection $u_{D,h}$ into the space

$$S_1 := \operatorname{span}\{\psi_j : j\in\mathcal{J}\}\subseteq H^{1/2}(\Gamma).$$

This projection can be computed by solving the variational equation

$$\langle v_h,u_{D,h}\rangle_{0,\Gamma} = \langle v_h,u_D\rangle_{0,\Gamma} \quad \text{for all } v_h\in S_1, \tag{10.12}$$

and the corresponding system of linear equations is well-conditioned and can be solved efficiently by the conjugate gradient method.

Now we proceed as before: we replace the infinite-dimensional space $H^{-1/2}(\Gamma)$ by the finite-dimensional space S_0 and the exact Dirichlet value u_D by $u_{D,h}$ and get the system of linear equations

$$Vx = \left(K+\frac{1}{2}M\right)y \tag{10.13}$$

for the matrix $V\in\mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ already introduced in (10.6), the matrix $K\in\mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ defined by

$$K_{ij} := \int_\Gamma \varphi_i(x)\int_\Gamma \frac{\langle x-y,n(y)\rangle_2\psi_j(y)}{4\pi\|x-y\|_2^3}\,dy\,dx \quad \text{for all } i\in\mathcal{I}, j\in\mathcal{J},$$

the mass matrix $M \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ defined by

$$M_{ij} := \int_\Gamma \varphi_i(x)\psi_j(y)\,dy\,dx \quad \text{for all } i \in \mathcal{I}, j \in \mathcal{J},$$

and the coefficient vector $y \in \mathbb{R}^{\mathcal{J}}$ corresponding to $u_{D,h}$, which is defined by

$$u_{D,h} = \sum_{j\in\mathcal{J}} y_j \psi_j.$$

Since a_V is $H^{-1/2}(\Gamma)$-coercive, the matrix V is symmetric and positive definite, so (10.13) has a unique solution $x \in \mathbb{R}^{\mathcal{I}}$ which corresponds to the Galerkin approximation

$$u_{N,h} := \sum_{i\in\mathcal{I}} x_i \varphi_i$$

of the Neumann boundary values u_N.

Approximation of the matrices

The mass matrix M and the matrix corresponding to (10.12) are sparse and can be computed exactly by standard quadrature. The matrices V and K are not sparse, therefore we have to replace them by efficient approximations. For V we can use interpolation (cf. Section 4.4), while for K the equation

$$\frac{\langle x-y, n(y)\rangle_2}{4\pi\|x-y\|_2^3} = \frac{\partial}{\partial n(y)}\frac{1}{4\pi\|x-y\|_2}$$

suggests using derivatives of interpolants (cf. Section 4.5). Replacing V and K by their respective $\mathcal{H}^2$-matrix approximations $\tilde{V}$ and $\tilde{K}$ leads to the perturbed system

$$\tilde{V}\tilde{x} = \left(\tilde{K} + \frac{1}{2}M\right)y. \tag{10.14}$$

The solution of the corresponding perturbed system corresponds to the function

$$\tilde{u}_{N,h} := \sum_{i\in\mathcal{I}} \tilde{x}_i \varphi_i$$

and can be characterized by the variational equation

$$\tilde{a}_V(v_h, \tilde{u}_{N,h}) = \tilde{a}_K(v_h, u_{D,h}) + \frac{1}{2}\langle v_h, u_{D,h}\rangle_{0,\Gamma} \quad \text{for all } v_h \in S_0$$

for the perturbed bilinear forms $\tilde{a}_V$ defined by (10.8) and $\tilde{a}_K$ defined by

$$\tilde{a}_K(\varphi_i, \psi_j) = \tilde{K}_{ij} \quad \text{for all } i \in \mathcal{I}, j \in \mathcal{J}. \tag{10.15}$$

Strang's first lemma (e.g., Satz III.1.1 in [26]) yields

$$\|u_N - \tilde{u}_{N,h}\|_{-1/2,\Gamma} \lesssim \inf_{v_h \in S_0} \left(\|u_N - v_h\|_{-1/2,\Gamma} + \sup_{w_h \in S_0} \frac{|a_{\mathcal{V}}(v_h, w_h) - \tilde{a}_{\mathcal{V}}(v_h, w_h)|}{\|w_h\|_{-1/2,\Gamma}} \right) + \sup_{w_h \in S_0} \frac{|\lambda(w_h) - \tilde{\lambda}(w_h)|}{\|w_h\|_{-1/2,\Gamma}}$$

for the right-hand side functional

$$\lambda(w_h) := a_{\mathcal{K}}(w_h, u_D) + \frac{1}{2}\langle w_h, u_D \rangle_{0,\Gamma} \quad \text{for all } w_h \in S_0$$

and its perturbed counterpart

$$\tilde{\lambda}(w_h) := \tilde{a}_{\mathcal{K}}(w_h, u_{D,h}) + \frac{1}{2}\langle w_h, u_{D,h} \rangle_{0,\Gamma} \quad \text{for all } w_h \in S_0.$$

We have already given estimates for the first two terms of the error estimate in the previous section, so we now only have to find a suitable bound for the difference of the right-hand sides. To this end, we separate the approximation of u_D and the approximation of $a_{\mathcal{K}}$:

$$\begin{aligned}\lambda(w_h) - \tilde{\lambda}(w_h) &= a_{\mathcal{K}}(w_h, u_D) - \tilde{a}_{\mathcal{K}}(w_h, u_{D,h}) + \frac{1}{2}\langle w_h, u_D - u_{D,h} \rangle_{0,\Gamma} \\ &= a_{\mathcal{K}}(w_h, u_D - u_{D,h}) + a_{\mathcal{K}}(w_h, u_{D,h}) \\ &\quad - \tilde{a}_{\mathcal{K}}(w_h, u_{D,h}) + \frac{1}{2}\langle w_h, u_D - u_{D,h} \rangle_{0,\Gamma}.\end{aligned}$$

Since $u_{D,h}$ is computed by the L^2-projection, we have

$$\|u_D - u_{D,h}\|_{1/2,\Gamma} \lesssim h^{3/2}$$

according to [97], equation (12.19), in the case of optimal regularity $u_D \in H^2(\Gamma)$. This implies

$$\langle w_h, u_D - u_{D,h} \rangle_{0,\Gamma} \leq \|w_h\|_{-1/2,\Gamma} \|u_D - u_{D,h}\|_{1/2,\Gamma} \lesssim h^{3/2} \|w_h\|_{-1/2,\Gamma}.$$

According to Satz 6.11 in [97], we have

$$a_{\mathcal{K}}(w_h, u_D - u_{D,h}) \lesssim \|w_h\|_{-1/2,\Gamma} \|u_D - u_{D,h}\|_{1/2,\Gamma} \lesssim h^{3/2} \|w_h\|_{-1/2,\Gamma}.$$

For the perturbed bilinear form, combining the technique used in the proof of Lemma 10.1 with the inverse estimate ([38], Theorem 4.6) yields

$$\begin{aligned}|a_{\mathcal{K}}(w_h, u_{D,h}) - \tilde{a}_{\mathcal{K}}(w_h, u_{D,h})| &\lesssim h^{-d_\Gamma} \|K - \tilde{K}\|_2 \|w_h\|_{0,\Gamma} \|u_{D,h}\|_{0,\Gamma} \\ &\lesssim h^{-d_\Gamma - 1/2} \|K - \tilde{K}\|_2 \|w_h\|_{-1/2,\Gamma} \|u_{D,h}\|_{0,\Gamma}.\end{aligned}$$

Due to $u_D \in H^{1/2}(\Gamma) \subseteq L^2(\Gamma)$, we can assume that $\|u_{D,h}\|_{0,\Gamma}$ is bounded independently of h and conclude

$$|\lambda(w_h) - \tilde{\lambda}(w_h)| \lesssim (h^{-d_\Gamma - 1/2}\|K - \tilde{K}\|_2 + h^{3/2})\|w_h\|_{-1/2,\Gamma},$$

i.e., if we can ensure

$$\|K - \tilde{K}\|_2 \lesssim h^{d_\Gamma + 2}, \tag{10.16}$$

we get

$$\|u_N - \tilde{u}_{N,h}\|_{-1/2,\Gamma} \lesssim h^{3/2}.$$

Since the kernel function corresponding to K is based on first derivatives of a kernel function with singularity order $\sigma = 1$, Corollary 4.32 yields an approximation error of

$$\|g - \tilde{g}\|_{\infty,\Omega_t \times \Omega_s} \lesssim \frac{q^m}{\operatorname{dist}(\mathcal{Q}_t, \mathcal{Q}_s)^2} \quad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Assuming again

$$|\Omega_t| \lesssim \operatorname{diam}(\mathcal{Q}_t)^{d_\Gamma}, \quad |\Omega_s| \lesssim \operatorname{diam}(\mathcal{Q}_s)^{d_\Gamma} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},\ s \in \mathcal{T}_\mathcal{J}$$

and applying Theorem 4.49 gives us

$$\|K - \tilde{K}\|_2 \lesssim C_{\mathrm{ov}} C_\mathcal{I} C_\mathcal{J} q^m \max\{p_\mathcal{I}, p_\mathcal{J}\}. \tag{10.17}$$

For our choice of basis functions, we have $C_{\mathrm{ov}} = 3$, $C_\mathcal{I}, C_\mathcal{J} \lesssim h^{d_\Gamma/2}$ and get

$$\|K - \tilde{K}\|_2 \lesssim h^{d_\Gamma} q^m \max\{p_\mathcal{I}, p_\mathcal{J}\}.$$

In order to guarantee (10.16), we have to choose the order m large enough to yield $q^m \max\{p_\mathcal{I}, p_\mathcal{J}\} \lesssim h^2$.

Experiments

We can assume $p_\mathcal{I} \lesssim \log n_\mathcal{I}$ and $p_\mathcal{J} \lesssim \log n_\mathcal{J}$, therefore our previous experiments (cf. Table 4.4) suggest that increasing the interpolation order by one if the mesh width h is halved should be sufficient to ensure $q^m \max\{p_\mathcal{I}, p_\mathcal{J}\} \lesssim h^2$ and therefore the desired error bound $\|u_N - \tilde{u}_{N,h}\|_{-1/2,\Gamma} \lesssim h^{3/2}$.

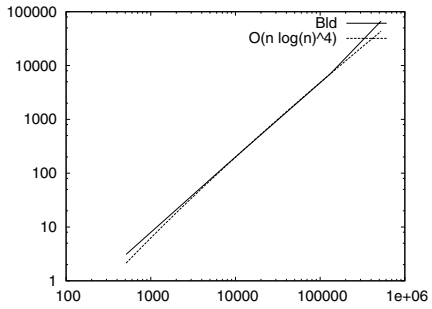
We once more test the method for the unit sphere and the harmonic functions u_1, u_2 and u_3 introduced in the previous section. For these functions, the Neumann boundary values $u_{1,N}$, $u_{2,N}$ and $u_{3,N}$ can be computed and compared to their discrete approximations $u_{1,N,h}$, $u_{2,N,h}$ and $u_{3,N,h}$. Since computing a sufficiently accurate approximation of the norm of $H^{-1/2}(\Gamma)$ is too complicated, we measure the L^2-norm instead. According to the inverse estimate [38], Theorem 4.6, we can expect

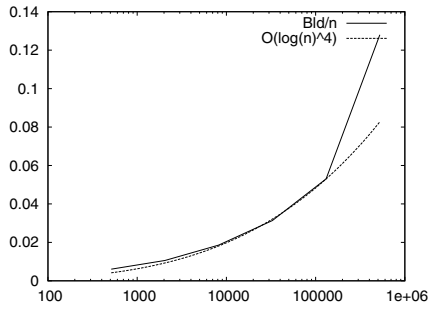
$$\epsilon_i := \|u_{N,i} - u_{N,i,h}\|_{0,\Gamma} \lesssim h \quad \text{for all } i \in \{1,2,3\}.$$

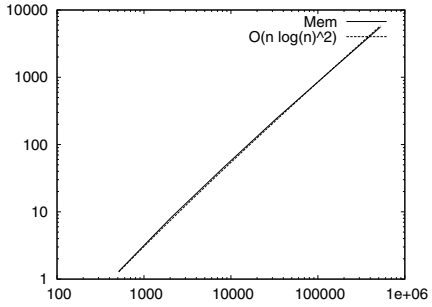
The results in Table 10.3 match our expectations: the errors decrease at least like h, the faster convergence of ϵ_1 is due to the fact that $u_{N,1}$ can be represented *exactly* by the discrete space, therefore the discretization error is zero and the error ϵ_1 is only caused by the quadrature and the matrix approximation, and its fast convergence is owed to the fact that we choose the error tolerance $\hat{\epsilon}$ slightly smaller than strictly necessary.

