

ulm university universität
uulm

Fakultät für Mathematik und Wirtschaftswissenschaften
Institut für Numerische Mathematik

Dissertation

On hp-Boundary Element Methods for the Laplace Operator in Two Dimensions

Implementation, hp-Adaptive Algorithms, and Data Compression

zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Mathematik und Wirtschaftswissenschaften
der Universität Ulm

vorgelegt von
Markus Bantle
aus Ulm

April 2013

Amtierender Dekan:	Prof. Dr. Dieter Rautenbach (Universität Ulm)
1. Gutachter:	Prof. Dr. Stefan A. Funken (Universität Ulm)
2. Gutachter:	Prof. Dr. Dirk Praetorius (Technische Universität Wien)
Tag der Promotion:	11. Juli 2013

Abstract

This work expands on three aspects of the hp-boundary element method (BEM) for the Laplace operator in two dimensions.

First, we provide a new stable and efficient implementation of hp-BEM. Using Legendre functions and their antiderivatives as local bases for the discrete ansatz spaces, we are able to reduce both the evaluation of potentials and the computation of Galerkin entries to the evaluation of basic integrals closely related to associated Legendre functions of the second kind. For the computation of these integrals we derive recurrence relations and discuss their accurate evaluation in detail. Our implementation of p- and hp-BEM is the first that produces accurate results even for large polynomial degrees ($p > 1000$) while still being efficient.

The next aspect considered is the construction of a posteriori error estimators for Symm's integral equation. We generalize the idea of $h - h/2$ error estimators to obtain error estimators for hp-BEM. By a combination of theoretical considerations and numerical experiments, we show that weighted L^2 norms and projections have to be considered in order to obtain efficient and reliable estimators. Based on these estimators, we construct two hp-adaptive algorithms and prove their convergence. Numerical experiments show that both algorithms converge exponentially.

Finally, we develop a new data compression scheme for the single layer Galerkin matrix arising in p- and hp-BEM. This novel approach uses different compression techniques for different sub-blocks of the Galerkin matrix, which results in an $\mathcal{H}$-Matrix approximation. Complexity estimates show that the amount of storage needed by the Galerkin matrix and the effort for the matrix vector multiplication are of the order $\mathcal{O}(\mathcal{N}\log(\mathcal{N})^\alpha)$ with $\alpha \leq 3$.

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit drei Aspekten der hp-Randelementmethode (REM) für den Laplace Operator in zwei Dimensionen.
Im ersten Teil diskutieren wir eine neue, stabile und effiziente Implementierung der hp-REM. Wir verwenden Legendre-Polynome und deren Stammfunktionen als Elementbasisfunktionen für die diskreten Ansatzräume. Diese Methode ermöglicht es, die Auswertung von Potentialen und Berechnung von Galerkin-Matrizen auf das Auswerten einiger einfacher Integrale, die eng mit Legendre-Funktionen zweiter Art verwandt sind, zu reduzieren. Für die Berechnung der Integrale leiten wir Rekursionsformeln her und beschreiben im Detail, wie diese Rekursionen mit hoher Genauigkeit ausgewertet werden können. Unsere Implementierung der hp-REM ist die erste, die auch für sehr hohe Polynomgrade ($p > 1000$) akkurate Ergebnisse liefert und trotzdem noch effizient ist.
Der zweiten Teil beschäftigt sich mit der Konstruktion von a posteriori Fehlerschätzern für die Symm'sche Integralgleichung. Wir verallgemeinern die Idee von $h - h/2$ Fehlerschätzern um Fehlerschätzer für die hp-REM zu erhalten. Durch theoretische Betrachtungen und numerische Experimente wird gezeigt, dass gewichtete L^2-Normen und -Projektionen verwendet werden müssen, um effiziente und zuverlässige Schätzer zu erhalten. Diese Schätzer werden dann für die Konstruktion von zwei hp-adaptiven Algorithmen verwendet, deren Konvergenz wir beweisen. Numerische Experimente zeigen schließlich, dass beide Algorithmen sogar exponentiell konvergieren.
Im letzten Teil der Arbeit beschäftigen wir uns mit der Konstruktion einer neuen Datenkompressionsmethode für die Galerkin-Matrix des Einfachschichtpotentials in der p- und hp-REM. Für verschiedene Blöcke der Galerkin-Matrix werden dabei verschiedene Kompressionsansätze verwendet was zu einer $\mathcal{H}$-Matrix approximation führt. Wir leiten Abschätzungen her, die zeigen, dass sowohl der Speicheraufwand als auch der Aufwand für die Matrix-Vektor Multiplikation von der Ordnung $\log(\mathcal{N}\log(\mathcal{N})^\alpha)$ für $\alpha \leq 3$ sind.

Acknowledgements

Though my name is listed as the sole author, there were many people whose efforts and consideration were essential to make this work possible.

First and foremost, I owe my deepest gratitude to my advisor, teacher, and - above all - mentor Prof. Dr. Stefan Funken. His mentorship has deeply influenced who I am as both a researcher and a person. I thank him for his friendliness and patience, his tireless support in every possible way, and his steadfast collaboration whenever I had questions or needed to discuss problems.

I would also like to thank Prof. Dr. Dirk Praetorius for the collaboration and inspiring discussions during the last three years. I am honored to have him as the second reviewer for this dissertation.

Thank you to all my colleagues at the Institute for Numerical Analysis and the Scientific Computing Centre Ulm. You have made it a pleasure to be a member of this community. In particular, I would like to thank Christoph Erath, Sebastian Kestler, and Kristina Steih for always being available for spontaneous questions and discussions.

To my brother Andreas Bantle and my girlfriend Natalie Selinski, I would like to offer my gratitude for supporting me both personally and professionally, in proofreading this work and for listening whenever problems arose.

Last but not least, thank you to my entire family, Gisela, Gerhard, Andreas, and Daniela Bantle and Natalie Selinski for always supporting and encouraging me. Without your love and support, the following pages would not exist.

Contents

Introduction iii

1. Analytical Basics and Notation 1

1.1. Function Spaces . . . 1
1.2. Boundary Integral Equations . . . 5
1.2.1. The Dirichlet Problem and Symm's Integral Equation . . . 9
1.2.2. The Neumann Problem and the Hypersingular Integral Equation . . . 10
1.2.3. The Mixed Problem . . . 11
1.3. hp-Boundary Element Methods . . . 12
1.4. Boundary Meshes and Refinement . . . 15
1.5. A Priori Error Analysis . . . 16

2. Efficient and Stable Evaluation of Legendre Functions and Their Integrals 21

2.1. Legendre Polynomials and Lobatto Shape Functions . . . 22
2.2. Associated Legendre Functions . . . 25
2.3. Modified Associated Legendre Functions of the Second Kind . . . 26
2.3.1. Definition and Recurrence Relations . . . 26
2.3.2. Asymptotic Estimates for $\widetilde{Q}_k^m$ . . . 33
2.3.3. Minimal Solutions . . . 38
2.3.4. Implementation . . . 42
2.4. Integrals of Modified Associated Legendre Functions of the Second Kind . . . 52
2.4.1. Definition and Recurrence Relations . . . 53
2.4.2. Implementation . . . 56

3. Implementation of hp-Boundary Element Methods 69

3.1. Overview of Existing Software Packages . . . 70
3.2. Discrete Spaces for hp-BEM . . . 71
3.3. Evaluation of the Potentials . . . 75
3.4. Computation of Galerkin Entries . . . 78
3.4.1. Theory . . . 78
3.4.2. Implementation . . . 83

3.5. The epsBEM Software Package 86
3.6. Numerical Experiments 88

4. Error Estimators and hp-Adaptive Algorithms 95
4.1. $h - h/2$ Error Estimators for Lowest Order BEM 96
4.2. Error Estimators for hp-BEM 98
4.3. hp-Adaptive Algorithms 118
4.4. Implementation and Numerical Experiments 124

5. Data Compression Techniques for the Single Layer Galerkin Matrix in p-BEM 135
5.1. Existing Compression Schemes 137
5.2. Perturbations in the Galerkin Matrix 146
5.3. Diagonal Blocks 148
5.4. Far Field Blocks 153
5.5. Off-Diagonal Blocks 159
5.6. Extension of the Compression Scheme 161
5.7. Implementation and Numerical Experiments 166
5.7.1. Aspects of Implementation 166
5.7.2. Uniform p-Method 169
5.7.3. Uniform h- and hp-Method 173

Summary and Outlook 177

A. Formulas for the Calculation of Initial Values for $\widetilde{Q}_k^m$ and R_k^m 181

B. Evaluation of Potentials 187

C. Implementation of the Compression Scheme from Chapter 5 193

Bibliography 197

List of Symbols 205

Curriculum Vitae 209

Erklärung 211

Introduction

The boundary element method (BEM) is a well-established numerical method for the solution of certain types of partial differential equations (PDEs) arising in a variety of fields including potential theory, linear elasticity, acoustic scattering and electromagnetism. Because it offers the possibility to treat unbounded domains in a natural way and requires the discretization of only the domain's boundary, it is an attractive alternative to other numerical methods for partial differential equations such as the finite element, finite volume, or the finite difference methods. The idea of the boundary element method is to rewrite the PDE into an equivalent boundary integral equation and then solve this boundary integral equation numerically using a Galerkin method. For this, the variational formulation of the equation is considered in a discrete space, e.g. a space of piecewise polynomials with respect to a given mesh. This results in a large system of linear equations that is assembled and solved numerically.

There exist different versions of BEM. In h-BEM, the unknown solution is approximated by piecewise polynomials of low order, typically orders $p = 0$ or $p = 1$, on fine boundary meshes. If the solution is approximated with high order polynomials, we call this p-BEM. A combination of h- and p-BEM is called hp-BEM. It has been shown that the h- and p-versions of BEM yield algebraic convergence with respect to the number of degrees of freedom $\mathcal{N}$ [StSu91, GuHe06]. Here, the p-version has twice the convergence rate of the h-version. Furthermore, certain types of hp-versions, where the meshes and polynomial degrees are adapted to the smoothness of the solution, result in exponential convergence, i.e. the errors are proportional to $\exp(-b\mathcal{N}^{\alpha})$ where b and α are constants independent of $\mathcal{N}$ [HS96].

While most aspects of the h-version of BEM are understood well from both a theoretical and practical point of view, there are still several open questions in p- and hp-BEM. In this work we consider some of these challenges for the Laplace problem in two dimensions. Specifically, we discuss the following aspects:

- Efficient and stable implementation of p- and hp-BEM, especially for high orders p,
- Error estimators and hp-adaptive algorithms, and
- Data compression techniques for p- and hp-BEM.

Before we elaborate on each of these items, we present the example of Symm's integral equation, which was introduced by Jaswon and Symm [Jas63, Sym63] to solve PDEs arising in potential

theory and became the starting point of modern boundary element methods. Even though this is a basic example, it shows some of the difficulties that arise in BEM.

Symm's Integral Equation

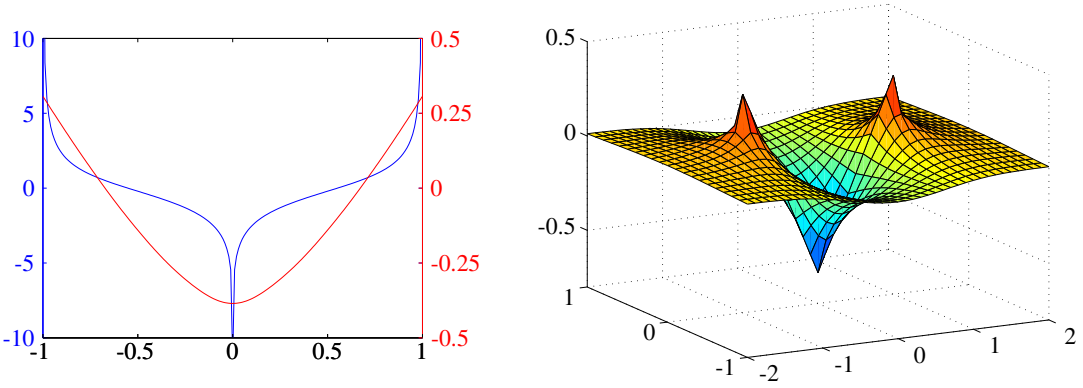
We consider Symm's integral equation

$$(\mathcal{V}\phi)(x) := -\frac{1}{2\pi}\int_{-1}^{1}\phi(y)\log|x-y|dy = f(x) \quad x \in [-1,1]. \tag{1}$$

If we choose $f(x) = |x| - 2/\pi$, the exact solution can be computed analytically and is given by

$$\phi(x) = \frac{\log|2x| - g(x)}{\sqrt{1-x^2}}$$

with a smooth function g. Figure 1 shows this function g and the solution ϕ as well as a surface plot of the potential $\mathcal{V}\phi$ on $[-2,2]\times[-1,1]$.



(a) Plot of the solution ϕ (blue) and the function g (red).

(b) Surface plot of the potential $\mathcal{V}\phi$.

Figure 1: Left picture: Solution ϕ of Symm's integral equation $(\mathcal{V}\phi)(x) = |x| - 2/\pi$ (blue) and the function g (red) on $[-1,1]$. Right picture: Surface plot of the potential $\mathcal{V}\phi$ on $[-2,2]\times[-1,1]$.