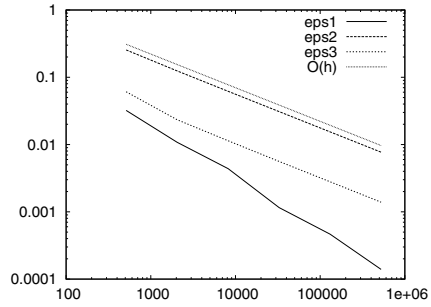
Table 10.3. Solving Poisson's equation using a direct boundary element formulation. The matrix V is constructed by applying the recompression Algorithm 30 to an initial $\mathcal{H}^2$-matrix approximation constructed by interpolation. The matrix K is approximated on the fly by interpolation and not stored.

n	m	$\hat{\epsilon}$	Build	Mem	Mem/n	ϵ_1	ϵ_2	ϵ_3
512	3	2_{-3}	3.1	1.3	2.7	3.2_{-2}	2.5_{-1}	5.1_{-2}
2048	4	5_{-4}	21.7	8.2	4.1	1.1_{-2}	1.2_{-1}	2.3_{-2}
8192	5	1_{-4}	152.1	45.3	5.7	4.4_{-3}	6.2_{-2}	1.5_{-2}
32768	6	2_{-5}	1027.8	238.1	7.4	1.2_{-3}	3.1_{-2}	5.6_{-3}
131072	7	5_{-6}	6950.0	1160.9	9.1	4.6_{-4}	1.5_{-2}	2.8_{-3}
524288	8	1_{-6}	67119.0	5583.8	10.9	1.4_{-4}	7.7_{-3}	1.4_{-3}









10.3 Preconditioners for integral equations

According to Lemma 4.5.1 in [92], the condition number of the matrix V can be expected to behave like h^{-1}, i.e., the linear systems (10.5) and (10.13) become ill-conditioned if the resolution of the discrete space grows. For moderate problem dimensions, this poses no problem, since the conjugate gradient method used for solving the linear systems is still quite fast, especially if V is approximated by an $\mathcal{H}^2$-matrix and Algorithm 8 is used to perform the matrix-vector multiplication required in one step of the solver. For large problem dimensions, the situation changes: due to the growing condition number, a large number of matrix-vector multiplications are required by the solver, and each of these multiplications requires an amount of time that is not negligible.

Preconditioned conjugate gradient method

If we still want to be able to solve the linear systems efficiently, we have to either speed up the matrix-vector multiplication or reduce the number of these multiplications required by the solver. The time per matrix-vector multiplication can be reduced by compressing the matrix as far as possible, e.g., by applying the Algorithms 30 or 33, since reducing the storage required for the $\mathcal{H}^2$-matrix also reduces the time needed to process it, or by parallelizing Algorithm 8.

The number of multiplications required by the conjugate gradient method can be reduced by using a preconditioner, i.e., by solving the system

$$\hat{V}\hat{x} = \hat{b}$$

for the transformed quantities

$$\hat{V} := P^{-1/2} V P^{-1/2}, \quad \hat{x} := P^{1/2} x, \quad \hat{b} := P^{-1/2} b,$$

where $P \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ is a symmetric positive definite matrix, called the *preconditioning matrix* for V. The idea is to construct a matrix P which is, in a suitable sense, "close" to V, since then we can expect $\hat{V}$ to be "close" to the identity, i.e., to be well-conditioned.

The matrix P can be constructed by a number of techniques: wavelet methods [74], [40], [36] apply a basis transformation and suitable diagonal scaling, the matrix P can be constructed by discretizing a suitable pseudo-differential operator [98], but we can also simply rely on $\mathcal{H}^2$-matrix arithmetic operations introduced in Chapter 8 by computing an approximative LU factorization of V.

LU decomposition

In order to do so, let us first consider a suitable recursive algorithm for the computation of the *exact* LU factorization of V. For all $t, s \in \mathcal{T}_{\mathcal{I}}$, we denote the submatrix corresponding to the block (t, s) by $V_{t,s} := V|_{\hat{t}\times\hat{s}}$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ be a cluster. The LU factorization of $V_{t,t}$ can be computed by a recursive algorithm: if t is a leaf cluster, the admissibility condition (4.49) and, in fact, all other admissibility conditions we have introduced, implies that $(t, t) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}$ is an inadmissible leaf of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$. Therefore it is stored in one of the standard representations for densely populated matrices and we can compute the LU factorization of $V_{t,t}$ by standard algorithms.

If t is not a leaf cluster, the admissibility condition guarantees that (t, t) is inadmissible, i.e., that $V_{t,t}$ is a block matrix. We consider only the simple case $\#\operatorname{sons}(t) = 2$, the general case $\#\operatorname{sons}(t) \in \mathbb{N}$ can be handled by induction. Let $\operatorname{sons}(t) = \{t_1, t_2\}$. We have

$$V_{t,t} = \begin{pmatrix} V_{t_1,t_1} & V_{t_1,t_2} \\ V_{t_2,t_1} & V_{t_2,t_2} \end{pmatrix}$$

and the LU factorization $V_{t,t} = L^{(t)}U^{(t)}$ takes the form

$$\begin{pmatrix} V_{t_1,t_1} & V_{t_1,t_2} \\ V_{t_2,t_1} & V_{t_2,t_2} \end{pmatrix} = \begin{pmatrix} L^{(t)}_{t_1,t_1} & \\ L^{(t)}_{t_2,t_1} & L^{(t)}_{t_2,t_2} \end{pmatrix} \begin{pmatrix} U^{(t)}_{t_1,t_1} & U^{(t)}_{t_1,t_2} \\ & U^{(t)}_{t_2,t_2} \end{pmatrix},$$

which is equivalent to the four equations

$$V_{t_1,t_1} = L^{(t)}_{t_1,t_1} U^{(t)}_{t_1,t_1}, \tag{10.18a}$$

$$V_{t_1,t_2} = L^{(t)}_{t_1,t_1} U^{(t)}_{t_1,t_2}, \tag{10.18b}$$

$$V_{t_2,t_1} = L^{(t)}_{t_2,t_1} U^{(t)}_{t_1,t_1}, \tag{10.18c}$$

$$V_{t_2,t_2} = L^{(t)}_{t_2,t_1} U^{(t)}_{t_1,t_2} + L^{(t)}_{t_2,t_2} U^{(t)}_{t_2,t_2}. \tag{10.18d}$$

According to (10.18a), we compute $L^{(t_1)}$ and $U^{(t_1)}$ by recursively constructing the LU factorization $V_{t_1,t_1} = L^{(t_1)}U^{(t_1)}$ of the submatrix V_{t_1,t_1} and setting $L^{(t)}_{t_1,t_1} := L^{(t_1)}$ and $U^{(t)}_{t_1,t_1} := U^{(t_1)}$.

Once we have $L^{(t)}_{t_1,t_1}$ and $U^{(t)}_{t_1,t_1}$, we compute $U^{(t)}_{t_1,t_2}$ and $L^{(t)}_{t_2,t_1}$ by using equations (10.18b) and (10.18c).

Then equation (10.18d) yields

$$V_{t_2,t_2} - L^{(t)}_{t_2,t_1} U^{(t)}_{t_1,t_2} = L^{(t)}_{t_2,t_2} U^{(t)}_{t_2,t_2},$$

and we compute the left-hand side by constructing $V'_{t_2,t_2} := V_{t_2,t_2} - L^{(t)}_{t_2,t_1} U^{(t)}_{t_1,t_2}$, using a recursion to find its LU factorization $V'_{t_2,t_2} = L^{(t_2)}U^{(t_2)}$, and setting $L^{(t)}_{t_2,t_2} := L^{(t_2)}$ and $U^{(t)}_{t_2,t_2} := U^{(t_2)}$.

Algorithm 55. LU decomposition.

procedure LUDecomposition(t, X, **var** L, U);
if sons(t) $= \emptyset$ **then**
  Compute factorization $X = LU$
else
  $\tau \leftarrow \#\,\mathrm{sons}(t)$; $\{t_1, \ldots, t_\tau\} \leftarrow \mathrm{sons}(t)$;
  for $i, j \in \{1, \ldots, \tau\}$ **do**
    $X_{ij} \leftarrow X|_{\hat{t}_i \times \hat{t}_j}$
  end for;
  for $i = 1$ **to** τ **do**
    LUDecomposition(t_i, X_{ii}, **var** L_{ii}, U_{ii});
    for $j \in \{i+1, \ldots, \tau\}$ **do**
      $U_{ij} \leftarrow X_{ij}$;
      LowerBlockSolve(t_i, L_{ii}, U_{ij}); {Algorithm 56}
      $L_{ji} \leftarrow X_{ji}$;
      UpperBlockSolve(t_i, U_{ii}, L_{ji}) {Algorithm 57}
    end for;
    for $j, k \in \{i+1, \ldots, \tau\}$ **do**
      $X_{jk} \leftarrow X_{jk} - L_{ji} U_{ik}$ {Use approximative multiplication}
    end for
  end for;
  for $i, j \in \{1, \ldots, \tau\}$ **do**
    if $i > j$ **then**
      $L|_{\hat{t}_i, \hat{t}_j} \leftarrow L_{ij}$; $U|_{\hat{t}_i, \hat{t}_j} \leftarrow 0$ {Unify resulting matrices by Algorithm 32}
    else if $i < j$ **then**
      $L|_{\hat{t}_i, \hat{t}_j} \leftarrow 0$; $U|_{\hat{t}_i, \hat{t}_j} \leftarrow U_{ij}$ {Unify resulting matrices by Algorithm 32}
    else
      $L|_{\hat{t}_i, \hat{t}_j} \leftarrow L_{ij}$; $U|_{\hat{t}_i, \hat{t}_j} \leftarrow U_{ij}$ {Unify resulting matrices by Algorithm 32}
    end if
  end for
end if

The equations (10.18b) and (10.18c) are solved by recursive forward substitution: for an arbitrary finite index set $\mathcal{K}$ and a matrix $Y \in \mathbb{R}^{\hat{I} \times \mathcal{K}}$, we can find the solution $X \in \mathbb{R}^{\hat{I} \times \mathcal{K}}$ of $Y = L^{(t)} X$ by again using the block representation. We let $X_{t_1} := X|_{\hat{t}_1 \times \mathcal{K}}$, $X_{t_2} := X|_{\hat{t}_2 \times \mathcal{K}}$, $Y_{t_1} := Y|_{\hat{t}_1 \times \mathcal{K}}$ and $Y_{t_2} := Y|_{\hat{t}_2 \times \mathcal{K}}$ and get

$$\begin{pmatrix} Y_{t_1} \\ Y_{t_2} \end{pmatrix} = \begin{pmatrix} L^{(t)}_{t_1,t_1} & \\ L^{(t)}_{t_2,t_1} & L^{(t)}_{t_2,t_2} \end{pmatrix} \begin{pmatrix} X_{t_1} \\ X_{t_2} \end{pmatrix},$$

which is equivalent to

$$Y_{t_1} = L^{(t)}_{t_1,t_1} X_{t_1}, \tag{10.19a}$$

$$Y_{t_2} = L^{(t)}_{t_2,t_1} X_{t_1} + L^{(t)}_{t_2,t_2} X_{t_2}. \tag{10.19b}$$

The first equation (10.19a) is solved by recursion. For the second equation (10.19b), we compute the matrix $Y'_{t_2} := Y_{t_2} - L^{(t)}_{t_2,t_1} X_{t_1}$ and then solve the equation $Y'_{t_2} = L^{(t)}_{t_2,t_2} X_{t_2}$ by recursion. For the general case $\#\operatorname{sons}(t) \in \mathbb{N}$, the entire recursion is summarized in Algorithm 56.

Algorithm 56. Forward substitution for a lower triangular block matrix.

```
procedure LowerBlockSolve(t, L, var X);
if sons(t) = ∅ then
  Y ← X;
  Apply standard backward substitution to solve LX = Y
else
  τ ← #sons(t); {t_1, ..., t_τ} ← sons(t);
  for i, j ∈ {1, ..., τ} do
    L_ij ← L|_{t̂_i × t̂_j}
  end for;
  for i ∈ {1, ..., τ} do
    X_i ← X|_{t̂_i × K}
  end for;
  for i = 1 to τ do
    LowerBlockSolve(t_i, L_ii, X_i);
    for j ∈ {i + 1, ..., τ} do
      X_j ← X_j − L_ji X_i          {Use approximative multiplication}
    end for
  end for;
  for i ∈ {1, ..., τ} do
    X|_{t̂_i × K} ← X_i              {Unify resulting matrix by Algorithm 32}
  end for
end if
```

A similar procedure can be applied to (10.18c): for a finite index set $\mathcal{K}$ and a matrix $Y \in \mathbb{R}^{\mathcal{K}\times\hat{t}}$, we compute the solution $X \in \mathbb{R}^{\mathcal{K}\times\hat{t}}$ of $Y = XU^{(t)}$ by considering the subblocks $X_{t_1} := X|_{\mathcal{K}\times\hat{t}_1}$, $X_{t_2} := X|_{\mathcal{K}\times\hat{t}_2}$, $Y_{t_1} := Y|_{\mathcal{K}\times\hat{t}_1}$ and $Y_{t_2} := Y|_{\mathcal{K}\times\hat{t}_2}$ and the corresponding block equation

$$\begin{pmatrix} Y_{t_1} & Y_{t_2} \end{pmatrix} = \begin{pmatrix} X_{t_1} & X_{t_2} \end{pmatrix} \begin{pmatrix} U^{(t)}_{t_1,t_1} & U^{(t)}_{t_1,t_2} \\ & U^{(t)}_{t_2,t_2} \end{pmatrix}.$$

The resulting equations

$$Y_{t_1} = X_{t_1} U^{(t)}_{t_1,t_1}, \tag{10.20a}$$

$$Y_{t_2} = X_{t_1} U^{(t)}_{t_1,t_2} + X_{t_2} U^{(t)}_{t_2,t_2} \tag{10.20b}$$

are solved by applying recursion to (10.20a), computing $Y'_{t_2} := Y_{t_2} - X_{t_1} U^{(t)}_{t_1,t_2}$ and then applying recursion to $Y'_{t_2} = X_{t_2} U^{(t)}_{t_2,t_2}$. The entire recursion is given in Algorithm 57.

Algorithm 57. Forward substitution for an upper triangular block matrix.

```
procedure UpperBlockSolve(t, U, var X);
if sons(t) = ∅ then
  Y ← X;
  Apply standard backward substitution to solve XU = Y
else
  τ ← #sons(t); {t_1, ..., t_τ} ← sons(t);
  for i, j ∈ {1, ..., τ} do
    U_ij ← U|_{t̂_i × t̂_j}
  end for;
  for i ∈ {1, ..., τ} do
    X_i ← X|_{K × t̂_i}
  end for;
  for i = 1 to τ do
    UpperBlockSolve(t_i, U_ii, X_i);
    for j ∈ {i + 1, ..., τ} do
      X_j ← X_j − X_i U_ij          {Use approximative multiplication}
    end for
  end for;
  for i ∈ {1, ..., τ} do
    X|_{K × t̂_i} ← X_i              {Unify resulting matrix by Algorithm 32}
  end for
end if
```

The Algorithms 55, 56 and 57 are given in the standard notation for dense matrices and will compute the *exact* LU factorization of a given matrix. An excellent preconditioner for our system $Vx = b$ would then be the matrix $P = LU$, since $P^{-1} = U^{-1}L^{-1}$ can be easily computed by forward and backward substitution and $P^{-1/2}VP^{-1/2} = I$ means that the resulting preconditioned method would converge in one step.

Approximate $\mathcal{H}^2$-LU decomposition

This strategy is only useful for small problems, i.e., in situations when the matrices can be stored in standard array representation. For large problems, we cannot treat V in this way, and we also cannot compute the factors L and U in this representation, either. Therefore we use the $\mathcal{H}^2$-matrix approximation $\widetilde{V}$ of V and try to compute $\mathcal{H}^2$-matrix approximations $\widetilde{L}$ and $\widetilde{U}$ of the factors L and U.

Fortunately, switching from the original Algorithms 55, 56 and 57 to their approximative $\mathcal{H}^2$-matrix counterparts is straightforward: we replace the exact matrix-matrix multiplication by the approximative multiplication algorithms introduced in Chapter 7 or Chapter 8. This allows us to compute all submatrices L_{ij}, U_{ij} and X_{ij} in the algorithms efficiently. Since we are working with $\mathcal{H}^2$-matrices, we cannot simply combine the submatrices to form L, U or X, but we have to use the unification Algorithm 32 to construct uniform row and column cluster bases for the results.

Assuming that $\tilde{L}\tilde{U} \approx \tilde{V}$ is a sufficiently good approximation, we can now use $P := \tilde{L}\tilde{U}$ as a preconditioner if we are able to provide efficient algorithms for evaluating $P^{-1} = \tilde{U}^{-1}\tilde{L}^{-1}$, which is equivalent to solving $\tilde{L}x = y$ and $\tilde{U}x = y$ for vectors $x, y \in \mathbb{R}^{\mathcal{I}}$.

We consider only the forward substitution, i.e., the computation of $x \in \mathbb{R}^{\mathcal{I}}$ solving $Lx = y$ for a vector $y \in \mathbb{R}^{\mathcal{I}}$ and a lower triangular $\mathcal{H}^2$-matrix for a cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ and nested cluster bases $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $W = (W_s)_{s\in\mathcal{T}_{\mathcal{I}}}$ with transfer matrices $(E_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(F_s)_{s\in\mathcal{T}_{\mathcal{I}}}$, respectively. Let us first investigate the simple case $\#\operatorname{sons}(t) = 2$ and $\{t_1, t_2\} = \operatorname{sons}(t)$. As before, we let $L_{11} := L|_{\hat{t}_1\times\hat{t}_1}$, $L_{21} := L|_{\hat{t}_2\times\hat{t}_1}$, $L_{22} := L|_{\hat{t}_2\times\hat{t}_2}$, $x_1 := x|_{\hat{t}_1}$, $x_2 := x|_{\hat{t}_2}$, $y_1 := y|_{\hat{t}_1}$ and $y_2 := y|_{\hat{t}_2}$ and observe that $Lx = y$ is equivalent to

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

i.e., to the two equations

$$\begin{aligned} y_1 &= L_{11}x_1, \\ y_2 &= L_{21}x_1 + L_{22}x_2. \end{aligned}$$

The first of these equations can be solved by recursion, for the second one we introduce $y_2' := y_2 - L_{21}x_1$ and solve $L_{22}x_2 = y_2'$, again by recursion. So far, the situation is similar to the one encountered in the discussion of Algorithm 56.

Now we have to take a look at the computation of y_2'. If we would use the standard matrix-vector multiplication Algorithm 8 to compute $L_{21}x_1$, the necessary forward and backward transformations would lead to a suboptimal complexity, since some coefficient vectors would be recomputed on all levels of the cluster tree. In order to reach the optimal order of complexity, we have to avoid these redundant computations.

Fortunately, this is possible: in the simple case of our 2×2 block matrix, we can see that the computation of x_1 requires no matrix-vector multiplication, and therefore no forward or backward transformations. The computation of $L_{21}x_1$ requires the coefficient vectors $\hat{x}_{s^*} := W_{s^*}x$ for all $s^* \in \operatorname{sons}^*(t_1)$ and computes contributions to y only in the coefficient vectors $\hat{y}_{s^*}$ for all $s^* \in \operatorname{sons}^*(t_2)$.

This observation suggests a simple algorithm: we assume that y is given in the form

$$y = y_0 + \sum_{t^*\in\operatorname{sons}^*(t)} V_{t^*}\hat{y}_{t^*}$$

and that we have to compute $x|_{\hat{t}}$ and all coefficient vectors $\hat{x}_{s^*} := W_{s^*}^* x$ for $s^* \in \operatorname{sons}^*(t)$.

If t is a leaf, we have $\operatorname{sons}^*(t) = \{t\}$ and compute $y = y_0 + V_t \hat{y}_t$ directly. Since t is a leaf, $L|_{\hat{t}\times\hat{t}}$ is a dense matrix and we use the standard forward substitution algorithm for dense matrices. Then we can compute $\hat{x}_t := W_t^* x$ directly.

If t is not a leaf, we use the equation

$$V_t \hat{y}_t = \sum_{t' \in \operatorname{sons}(t)} V_{t'} E_{t'} \hat{y}_t$$

to express $\hat{y}_t$ in terms of the sons of t, perform the recursion described above, and then assemble the coefficient vector

$$\hat{x}_t = \sum_{t' \in \operatorname{sons}(t)} F_{t'}^* \hat{x}_{t'}.$$

In short, we mix the elementary steps of the forward and backward transformations with the computation of the solution.

Algorithm 58. Forward substitution.

```
procedure SolveLower(t, L, var x, y, x̂, ŷ);
if sons(t) = ∅ then
  y|_t̂ ← y|_t̂ + (V_t ŷ_t)|_t̂;
  Solve L|_{t̂×t̂} x|_t̂ = y|_t̂;
  x̂_t ← W_t^* x
else
  τ ← #sons(t); {t_1, ..., t_τ} ← sons(t);
  x̂_t ← 0;
  for i = 1 to τ do
    ŷ_{t_i} ← ŷ_{t_i} + E_{t_i} ŷ_t;
    for j ∈ {1, ..., i − 1} do
      b' ← (t_i, t_j);
      for b^* = (t^*, s^*) ∈ sons^*(b') do
        if b^* ∈ L^-_{I×I} then
          y|_{t̂^*} ← y|_{t̂^*} − L|_{t̂^*×ŝ^*} x|_{ŝ^*}
        else if b^* ∈ L^+_{I×I} then
          ŷ_{t^*} ← ŷ_{t^*} − S_{b^*} x̂_{s^*}
        end if
      end for
    end for;
    SolveLower(t_i, L, x, y, x̂, ŷ);
    x̂_t ← x̂_t + F_{t_i}^* x̂_{t_i}
  end for
end if
```

The resulting recursive procedure is given in Algorithm 58. An efficient algorithm for the backward substitution to solve the system $Ux = y$ can be derived in a very similar way. A closer investigation reveals that both algorithms have the same complexity as the standard matrix-vector multiplication Algorithm 8, i.e., that their complexity can be considered to be of optimal order.

Experiments

In the first experiment, we construct an approximation $\widetilde{V}$ of the matrix V for a polygonal approximation of the unit sphere with $n = 32768$ degrees of freedom using the same parameters as before, i.e., with an interpolation order of $m = 6$, a tolerance $\hat{\epsilon}_{\mathrm{rc}} = 10^{-6}$ for the recompression and a quadrature order of 3. We have already seen that these parameters ensure that the error introduced by the matrix approximation is sufficiently small. The approximation of $\widetilde{V}$ requires 336.8 MB, its nearfield part accounts for 117.5 MB, and the construction of the matrix is accomplished in 1244.7 seconds.

The preconditioner is constructed by computing an approximate $\mathcal{H}^2$-matrix LU factorization of $\widetilde{V}$ with an error tolerance $\hat{\epsilon}$ for the blockwise relative spectral error. Table 10.4 lists the experimental results for different values of $\hat{\epsilon}$: the first column contains the value of $\hat{\epsilon}$, the second the time in seconds required for the construction of $\widetilde{L}$ and $\widetilde{U}$, the third the storage requirements in MB, the fourth the storage requirements per degree of freedom in KB, the fifth an estimate for $\epsilon := \|\widetilde{V} - \widetilde{L}\widetilde{U}\|_2$ computed by a number of steps of the power iteration, the sixth gives the maximal number of steps that where required to solve the problems (10.5) and (10.13), and the last column gives the time required for solving the system.

Without preconditioning, up to 321 steps of the conjugate gradient method are required to solve one of the problems (10.5) or (10.13), and due to the corresponding large number of matrix-vector multiplications the solver takes 270 seconds.