First, it can be seen that Symm's integral equation (1) involves a singular integral that is challenging to accurately evaluate using numerical methods. In general, the implementation of BEM is not straight forward.

Second, Figure 1 (a) shows that the solution of an integral equation may have singularities even though the right-hand side is continuous. In order to accurately compute solutions with singularities, the discretization has to be adjusted to fit the behavior of the solution. Specifically, the discretization has to be finer where the solution has singularities or discontinuities. Since the solution is generally unknown, a posteriori error estimators and adaptive algorithms, which automatically refine the discretization around singularities, are needed.

Third, in Figure 1 (b) we see that boundary integral operators are nonlocal operators. Even

though the function ϕ is only defined on $[-1, 1]$, the support of the potential $\mathcal{V}\phi$ is not limited to this interval. A consequence for the numerical treatment of boundary integral equations is that the linear systems obtained in a Galerkin method have densely populated matrices. In order to be able to handle fine discretizations with large numbers of degrees of freedom, data compression techniques have to be developed.
In the following sections we discuss each of these aspects in more detail and elaborate on the contributions that are made by this work.

Implementation of BEM

For the computation of Galerkin entries and the evaluation of potentials, singular integrals have to be evaluated. Therefore, the implementation of the boundary element method is not straight forward.

Common approaches for the evaluation of the singular integrals arising in h-BEM are adapted quadrature formulas [SW92, SaSch04] as well as semi-analytic and fully-analytic approaches [SaSch04]. Using these approaches, h-BEM has been implemented in a stable and efficient way for different problems, mainly Laplace, Lamé, and Helmholtz problems in two and three dimensions. Implementations are collected in various software packages, such as BEM++ [ŚAB$^+$12], HILBERT [AEF$^+$11], and HyENA [MMRU10], to mention a few but not all.
For the evaluation of singular integrals arising in p- and hp-BEM, adapted quadrature rules have been proposed; see [SaSch97] and the references therein. Furthermore, analytical formulas for integrals arising when monomials are used as basis functions for the ansatz space are derived in e.g. [Mai96]. However, both the use of quadrature rules and analytical formulas for monomials have limitations with respect to the polynomial degrees that can be used. Since polynomials of high orders have to be integrated, quadrature becomes inaccurate. If monomials are used as basis functions for the discrete spaces, the condition numbers of Galerkin matrices arising in p-BEM grow exponentially with respect to p. Overall, this means that with double precision computations the polynomial degrees are limited to $p < 20$. Using a multiple precision library, Maischak was able to perform numerical experiments with polynomial degrees up to $p = 50$ [Mai12].

Thus far, there exists no implementation of p- and hp-BEM that produces accurate results for large polynomial degrees. This is the first problem we address in this work. We discuss the evaluation of potentials and assembly of Galerkin matrices for high polynomial degrees. As a basis for the discrete ansatz spaces, we use Legendre polynomials and their antiderivatives. This approach allows us to reduce the computation of potentials and Galerkin entries to the evaluation of certain basic integrals that are closely related to Legendre functions of the second kind. Exploiting properties of Legendre functions, we are able to derive recurrence relations for these

basic integrals. In order to evaluate the integrals related to the potentials, we adapt Gautschi's continued fraction algorithm [Gau67]. This algorithm results in a stable and efficient method for the evaluation of potentials. For the integrals needed in computing the Galerkin entries, we evaluate the recurrence relations in the forward direction. But this evaluation is generally unstable. Therefore, in order to guarantee the accurate computation of Galerkin entries, we use the multiple precision libraries `MPFR` [FHL$^+$07] and `MPC` [EGTZ12] and derive a heuristic upper bound for the number of digits needed for the computation using numerical experiments. The evaluation of potentials and the computation of Galerkin entries are parallelized with `openMP`.
Overall, we obtain an efficient and accurate implementation of p- and hp-BEM. All implementations are part of the software package epsBEM [BBF13] that was co-developed by the author and implements two dimensional hp-BEM for Laplace and Lamé problems.
As an example, we consider Symm's integral equation $(\mathcal{V}\phi)(x) = |x| - 2/\pi$ on $[-1, 1]$. Figure 2 shows the Galerkin errors over the number of degrees of freedom $\mathcal{N}$ of the discrete ansatz space for uniform h- and p-BEM as well as an adaptive hp-BEM. For the uniform h-version, we start with four elements and use $p = 7$ on all elements. The uniform p-version is performed on a mesh with 4 elements, and we start with $p = 7$. We demonstrate that our implementation produces accurate results even for high polynomial degrees (for the uniform p-method, the largest polynomial degree used is $p = 4095$) and hp-methods on adaptive meshes. The computation times in Table 1 show that the computational effort for the assembly of the Galerkin matrix grows proportional to $\mathcal{N}^\alpha$ with $2 \leq \alpha \leq 2.5$. Since we have to compute $\mathcal{N}^2$ matrix entries, this shows that the computational effort scales almost optimally with respect to the degrees of freedom $\mathcal{N}$.

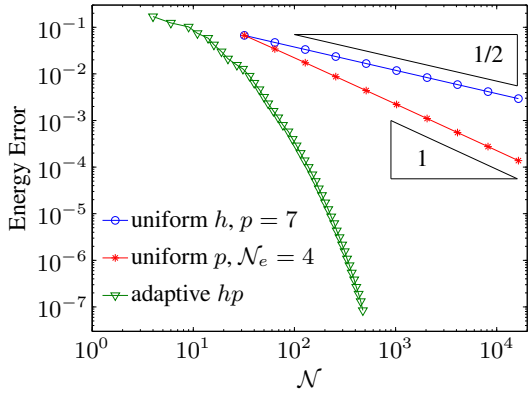


Figure 2: Energy error over the the degrees of freedom $\mathcal{N}$ for uniform h- and p-methods and an adaptive hp-method.

degrees of freedom $\mathcal{N}$	type of refinement uniform-h	uniform-p
1024	0.42	0.20
2048	1.76	0.92
4096	7.89	6.17
8192	35.94	40.65
16384	169.12	281.70

Table 1: Computation times in seconds for the assembly of the Galerkin matrix for different values of $\mathcal{N}$.

A Posteriori Error Estimators and Adaptive Algorithms

In order to achieve highly accurate numerical solutions, a fine discretization of the problem has to be chosen. This results in linear systems of equations with millions of degrees of freedom. In order to reduce the dimensionality, the discretization has to be adapted to the problem under consideration. In areas where the solution has singularities or discontinuities, the discretization has to be finer than in areas where the solution is smooth. If the solution and its behavior, e.g. singularities or discontinuities, are known a priori, one can create such adapted discretizations, which can be locally refined meshes and/or adapted polynomial degree distributions. However, for most applications, the exact solution is not known a priori. In this case algorithms that adapt the discretization automatically are required. For the construction of such algorithms, one needs a posteriori error estimators, which are computable quantities that estimate the error of the discretization. These estimators have to be reliable and efficient, i.e. a lower and upper bound to the error up to a constant that is independent of the chosen discretization. Furthermore, the error estimators have to be local, i.e. provide a local estimate of the error on each element of the mesh.

For h-BEM, different types of error estimators have been introduced and analyzed, for example residual-based error estimators (see e.g. [CS95, CMS01, CMPS04] and the references therein), and estimators created by comparing solutions from different meshes called $h - h/2$ error estimators [FLP08, EFFP08, EFGP12]. The convergence of h-adaptive algorithms is analyzed in [AFLP12, Kar12, FKMP13]. Recently, the quasi-optimal convergence of an h-adaptive algorithm was shown in [FKMP13].
For p- and hp-BEM, residual type error estimators and hp-adaptive algorithms were analyzed in [CFS96]. Furthermore, in [HMS01, Heu02] two-level error estimators are introduced and used for the design of hp-adaptive algorithms. However, the convergence analysis for hp-adaptive algorithms for BEM has not yet been done.

In this work we analyze the generalization of $h - h/2$ error estimators to hp-BEM. Theoretical results show that a generalization using standard L^2 norms is not possible. A generalization with weighted L^2 norms and weighted projections is analyzed numerically using eigenvalue problems. These numerical experiments indicate that the choice of weighted norms and projections results in an error estimator that is efficient and reliable. Using this error estimator, we construct two hp-adaptive algorithms based on the Dörfler marking strategy [Dör96] and prove the convergence of these algorithms using estimator reduction as in [AFLP12]. Numerical experiments show that both algorithms yield exponential convergence with respect to the degrees of freedom; see the green line with triangles in Figure 2.

Data Compression Techniques

As mentioned above, the integral operators arising in BEM are nonlocal. Hence, the discretization of boundary integral equations generally leads to dense Galerkin matrices. Note that the storage requirement for a dense matrix is $\mathcal{O}(\mathcal{N}^2)$ and the computational cost for the matrix vector multiplication is of the same order. To reduce the storage requirement of Galerkin matrices and the computational cost for the solution of the arising linear systems of equations, data compression techniques have to be employed.

The most commonly used data compression methods for h-BEM are $\mathcal{H}$- and $\mathcal{H}^2$-Matrices (see e.g. [Hac99, HKS00, GH03, BÖ6]) and adaptive cross approximation (see e.g. [Beb00]). Here, the idea is to split the Galerkin matrix hierarchically into sub-blocks and use low rank approximations on certain sub-blocks. This results in almost linear complexities ($\mathcal{O}(\mathcal{N}\ \log(\mathcal{N})^\alpha\ \nu)$) for $\mathcal{H}$-Matrices and linear complexities ($\mathcal{O}(\mathcal{N}\ \nu)$) for $\mathcal{H}^2$-Matrices for the assembly and storage of $\mathcal{N} \times \mathcal{N}$ Galerkin matrices and the matrix vector multiplication. Here, ν is the rank of the low rank approximations. Only little research has been done to date on data compression techniques for p- and hp-BEM. Heuer discussed the idea of obtaining a sparse Galerkin matrix by neglecting matrix entries of small magnitude in [Heu96].

We develop a data compression scheme for the single layer Galerkin matrix arising in p- and hp-BEM. The whole compression scheme is based on small perturbations in the Galerkin matrix not affecting the overall error. We generalize the results from [GHS05] for hp-BEM and obtain an upper bound for the perturbations in the Galerkin matrix. In order to obtain a compression scheme for p-BEM, we employ different compression techniques for different sub-blocks of the Galerkin matrix (see Figure 3). For diagonal blocks, we combine the $\mathcal{H}$-Matrix framework [Hac99] with the results for Legendre expansions from [AR91] and obtain a hierarchical structure. We show that the storage requirement for one $n \times n$ diagonal block and the complexity of the multiplication of a diagonal block with a vector are proportional to $n \log(n)^2$. For off-diagonal blocks, we use a cross approximation scheme [Beb00] to create low rank approximations. We show numerically that the ranks needed in the approximation of an $n \times n$ block are of the order $\mathcal{O}(\log(n)^2)$. This means that the amount of storage for one off-diagonal block and the complexity for the matrix vector multiplication is $\mathcal{O}(n \log(n)^2)$. For all other blocks, which we call far field blocks, we use an idea from [Heu96], which is to neglect small matrix entries. Using asymptotic estimates for Legendre functions, we are able to derive a simple upper bound for the magnitude of Galerkin entries that can be computed a priori. Combining this bound with the estimate for perturbations, we develop an algorithm for the compression of far field blocks. Complexity estimates show that for one $n \times n$ far field block we only need to compute and store $\mathcal{O}(\log(n)^2)$ matrix entries.
The compression scheme is extended for Galerkin matrices arising in hp-BEM with quasi-

uniform meshes and polynomial degree distributions. We create low rank approximations of clusters of far field blocks by generalizing the $\mathcal{H}$-Matrix techniques. Extending the complexity estimates from $\mathcal{H}$-Matrices shows that the compressed matrix requires an amount of storage proportional to $\mathcal{N}\log(\mathcal{N})^3$ for the $\mathcal{N}\times\mathcal{N}$ Galerkin matrix, and the matrix vector multiplication can be done with $\mathcal{O}(\mathcal{N}\log(\mathcal{N}))$ operations. This is similar to the complexity of $\mathcal{H}$-Matrices.

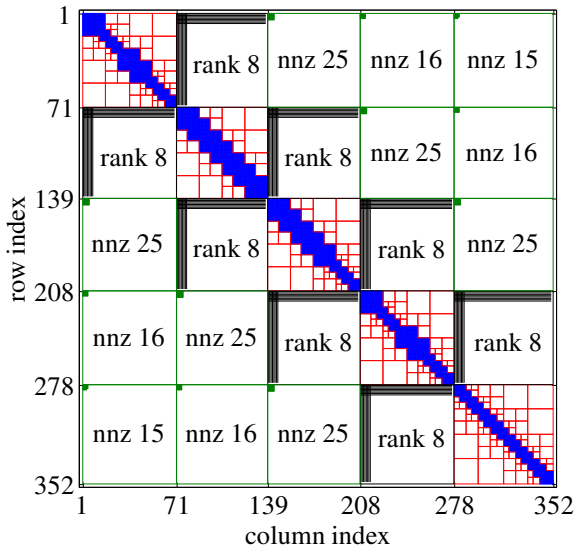


Figure 3: Structure of the compressed Galerkin matrix $\mathbf{V}$ for a boundary mesh with 5 elements and polynomial degrees between 67 and 73. The number of nonzero entries in the respective matrix block is denoted by "nnz".