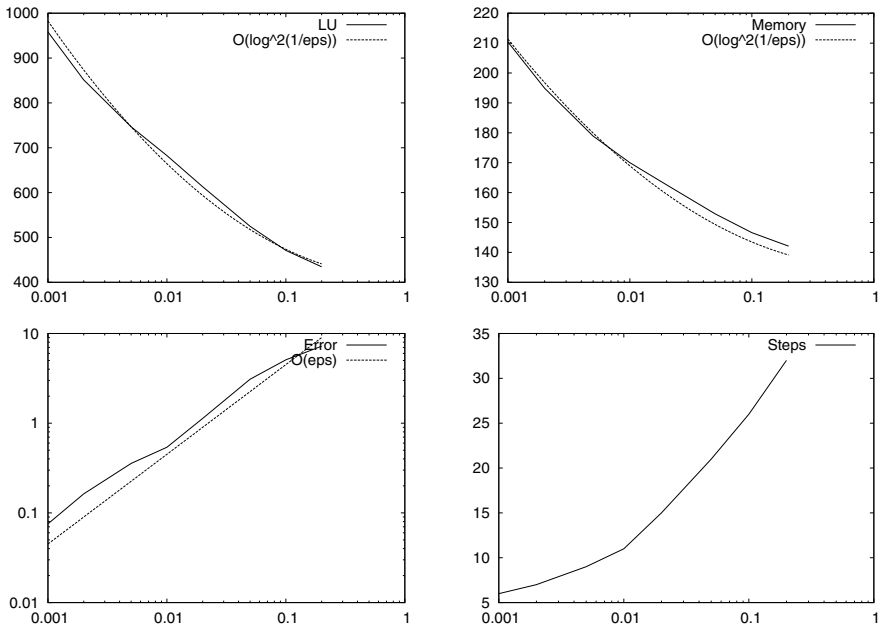
Even with the lowest accuracy $\hat{\epsilon} = 0.2$, the number of iterations is reduced significantly, and only 44.4 seconds are required by the solver, and the storage requirements of 142.1 MB are clearly dominated by the nearfield part of 117.5 MB and less than half of the amount needed by $\widetilde{V}$. Although the estimate for ϵ seems to indicate that the approximate LU factors are too inaccurate, they are well-suited for the purpose of preconditioning. The construction of the approximate LU factors requires 434.6 seconds, i.e., approximately a third of the time required to find the approximate matrix $\widetilde{V}$.

If we use smaller values of $\hat{\epsilon}$, we can see that the error ϵ is proportional to the error tolerance $\hat{\epsilon}$ and that the storage requirements and the time for preparing the preconditioner seem to grow like $\log^2(1/\hat{\epsilon})$.

Now let us consider the behaviour of the preconditioner for different matrix dimensions n. We have seen in the previous experiment that the error ϵ is proportional to the error tolerance $\hat{\epsilon}$. Since the condition number can be expected to grow like h^{-1} as the mesh parameter h decreases, we use the choice $\hat{\epsilon} \approx h$ to ensure stable convergence rates.

Table 10.4. Preconditioners of different quality for the single layer potential matrix V and the matrix dimension $n = 32768$.

$\hat{\epsilon}$	LU	Mem	Mem/n	ϵ	Steps	Solve
2_{-1}	434.6	142.1	4.4	7.2_{0}	32	44.4
1_{-1}	471.1	146.6	4.6	5.1_{0}	26	36.1
5_{-2}	524.8	152.9	4.8	3.1_{0}	21	30.2
2_{-2}	613.1	162.6	5.1	1.1_{0}	15	22.7
1_{-2}	682.6	170.0	5.3	5.4_{-1}	11	17.3
5_{-3}	746.7	178.9	5.6	3.6_{-1}	9	15.1
2_{-3}	851.3	194.9	6.1	1.6_{-1}	7	12.2
1_{-3}	958.3	210.4	6.6	7.6_{-2}	6	11.0

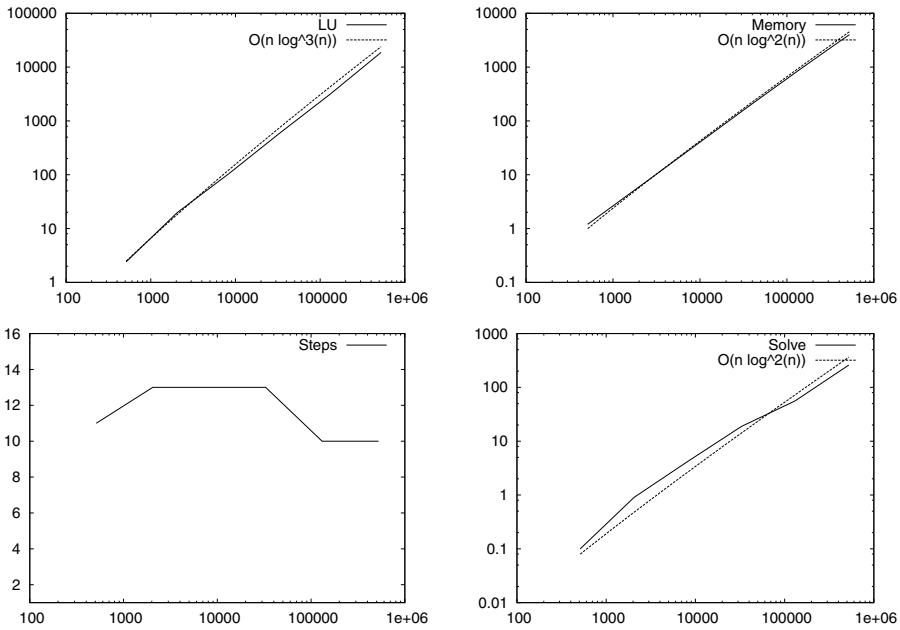


The results in Table 10.5 suggest that this approach works: the number of iteration steps required for solving the system is bounded and even seems to decay for large problem dimensions, while the time for the construction of the preconditioner seems to grow like $n \log^3(n)$, i.e., the ratio between the time required for the construction of $\widetilde{V}$ and the preconditioner improves as n grows. Both the storage requirements and the time for solving the linear system grow like $n \log^2(n)$, i.e., at the same rate as the storage requirements for the matrix $\widetilde{V}$.

We can conclude that the preconditioner works well: the time for solving the linear systems (10.5) and (10.13) is proportional to the storage requirements for the matrix

Table 10.5. Preconditioners for the single layer potential matrix V for different matrix dimensions n and $\hat{\epsilon} \approx h$.

n	$\hat{\epsilon}$	Build	LU	Mem	Mem/n	ϵ	Steps	Solve
512	1.0_{-1}	3.1	2.4	1.2	2.4	2.0_{-2}	11	0.1
2048	6.7_{-2}	21.2	19.7	6.1	3.1	5.5_{-1}	13	0.9
8192	3.2_{-2}	151.9	102.2	32.1	4.0	9.1_{-1}	13	4.2
32768	1.6_{-2}	1038.3	582.1	165.1	5.2	7.5_{-1}	13	18.8
131072	7.9_{-3}	6953.5	3131.9	825.9	6.5	9.9_{-1}	10	56.1
524288	3.9_{-3}	66129.6	18771.4	4009.8	7.8	1.2_{0}	10	259.8



approximation $\widetilde{V}$, which is the best result we can hope for, and the construction of the preconditioner takes less time than the construction of $\widetilde{V}$, it even becomes more efficient as the problem dimension grows.

Remark 10.3 (Symmetry). If $\widetilde{V}$ is symmetric and positive definite, it is advisable to use the Cholesky decomposition instead of the general LU decomposition. In this way, the computation time can be halved, and the Cholesky factorization can be computed even if only the lower half of $\widetilde{V}$ has been stored (cf. Remark 10.2). □

10.4 Application to realistic geometries

Until now, we have only considered problems on "academical" domains: the unit circle or sphere and the unit square. Since these geometries are very smooth, we expect the discretization error to be small, at least for those solutions that are of practical interest. According to Strang's first lemma, this means that we have to ensure that the approximation of the matrices V and K is very accurate if we intend to get the optimal result.

For geometries appearing in practical applications, a lower accuracy may be acceptable, since the discretization error will usually be quite large, but the algorithm has to be able to handle non-uniform meshes, possibly with fine geometric details.

In order to investigate the performance of our techniques under these conditions, we now consider geometries that are closer to practical applications: Figure 10.1 contains a surface resembling a crank shaft[1], a sphere[2] pierced with cylindrical "drill holes" and an object with foam-like structure[3].

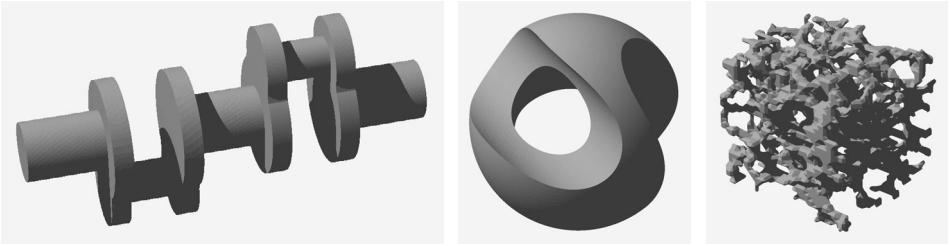


Figure 10.1. Examples of practical computational domains.

We test our algorithms in the same way as in the previous section: we construct an $\mathcal{H}^2$-matrix approximation of the matrix V on the surface mesh by compressing an initial approximation based on interpolation, we compute the L^2-projection of the Dirichlet values and apply an $\mathcal{H}^2$-matrix approximation of the matrix K to compute the right-hand side of equation (10.14). Then we solve the equation by a conjugate gradient method using an approximate LU factorization of $\widetilde{V}$ as a preconditioner.

Since the triangulation is less regular than the ones considered before, we use higher quadrature orders $q \in \{4, 5, 6\}$ to compute the far- and nearfield entries and determine the accuracy of the matrix approximation by trial and error: the matrices are constructed for the interpolation orders $m \in \{6, 7, 8\}$ and the error tolerances $\hat{\epsilon} \in \{10^{-6}, 10^{-7}, 10^{-8}\}$, and we check whether the approximation of the Neumann data changes.

The results for the "crank shaft" problem are given in Table 10.6: switching from $m = 6$ to $m = 7$ hardly changes the errors ϵ_2 and ϵ_3, only the error ϵ_1 is affected.

[1] This mesh was created by the NGSolve package by Joachim Schöberl.
[2] Also created by NGSolve.
[3] This mesh is used by courtesy of Heiko Andrä and Günther Of.

Table 10.6. Matrix approximations of different degrees of accuracy for the "crank shaft" geometry.

			Matrix V		Matrices LU					
m	q	$\hat{\epsilon}$	Build	Mem	Build	Mem	Slv.	ϵ_1	ϵ_2	ϵ_3
6	4	1_{-6}	1543.3	456.3	1011.5	180.1	20.6	5.7_{-3}	1.1_{-2}	5.5_{-3}
7	4	1_{-7}	2478.7	597.1	881.4	213.1	18.9	3.1_{-3}	1.1_{-2}	4.1_{-3}
8	4	1_{-8}	5009.6	753.3	893.3	248.9	20.1	2.9_{-3}	1.1_{-2}	4.0_{-3}
6	5	1_{-6}	2268.6	456.3	1016.3	180.1	21.1	5.1_{-3}	1.1_{-2}	4.3_{-3}
7	5	1_{-7}	3505.1	597.1	890.6	213.1	18.9	1.7_{-3}	1.1_{-2}	2.2_{-3}
8	5	1_{-8}	6319.0	753.3	890.3	248.9	20.1	1.3_{-3}	1.1_{-2}	2.1_{-3}
7	6	1_{-7}	5311.9	597.1	885.3	213.1	19.2	1.4_{-3}	1.1_{-2}	2.0_{-3}
8	6	1_{-8}	8660.5	753.3	896.0	248.9	20.2	9.7_{-4}	1.1_{-2}	1.8_{-3}

If we increase the quadrature order, the errors ϵ_1 and ϵ_3 decrease, the error ϵ_2 is not affected. As mentioned before, ϵ_1 is only determined by the quadrature and matrix approximation errors, since the corresponding solution $u_{N,1}$ is contained in the discrete space and therefore would be represented exactly by an exact Galerkin scheme.

If we increase the interpolation error to $m = 8$, all errors remain almost unchanged. The fact that ϵ_1 changes only slightly suggests that the remaining errors are caused by the quadrature scheme, not by the matrix approximation.

Table 10.7. Matrix approximations of different degrees of accuracy for the "pierced sphere" problem.

			Matrix V		Matrices LU					
m	q	$\hat{\epsilon}$	Build	Mem	Build	Mem	Slv.	ϵ_1	ϵ_2	ϵ_3
6	4	1_{-6}	1682.1	440.4	955.1	187.2	24.2	7.8_{-2}	4.4_{-2}	6.0_{-2}
7	4	1_{-7}	2696.0	577.2	873.1	213.9	24.2	7.8_{-2}	4.4_{-2}	6.0_{-2}
8	4	1_{-8}	5155.2	729.5	883.5	249.3	23.8	7.8_{-2}	4.4_{-2}	6.0_{-2}
6	5	1_{-6}	2444.1	440.4	959.2	187.2	24.3	4.3_{-2}	3.8_{-2}	3.3_{-2}
7	5	1_{-7}	3705.9	577.2	871.4	213.9	22.8	4.3_{-2}	3.8_{-2}	3.3_{-2}
5	6	5_{-6}	2595.6	359.2	1168.0	174.7	25.7	2.7_{-2}	3.7_{-2}	2.0_{-2}
6	6	1_{-6}	3758.8	440.4	962.1	187.2	24.2	2.2_{-2}	3.6_{-2}	1.8_{-2}
7	6	1_{-7}	5497.8	577.2	874.8	213.9	22.8	2.2_{-2}	3.6_{-2}	1.8_{-2}

We can conclude that an interpolation order of $m = 7$ and a quadrature order of $q = 5$ are sufficient to approximate the solutions $u_{N,2}$ and $u_{N,3}$ up to their respective discretization errors.

For the "pierced sphere" problem, the results are collected in Table 10.7: even an interpolation order of $m = 6$ seems to be sufficient, since changing to $m = 7$ or $m = 8$ does not significantly change the approximation error. Increasing the quadrature order q, on the other hand, reduces the errors ϵ_1 and ϵ_3, therefore we can conclude that the quadrature error dominates and the $\mathcal{H}^2$-matrix approximation is sufficiently accurate.

For the "foam block" problem, the situation is similar: Table 10.8 shows that an interpolation order of $m = 5$ seems to be sufficient, the quadrature error dominates. The approximations of $u_{N,2}$ and $u_{N,3}$ are only slightly changed if the interpolation or the quadrature order are increased.

Table 10.8. Matrix approximations of different degrees of accuracy for the "foam block" problem.

			Matrix V		Matrices LU					
m	q	$\hat{\epsilon}$	Build	Mem	Build	Mem	Slv.	ϵ_1	ϵ_2	ϵ_3
5	4	5_{-6}	1926.0	817.3	2151.0	290.0	91.1	6.9_{-2}	1.4_{-1}	7.2_{-2}
6	4	1_{-6}	2969.7	976.7	1807.4	317.8	85.5	6.3_{-2}	1.4_{-1}	7.1_{-2}
7	4	1_{-7}	4693.8	1240.3	1703.4	382.7	78.7	6.2_{-2}	1.4_{-1}	7.1_{-2}
8	4	1_{-8}	8017.0	1521.0	1767.4	469.4	78.6	6.2_{-2}	1.4_{-1}	7.1_{-2}
5	5	5_{-6}	2852.8	817.3	2180.5	290.0	90.9	4.1_{-2}	1.4_{-1}	6.6_{-2}
6	5	1_{-6}	4295.7	976.7	1822.2	317.8	83.4	2.8_{-2}	1.4_{-1}	6.4_{-2}
7	5	1_{-7}	6561.9	1240.3	1693.7	382.6	79.3	2.7_{-2}	1.4_{-1}	6.4_{-2}
8	5	1_{-8}	10861.6	1521.0	1769.5	469.4	78.8	2.7_{-2}	1.4_{-1}	6.4_{-2}
5	6	5_{-6}	4503.1	817.3	2164.0	290.0	92.3	3.3_{-2}	1.4_{-1}	6.4_{-2}
6	6	1_{-6}	6636.4	976.7	1816.3	317.9	88.3	1.3_{-2}	1.4_{-1}	6.2_{-2}

10.5 Solution operators of elliptic partial differential equations

All examples we have considered so far have been, in one way or another, based on polynomial expansion: even the hybrid cross approximation relies on interpolation. Now we investigate a problem class that cannot be treated in this way: the approximation of the inverse of a finite element stiffness matrix corresponding to an elliptic partial differential equation (cf. Chapter 9).

Finite element discretization

We consider the equation

$$-\operatorname{div} C(x) \operatorname{grad} u(x) = f(x) \quad \text{for all } x \in \Omega, \tag{10.21a}$$

$$u(x) = 0 \quad \text{for all } x \in \partial\Omega, \tag{10.21b}$$

where $\Omega \subseteq \mathbb{R}^2$ is a bounded domain and $C \in L^\infty(\Omega, \mathbb{R}^{2\times 2})$ is a mapping that assigns a 2×2-matrix to each point $x \in \Omega$. We assume that C is symmetric and that there are constants $\alpha, \beta \in \mathbb{R}$ with $0 < \alpha \leq \beta$ and $\sigma(C(x)) \subseteq [\alpha, \beta]$ for all $x \in \Omega$, i.e., that the partial differential operator is strongly elliptic.

The variational formulation of the partial differential equation (10.21) is given by

$$\int_\Omega \langle \nabla v(x), C(x)\nabla u(x)\rangle_2 \, dx = \int_\Omega v(x) f(x) \, dx \quad \text{for all } v \in H_0^1(\Omega), \tag{10.22}$$

where $u \in H_0^1(\Omega)$ is now only a weak solution (which nevertheless coincides with u if a classical solution exists). The variational formulation is discretized by a Galerkin scheme: as in Section 10.2, we use the space S_1 spanned by the family $(\varphi_i)_{i\in\mathcal{I}}$ of piecewise linear nodal basis functions on a conforming triangulation $\mathcal{G}$ and consider the equation

$$Ax = b, \tag{10.23}$$

where the matrix $A \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ is defined by

$$A_{ij} = \int_\Omega \langle \nabla\varphi_i(x), C(x)\nabla\varphi_j(x)\rangle_2 \, dx \quad \text{for all } i, j \in \mathcal{I}, \tag{10.24}$$

the vector $b \in \mathbb{R}^{\mathcal{I}}$ is defined by

$$b_i = \int_\Omega \varphi_i(x) f(x) \, dx \quad \text{for all } i \in \mathcal{I}$$

and the solution vector $x \in \mathbb{R}^{\mathcal{I}}$ corresponds to the discrete solution

$$u_h := \sum_{i\in\mathcal{I}} x_i \varphi_i$$

of the variational equation. A is a symmetric positive definite matrix, therefore we could compute approximations of its Cholesky or LU factorization in order to construct efficient preconditioners (cf. [52], [79], [56]).

A factorization is very efficient if we are only interested in solving a system of linear equations, but there are tasks, e.g., the computation of the Schur complement of a saddle point problem or the treatment of a matrix equation, when we require a sufficiently accurate approximation of the inverse $B := A^{-1}$ of the matrix A.

Inversion

As in the previous section, we reduce this task to a blockwise recursion using matrix-matrix multiplications. For all $t, s \in \mathcal{T}_{\mathcal{I}}$, we denote the submatrices corresponding to the block (t, s) by $A_{t,s} := A|_{\hat{t}\times\hat{s}}$ and $B_{t,s} := B|_{\hat{t}\times\hat{s}}$. Let $t \in \mathcal{T}_{\mathcal{I}}$ be a cluster. If t is a leaf cluster, the admissibility condition implies that $(t, t) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}$ is an inadmissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, and A_{tt} is stored in standard array representation. Its inverse $B^{(t)}$ can be computed by the well-known algorithms from linear algebra.

If t is not a leaf cluster, $A_{t,t}$ is a block matrix. As in the previous section, we consider only the simple case $\#\operatorname{sons}(t) = 2$ with $\operatorname{sons}(t) = \{t_1, t_2\}$. The defining equation $A_{t,t}B^{(t)} = I$ of $B^{(t)}$ takes the form

$$\begin{pmatrix} A_{t_1,t_1} & A_{t_1,t_2} \\ A_{t_2,t_1} & A_{t_2,t_2} \end{pmatrix} \begin{pmatrix} B^{(t)}_{t_1,t_1} & B^{(t)}_{t_1,t_2} \\ B^{(t)}_{t_2,t_1} & B^{(t)}_{t_2,t_2} \end{pmatrix} = \begin{pmatrix} I & \\ & I \end{pmatrix}.$$

We compute the inverse $B^{(t_1)}$ of A_{t_1,t_1} by recursion and multiply the equation by the matrix

$$\begin{pmatrix} B^{(t_1)} & \\ -A_{t_2,t_1}B^{(t_1)} & I \end{pmatrix}$$

in order to get

$$\begin{pmatrix} I & B^{(t_1)}A_{t_1,t_2} \\ 0 & A_{t_2,t_2} - A_{t_2,t_1}B^{(t_1)}A_{t_1,t_2} \end{pmatrix} \begin{pmatrix} B^{(t)}_{t_1,t_1} & B^{(t)}_{t_1,t_2} \\ B^{(t)}_{t_2,t_1} & B^{(t)}_{t_2,t_2} \end{pmatrix} = \begin{pmatrix} B^{(t_1)} & 0 \\ -A_{t_2,t_1}B^{(t_1)} & I \end{pmatrix}.$$

By denoting the *Schur complement* by $S^{(t_2)} := A_{t_2,t_2} - A_{t_2,t_1}B^{(t_1)}A_{t_1,t_2}$, we can write this equation in the short form

$$\begin{pmatrix} I & B^{(t_1)}A_{t_1,t_2} \\ 0 & S^{(t_2)} \end{pmatrix} \begin{pmatrix} B^{(t)}_{t_1,t_1} & B^{(t)}_{t_1,t_2} \\ B^{(t)}_{t_2,t_1} & B^{(t)}_{t_2,t_2} \end{pmatrix} = \begin{pmatrix} B^{(t_1)} & 0 \\ -A_{t_2,t_1}B^{(t_1)} & I \end{pmatrix}.$$

We compute the inverse $T^{(t_2)}$ of $S^{(t_2)}$ by recursion and multiply the equation by the matrix

$$\begin{pmatrix} I & -B^{(t_1)}A_{t_1,t_2}T^{(t_2)} \\ & T^{(t_2)} \end{pmatrix},$$

which yields the desired result

$$\begin{aligned} \begin{pmatrix} B^{(t)}_{t_1,t_1} & B^{(t)}_{t_1,t_2} \\ B^{(t)}_{t_2,t_1} & B^{(t)}_{t_2,t_2} \end{pmatrix} &= \begin{pmatrix} I & -B^{(t_1)}A_{t_1,t_2}T^{(t_2)} \\ 0 & T^{(t_2)} \end{pmatrix} \begin{pmatrix} B^{(t_1)} & 0 \\ -A_{t_2,t_1}B^{(t_1)} & I \end{pmatrix} \\ &= \begin{pmatrix} B^{(t_1)} + B^{(t_1)}A_{t_1,t_2}T^{(t_2)}A_{t_2,t_1}B^{(t_1)} & -B^{(t_1)}A_{t_1,t_2}T^{(t_2)} \\ -T^{(t_2)}A_{t_2,t_1}B^{(t_1)} & T^{(t_2)} \end{pmatrix}. \end{aligned}$$

This construction of the inverse of $A_{t,t}$ gives rise to the recursive Algorithm 59: we start with the upper left block, invert it by recursion, and use it to compute the matrices $X_{1,2} := B^{(t_1)} A_{t_1,t_2}$ and $X_{2,1} := A_{t_2,t_1} B^{(t_1)}$. Then we overwrite A_{t_2,t_2} by the Schur complement, given by $A_{t_2,t_2} - A_{t_2,t_1} X_{1,2}$. In the next iteration of the loop, A_{t_2,t_2} is inverted, which gives us the lower right block of the inverse. The remaining blocks can be computed by multiplying this block by $X_{1,2}$ and $X_{2,1}$.