Outline

In the first of five chapters we introduce the function spaces, integral operators, and boundary integral equations relevant for this work. We also summarize the main results of hp-boundary element methods, introduce geometric boundary meshes, and present standard a priori convergence results for hp-BEM.

The second chapter serves as a basis for the stable implementation of hp-BEM. After having introduced Legendre functions and reviewed their main properties, we define modified Legendre functions. We derive three-term recurrence relations and show that the modified Legendre functions are so-called minimal solutions of these recurrence relations. The stable computation of minimal solutions is discussed in detail. Furthermore, we derive asymptotic estimates for modified associated Legendre functions that are needed for the compression scheme introduced in Chapter 5. We also introduce integrals of modified Legendre functions and derive four-term recurrence relations for them. For these recurrence relations, we do not have an algorithm that is stable in all cases. To overcome this problem, we evaluate the recurrence relation in the forward direction, which is unstable, but with higher precision than the usual machine precision of 15 significant digits by using the multiple precision libraries `mpfr` and `mpc` for the computations. To ensure the accuracy of the computational results, we derive an upper bound for the required

precision using numerical experiments.

The evaluation of potentials and computation of entries of Galerkin matrices is discussed in Chapter 3. We show that both can be reduced to the evaluation of modified Legendre functions and their integrals. Numerical experiments show the code is efficient and the results are accurate even for high polynomial degrees.

Chapter 4 is concerned with a posteriori error estimators, the construction of hp-adaptive algorithms, and the convergence of these algorithms. First, we recap the $h - h/2$ error estimators from [FLP08] and review the proof of their reliability and efficiency. Then we investigate if and how the error estimators can be generalized for hp-BEM. Theoretical results show that a generalization using standard L^2 norms is not possible. A generalization with weighted L^2 norms is discussed and analyzed numerically using eigenvalue problems. In Section 3 we introduce two hp-adaptive algorithms that are based on the Dörfler marking strategy [Dör96] and show the convergence of these algorithms. Finally, numerical experiments show that both algorithms yield exponential convergence.

The last chapter discusses data compression techniques for p- and hp-BEM. After having reviewed existing compression schemes and their main properties, we present a data compression scheme for the single layer Galerkin matrix. The basis of the compression scheme is an estimate for perturbations in the Galerkin matrix, which is presented in the second section. The compression scheme uses different compression techniques for different sub-blocks of the Galerkin matrix, which are discussed in Sections 3-5. In Section 6 we discuss an extension of the compression scheme for hp-BEM with quasi-uniform meshes. Here, the idea is to create low rank approximations of clusters of sub-blocks, which can be seen as generalization of the $\mathcal{H}$-Matrix approach. Finally, the numerical experiments presented in the last section confirm the theoretical complexity estimates and show that the compression scheme is both stable and efficient.

Figure 4 gives an overview of the chapters in this dissertation and shows how they are related.

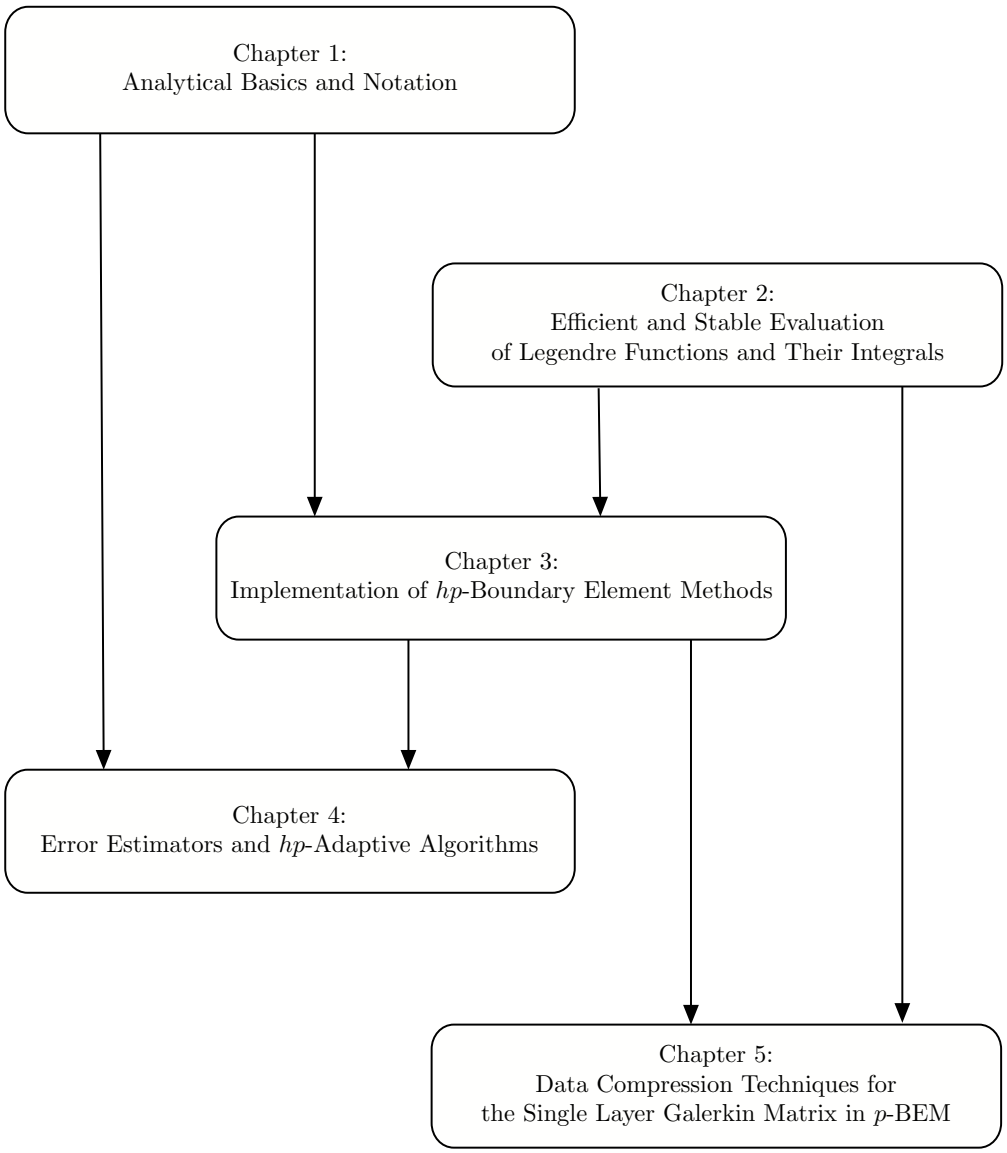


Figure 4: Overview of the chapters in this dissertation and the relationships between chapters.

1. Analytical Basics and Notation

The goal of this chapter is to introduce some notation and basic theoretical results used throughout this work. We start by introducing function spaces. In the second section we give a short introduction to boundary integral equations, specifically the main results for the Laplace operator in two dimensions. In the third section we discuss hp-boundary element methods. The fourth section introduces different methods that can be used to increase the dimension of the discrete ansatz spaces in hp-BEM and defines geometric grids. Finally, in the last section, we summarize some results concerning the a priori error analysis for hp-BEM.

1.1. Function Spaces

We begin by introducing function spaces, namely Lebesgue spaces and Sobolev spaces for domains and boundaries of domains. Here, we only state the main results. For an in-depth discussion of Lebesgue- and Sobolev spaces, see e.g. [AF03].
In the following let $\Omega \subset \mathbb{R}^2$ be a Lipschitz domain and $\Gamma \subseteq \partial\Omega$ be the boundary of Ω or a connected subset of the boundary of Ω. Since Ω is a Lipschitz domain, its boundary is locally the graph of a Lipschitz continuous function. Furthermore, Ω is locally only on one side of Γ. For a detailed definition, see e.g. [SaSch04].
Writing $C^k(\Omega)$ ($k \in \mathbb{N}_0$) for the space of all functions on Ω that are k times continuously differentiable, we can define the space $C^\infty(\Omega)$ of infinitely differentiable functions by

$$C^\infty(\Omega) = \bigcap_{k\in\mathbb{N}_0} C^k(\Omega).$$

Furthermore, for $k \in \mathbb{N}_0 \cup \{\infty\}$, we define the spaces

$$C_0^k(\Omega) := \{v \in C^k(\Omega) \ : \ v \text{ has compact support in } \Omega\}.$$

The space $L^2(\Omega)$ is the space of all measurable functions $u : \Omega \to \mathbb{R}$ with

$$\int_\Omega |u(x)|^2 \, dx < \infty.$$

Together with the L^2 scalar product

$$(u, v)_{L^2(\Omega)} = (v, u)_{L^2(\Omega)} = \int_\Omega u(x) \; v(x) \, dx \quad \forall u, v \in L^2(\Omega),$$

the space $L^2(\Omega)$ is a Hilbert space. Furthermore, the L^2 scalar product induces the norm

$$\|u\|^2_{L^2(\Omega)} := (u,u)_{L^2(\Omega)} = \int_\Omega u(x)^2\,dx.$$

We define the space $L^2_{loc}(\Omega)$ as the space of all measurable functions v such that the restriction $v|_K$ is in $L^2(K)$ for all compact subsets $K \subset \Omega$.
The space of all functions u that are essentially bounded is called $L^\infty(\Omega)$, and a norm on $L^\infty(\Omega)$ is given by

$$\|u\|_{L^\infty(\Omega)} := \text{ess sup}_{x\in\Omega}|u(x)| = \inf\{C > 0 \,:\, |u(x)| < C \text{ almost everywhere in } \Omega\}.$$

Sobolev spaces. Since functions in $L^2(\Omega)$ do not necessarily have a classical derivative, we introduce the weak derivative.

Definition 1.1.1. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2) \in \mathbb{N}_0^2$ be a multi-index of order $|\boldsymbol{\alpha}| := \alpha_1 + \alpha_2$ and let

$$D^{\boldsymbol{\alpha}} u := \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}}$$

be the $|\boldsymbol{\alpha}|$-th derivative of u.
For a function $u \in L^2(\Omega)$, the $\boldsymbol{\alpha}$-th weak derivative $D^{\boldsymbol{\alpha}}_w u$ is defined by

$$(D^{\boldsymbol{\alpha}}_w u, v)_{L^2(\Omega)} = (-1)^{|\boldsymbol{\alpha}|}(u, D^{\boldsymbol{\alpha}} v)_{L^2(\Omega)} \quad \forall v \in C_0^\infty(\Omega).$$

Remark 1.1.2. If $u \in L^2(\Omega)$ has the weak derivative $D^{\boldsymbol{\alpha}}_w u$ and if the derivative $D^{\boldsymbol{\alpha}} u$ exists on a subset of Ω, then the derivative and the weak derivative coincide almost everywhere in this subset. Therefore, in the following we use the notation $D^{\boldsymbol{\alpha}} u$ for the weak derivative.

Definition 1.1.3. (i) For $k \in \mathbb{N}_0$, the Sobolev-space $H^k(\Omega)$ is defined by

$$H^k(\Omega) := \{u \in L^2(\Omega) \,:\, u \text{ is } \boldsymbol{\alpha} \text{ times weakly differentiable and } D^{\boldsymbol{\alpha}} u \in L^2(\Omega)\}.$$

The space $H^k(\Omega)$ ($k \in \mathbb{N}$) is equipped with the scalar product

$$(u,v)_{H^k(\Omega)} := \sum_{0\le|\boldsymbol{\alpha}|\le k} (D^{\boldsymbol{\alpha}} u, D^{\boldsymbol{\alpha}} v)_{L^2(\Omega)},$$

which induces the norm

$$\|u\|^2_{H^k(\Omega)} := \sum_{0\le|\boldsymbol{\alpha}|\le k} \|D^{\boldsymbol{\alpha}} u\|^2_{L^2(\Omega)}.$$

(ii) We define

$$H^k_{loc}(\Omega) := \left\{u :\, \Omega \to \mathbb{R} \,:\, u|_K \in H^k(K) \text{ for all compact } K \subset \Omega\right\}.$$

Remark 1.1.4. It can be seen directly that $H^0(\Omega) = L^2(\Omega)$. Furthermore, the space $H^k(\Omega)$ together with the scalar product $(u,v)_{H^k(\Omega)}$ is a Hilbert space.

For noninteger orders $s = k + \sigma$ with $k \in \mathbb{N}_0$ and $\sigma \in (0,1)$, we define the Sobolev-space $H^s(\Omega)$ using the Sobolev-Slobodeckij semi-norm

$$|u|^2_{H^\sigma(\Omega)} := \int_\Omega \int_\Omega \frac{|u(x) - u(y)|^2}{|x-y|^{2+2\sigma}}\, dx\, dy.$$

Definition 1.1.5. Let $k \in \mathbb{N}_0$, $\sigma \in (0,1)$, and $s = k + \sigma$.

(i) We define the fractional order Sobolev-space

$$H^s(\Omega) := \{u \in H^k \,:\, |D^{\boldsymbol{\alpha}} u|_{H^\sigma(\Omega)} < \infty \text{ for } |\boldsymbol{\alpha}| = k\}$$

and the corresponding norm

$$\|u\|^2_{H^s(\Omega)} := \|u\|^2_{H^k(\Omega)} + \sum_{|\boldsymbol{\alpha}|=k} |D^{\boldsymbol{\alpha}} u|^2_{H^\sigma(\Omega)}.$$

(ii) The space $\widetilde{H}^s(\Omega)$ is defined as the closure of $C_0^\infty(\Omega)$ in $H^s(\mathbb{R}^n)$ and the space $H_0^s(\Omega)$ as the closure of $C_0^\infty(\Omega)$ in $H^s(\Omega)$. Note that $\widetilde{H}^s(\Omega) \subseteq H_0^s(\Omega)$.

Negative order Sobolev spaces. Negative order Sobolev spaces are defined as dual spaces of the corresponding Sobolev spaces of positive order. Using the Riesz representation theorem, we can define dual spaces using the extended L^2 scalar product.

Definition 1.1.6. For $k \geq 0$, the space $H^{-k}(\Omega)$ is the dual space of $\widetilde{H}^k(\Omega)$ with respect to the extended L^2 scalar product $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_{H^{-k}(\Omega) \times \widetilde{H}^k(\Omega)}$. The norm on $H^{-k}(\Omega)$ is defined as the dual norm

$$\|u\|_{H^{-k}(\Omega)} := \sup_{0 \neq v \in \widetilde{H}^k(\Omega)} \frac{|\langle u, v \rangle|}{\|v\|_{\widetilde{H}^k(\Omega)}}.$$

Similarly, the space $\widetilde{H}^{-k}(\Omega)$ can be defined as the dual space of $H^k(\Omega)$. For a detailed discussion, see [McL00].
In the following we write $\langle \cdot, \cdot \rangle$ for the extended L^2 scalar product whenever the spaces involved are clear. Otherwise, we write the spaces as index, e.g. $\langle \cdot, \cdot \rangle_{H^{-1}(\Omega) \times \widetilde{H}^1(\Omega)}$.

Sobolev spaces on boundaries. Sobolev spaces on the boundary $\Gamma = \partial\Omega$ or a subset of the boundary can be defined using local parametrizations of Γ by lifting up. We only state the main facts and refer to [SaSch04, Chapter 2.4] for more details. For a Lipschitz domain Ω, the boundary $\Gamma = \partial\Omega$ can be locally parametrized by a Lipschitz continuous function and one can construct the spaces $H^s(\Gamma)$ for $s \in [0,1]$. For higher orders $s \in [0,k]$, the space $H^s(\Gamma)$ can be

defined for the boundary Γ of a C^k domain Ω, i.e. for a Lipschitz domain whose boundary has a local Hölder continuous parametrization of order $(k-1,1)$. Here, we are only interested in the space $H^s(\Gamma)$ for $s \in [0,1]$. We also introduce the space

$$H^s_*(\Gamma) := \left\{ u \in H^s(\Gamma) \ : \ \int_\Gamma u(x)\, ds_x = 0 \right\},$$

i.e. the space of all functions in $H^s(\Gamma)$ having integral mean zero. For the definition of Sobolev spaces of negative orders, we again use the Riesz representation theorem. For $s \in (0,1)$, we define the space $H^{-s}(\Gamma)$ as the dual space of $\widetilde{H}^s(\Gamma)$ and the norm on $H^{-s}(\Gamma)$ as the dual norm

$$\|u\|_{H^{-s}(\Gamma)} := \sup_{0 \neq v \in H^s(\Gamma)} \frac{|\langle u, v \rangle|}{\|v\|_{\widetilde{H}^s(\Gamma)}}$$

with the extended L^2 scalar product $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_{H^{-s}(\Gamma) \times \widetilde{H}^s(\Gamma)}$.

An alternative way to define fractional order Sobolev spaces, both for domains and boundaries of domains, is by defining them as interpolation spaces. We do not give a detailed definition of interpolation spaces here but refer to [McL00, Appendix B]. The following remark summarizes the properties that we need in later chapters.

Remark 1.1.7. (i) For $s \in (0,1)$ and $\Gamma \subseteq \partial\Omega$, the space $\widetilde{H}^s(\Gamma)$ can be defined as the complex interpolation space between $H^1_0(\Gamma)$ and $L^2(\Gamma)$, i.e. $\widetilde{H}^s(\Gamma) := [L^2(\Gamma), H^1_0(\Gamma)]_s$. The space $H^s(\Gamma)$ can be defined as $H^s(\Gamma) := [L^2(\Gamma), H^1(\Gamma)]_s$.

(ii) The following properties of interpolation spaces are taken from [CP06]. Let $s \in (0,1)$. For two normed spaces $X_0 \subseteq X_1$, the norm of the interpolation space $X = [X_0, X_1]_s$ satisfies

$$\|\cdot\|_X \leq \|\cdot\|^{1-s}_{X_0} \, \|\cdot\|^s_{X_1}.$$

Furthermore, if $Y := [Y_0, Y_1]_s$ is the complex interpolation space of the normed spaces $Y_0 \subseteq Y_1$, and if the linear bounded operator $E : X_0 \to Y_0$ (i.e. $E \in L(X_0, Y_0)$) can also be viewed as a linear bounded operator from $X_1 \to Y_1$, then $E : X \to Y$ is well defined and continuous. Furthermore, there holds

$$\|E\|_{L(X,Y)} \leq \|E\|^{1-s}_{L(X_0,Y_0)} \|E\|^s_{L(X_1,Y_1)}.$$

(iii) If $H^s(\Gamma)$ is defined as complex interpolation space between $L^2(\Gamma)$ and $H^1(\Gamma)$, we have the following property. Let Γ be partitioned into elements $T_1, \ldots, T_n$. Then for $u \in H^s(\Gamma)$ there holds $u|_{T_i} \in H^s(T_i)$ $(i = 1, \ldots, n)$ and

$$\sum_{i=1}^n \left\| u|_{T_i} \right\|^2_{H^s(T_i)} \leq \|u\|^2_{H^s(\Gamma)}.$$

If $H^s(\Gamma)$ is defined using the Sobolev-Slobodeckij norm, this inequality holds only up to a constant depending on the partition $T_1, \ldots, T_n$.

Trace operators. On Lipschitz domains we can define a trace operator in order to obtain an analytic representation of Sobolev spaces on boundaries.

Theorem 1.1.8. ([McL00, Theorems 3.37, 3.38]) *For a Lipschitz domain Ω with boundary $\Gamma = \partial\Omega$ and $s \in (1/2, 3/2)$, there exists a linear bounded operator $\gamma_0 : H^s(\Omega) \to H^{s-1/2}(\Gamma)$ with*

$$\gamma_0 u = u|_\Gamma \quad \forall u \in C^\infty(\overline{\Omega}).$$

Furthermore, we can define the one sided trace operators $\gamma_0^{int} : H^s(\Omega) \to H^{s-1/2}(\Gamma)$ and $\gamma_0^{ext} : H^s(\mathbb{R}^2 \setminus \overline{\Omega}) \to H^{s-1/2}(\Gamma)$.

Remark 1.1.9. In the following, we are interested in the space $H^{1/2}(\Gamma)$. Using the previous theorem, this space can be defined as the trace space of $H^1(\Omega)$, i.e.

$$H^{1/2}(\Gamma) = \left\{ u|_\Gamma \,:\, u \in H^1(\Omega) \right\}.$$

For the definition of boundary integral operators, we also need the co-normal derivative. This can be defined using the integration by parts formula as follows.

Theorem 1.1.10. ([McL00, Lemma 4.3]) *Let Ω be a bounded Lipschitz domain with boundary $\Gamma = \partial\Omega$. For $u \in H^1(\Omega)$ and $f \in \widetilde{H}^{-1}(\Omega)$ with $-\Delta u = f$ in a weak sense, there exists $g \in H^{-1/2}(\Gamma)$ such that*

$$\langle g, \gamma_0^{int} v \rangle_{H^{-1/2}(\Gamma) \times H^{1/2}(\Gamma)} = (\nabla u, \nabla v)_{L^2(\Omega)} - \langle f, v \rangle_{\widetilde{H}^{-1}(\Omega) \times H^1(\Omega)} \quad \forall v \in H^1(\Omega).$$

We call $\gamma_1^{int} u := g$ the co-normal derivative of u, because, for $u \in H^2(\Omega)$, there holds for the normal derivative of u that $\partial_n u = \gamma_1^{int} u$. Analogously to the exterior trace operator γ_0^{ext}, we can define the exterior co-normal derivative γ_1^{ext}.

1.2. Boundary Integral Equations

In order to apply the boundary element method to an elliptic PDE, we have to rewrite the PDE into an equivalent boundary integral equation. This can be done for all partial differential operators for which a fundamental solution is known. In this work we are interested in the two dimensional Laplace operator whose fundamental solution is called the Newton kernel and given by

$$-\frac{1}{2\pi} \log |x|.$$

The main result is the Green representation formula, which gives the relationship between the Poisson equation and a boundary integral equation and is stated in the following theorem.

Theorem 1.2.1. (Green's Representation Formula [Jas63]) *Let $u \in H^1(\Omega)$ and $f \in H^{-1}(\Omega)$ with $-\Delta u = f$. Then there holds almost everywhere in Ω*

$$\begin{aligned} u(x) = \quad & -\frac{1}{2\pi}\int_\Omega f(y)\log|x-y|\,dy \\ & -\frac{1}{2\pi}\int_\Gamma (\gamma_1^{int}u)(y)\log|x-y|\,ds_y \\ & +\frac{1}{2\pi}\int_\Gamma (\gamma_0^{int}u)(y)\frac{\partial}{\partial \mathbf{n}_y}\log|x-y|\,ds_y. \end{aligned} \tag{1.1}$$

The three integral operators in the representation formula are the **Newton potential**

$$(\widetilde{\mathcal{N}}g)(x) := -\frac{1}{2\pi}\int_{\mathbb{R}^2} g(y)\log|x-y|\,dy \quad x \in \mathbb{R}^2, \tag{1.2}$$

which is defined for all $g : \mathbb{R}^2 \to \mathbb{R}$ with compact support, the **single layer potential**

$$(\widetilde{\mathcal{V}}\phi)(x) := -\frac{1}{2\pi}\int_\Gamma \phi(y)\log|x-y|\,ds_y \quad x \in \mathbb{R}^2 \setminus \Gamma, \tag{1.3}$$

which is defined for all $\phi : \Gamma \to \mathbb{R}$, and the **double layer potential**

$$(\widetilde{\mathcal{K}}\psi)(x) := -\frac{1}{2\pi}\int_\Gamma \psi(y)\frac{\partial}{\partial \mathbf{n}_y}\log|x-y|\,ds_y \quad x \in \mathbb{R}^2 \setminus \Gamma, \tag{1.4}$$

which is defined for all $\psi : \Gamma \to \mathbb{R}$, where $\mathbf{n}_y$ is the normal vector with respect to y. For some applications, not only the Cauchy data $\psi = \gamma_0^{int}u$ and $\phi = \gamma_1^{int}u$ but also derivatives of ϕ and ψ as well as their potentials are of interest. Therefore, for a given point $x \in \mathbb{R}^2 \setminus \Gamma$ and a direction $\mathbf{n}_x$ we define the **adjoint double layer operator**

$$(\widetilde{\mathcal{K}}'\phi)(x) := -\frac{1}{2\pi}\int_\Gamma \phi(y)\frac{\partial}{\partial \mathbf{n}_x}\log|x-y|\,ds_y \quad x \in \mathbb{R}^2 \setminus \Gamma, \tag{1.5}$$

and the **hypersingular operator**

$$(\widetilde{\mathcal{W}}\psi)(x) := -\frac{1}{2\pi}\int_\Gamma \phi(y)\frac{\partial}{\partial \mathbf{n}_y}\frac{\partial}{\partial \mathbf{n}_x}\log|x-y|\,ds_y \quad x \in \mathbb{R}^2 \setminus \Gamma. \tag{1.6}$$

In this work we only consider the Laplace equation $\Delta u = 0$. Therefore, we do not discuss the Newton potential. We collect some important properties of the single and double layer potential in the following theorems. For more details and proofs, see e.g. [SaSch04, Chapter 3] or [McL00, Chapter 7].

Theorem 1.2.2. (Single Layer Potential)

(i) The single layer potential is a linear bounded operator from $H^{-1/2}(\Gamma)$ to $H^1_{loc}(\mathbb{R}^2)$.

(ii) The traces $\gamma_0^{int}\widetilde{\mathcal{V}} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ and $\gamma_0^{ext}\widetilde{\mathcal{V}} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ are linear bounded operators, and there holds

$$[\gamma_0\widetilde{\mathcal{V}}\phi] := \gamma_0^{ext}\widetilde{\mathcal{V}}\phi - \gamma_0^{int}\widetilde{\mathcal{V}}\phi = 0. \tag{1.7}$$

(iii) The co-normal derivatives $\gamma_1^{int}\widetilde{\mathcal{V}} : H^{-1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ *and* $\gamma_1^{ext}\widetilde{\mathcal{V}} : H^{-1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ *are linear bounded operators, and there holds*

$$[\gamma_1\widetilde{\mathcal{V}}\phi] := \gamma_1^{ext}\widetilde{\mathcal{V}}\phi - \gamma_1^{int}\widetilde{\mathcal{V}}\phi = -\phi.$$

Theorem 1.2.3. *(*Double Layer Potential*)*

(i) The double layer potential is a linear bounded operator from $H^{1/2}(\Gamma)$ *to* $H^1(\Omega)\cap H^1_{loc}(\mathbb{R}^2)$.