Algorithm 59. Matrix inversion, overwrites A with its inverse A^{-1} and uses X for temporary matrices.

```
procedure Inversion(t, var A, X);
if sons(t) = ∅ then
  Compute A^{-1} directly
else
  τ ← #sons(t); {t_1, ..., t_τ} ← sons(t);
  for i, j ∈ {1, ..., τ} do
    A_ij ← A|_{t̂_i × t̂_j}
  end for;
  for i = 1 to τ do
    Inversion(t_i, A_ii, X_ii);
    for j ∈ {i + 1, ..., τ} do
      X_ij ← A_ii A_ij;          {Use approximative multiplication}
      X_ji ← A_ji A_ii           {Use approximative multiplication}
    end for;
    for j, k ∈ {i + 1, ..., τ} do
      A_jk ← A_jk − A_ji X_ik    {Use approximative multiplication}
    end for
  end for;
  for i = τ downto 1 do
    for j ∈ {i + 1, ..., τ} do
      A_ij ← 0; A_ji ← 0;
      for k ∈ {i + 1, ..., τ} do
        A_ij ← A_ij − X_ik A_kj;  {Use approximative multiplication}
        A_ji ← A_ji − A_jk X_ki;  {Use approximative multiplication}
      end for;
      A_ii ← A_ii − X_ij A_ji    {Use approximative multiplication}
    end for
  end for;
  for i, j ∈ {1, ..., τ} do
    A|_{t_i,t_j} ← A_ij          {Unify resulting matrix by Algorithm 32}
  end for
end if
```

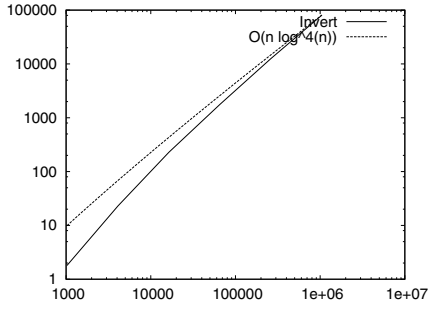
As in the case of the LU factorization, we only need matrix-matrix multiplications and recursion, therefore we can construct an approximate inverse by replacing the matrix-matrix products by their $\mathcal{H}^2$-matrix approximations, which can be computed by the algorithms introduced in the Chapters 7 and 8. Using the unification Algorithm 32, the blocks of the intermediate representation of the inverse can then be combined into the final $\mathcal{H}^2$-matrix representation.

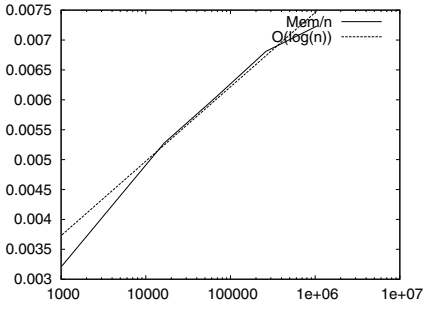
Experiments

We use the inversion Algorithm 59 to construct $\mathcal{H}^2$-matrix approximations of the stiffness matrix (10.24) for different types of coefficients. In our first experiment, we consider the simple case $C = I$, i.e., we discretize Poisson's equation on the square domain $[-1, 1]^2$ with a regular triangulation. Table 10.9 gives the computation times, storage requirements and approximation errors $\epsilon := \|I - BA\|_2$ for matrix dimensions ranging from $n = 1024$ to 1048576. The tolerance $\hat{\epsilon}$ for the approximative arithmetic operations is chosen to ensure $\epsilon \approx 2 \times 10^{-4}$. Since we can expect the condition number of A to grow like n, using $\hat{\epsilon} \sim 1/n$ should ensure a good approximation, and the experiment shows this leads to good results.

Table 10.9. Approximate inverse of Poisson's equation in two spatial dimensions for different matrix dimensions n.

n	$\hat{\epsilon}$	Build	Mem	Mem$/n$	ϵ
1024	4_{-6}	1.8	3.3	3.3	6.3_{-5}
4096	1_{-6}	23.0	17.4	4.3	2.0_{-4}
16384	1_{-7}	229.0	86.4	5.4	1.2_{-4}
65536	2_{-8}	1768.7	395.1	6.2	1.4_{-4}
262144	4_{-9}	12588.0	1785.2	7.0	1.5_{-4}
1048576	1_{-9}	80801.3	7585.0	7.4	2.2_{-4}





Combining Theorem 9.13 (cf. Theorem 2.8 in [6] for a related result for $\mathcal{H}$-matrices) with Lemma 9.10 yields that a rank of $\log(n)^3$ should be sufficient to approximate the inverse B of A, but the numerical experiment shows that in fact $\log(n)$ is sufficient, and Lemma 3.38 yields that the storage requirements are in $\mathcal{O}(n \log(n))$.

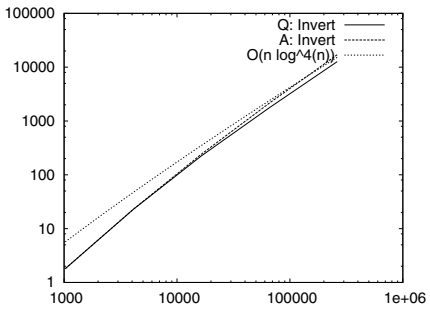
Our estimates also apply to strongly elliptic partial differential equations with non-smooth coefficients in L^∞, and even to anisotropic coefficients. We therefore consider two examples for non-smooth coefficients: in the first case, we split $\Omega = [-1, 1]^2$ into four equal quarters and choose $C = 100\, I$ in the lower left and upper right quarter and $C = I$ in the remaining quarters:

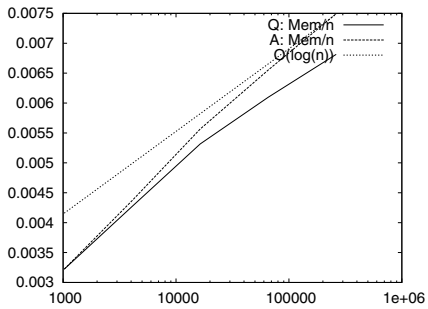
$$C_Q : \Omega \to \mathbb{R}^{2\times 2}, \quad x \mapsto \begin{cases} \begin{pmatrix} 100 & \\ & 100 \end{pmatrix} & \text{if } x \in [-1,0) \times [-1,0) \cup [0,1] \times [0,1], \\ \begin{pmatrix} 1 & \\ & 1 \end{pmatrix} & \text{otherwise.} \end{cases}$$

Due to the jump in the coefficients, the Green function for the corresponding elliptic differential operator will not be smooth, and approximating it by polynomials is not an option. The general theory of Section 6.3 and Chapter 9, on the other hand, can still be applied, therefore we expect to be able to approximate the inverse of A by an $\mathcal{H}^2$-matrix.

Table 10.10. Approximate inverse of elliptic partial differential equations with coefficients in L^∞ for different matrix dimensions n.

		Quartered			Anisotropy		
n	$\hat{\epsilon}$	Build	Mem	ϵ	Build	Mem	ϵ
1024	4_{-6}	1.8	3.3	9.2_{-5}	1.8	3.3	2.3_{-4}
4096	1_{-6}	22.7	17.5	2.6_{-4}	22.8	17.9	4.5_{-4}
16384	1_{-7}	223.6	87.1	1.7_{-4}	240.4	91.2	2.1_{-4}
65536	2_{-8}	1756.2	399.6	1.9_{-4}	2137.2	429.4	2.8_{-4}
262144	4_{-9}	12670.2	1786.4	2.2_{-4}	17021.6	1964.7	3.9_{-4}





We can conclude that $\mathcal{H}^2$-matrices can be used in situations where the underlying operator cannot be approximated by polynomials and that the $\mathcal{H}^2$-matrix arithmetic algorithms will find suitable approximations in almost linear complexity.

In a second example, we investigate the influence of anisotropic coefficients. We split Ω into two halves and use an anisotropic coefficient matrix in one of them:

$$C_A : \Omega \to \mathbb{R}^{2\times 2}, \quad x \mapsto \begin{cases} \begin{pmatrix} 100 & \\ & 1 \end{pmatrix} & \text{if } x \in [-1,1] \times [0,1], \\ \begin{pmatrix} 1 & \\ & 1 \end{pmatrix} & \text{otherwise.} \end{cases}$$

The approximate inverse is computed by using the recursive inversion Algorithm 59 in combination with the adaptive matrix-matrix multiplication introduced in Chapter 8 and yields the results given in Table 10.10. We can see that, compared to the case of Poisson's equation, the storage requirements are only slightly increased and still proportional to $n \log(n)$. For the anisotropic problem, the computation time is increased by approximately 35%, but it is still proportional to $n \log(n)^4$.

Bibliography

The numbers at the end of each item refer to the pages on which the respective work is cited.

[1] S. Amini and A. T. J. Profit, Multi-level fast multipole solution of the scattering problem. *Eng. Anal. Bound. Elem.* **27** (2003), 547–564. 3

[2] C. R. Anderson, An implementation of the fast multipole method without multipoles. *SIAM J. Sci. Statist. Comput.* **13** (1992), 923–947. 2

[3] L. Banjai and W. Hackbusch, Hierarchical matrix techniques for low- and high-frequency Helmholtz problems. *IMA J. Numer. Anal.* **28** (2008), 46–79. 3

[4] M. Bebendorf, Approximation of boundary element matrices. *Numer. Math.* **86** (2000), 565–589. 2, 248, 392

[5] M. Bebendorf, Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang-Matrizen. PhD thesis, Universität Saarbrücken, Saarbrücken 2000; dissertation.de -Verlag im Internet GmbH, Berlin 2001. 2

[6] M. Bebendorf and W. Hackbusch, Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numer. Math.* **95** (2003), 1–28. 2, 3, 6, 363, 367, 368, 370, 375, 376, 418

[7] M. Bebendorf and S. Rjasanow, Adaptive low-rank approximation of collocation matrices. *Computing* **70** (2003), 1–24. 2, 227, 248

[8] C. Bernardi and V. Girault, A local regularization operator for triangular and quadrilateral finite elements. *SIAM J. Numer. Anal.* **35** (1998), 1893–1916. 377

[9] G. Beylkin, R. Coifman, and V. Rokhlin, Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.* **44** (1991), 141–183. 2

[10] S. Börm, Approximation of integral operators by $\mathcal{H}^2$-matrices with adaptive bases. *Computing* **74** (2005), 249–271. 163, 212

[11] S. Börm, $\mathcal{H}^2$-matrix arithmetics in linear complexity. *Computing* **77** (2006), 1–28. 281, 329

[12] S. Börm, Adaptive variable-rank approximation of dense matrices. *SIAM J. Sci. Comput.* **30** (2007), 148–168. 212

[13] S. Börm, Data-sparse approximation of non-local operators by $\mathcal{H}^2$-matrices. *Linear Algebra Appl.* **422** (2007), 380–403. 6, 211

[14] S. Börm, Construction of data-sparse $\mathcal{H}^2$-matrices by hierarchical compression. *SIAM J. Sci. Comput.* **31** (2009), 1820–1839. 257

[15] S. Börm, Approximation of solution operators of elliptic partial differential equations by $\mathcal{H}$- and $\mathcal{H}^2$-matrices. *Numer. Math.* **115** (2010), 165–193. 3, 6, 363, 366, 376

[16] S. Börm and L. Grasedyck, Low-rank approximation of integral operators by interpolation. *Computing* **72** (2004), 325–332. 2

[17] S. Börm and L. Grasedyck, Hybrid cross approximation of integral operators. *Numer. Math.* **101** (2005), 221–249. 2, 75, 248, 394

[18] S. Börm, L. Grasedyck, and W. Hackbusch, Hierarchical matrices. Lecture Note 21 of the Max Planck Institute for Mathematics in the Sciences, Leipzig 2003. 49, 269, 366
http://www.mis.mpg.de/de/publications/andere-reihen/ln/lecturenote-2103.html

[19] S. Börm, L. Grasedyck, and W. Hackbusch, Introduction to hierarchical matrices with applications. *Eng. Anal. Bound. Elem.* **27** (2003), 405–422. 49

[20] S. Börm and W. Hackbusch, Approximation of boundary element operators by adaptive $\mathcal{H}^2$-matrices. In *Foundations of computational mathematics: Minneapolis 2002*, London Math. Soc. Lecture Note Ser. 312, Cambridge University Press, Cambridge 2004, 58–75. 163

[21] S. Börm and W. Hackbusch, Hierarchical quadrature of singular integrals. *Computing* **74** (2005), 75–100. 272

[22] S. Börm, M. Löhndorf, and J. M. Melenk, Approximation of integral operators by variable-order interpolation. Preprint 82/2002, Max Planck Institute for Mathematics in the Sciences, Leipzig 2002.
http://www.mis.mpg.de/publications/preprints/2002/prepr2002-82.html 129

[23] S. Börm, M. Löhndorf, and J. M. Melenk, Approximation of integral operators by variable-order interpolation. *Numer. Math.* **99** (2005), 605–643. 2, 3, 75, 94, 125, 129, 135, 161, 278

[24] S. Börm and J. Ostrowski, Fast evaluation of boundary integral operators arising from an eddy current problem. *J. Comput. Phys.* **193** (2004), 67–85. 6