(ii) The traces $\gamma_0^{int}\widetilde{\mathcal{K}} : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ *and* $\gamma_0^{ext}\widetilde{\mathcal{K}} : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ *are linear bounded operators, and there holds*

$$[\gamma_0\widetilde{\mathcal{K}}\psi] := \gamma_0^{ext}\widetilde{\mathcal{K}}\psi - \gamma_0^{int}\widetilde{\mathcal{K}}\psi = \psi. \tag{1.8}$$

(iii) The co-normal derivatives $\gamma_1^{int}\widetilde{\mathcal{K}} : H^{1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ *and* $\gamma_1^{ext}\widetilde{\mathcal{K}} : H^{1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ *are linear bounded operators, and there holds*

$$[\gamma_1\widetilde{\mathcal{K}}\psi] := \gamma_1^{ext}\widetilde{\mathcal{K}}\psi - \gamma_1^{int}\widetilde{\mathcal{K}}\psi = 0.$$

From the representation formula (1.1), it can be seen that the solution u of the Laplace equation $\Delta u = 0$ in the domain Ω can be computed once the complete Cauchy data, i.e. the trace and the co-normal derivative of u on $\Gamma = \partial\Omega$, are known. However, for Dirichlet problems the trace of u is known on Γ, and for Neumann problems the normal derivative of u is given on Γ. For mixed problems, where the boundary Γ is split into Neumann boundary Γ_N and Dirichlet boundary Γ_D with $\Gamma = \Gamma_N \cap \Gamma_D$, the trace of u is given only on Γ_D, and the normal derivative of u is known only on Γ_N. In order to compute the solution u in Ω, one has to compute the missing Cauchy data. For this purpose, we derive boundary integral equations by applying the trace and the co-normal derivative to the representation formula (1.1). With $\phi := \gamma_1^{int}u \in H^{-1/2}(\Gamma)$ and $\psi := \gamma_0^{int}u \in H^{1/2}(\Gamma)$, we obtain

$$\begin{aligned} \gamma_0^{int}u &= \gamma_0^{int}\widetilde{\mathcal{V}}\phi - \gamma_0^{int}\widetilde{\mathcal{K}}\psi, \\ \gamma_1^{int}u &= \gamma_1^{int}\widetilde{\mathcal{V}}\phi - \gamma_1^{int}\widetilde{\mathcal{K}}\psi. \end{aligned} \tag{1.9}$$

Together with the jump relations (1.7) and (1.8), this motivates us to define the following boundary integral operators. The **single layer operator** is defined by

$$(\mathcal{V}\phi)(x) := -\frac{1}{2\pi}\int_\Gamma \phi(y)\log|x-y|\,ds_y,$$

and there holds for all $x \in \Gamma$

$$\mathcal{V} = \gamma_0^{int}\widetilde{\mathcal{V}}.$$

The **double layer operator** is given by

$$(\mathcal{K}\psi)(x) := \lim_{\varepsilon\to 0} -\frac{1}{2\pi}\int\limits_{y\in\Gamma\ |x-y|>\varepsilon} \psi(y)\frac{\partial}{\partial\mathbf{n}_y}\log|x-y|\,ds_y,$$

and there holds

$$(\mathcal{K}\psi)(x) = \left(1 - \frac{\beta(x)}{2\pi} + \gamma_0^{int}\widetilde{\mathcal{K}}\right)\psi(x),$$

where $\beta(x)$ is the inner angle of the boundary Γ in $x \in \Gamma$. The **adjoint single layer operator** is given by

$$(\mathcal{K}'\phi)(x) := \lim_{\varepsilon\to 0} -\frac{1}{2\pi} \int\limits_{y\in\Gamma\ |x-y|>\varepsilon} \phi(y)\frac{\partial}{\partial \mathbf{n}_x}\log|x-y|\, ds_y,$$

and there holds

$$(\mathcal{K}'\phi)(x) = \left(-\frac{\beta(x)}{2\pi} + \gamma_1^{int}\widetilde{\mathcal{V}}\right)\phi(x),$$

where $\beta(x)$ is the inner angle of the boundary Γ in $x \in \Gamma$. Finally, the **hypersingular operator** is given by

$$(\mathcal{W}\psi)(x) := -\gamma_1^{int}(\mathcal{K}\psi)(x).$$

Using these boundary integral operators we can rewrite (1.9) to the so-called Calderon system

$$\begin{pmatrix}\phi\\ \psi\end{pmatrix} = \begin{pmatrix}1/2 - \mathcal{K} & \mathcal{V}\\ \mathcal{W} & 1/2 + \mathcal{K}'\end{pmatrix}\begin{pmatrix}\phi\\ \psi\end{pmatrix}. \tag{1.10}$$

The following lemma collects some important properties of the four boundary integral operators. More details and the proofs of these properties can be found in e.g. [McL00] and [SaSch04].

Lemma 1.2.4. *Let $\Gamma = \partial\Omega$ be the boundary of a Lipschitz domain $\Omega \subset \mathbb{R}^2$.*

(i) There holds

$$\begin{aligned}&\mathcal{V} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma), \qquad &&\mathcal{K} : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma),\\ &\mathcal{K}' : H^{-1/2}(\Gamma) \to H^{-1/2}(\Gamma), \qquad &&\mathcal{W} : H^{1/2}(\Gamma) \to H^{-1/2}(\Gamma),\end{aligned}$$

and all four operators are linear and bounded.

(ii) The operator $\mathcal{V}$ is symmetric, i.e.

$$\langle \mathcal{V}\phi_1, \phi_2\rangle = \langle \phi_1, \mathcal{V}\phi_2\rangle$$

for all ϕ_1, ϕ_2 in $H^{-1/2}(\Gamma)$. Furthermore, if the capacity of Ω, cap(Ω), is smaller than one, the operator $\mathcal{V}$ is $H^{-1/2}$-elliptic, i.e.

$$\langle V\phi, \phi\rangle \geq C\|\phi\|^2_{H^{-1/2}(\Gamma)}$$

for all ϕ in $H^{-1/2}(\Gamma)$ and a constant $C > 0$.

(iii) The operator $\mathcal{W}$ is symmetric and $H^{1/2}_*$-elliptic.

(iv) The operator $\mathcal{K}'$ is the adjoint of $\mathcal{K}$, i.e.

$$\langle \mathcal{K}\psi, \phi \rangle = \langle \psi, \mathcal{K}'\phi \rangle$$

for all $\psi \in H^{1/2}(\Gamma)$ and $\phi \in H^{-1/2}(\Gamma)$.

Remark 1.2.5. The capacity of a domain $\mathrm{cap}(\Omega)$ is defined in [SlSp88] or [Dij08]. It is also known as the transfinite diameter of a domain and analyzed in [PS09]. We do not state the detailed definition here, but we do mention that the capacity of a domain is always smaller than its Euclidian diameter. Thus, if $\mathrm{diam}(\Omega)$ is smaller than one, the operator $\mathcal{V}$ is $H^{-1/2}$-elliptic. This can always be achieved by scaling Ω.
For some domains, the capacity can be computed explicitly (see [Dij08], [PS09]):

- For a circle with radius R, the capacity is R.
- For a square with side L, the capacity is approximately $0.59\,L$.
- For an ellipse with semi axes a and b, the capacity is $(a+b)/2$.
- For an interval of length L, the capacity is $L/4$.
- For a half disk with radius R, the capacity is $4R/(3\sqrt{3})$.

1.2.1. The Dirichlet Problem and Symm's Integral Equation

We consider the Dirichlet problem

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega, \\ \gamma_0 u &= \psi \quad \text{on } \Gamma = \partial\Omega, \end{aligned}$$

where the equations hold in a weak sense and $\psi \in H^{1/2}(\Gamma)$ is the given Dirichlet data. In order to compute the Neumann data $\phi = \gamma_1 u$, we rearrange the first equation of the Calderon system (1.10) and obtain Symm's integral equation

$$\mathcal{V}\phi = \left(\mathcal{K} + \frac{1}{2}\right)\psi. \tag{1.11}$$

We also consider a generalized version of Symm's integral equation, which reads

$$\mathcal{V}\phi = f, \quad \text{for } f \in H^{1/2}(\Gamma). \tag{1.12}$$

In order to solve these equations numerically with a Galerkin method we need the variational formulations

$$\langle \mathcal{V}\phi, w \rangle = \langle (\mathcal{K} + 1/2)\psi, w \rangle \qquad \forall w \in H^{-1/2}(\Gamma) \tag{1.13}$$

and

$$\langle \mathcal{V}\phi, w \rangle = \langle f, w \rangle \qquad \forall w \in H^{-1/2}(\Gamma). \tag{1.14}$$

Since the operator $\mathcal{V}$ is $H^{-1/2}$-elliptic and bounded, the Lax-Milgram theorem ensures the unique solvability of Symm's integral equation and the equivalent variational formulations.

1.2.2. The Neumann Problem and the Hypersingular Integral Equation

For the Neumann problem

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega, \\ \gamma_1 u &= \phi \quad \text{on } \Gamma = \partial\Omega, \end{aligned}$$

with given Neumann data $\phi \in H^{-1/2}(\Gamma)$, we consider the second equation of the Calderon system (1.10) and obtain the hypersingular integral equation

$$\mathcal{W}\psi = \left(\frac{1}{2} - \mathcal{K}'\right)\phi. \tag{1.15}$$

More generally, we consider

$$\mathcal{W}\psi = f, \quad \text{for } f \in H^{-1/2}(\Gamma). \tag{1.16}$$

In Lemma 1.2.4 we state that the operator $\mathcal{W}$ is bounded and elliptic on $H_*^{-1/2}(\Gamma)$, the space of all functions in $H^{-1/2}$ that have integral mean zero. Thus, according to the Lax-Milgram theorem, in this space the hypersingular integral equations as well as the equivalent variational formulations

$$\langle \mathcal{W}\psi, w\rangle = \langle (1/2 - \mathcal{K}')\phi, w\rangle \qquad \forall w \in H_*^{-1/2}(\Gamma) \tag{1.17}$$

and

$$\langle \mathcal{W}\psi, w\rangle = \langle f, w\rangle \qquad \forall w \in H_*^{-1/2}(\Gamma) \tag{1.18}$$

have a unique solution. However, for the implementation, this variational formulation is inconvenient, since one has to consider ansatz functions with integral mean zero. Therefore, we introduce the bilinear form

$$\langle\!\langle \psi, w\rangle\!\rangle_{\mathcal{W}+S} := \langle \mathcal{W}\psi, w\rangle + \left(\int_\Gamma \psi \, ds\right)\left(\int_\Gamma w \, ds\right)$$

and consider the variational problems to find $\psi \in H^{1/2}(\Gamma)$ with

$$\langle\!\langle \psi, w\rangle\!\rangle_{\mathcal{W}+S} = \langle (1/2 - \mathcal{K}')\phi, w\rangle \qquad \forall w \in H^{1/2}(\Gamma) \tag{1.19}$$

and

$$\langle\!\langle \psi, w\rangle\!\rangle_{\mathcal{W}+S} = \langle f, w\rangle \qquad \forall w \in H^{1/2}(\Gamma), \tag{1.20}$$

respectively.

Lemma 1.2.6. *The variational formulation (1.17) is equivalent to (1.19).*

Proof. Let $\psi \in H_*^{1/2}(\Gamma)$ fulfill (1.17). Then there holds for all $w \in H^{1/2}(\Gamma)$

$$\langle W\psi, w\rangle = \langle W\psi, w\rangle + \underbrace{\left(\int_\Gamma \psi \, ds\right)}_{=0} \left(\int_\Gamma w \, ds\right) = \langle (1/2 - \mathcal{K}')\phi, w\rangle.$$

Thus, ψ fulfills (1.19).
If $\psi \in H^{1/2}(\Gamma)$ fulfills (1.19), then ψ fulfills (1.17). This is because $\int_\Gamma w \, ds = 0$ for all $w \in H_*^{-1/2}(\Gamma)$. Furthermore, there holds for all $\phi \in H_*^{-1/2}$

$$\langle (1/2 - \mathcal{K}')\phi, 1\rangle = \frac{1}{2}\underbrace{\langle \phi, 1\rangle}_{=0} - \langle \phi, (1/2 + \mathcal{K})1\rangle = \langle \phi, \underbrace{\mathcal{V}(\gamma_1 1)}_{=0}\rangle = 0.$$

Thus, testing (1.19) with $w = 1$ yields

$$\left(\int_\Gamma \psi \, ds\right) |\Gamma| = \underbrace{\langle (1/2 - \mathcal{K}')\phi, 1\rangle}_{=0} - \underbrace{\langle \mathcal{W}\psi, 1\rangle}_{=0} = 0,$$

where $\langle \mathcal{W}\psi, 1\rangle = 0$ follows from the symmetry of $\mathcal{W}$ and the kernel of $\mathcal{W}$ consisting of all constant functions. □