[25] S. Börm and S. A. Sauter, BEM with linear complexity for the classical boundary integral operators. *Math. Comp.* **74** (2005), 1139–1177. 126

[26] D. Braess, *Finite Elemente*. 4th ed., Springer-Verlag, Berlin 2007. 390, 398

[27] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Comput. Phys. Comm.* **65** (1991), 24–38. 83

[28] A. Brandt and A. A. Lubrecht, Multilevel matrix multiplication and fast solution of integral equations. *J. Comput. Phys.* **90** (1990), 348–370. 83

[29] S. Chandrasekaran, M. Gu, and W. Lyons, A fast adaptive solver for hierarchically semiseparable representations. *Calcolo* **42** (2005), 171–185. 59

[30] S. Chandrasekaran, M. Gu, and T. Pals, Fast and stable algorithms for hierarchically semi-separable representations. Tech. rep., Department of Mathematics, University of California, Berkeley, 2004. 59

[31] S. Chandrasekaran, M. Gu, and T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM J. Matrix Anal. Appl.* **28** (2006), 603–622. 59

[32] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, A superfast algorithm for Toeplitz systems of linear equations. *SIAM J. Matrix Anal. Appl.* **29** (2007), 1247–1266. 59

[33] S. Chandrasekaran and I. C. F. Ipsen, On rank-revealing factorisations. *SIAM J. Matrix Anal. Appl.* **15** (1994), 592–622. 227

[34] P. G. Ciarlet, *The finite element method for elliptic problems*. Classics Appl. Math. 40, Society for Industrial and Applied Mathematics (SIAM), Philadelphia 2002. 365, 390

[35] P. Clément, Approximation by finite element functions using local regularization. *RAIRO Anal. Numér.* **9** (1975), 77–84. 376

[36] A. Cohen, W. Dahmen, and R. DeVore, Adaptive wavelet methods for elliptic operator equations: convergence rates. *Math. Comp.* **70** (2001), 27–75. 401

[37] J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series. *Math. Comp.* **19** (1965), 297–301. 1

[38] W. Dahmen, B. Faermann, I. G. Graham, W. Hackbusch, and S. A. Sauter, Inverse inequalities on non-quasi-uniform meshes and application to the mortar element method. *Math. Comp.* **73** (2004), 1107–1138. 376, 379, 390, 391, 398, 399

[39] W. Dahmen, H. Harbrecht, and R. Schneider, Compression techniques for boundary integral equations—asymptotically optimal complexity estimates. *SIAM J. Numer. Anal.* **43** (2006), 2251–2271. 2

[40] W. Dahmen, S. Prössdorf, and R. Schneider, Multiscale methods for pseudo-differential equations on smooth closed manifolds. In *Wavelets: theory, algorithms, and applications* (Taormina, 1993), Wavelet Anal. Appl. 5, Academic Press, San Diego 1994, 385–424. 401

[41] W. Dahmen and R. Schneider, Wavelets on manifolds I: Construction and domain decomposition. *SIAM J. Math. Anal.* **31** (1999), 184–230. 2

[42] R. A. DeVore and G. G. Lorentz, *Constructive approximation*. Grundlehren Math. Wiss. 303, Springer-Verlag, Berlin 1993. 95, 106

[43] S. Erichsen and S. A. Sauter, Efficient automatic quadrature in 3-d Galerkin BEM. *Comput. Methods Appl. Mech. Engrg.* **157** (1998), 215–224. 156, 392

[44] P. P. Ewald, Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Physik 369* (1921), 253–287. 2

[45] K. Giebermann, Multilevel approximation of boundary integral operators. *Computing* **67** (2001), 183–207. 1, 3, 74

[46] Z. Gimbutas and V. Rokhlin, A generalized fast multipole method for nonoscillatory kernels. *SIAM J. Sci. Comput.* **24** (2002), 796–817. 2

[47] G. Golub, Numerical methods for solving linear least squares problems. *Numer. Math.* **7** (1965), 206–216. 227

[48] G. H. Golub and C. F. Van Loan, *Matrix computations*. 3rd. ed., The Johns Hopkins University Press, Baltimore 1996. 180, 181, 186, 187, 188, 233

[49] L. Grasedyck, Theorie und Anwendungen Hierarchischer Matrizen. PhD thesis, Universität Kiel, Kiel 2001. 2, 50, 122, 262, 361, 376
http://eldiss.uni-kiel.de/macau/receive/dissertation_diss_00000454

[50] L. Grasedyck, Adaptive recompression of $\mathcal{H}$-matrices for BEM. *Computing* **74** (2005), 205–223. 248, 393

[51] L. Grasedyck, Existence of a low-rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation. *Numer. Linear Algebra Appl.* **11** (2004), 371–389. 2

[52] L. Grasedyck and W. Hackbusch, Construction and arithmetics of $\mathcal{H}$-matrices. *Computing* **70** (2003), 295–334. 2, 4, 6, 29, 37, 38, 50, 347, 361, 366, 414

[53] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij, Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing* **70** (2003), 121–165. 2

[54] L. Grasedyck, W. Hackbusch, and R. Kriemann, Performance of H-LU preconditioning for sparse matrices. *Comput. Methods Appl. Math.* **8** (2008), 336–349. 2

[55] L. Grasedyck, R. Kriemann, and S. Le Borne, Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems. *Comput. Visual Sci.* **11** (2008), 273–291. 2

[56] L. Grasedyck, R. Kriemann, and S. Le Borne, Domain decomposition based $\mathcal{H}$-LU preconditioning. *Numer. Math.* **112** (2009), 565–600. 2, 414

[57] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, Fast direct solvers for integral equations in complex three-dimensional domains. *Acta Numer.* **18** (2009), 243–275. 2

[58] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations. *J. Comput. Phys.* **73** (1987), 325–348. 1, 3, 5, 74

[59] L. Greengard and V. Rokhlin, On the numerical solution of two-point boundary value problems. *Comm. Pure Appl. Math.* **44** (1991), 419–452. 3, 36, 59

[60] L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace in three dimensions. *Acta Numer.* **6** (1997), 229–269. 1, 3, 5, 74

[61] W. Hackbusch, *Multigrid methods and applications*. Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin 1985. 366

[62] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing* **62** (1999), 89–108. v, 2, 6, 29, 36, 49, 366

[63] W. Hackbusch, *Hierarchische Matrizen*. Springer-Verlag, Dordrecht 2009. 2, 6, 29, 49, 84, 104, 328

[64] W. Hackbusch and S. Börm, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing* **69** (2002), 1–35. v, 2, 211, 212

[65] W. Hackbusch and S. Börm, $\mathcal{H}^2$-matrix approximation of integral operators by interpolation. *Appl. Numer. Math.* **43** (2002), 129–143. 3, 74, 75

[66] W. Hackbusch and B. N. Khoromskij, $\mathcal{H}$-matrix approximation on graded meshes. In *The mathematics of finite elements and applications X* (MAFELAP 1999), Elsevier, Kidlington 2000, 307–316. 274

[67] W. Hackbusch and B. N. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic: general complexity estimates. *J. Comp. Appl. Math.* **125** (2000), 479–501. 2, 29

[68] W. Hackbusch and B. N. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing* **64** (2000), 21–47. 2, 29

[69] W. Hackbusch, B. N. Khoromskij, and R. Kriemann, Hierarchical matrices based on a weak admissibility criterion. *Computing* **73** (2004), 207–243. 36, 59

[70] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter, On $\mathcal{H}^2$-matrices. In *Lectures on applied mathematics*, Springer-Verlag, Berlin 2000, 9–29. v, 2, 29, 125

[71] W. Hackbusch and Z. P. Nowak, *O cloznosti metoda panelej* (On complexity of the panel method, Russian). In *Vychislitelnye protsessy i sistemy* 6, Nauka, Moscow 1988, 233–244. 1

[72] W. Hackbusch and Z. P. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.* **54** (1989), 463–491. 1, 3, 5, 9, 74

[73] H. Harbrecht and R. Schneider, Wavelet Galerkin schemes for boundary integral equations—implementation and quadrature. *SIAM J. Sci. Comput.* **27** (2006), 1347–1370. 2

[74] S. Jaffard, Wavelet methods for fast resolution of elliptic problems. *SIAM J. Numer. Anal.* **29** (1992), 965–986. 401

[75] S. Lang, *Real and functional analysis*. 3rd ed., Graduate Texts in Math. 142, Springer-Verlag, New York 1993. 10, 12

[76] S. Le Borne, $\mathcal{H}$-matrices for convection-diffusion problems with constant convection. *Computing* **70** (2003), 261–274. 2

[77] S. Le Borne, Hierarchical matrices for convection-dominated problems. In *Domain decomposition methods in science and engineering* (Berlin, 2003), Lect. Notes Comput. Sci. Eng. 40, Springer-Verlag, Berlin 2005, 631–638. 2

[78] S. Le Borne and L. Grasedyck, $\mathcal{H}$-matrix preconditioners in convection-dominated problems. *SIAM J. Matrix Anal. Appl.* **27** (2006), 1172–1183. 2

[79] S. Le Borne, L. Grasedyck, and R. Kriemann, Domain-decomposition based $\mathcal{H}$-LU preconditioners. In *Domain decomposition methods in science and engineering XVI*, Lect. Notes Comput. Sci. Eng. 55, Springer-Verlag, Berlin 2007, 667–674. 414

[80] M. Löhndorf, Effiziente Behandlung von Integraloperatoren mit $\mathcal{H}^2$-Matrizen variabler Ordnung. PhD thesis, Universität Leipzig, Leipzig 2003. http://www.mis.mpg.de/scicomp/Fulltext/Dissertation_Loehndorf.pdf 65, 161

[81] M. Löhndorf and J. M. Melenk, Approximation of integral operators by variable-order approximation. Part II: Non-smooth domains. In preparation. 65, 274

[82] J. Makino, Yet another fast multipole method without multipoles—pseudoparticle multipole method. *J. Comput. Phys.* **151** (1999), 910–920. 2

[83] P.-G. Martinsson, A fast direct solver for a class of elliptic partial differential equations. *J. Sci. Comput.* **38** (2009), 316–330. 59

[84] E. Michielssen, A. Boag, and W. C. Chew, Scattering from elongated objects: direct solution in $O(N \log^2 N)$ operations. *IEE Proc.-Microw. Antennas Propag.* **143** (1996),277–283. 36

[85] G. Of, O. Steinbach, and W. L. Wendland, The fast multipole method for the symmetric boundary integral formulation. *IMA J. Numer. Anal.* **26** (2006), 272–296. 2

[86] J. Ostrowski, Boundary element methods for inductive hardening. PhD thesis, Universität Tübingen, Tübingen 2003. http://tobias-lib.uni-tuebingen.de/dbt/volltexte/2003/672/ 2

[87] T. J. Rivlin, *The Chebyshev polynomials*. Wiley-Interscience, New York 1974. 94

[88] V. Rokhlin, Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.* **60** (1985), 187–207. 2, 74

[89] S. A. Sauter, Cubature techniques for 3-D Galerkin BEM. In *Boundary elements: implementation and analysis of advanced algorithms* (Kiel, 1996), Notes Numer. Fluid Mech. 54, Vieweg, Braunschweig 1996, 29–44. 156, 392

[90] S. A. Sauter, Variable order panel clustering (extended version). Preprint 52/1999, Max Planck Institute for Mathematics in the Sciences, Leipzig 1999. 37, 125, 126, 278
http://www.mis.mpg.de/publications/preprints/1999/prepr1999-52.html

[91] S. A. Sauter, Variable order panel clustering. *Computing* **64** (2000), 223–261. 1, 2, 5, 37, 75, 125, 126

[92] S. A. Sauter and C. Schwab, *Randelementmethoden*. B. G. Teubner, Wiesbaden 2004. 37, 388, 389, 390, 401

[93] O. Schenk, K. Gärtner, W. Fichtner, and A. Stricker, PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generat. Comput. Syst.* **18** (2001), 69–78. 366

[94] R. Schneider, *Multiskalen- und Wavelet-Matrixkompression*. B. G. Teubner, Stuttgart 1998. 2

[95] L. R. Scott and S. Zhang, Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Math. Comp.* **54** (1990), 483–493. 377

[96] P. Starr and V. Rokhlin, On the numerical solution of two-point boundary value problems II. *Comm. Pure Appl. Math.* **47** (1994), 1117–1159. 3, 36, 59

[97] O. Steinbach, Numerische Näherungsverfahren für elliptische Randwertprobleme. B. G. Teubner, Wiesbaden 2003. 392, 396, 398

[98] O. Steinbach and W. L. Wendland, The construction of some efficient preconditioners in the boundary element method. *Adv. Comput. Math.* **9** (1998), 191–216. 401

[99] E. Stiefel, Über einige Methoden der Relaxationsrechnung. *Z. Angew. Math. Physik* **3** (1952), 1–33. 393

[100] J. Tausch, The variable order fast multipole method for boundary integral equations of the second kind. *Computing* **72** (2004), 267–291. 3, 75, 125, 278

[101] J. Tausch, A variable order wavelet method for the sparse representation of layer potentials in the non-standard form. *J. Numer. Math.* **12** (2004), 233–254. 125

[102] J. Tausch and J. White, Multiscale bases for the sparse representation of boundary integral operators on complex geometry. *SIAM J. Sci. Comput.* **24** (2003), 1610–1629. 2, 191

[103] E. Tyrtyshnikov, Mosaic-skeleton approximations. *Calcolo* **33** (1996), 47–57. 2

[104] E. E. Tyrtyshnikov, Incomplete cross approximation in the mosaic-skeleton method. *Computing* **64** (2000), 367–380. 227, 392

[105] T. von Petersdorff and C. Schwab, Fully discrete multiscale Galerkin BEM. In *Multiscale wavelet methods for partial differential equations*, Wavelet Anal. Appl. 6, Academic Press, San Diego 1997, 287–346. 2

[106] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, Fast algorithms for hierarchically semiseparable matrices. Numer. Linear Algebra Appl., to appear. Article first published online: 22 DEC 2009, DOI: 10.1002/nla.691 59

[107] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, Superfast multifrontal method for large structured linear systems of equations, *SIAM J. Matrix Anal. Appl.* **31** (2009), 1382–1411. 59

[108] L. Ying, G. Biros, and D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.* **196** (2004), 591–626. 2

Algorithm index

Adaptive bases for $\mathcal{H}$-matrices, 238
Adaptive bases for $\mathcal{H}^2$-matrices, 246
Adaptive bases for dense matrices, 230
Adaptive bases for dense matrices, theoretical, 229
Adaptive bases for weighted dense matrices, 263

Backward transformation, 61
Block backward transformation, 339
Block cluster tree, 43
Block forward transformation, 169
Blockwise conversion of an $\mathcal{H}$-matrix, 257
Bounding boxes, 48

Check for orthogonality, 166
Cluster basis product, 176
Coarsening, 354
Construction of a nested cluster basis from a general one, 224
Construction of weights for the total cluster basis, 243
Conversion, $\mathcal{H}$ to $\mathcal{H}^2$, 173
Conversion, $\mathcal{H}$ to $\mathcal{H}^2$, with error norm, 202
Conversion, $\mathcal{H}^2$ to $\mathcal{H}^2$, 178
Conversion, $\mathcal{H}^2$ to $\mathcal{H}^2$, with error norm, 203
Conversion, dense to $\mathcal{H}^2$, 170
Conversion, dense to $\mathcal{H}^2$, with error norm, 201
Covering balls, 46

Expansion of a cluster basis, 338

Find optimal rank, 187
Forward substitution, 407
Forward transformation, 61

Geometric cluster trees, 42

Householder factorization, 181

Low-rank approximation of leaf blocks, 351
Low-rank approximation of matrix blocks, 349
Low-rank approximation of subdivided blocks, 353
Low-rank projection, 188
Lower forward substitution, 404
LU decomposition, 403

Matrix addition, 296
Matrix backward transformation, 291
Matrix backward transformation, elementary step, 290
Matrix forward transformation, 286
Matrix forward transformation, elementary step, 285
Matrix inversion, 416
Matrix multiplication, 314
Matrix multiplication, b_A admissible, 312
Matrix multiplication, b_B admissible, 312
Matrix multiplication, b_C admissible, 313
Matrix multiplication, inadmissible, 314
Matrix multiplication, recursion, 311
Matrix-vector multiplication, 62

Orthogonalization of cluster bases, 184

Recursive construction of cluster bases for $\mathcal{H}$-matrices, 237

Recursive construction of cluster bases for $\mathcal{H}^2$-matrices, 246
Recursive construction of unified cluster bases, 252
Regular cluster trees, 40

Semi-uniform matrix forward transformation, 338, 340
Semi-uniform matrix product, multiplication step, 342

Truncation of cluster bases, 190

Unification of submatrices, 255
Upper forward substitution, 405

Subject index

Admissibility condition, 36
Admissibility condition, weak, 36
Anisotropic coefficients, 419
Approximation of derivatives, 104
Asymptotically smooth function, 83

Backward transformation, blocks, 339
Backward transformation, matrices, 287
Block backward transformation, 339
Block cluster tree, 34
Block cluster tree, admissible, 37
Block cluster tree, consistent, 144
Block cluster tree, induced, 298, 325
Block cluster tree, sparse, 51
Block cluster tree, transposed, 215
Block columns, 50
Block columns, admissible, 59
Block forward transformation, 169
Block restriction, 167
Block rows, 50
Block rows, admissible, 59
Block rows, extended, 170
Boundary integral formulation, direct, 395
Boundary integral formulation, indirect, 388
Bounding boxes, 45, 90

Cacciopoli inequality, 367
Call tree for multiplication, 315
Characteristic point, 38
Chebyshev interpolation, 93
Clément interpolation operator, 377
Cluster basis, 54
Cluster basis product, 176
Cluster basis, conversion into nested basis, 223
Cluster basis, induced, 299, 321, 325
Cluster basis, nested, 54
Cluster basis, orthogonal, 164
Cluster basis, orthogonalization, 180
Cluster basis, truncation, 185
Cluster basis, unified, 248
Cluster operator, 175
Cluster tree, 30
Cluster tree, bounded, 63
Cluster tree, depth, 34
Cluster tree, quasi-balanced, 129
Cluster tree, regular, 69
Cluster, descendants, 30
Cluster, father, 30
Cluster, level, 31
Cluster, predecessors, 30
Coarsening the block structure, 354
Complexity of block forward transformation, 169
Complexity of cluster basis expansion, 342
Complexity of coarsening, 357
Complexity of collecting submatrices, 284
Complexity of compression of dense matrices, 229
Complexity of compression of $\mathcal{H}$-matrices, 236
Complexity of compression of $\mathcal{H}^2$-matrices, 245
Complexity of conversion of dense matrices, 171
Complexity of conversion of $\mathcal{H}$-matrices, 173
Complexity of conversion of $\mathcal{H}^2$-matrices, 178
Complexity of finding cluster weights, 243

Complexity of hierarchical compression, 258
Complexity of matrix backward transformation, 292
Complexity of matrix forward transformation, 286
Complexity of orthogonalization, 183
Complexity of projected matrix addition, 295
Complexity of projected multiplication, 317
Complexity of semi-uniform forward transformation, 343
Complexity of semi-uniform matrix multiplication, 346
Complexity of splitting into submatrices, 289
Complexity of the cluster basis product, 177
Complexity of truncation, 190
Complexity of unification of cluster bases, 250
Complexity of unification of submatrices, 255
Condensation of $\mathcal{H}$-matrix blocks, 234
Condensation of total cluster bases, 240, 249
Conjugate gradient method, 401
Control of blockwise relative error, 262
Control of spectral error by variable rank compression, 264
Covering a regularity ellipse with circles, 152

Density function, 388
Direct boundary integral formulation, 395
Directional interpolation, 98
Double layer potential, 155

Elliptic partial differential equation, 363, 413
Error decomposition for truncation, 194
Error estimate for asymptotically smooth functions, 102
Error estimate for coarsening, 354
Error estimate for compression, 232
Error estimate for derived asymptotically smooth functions, 113
Error estimate for discrete solution operators, 385
Error estimate for isotropic interpolation, 101
Error estimate for multi-dimensional derived interpolation, 111
Error estimate for multi-dimensional interpolation, 100
Error estimate for multi-dimensional re-interpolation, 141
Error estimate for one-dimensional derived interpolation, 108
Error estimate for one-dimensional interpolation, 97
Error estimate for one-dimensional re-interpolation, 136
Error estimate for re-interpolation of analytic functions, 137
Error estimate for re-interpolation of asymptotically smooth functions, 142
Error estimate for semi-uniform projection, 224
Error estimate for solution operators, 372
Error estimate for submatrices of solution operators, 382
Error estimate for the double layer potential, 399
Error estimate for the single layer potential, 391
Error estimate for the Taylor expansion, 85
Error estimate for truncation, 195
Error estimate for variable-order interpolation, 147
Error orthogonality, 192
Expansion matrices, 54

Expansion system, 77

Farfield, 37
Finite element method, 76, 365, 389, 414
Forward substitution, 406
Forward substitution, blocks, 402
Forward transformation, blocks, 169
Forward transformation, matrices, 285
Forward transformation, semi-uniform matrices, 338, 340
Frobenius error, blockwise, 116
Frobenius error, integral operator, 120
Frobenius error, projection, 200
Frobenius error, total, 117
Frobenius inner product, 166
Frobenius norm, 115, 166, 200

Galerkin's method, 76, 365, 389, 414

$\mathcal{H}$-matrix, 49
$\mathcal{H}^2$-matrix, 56
Hierarchical compression, 257
Hierarchical matrix, 49
Holomorphic extension, 151
Householder factorization, 180
HSS-matrix, 59

Indirect boundary integral formulation, 388
Induced block cluster tree, 298, 325
Induced cluster basis, 299, 321, 325
Integral operator, double layer potential, 155
Integral operator, Frobenius error, 120
Integral operator, single layer potential, 155, 388
Integral operator, spectral error, 124
Integral operator, truncation, 198
Integral operator, variable-rank compression, 265
Interior regularity, 366, 367
Interpolation, 88
Interpolation scheme, 93
Interpolation, best approximation property, 94

Jumping coefficient, 418

Lebesgue constant, 93
Locally L-harmonic, 367
LU decomposition, 402

Markov's inequality, 106
Matrix addition, exact, 300
Matrix addition, projected, 293
Matrix backward transformation, 287
Matrix forward transformation, 285
Matrix inversion, 415
Matrix multiplication, exact, 327
Matrix multiplication, projected, 310
Matrix multiplication, semi-uniform, 340

Nearfield, 37

Operator norm, 120
Orthogonality criterion, 165
Orthogonalization of cluster bases, 180
Overlapping supports, 115

Partial differential equation, 364
Partial interpolation, 98
Polynomial approximation of analytic functions, 95
Preconditioner, 401
Projection into $\mathcal{H}^2$-matrix spaces, 167
Projection into semi-uniform matrix spaces, 214
Projection of $\mathcal{H}$-matrices, 172
Projection of dense matrices, 168

Quasi-optimality of compression, 233

Rank distribution, 54
Rank distribution, bounded, 64
Rank distribution, maximum, 176
Re-interpolated kernel, 132

Re-interpolation, 131
Re-interpolation cluster basis, 131
Restriction of cluster basis, 220

Schur complement, 415
Semi-uniform hierarchical matrix, 212, 334
Semi-uniform matrix forward transformation, 340
Semi-uniform matrix forward transformation, 338
Separable approximation by Taylor expansion, 81
Separable approximation by tensor interpolation, 90
Sequence of ξ-regular bounding boxes, 141
Sequence of ξ-regular boxes, 140
Sequence of ξ-regular intervals, 135
Single layer potential, 155, 388
Singular value decomposition, 185, 348
Spaces of semi-uniform hierarchical matrices, 213
Spectral error, blockwise, 120
Spectral error, factorized estimate, 124
Spectral error, integral operator, 124
Spectral error, projection, 216
Spectral norm, 120
Spectral norm, total, 121, 122
Stability of re-interpolation, 135
Stable interpolation scheme, 93
Strang's lemma, 390
Support, 38
Symm's equation, 388

Tausch/White wavelets, 191
Taylor expansion, 80
Tensor interpolation, 90
Total cluster basis, 218
Total cluster basis, geometric interpretation, 222
Total cluster basis, properties, 218
Total cluster basis, weighted, 261
Transfer matrices, 54
Transfer matrices, long-range, 193
Tree, 29, 30
Truncation of cluster bases, 185

Unification, 248

Variable-order approximation, 125
Variable-order interpolation of the kernel function, 142
Variable-rank compression, integral operator, 265