1.2.3. The Mixed Problem

The mixed problem is given by

$$\begin{aligned} -\Delta u &= 0 \quad && \text{in } \Omega, \\ \gamma_0 u &= \psi_D && \text{on } \Gamma_D \subset \partial\Omega, \\ \gamma_1 u &= \phi_N && \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D, \end{aligned}$$

where we assume that $|\Gamma_D| > 0$. Here, the missing Cauchy data are given by

$$\begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} = \begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix} - \begin{pmatrix} \overline{\psi}_D \\ \overline{\phi}_N \end{pmatrix},$$

where $\overline{\phi}_N \in H^{-1/2}(\Gamma)$ and $\overline{\psi}_D \in H^{1/2}(\Gamma)$ are arbitrary extensions of the given Dirichlet and Neumann data, respectively. The full Calderon system can be rearranged to

$$\mathcal{A}\begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} = \left(\frac{1}{2} - \mathcal{A}\right)\begin{pmatrix} \overline{\psi}_D \\ \overline{\phi}_N \end{pmatrix} \tag{1.21}$$

with the Calderon projector

$$\mathcal{A} := \begin{pmatrix} -\mathcal{K} & \mathcal{V} \\ \mathcal{W} & \mathcal{K}' \end{pmatrix},$$

which maps the space $\mathcal{H} := \widetilde{H}^{1/2}(\Gamma_N) \times \widetilde{H}^{-1/2}(\Gamma_D)$ to its dual space $\mathcal{H}^* := \widetilde{H}^{1/2}(\Gamma_D) \times \widetilde{H}^{-1/2}(\Gamma_N)$. The dual space is defined with the duality product

$$\left\langle \begin{pmatrix} \psi_D \\ \phi_N \end{pmatrix}, \begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} \right\rangle_{\mathcal{H}^* \times \mathcal{H}} := \langle \phi_D, \psi_D \rangle_{\Gamma_D} + \langle \phi_N, \psi_N \rangle_{\Gamma_N},$$

where $\langle \cdot, \cdot \rangle_{\Gamma_D}$ and $\langle \cdot, \cdot \rangle_{\Gamma_N}$ denote the extended L^2 scalar products on Γ_D and Γ_N, respectively. The variational formulation of the mixed problem reads

$$\left\langle \mathcal{A} \begin{pmatrix} \psi_D \\ \phi_N \end{pmatrix}, \begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} \right\rangle_{\mathcal{H}^* \times \mathcal{H}} = \left\langle \left(\frac{1}{2} - \mathcal{A} \right) \begin{pmatrix} \overline{\psi}_D \\ \overline{\phi}_N \end{pmatrix}, \begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} \right\rangle_{\mathcal{H}^* \times \mathcal{H}}. \tag{1.22}$$

To ensure that (1.21) has a unique solution, we need $\mathcal{A}$ to be $\mathcal{H}$-elliptic.

Lemma 1.2.7. *The Calderon projector $\mathcal{A}$ is $\mathcal{H}$-elliptic, i.e. there holds for all $(\psi_N, \phi_D) \in \mathcal{H}$*

$$\left\langle \mathcal{A} \begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix}, \begin{pmatrix} \psi_N \\ \phi_D \end{pmatrix} \right\rangle_{\mathcal{H}^* \times \mathcal{H}} \geq c \|(\psi_N, \phi_D)\|^2_{\mathcal{H}},$$

where

$$\|(\psi_N, \phi_D)\|^2_{\mathcal{H}} := \|\psi_N\|^2_{\widetilde{H}^{1/2}(\Gamma_N)} + \|\phi_D\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)}$$

defines a norm on $\mathcal{H}$.

Proof. See [Ste03, Lemma 7.2]. □

1.3. hp-Boundary Element Methods

In order to numerically solve the boundary integral equations introduced in the previous section, we consider finite dimensional subspaces of the function spaces on which the integral equations are defined. In the following we will always assume that $\Omega \subset \mathbb{R}^2$ is a Lipschitz domain with polygonal boundary $\Gamma = \partial\Omega$. Γ can be partitioned into a finite number of relative open and disjoint sub-boundaries. Within the scope of this work we consider the cases $\Gamma = \Gamma_D$ (Dirichlet problem), $\Gamma = \Gamma_N$ (Neumann problem) and $\Gamma = \Gamma_D \cap \overline{\Gamma}_N$ (mixed problem). For the discretization of the boundary Γ, we introduce a triangulation.

Definition 1.3.1. (i) A triangulation of Γ is a finite set of line segments $T_i = \mathrm{conv}\{A_i, B_i\} \subseteq \Gamma$, $A_i, B_i \in \Gamma$, $i = 1, \ldots, \mathcal{N}_e$, with $|T_i \cap T_j| = 0$ for $i \neq j$ and $\cup_{i=1}^{\mathcal{N}_e} T_i = \Gamma$. We write $\mathcal{T} := \{T_i \,:\, i = 1, \ldots, \mathcal{N}_e\}$.

If Γ is divided into Dirichlet and Neumann boundary, i.e. $\Gamma = \Gamma_D \cap \overline{\Gamma}_N$, we assume that each T_i belongs to exactly one of the two boundary parts. Therefore, we define

$$\mathcal{T}_D = \{T \in \mathcal{T} \,:\, T \cap \Gamma_D \neq \emptyset\} \qquad \text{and}$$
$$\mathcal{T}_N = \{T \in \mathcal{T} \,:\, T \cap \Gamma_N \neq \emptyset\}.$$

(ii) For a given triangulation $\mathcal{T}$, the set of all vertices is denoted by $\mathcal{K}$. Furthermore, $\mathcal{N}_v$ denotes the number of vertices.

(iii) Let $\mathbf{h} \in L^\infty(\Gamma)$ be a mesh width function with $\mathbf{h}|_{T_i} = |T_i| =: h_i$, where $|T_i|$ is the length of the element T_i, and let $\overline{h} := \max\{h_i, i = 1, \dots, \mathcal{N}_e\}$ and $\underline{h} := \min\{h_i, i = 1, \dots, \mathcal{N}_e\}$.

(iv) Let $\mathbf{p}_0 \in L^\infty(\Gamma)$ be a polynomial degree function with $\mathbf{p}_0|_{T_i} =: p_i \in \mathbb{N}_0$, and let $\overline{p} := \max\{p_i,\ i = 1, \dots, \mathcal{N}_e\}$ and $\underline{p} := \min\{p_i,\ i = 1, \dots, \mathcal{N}_e\}$.
Let $\mathbf{p}_1 \in L^\infty(\Gamma)$ be a polynomial degree function with $\mathbf{p}_1|_{T_i} =: q_i \in \mathbb{N}$, and let $\overline{q} := \max\{q_i,\ i = 1, \dots, \mathcal{N}_e\}$ and $\underline{q} := \min\{q_i,\ i = 1, \dots, \mathcal{N}_e\}$.

(v) We write $\mathbf{p}_D := \mathbf{p}|_{\Gamma_D}$, $\mathbf{p}_N := \mathbf{p}|_{\Gamma_N}$, $\mathbf{h}_D := \mathbf{h}|_{\Gamma_D}$, and $\mathbf{h}_N := \mathbf{h}|_{\Gamma_N}$.

(vi) For a mesh width function $\mathbf{h}$, we define the mesh width ratio

$$\sigma(\mathbf{h}) := \max\{\frac{h_i}{h_j} : T_i, T_j \in \mathcal{T},\ T_i \cap T_j \neq \emptyset\},$$

and for a polynomial degree function $\mathbf{p}$, we introduce the polynomial degree ratios

$$\pi(\mathbf{p}) := \max\{\frac{p_i}{\max\{p_j, 1\}} : T_i, T_j \in \mathcal{T},\ T_i \cap T_j \neq \emptyset\}.$$

We are now ready to define the discrete spaces that we use for the implementation of hp-BEM.

Definition 1.3.2. Let $\mathbf{h} \in L^\infty(\Gamma)$, $\mathbf{p}_0 \in L^\infty(\Gamma)$, $\mathbf{p}_0|_{T_i} = p_i \in \mathbb{N}_0$ and $\mathbf{p}_1 \in L^\infty(\Gamma)$, $\mathbf{p}_1|_{T_i} = q_i \in \mathbb{N}$ be defined as in Definition 1.3.1. We define the function spaces

$$S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0) = \{\phi \in L^2(\Gamma) :\ \phi|_{T_i} \in \mathcal{P}^{p_i}(T_i)\ \ \forall T_i \in \mathcal{T}\},$$
$$S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1) = \{u \in C^0(\Gamma) :\ u|_{T_i} \in \mathcal{P}^{q_i}(T_i)\ \ \forall T_i \in \mathcal{T}\}.$$

Here, $\mathcal{P}^n(T)$ is the space of all polynomials of degree less than or equal to n on a boundary element $T \in \mathcal{T}$ with respect to the arc length.

Note that $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0) \subset H^{-1/2}(\Gamma)$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1) \subset H^{1/2}(\Gamma)$, i.e. using these ansatz spaces we obtain a conforming Galerkin method. The basis of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is denoted by $\{\varphi_i,\ i = 1, \dots, \mathcal{N}_0\}$, where $\mathcal{N}_0$ is the dimension of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$. For the basis of $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$, we write $\{\psi_i,\ i = 1, \dots, \mathcal{N}_1\}$, where $\mathcal{N}_1$ is the dimension of $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$. Note that for both ansatz spaces we use the same boundary mesh $\mathcal{T}$. However, the polynomial degree functions can differ. For the implementation one usually uses $\mathbf{p}_1 = \mathbf{p}_0 + 1$.

Considering the variational formulations (1.13) and (1.14) of **Symm's integral equation** on the discrete spaces defined above, we obtain the discrete problem to find $\phi_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ such that

$$\langle \mathcal{V}\phi_h^p, \varphi_i \rangle = \langle (\mathcal{K} + 1/2)\psi, \varphi_i \rangle \qquad \forall i = 1, \dots \mathcal{N}_0 \tag{1.23}$$

and

$$\langle \mathcal{V}\phi_h^p, \varphi_i \rangle = \langle f, \varphi_i \rangle \qquad \forall i = 1, \dots \mathcal{N}_0, \tag{1.24}$$

respectively. To fully discretize (1.23) and (1.24) we use the projections $\psi_h^p = \sum_{j=1}^{\mathcal{N}_1} \Psi_j \psi_j$ and $f_h^p = \sum_{j=1}^{\mathcal{N}_1} f_j \psi_j$ of the given Dirichlet data ψ and the right-hand side f into the space $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$. Writing $\phi_h^p = \sum_{j=1}^{\mathcal{N}_0} \Phi_i \varphi_i$, we obtain the linear systems of equations

$$\mathbf{V}\mathbf{\Phi} = (\mathbf{K} + 1/2\, \mathbf{M})\mathbf{\Psi}, \tag{1.25}$$

$$\mathbf{V}\mathbf{\Phi} = \mathbf{M}\mathbf{f}, \tag{1.26}$$

where the Galerkin matrix for the single layer operator $\mathbf{V} \in \mathbb{R}^{\mathcal{N}_0 \times \mathcal{N}_0}$ is given by

$$\mathbf{V} = (V_{i,j}), \quad V_{i,j} = \langle \mathcal{V}\varphi_j, \varphi_i \rangle,$$

and the Galerkin matrix for the double layer operator $\mathbf{K} \in \mathbb{R}^{\mathcal{N}_0 \times \mathcal{N}_1}$ is given by

$$\mathbf{K} = (K_{i,j}), \quad K_{i,j} = \langle \mathcal{K}\psi_j, \varphi_i \rangle.$$

Finally, for the mass matrix $\mathbf{M} \in \mathbb{R}^{\mathcal{N}_0 \times \mathcal{N}_1}$, there holds

$$\mathbf{M} = (M_{i,j}), \quad M_{i,j} = \langle \psi_j, \varphi_i \rangle.$$

Alternatively, one could compute the integrals $b_i := \langle f, \varphi_i \rangle$ $(i = 1, \dots \mathcal{N}_0)$ numerically using quadrature. Then equation (1.26) would be replaced by $\mathbf{V}\mathbf{\Phi} = \mathbf{b}$ with $\mathbf{b} = (b_1, \dots, b_{\mathcal{N}_0})^T$.

For the numerical solution of the **hypersingular integral equation**, we consider the variational formulation (1.19) and (1.20), respectively, on the discrete spaces $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$. Thus, we must find $\psi_h \in S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ such that

$$\langle\!\langle \mathcal{W}\psi_h^p, \psi_i \rangle\!\rangle_{\mathcal{W}+S} = \langle (1/2 - \mathcal{K}')\phi, \psi_i \rangle \qquad \forall i = 1, \dots \mathcal{N}_1 \tag{1.27}$$

and

$$\langle\!\langle \mathcal{W}\psi_h^p, \psi_i \rangle\!\rangle_{\mathcal{W}+S} = \langle f, \psi_i \rangle \qquad \forall i = 1, \dots \mathcal{N}_1, \tag{1.28}$$

respectively. Replacing the Neumann data ϕ by its projection $\phi_h^p = \sum_{j=1}^{\mathcal{N}_1} \Phi_j \phi_j$ into the space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ and replacing $f \in H^{-1/2}(\Gamma)$ by its projection $f_h^p = \sum_{j=1}^{\mathcal{N}_0} f_j \varphi_j$ into $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$, we obtain the linear systems

$$(\mathbf{W} + \mathbf{S})\mathbf{\Psi} = (1/2\, \mathbf{M}^T - \mathbf{K}^T)\mathbf{\Phi}, \tag{1.29}$$

$$(\mathbf{W} + \mathbf{S})\mathbf{\Psi} = \mathbf{M}^T\mathbf{f}. \tag{1.30}$$

Here, $\mathbf{K}^T$ and $\mathbf{M}^T$ are the transposed of the matrices $\mathbf{K}$ and $\mathbf{M}$ that are introduced above. The matrix $\mathbf{W} \in \mathbb{R}^{\mathcal{N}_1 \times \mathcal{N}_1}$ is given by

$$\mathbf{W} = (W_{i,j}), \quad W_{i,j} = \langle \mathcal{W}\psi_j, \psi_i \rangle,$$

and for the stabilization matrix $\mathbf{S}$ there holds

$$\mathbf{S} = \mathbf{c} \cdot \mathbf{c}', \quad \mathbf{c} = (c_j), \quad c_j = \int_\Gamma \psi_j \, ds.$$

For the **mixed problem** we consider the variational formulation (1.22) on the discrete space

$$X_h := S_1(\mathcal{T}_N, \mathbf{p}_N, \mathbf{h}_{1,N}) \times S_0(\mathcal{T}_D, \mathbf{p}_D, \mathbf{h}_{0,D}) \subset \mathcal{H}.$$

By plugging in the representation of the functions $\psi^p_{N,h}$, $\phi^p_{D,h}$ as well as the projections $\overline{\psi}^p_{D,h}$ and $\overline{\phi}^p_{N,h}$ for the extensions of the given Cauchy data and testing the equation with all basis functions of the respective discrete spaces, we obtain the linear system

$$\begin{pmatrix} \mathbf{V}_{\Gamma_D \times \Gamma_D} & -\mathbf{K}_{\Gamma_D \times \Gamma_N} \\ \mathbf{K}^T_{\Gamma_N \times \Gamma_D} & \mathbf{W}_{\Gamma_N \times \Gamma_N} \end{pmatrix} \begin{pmatrix} \mathbf{\Phi}_D \\ \mathbf{\Psi}_N \end{pmatrix} = \begin{pmatrix} \left(\frac{1}{2}\mathbf{M}\mathbf{\Psi} + \mathbf{K}\mathbf{\Psi} - \mathbf{V}\mathbf{\Phi}\right)|_{\Gamma_D} \\ \left(\frac{1}{2}\mathbf{M}^T\mathbf{\Phi} - \mathbf{K}^T\mathbf{\Phi} - \mathbf{W}^T\Phi\right)|_{\Gamma_N} \end{pmatrix}.$$

Here, the indices on the left-hand side indicate that the corresponding Galerkin matrix is only computed with respect to the basis of the respective boundaries. On the right-hand side, we use the Galerkin matrices with respect to the discretization of the full boundary and restrict the sub-vectors to the basis functions that belong to the discrete spaces on Γ_D and Γ_N, respectively.

1.4. Boundary Meshes and Refinement

Throughout this work we consider different types of boundary meshes and different methods to increase the dimension of a discrete ansatz spaces. Here, we consider the space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$. For the space $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ the procedures are similar.

If in a refinement algorithm we create a sequence of discrete spaces $S_\ell = S_0(\mathcal{T}_\ell, \mathbf{h}_\ell, \mathbf{p}_\ell)$ by uniformly refining the boundary mesh $\mathcal{T}$, we call this a **uniform h-method**. Here, a boundary mesh is refined uniformly by splitting each element $T \in \mathcal{T}$ into two sons T' and T'' that have the same mesh width. Furthermore, the sons inherit the polynomial degree from T. Thus, there holds $S_{\ell+1} = S_0(\mathcal{T}_{\ell+1}, \mathbf{h}_\ell/2, \mathbf{p}_\ell)$.

If the boundary mesh is not refined uniformly but adaptively, i.e. in each step we use refinement indicators to determine which elements are refined, and if the polynomial degrees are held constant, we speak of an **adaptive h-method**.

If we create a sequence $S_\ell = S_0(\mathcal{T}_\ell, \mathbf{h}_\ell, \mathbf{p}_\ell)$ of discrete spaces by only increasing the polynomial degree in each step, e.g. $S_{\ell+1} = S_0(\mathcal{T}_\ell, \mathbf{h}_\ell, 2\mathbf{p}_\ell + 1)$, we call this a **uniform p-method**.

If in each refinement step first the polynomial degrees are increased uniformly and then each element is halved, we call this a **uniform hp-method**.
If in each refinement step we use one or more error indicators to single out elements that are halved and elements for which the polynomial degree is increased, we call this an **adaptive hp-method**.
Finally, we introduce a special type of mesh, namely a geometric mesh. The idea is to refine the mesh geometrically towards vertices where the solution of the integral equation has singularities. For the slit $\Gamma = [-1, 1] \times \{0\}$, we define for $\rho \in (0, 1)$

$$\begin{aligned} x_0 &= -1, \\ x_k &= -1 + \rho^{n-k}, \quad k = 1, \dots, n-1, \\ x_n &= 0, \\ x_{n+k} &= 1 - \rho^k, \qquad k = 1, \dots, n-1, \\ x_{2n} &= 1. \end{aligned}$$

Then, the mesh $\mathcal{T} = \{T_k = [x_{k-1}, x_k],\ k = 1, \dots, 2n\}$ is a geometric mesh for the slit. We obtain a **geometric hp-method** by using linearly growing polynomial degree vectors that start with $p = 0$ at the singularity. For the geometric mesh on the slit we could use the polynomial degree vector $p_k = k - 1$, $p_{k+n} = n - k$, $k = 1, \dots, n$. Figure 1.1 shows geometric grids and polynomial degree vectors for the slit and the boundary of the L-shape that can be used for geometric hp-methods.

1.5. A Priori Error Analysis

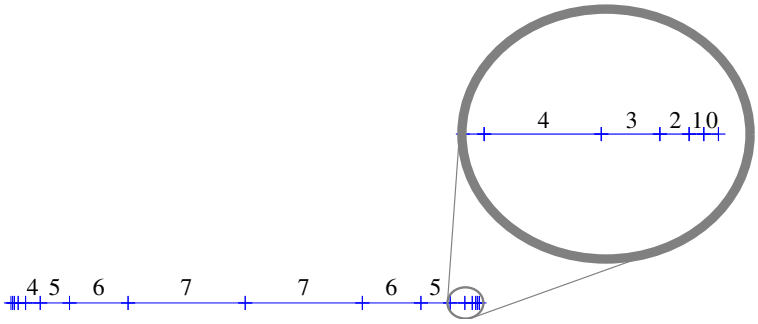
In this section, we give a priori error estimates for the numerical solutions of Symm's integral equation (1.11), the hypersingular integral equation (1.15), and the mixed problem (1.21).
We start with abstract a priori estimates for Galerkin methods. Let X be a Hilbert Space, $A : X \to X'$, and $f \in X'$. We consider the variational formulation

$$\langle Au, v\rangle = \langle f, v\rangle \quad \forall v \in X \tag{1.31}$$

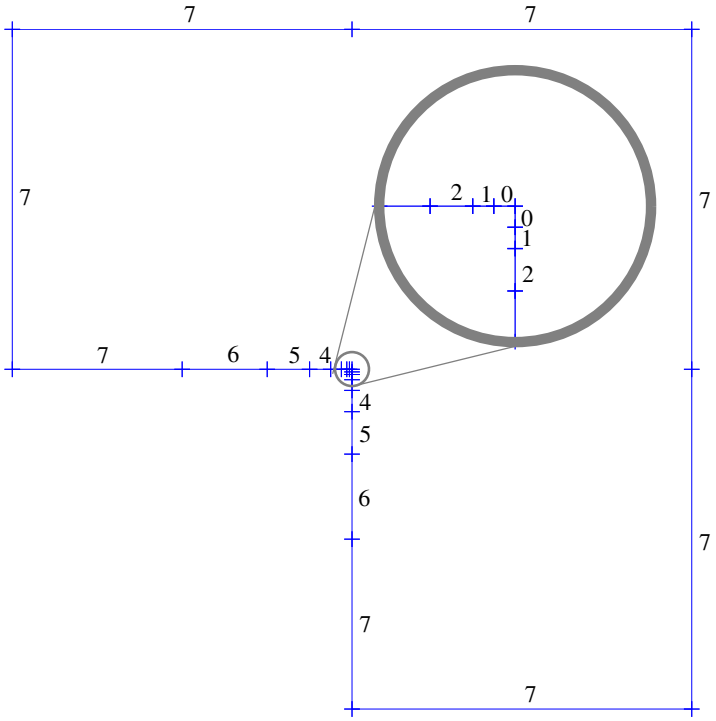
and the discrete variational formulation

$$\langle Au_h, v_h\rangle = \langle f, v_h\rangle \quad \forall v_h \in X_h, \tag{1.32}$$

where X_h is a finite dimensional subspace of X. The following lemma, called the Cea Lemma and stated in e.g. [Ste03], gives an estimate for the error between the solutions of (1.31) and (1.32).



(a) This figure shows a geometric grid on the slit with linearly growing polynomial degree vector. The numbers indicate the polynomial degree on each element. We zoom in on the right corner.



(b) This figure shows a geometric grid on the L-shape with linearly growing polynomial degree vector. The numbers indicate the polynomial degree on each element. We zoom in on the reentrant corner (see circles).

Figure 1.1.: Examples of boundary grids and polynomial degree vectors that can be used for geometric hp-methods.

Theorem 1.5.1. (Cea Lemma) *Let $A : X \to X'$ be bounded and X-elliptic and let $X_h \subset X$. Then, the variational formulation (1.32) has a unique solution $u_h \in X_h$, and there holds for the error between the solution u of (1.31) and the Galerkin solution u_h of (1.32)*

$$\|u - u_h\|_X \leq C \inf_{v_h \in X_h} \|u - v_h\|_X.$$

In practice, the variational formulation (1.32) can never be solved exactly, e.g. due to rounding errors in the computations. Furthermore, in Chapter 5 we present a data compression scheme for the single layer Galerkin matrix where we create low rank approximations of sub-blocks of the Galerkin matrix and neglect matrix entries of small magnitude. Working with the compressed Galerkin matrix corresponds to dealing with a perturbed variational formulation which reads

$$\langle A_h u_h, v_h \rangle = \langle f_h, v_h \rangle \quad \forall v_h \in X_h. \tag{1.33}$$

A generalization of the Cea Lemma is the Strang Lemma, which also takes approximation errors into account.

Theorem 1.5.2. (First Strang Lemma) *Let $A : X \to X'$ be bounded and X-elliptic. Let $u \in X$ be the solution of (1.31) and $\widetilde{u}_h \in X_h$ be the solution of (1.33). Then there holds*

$$\|u - \widetilde{u}_h\|_X \leq c \Bigg(\inf_{v_h \in X_h} \Bigg[\|u - v_h\|_X + \sup_{w_h \in X_h} \frac{|\langle A_h v_h, w_h \rangle - \langle A v_h, w_h \rangle|}{||w_h||_X} \Bigg] + \sup_{v_h \in X_h} \frac{|\langle f_h, v_h \rangle - \langle f, v_h \rangle|}{||v_h||_X} \Bigg). \tag{1.34}$$

Note that the Cea Lemma and Strang Lemma both hold for the three integral equations (1.11), (1.15), and (1.21), since the boundary integral operators are elliptic and bounded on the respective function spaces. Therefore, one obtains quasi optimal convergence when using a conforming Galerkin scheme for the solution of the integral equations. By analyzing the approximation properties of the respective discrete test and ansatz spaces, a priori error estimates can be obtained.

We start with results for quasi-uniform meshes and uniform polynomial degrees and use the short notations $S^0_{h,p}(\Gamma) := S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ as well as $S^1_{h,q}(\Gamma) := S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$. The following result for the mixed problem is proven in [StSu91].

Proposition 1.5.3. ([StSu91, Theorem 3.5]) *Let $(\psi_N, \phi_D) \in \mathcal{H}$ be the solution of (1.21) with $\psi_D \in H^s(\Gamma_D)$ and $\phi_N \in H^{s-1}(\Gamma_N)$, and let $(\psi^p_h, \phi^p_h) \in X_h = S^1_{h,p}(\Gamma_N) \times S^0_{h,p-1}(\Gamma_D)$ be the numerical solution of (1.22). Then for $p \in \mathbb{N}$ large enough, $h > 0$ small enough, and s large enough and for all $\varepsilon > 0$, there exists a constant $C = C(\varepsilon)$ independent of h and p such that*

$$\|\psi_N - \psi^p_h\|_{\widetilde{H}^{1/2}(\Gamma_N)} + \|\phi_D - \phi^p_h\|_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C\, h^{\alpha}\, p^{-2\alpha+\varepsilon} \left(\|\psi_D\|_{H^s(\Gamma_D)} + \|\phi_N\|_{H^{s-1}(\Gamma_N)} \right). \tag{1.35}$$

Here, $\alpha = \min\{\frac{\pi}{\beta_j},\ j = 1, \ldots \mathcal{N}\}$, where $\beta_j \neq \pi$ are the interior angles of the domain Ω.

Remark 1.5.4. (i) Similar estimates hold for Symm's integral equation (1.11) and the hypersingular integral equation (1.15), and the proofs can be found in [GuHe06, Theorem 4.2 and 4.6].

(ii) For the proof of such a priori estimates one proceeds as follows. First, it is shown that the exact solution behaves like x^α plus a smooth function at the corners of the boundary Γ, where α depends on the interior angle at the respective corner. Then one derives estimates for the approximation quality of x^α in the respective discrete spaces. Combining both results gives the error estimate.

(iii) For Symm's integral equation $\mathcal{V}\phi = f$ or the hypersingular integral equation $\mathcal{W}\psi = g$ with $f \in H^{1/2}(\Gamma)$ and $g \in H^{-1/2}(\Gamma)$, it can be shown (see [Heu92, Chapter 6]) that the solution behaves like x^α near corners. However, here α depends on the interior and the exterior angle at the respective corner. The reason for this is that f (or g) could be the right-hand side coming from an interior or an exterior Dirichlet (or Neumann) problem.

The estimate from Proposition 1.5.3 shows that a uniform p-version yields twice the rate of convergence of a uniform h-version. For hp-methods where a geometric grid is combined with an appropriate choice of the polynomial degrees, exponential convergence can be achieved [HS96].

Proposition 1.5.5. ([HS96]]) *Let $\mathcal{T}$ be a geometric grid on Γ and the polynomial degree function $\mathbf{p}$ be suitably chosen. If $\phi_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is the numerical solution of (1.13) and ϕ is the solution of Symm's integral equation (1.11), then there holds*

$$\|\phi - \phi_h^p\|_{H^{-1/2}(\Gamma)} \leq C\, e^{-b\sqrt{\mathcal{N}}}, \tag{1.36}$$

where $\mathcal{N} = \dim(S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0))$ and the constants C and b depend on the grading parameter ρ but not on $\mathcal{N}$.

Remark 1.5.6. In [HS96], a similar estimate for the hypersingular equation is proven.

2. Efficient and Stable Evaluation of Legendre Functions and Their Integrals

For our implementation of hp-BEM, we use Legendre polynomials and their antiderivatives as basis functions for the ansatz and test spaces. This choice is motivated by the following two observations. First, the condition numbers of Galerkin matrices grow linearly with respect to the polynomial degree p of the ansatz space as compared to exponential growth for the monomial basis functions (see Section 3.2). Second, with this approach we can evaluate potentials and compute Galerkin entries using simple recurrence relations (see Sections 3.3 and 3.4). In more detail, the evaluation of potentials can be reduced to the evaluation of basic integrals that are closely related to the associated Legendre functions of the second kind. Furthermore, the calculation of the entries in the Galerkin matrices can be reduced to the evaluation of certain integrals of associated Legendre functions.
Therefore, before we discuss the evaluation of potentials and the calculation of Galerkin entries, we devote a chapter to these basic integrals. We derive recurrence relations for these basic integrals and develop algorithms for the stable and efficient evaluation of the recurrence relations. The implementation of these algorithms serves as a foundation for the stable and efficient evaluation of potentials and Galerkin entries in hp-BEM.
In the first section, we introduce Legendre polynomials and Lobatto shape functions, and state some relevant properties. Associated Legendre functions of the first and second kind and their properties are introduced in the second section. The third section introduces modified associated Legendre functions of the second kind $\widetilde{Q}_k^m$ and R_k^m. These are the basic integrals to which we can reduce the evaluation of potentials. We derive recurrence relations, state asymptotic estimates, and give a detailed description of the stable and efficient evaluation of $\widetilde{Q}_k^m$, $m \in \{-2, -1, 0, 1\}$, and R_k^m, $m \in \{0, 1\}$, which are the cases relevant to hp-BEM.
Finally, we define integrals of the functions $\widetilde{Q}_k^m$ and R_k^m that are the foundation of the calculation of Galerkin entries. Again, we derive recurrence relations for these integrals and discuss their accurate computation.

In the subsequent sections we will see that ellipses play an important role in asymptotic estimates and series expansions of modified associated Legendre functions. There is a conformal complex mapping, namely the Joukowski transform, that maps points $z \in \mathbb{C}$, which are located

on a concentric circle with radius greater than one, to an ellipse with foci ± 1. The mapping is given by

$$J : \{w \in \mathbb{C} \ : \ |w| > 1\} \to \mathbb{C} \setminus [-1, 1] \tag{2.1a}$$

$$w \mapsto z := \frac{1}{2}\left(w + \frac{1}{w}\right). \tag{2.1b}$$

The inverse mapping is given by

$$J^{-1} : \mathbb{C} \setminus [-1, 1] \to \{w \in \mathbb{C} \ : \ |w| > 1\} \tag{2.2a}$$

$$z \mapsto w := z + \sqrt{z^2 - 1}. \tag{2.2b}$$

Remark 2.0.7. The complex square root $\sqrt{z^2 - 1}$ has to be chosen such that is has the same sign as $\mathrm{Re}(z)$. For $\mathrm{Re}(z) = 0$, the square root has the same sign as $\mathrm{Im}(z)$. In the following we will assume this choice of $\sqrt{z^2 - 1}$.

An example of the mapping J is displayed in Figure 2.1

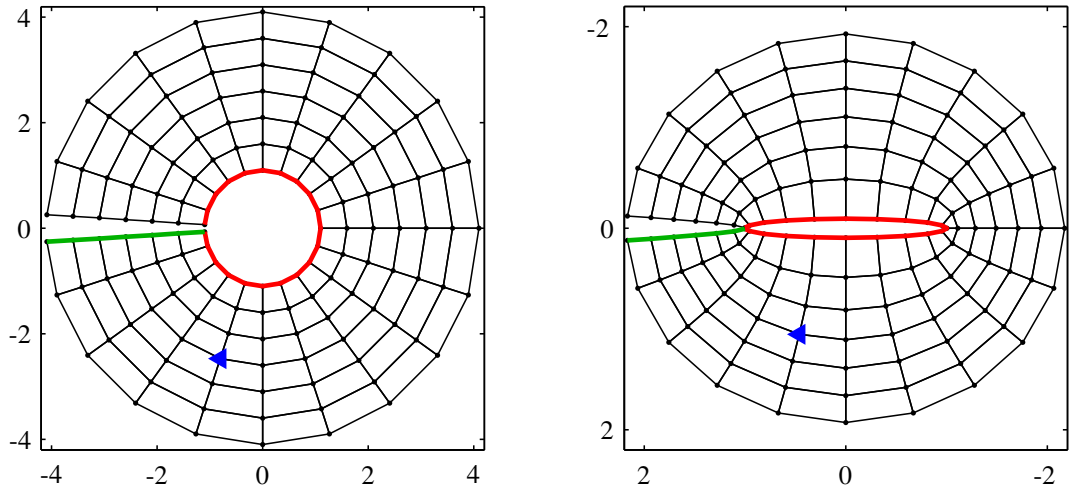


Figure 2.1.: Conformal mapping J of a circular grid to an elliptic grid. The blue triangle (green line, red circle) in the left picture is mapped to the blue triangle (green curve, red ellipse) in the right picture.

2.1. Legendre Polynomials and Lobatto Shape Functions

In this section we introduce Legendre polynomials and Lobatto shape functions. These functions serve as basis functions of the test and ansatz spaces for our implementation of hp-BEM. This choice of basis functions is motivated from hp-FEM, where scaled Legendre polynomials and scaled Lobatto shape functions, also known as Babushka-Shen functions in the FEM context

(see [CNV12]), are used.
We define the Legendre polynomials via their recurrence relation.

Definition 2.1.1. For $z \in \mathbb{C}$ and $k \in \mathbb{N}_0$, the Legendre polynomials P_k are defined by

$$\begin{aligned} P_0(z) &= 1, \quad P_1(z) = z, \\ (k+1)\, P_{k+1}(z) &= (2k+1)\, zP_k(z) - k\, P_{k-1}(z). \end{aligned} \tag{2.3}$$

The following Lemma summarizes some important properties of the Legendre polynomials.

Lemma 2.1.2. *The Legendre polynomials have the following properties.*

(i) ***(Rodrigues Formula)*** *For $k \in \mathbb{N}_0$ and $z \in \mathbb{C}$, there holds*

$$P_k(z) = \frac{1}{2^k k!} \frac{d^k}{dz^k}(1 - z^2). \tag{2.4}$$

(ii) ***(Orthogonality)*** *The Legendre polynomials are orthogonal with respect to the L^2 scalar product on $(-1, 1)$; i.e. there holds*

$$\int_{-1}^{1} P_k(t) P_\ell(t)\, dt = \delta_{k\ell} \frac{2}{2k+1}, \qquad k, \ell \in \mathbb{N}_0,$$

where $\delta_{k\ell}$ is the Kronecker Delta.

(iii) ***(Antiderivative)*** *For $k > 1$, the antiderivative of the Legendre polynomials can be computed by*

$$\int_{-1}^{t} P_k(s)\, ds = \frac{P_{k+1}(t) - P_{k-1}(t)}{2k+1}. \tag{2.5}$$

(iv) ***(Values at ± 1)*** *For $k \in \mathbb{N}_0$, there holds*

$$P_k(1) = 1, \qquad P_k(-1) = (-1)^k. \tag{2.6}$$

(v) ***(Integral)*** *For $k \in \mathbb{N}_0$, there holds*

$$\int_{-1}^{1} P_k(t)\, dt = \begin{cases} 2, & k = 0 \\ 0, & k \in \mathbb{N}. \end{cases} \tag{2.7}$$

Proof. The Rodrigues Formula (i) is proven in [Mac67, p. 82]. Using the Rodrigues Formula, the orthogonality properties (ii) can be proven (see [Mac67, pp. 85-86]). Part (iii) follows from the recurrence relation $(2k+1)P_k(t) = P_{k+1}(t)' - P_{k-1}(t)'$, which is shown for example in [Mac67, p. 91]. Part (iv) can be derived directly from the recurrence relation (2.3), and (v) follows from (ii) with $\ell = 0$. □

We introduce Lobatto shape functions as antiderivatives of Legendre polynomials.

Definition 2.1.3. For $k \in \mathbb{N}$, $k \geq 3$, and $t \in [-1, 1]$, we define the Lobatto shape functions

$$\begin{aligned} N_1(t) &= \frac{1-t}{2}, \quad N_2(t) = \frac{1+t}{2}, \\ N_k(t) &= \int_{-1}^{t} P_{k-2}(s)\, ds. \end{aligned} \tag{2.8}$$

The following properties follow directly from the properties of the Legendre polynomials.

Lemma 2.1.4. *Let $t \in [-1, 1]$. The Lobatto shape functions have the following properties.*

*(i) (**Recurrence Relation (1)**) For $k \geq 3$, there holds*

$$N_k(t) = \frac{1}{2k-3} \left(P_{k-1}(t) - P_{k-3}(t) \right). \tag{2.9}$$

*(ii) (**Recurrence Relation (2)**) With the additional initial values*

$$N_3(t) = \frac{1}{2}(t^2 - 1), \quad N_4(t) = \frac{1}{2} t (t^2 - 1), \tag{2.10}$$

there holds the three-term recurrence relation for $k \geq 4$

$$k N_{k+1}(t) = (2k-3)\, t\, N_k(t) - (k-3)\, N_{k-1}(t). \tag{2.11}$$

*(iii) (**Values at** ± 1) For $k \geq 3$, there holds*

$$N_k(\pm 1) = 0. \tag{2.12}$$

Proof. Lemma 2.1.2 (iii) implies the first statement. The second statement can be derived directly from the recurrence relation for the Legendre polynomials (Lemma 2.1.2 (ii)). Part (i) and Lemma 2.1.2 (v) yield the third statement. □

Remark 2.1.5. As mentioned before, Legendre polynomials and Lobatto shape functions have been used before for both one-dimensional hp-FEM and two-dimensional hp-BEM; see e.g. [Ste96], [CNV12] or [BS91]. However, in these publications the Lobatto shape functions, which are also called Babuska-Shen basis in the context of hp-FEM (see [CNV12]), are defined by

$$N_k(t) = \frac{1}{\|P_{k-2}\|} \int_{-1}^{t} P_{k-2}(s)\, ds$$

and have the property

$$\int_{-1}^{1} N_k'(t) N_\ell'(t)\, dt = \delta_{k\ell}.$$

Our choice of the Lobatto shape functions without the factor $1/\|P_{k-2}\|$ results in recurrence relations with integer coefficients (see Equation (2.11)) and therefore is preferred here.

2.2. Associated Legendre Functions

We consider the associated Legendre differential equation

$$\frac{d}{dz}\left[(1-z^2)\frac{du}{dz}\right]+\left[k(k+1)-\frac{m^2}{1-z^2}\right]u=0 \tag{2.13}$$

with $k \in \mathbb{N}_0$, $m \in \mathbb{Z}$, and $z \in \mathbb{C}$. The two independent sets of solutions are called associated Legendre functions of the first kind, denoted $P_k^m(z)$, and of the second kind, denoted $Q_k^m(z)$, respectively. Note that for $m = 0$, we obtain the Legendre differential equation, and the functions P_k^0 are the Legendre polynomials defined in (2.3). For $m \in \mathbb{N}$, we have the following representations.

Proposition 2.2.1. *(i) Let m be a positive integer and $z \in C \setminus [-1,1]$. Then, there holds*

$$P_k^m(z)=(z^2-1)^{\frac{m}{2}}\frac{d^m P_k(z)}{dz^m}, \tag{2.14}$$

$$Q_k^m(z)=(z^2-1)^{\frac{m}{2}}\frac{d^m Q_k(z)}{dz^m}, \tag{2.15}$$

where the arguments of $z+1$ and $z-1$ are set to zero if z is real and greater than one.

(ii) Let $m \in \mathbb{N}$, $k \geq m$, and $z \in C \setminus [-1,1]$. Then, there holds

$$Q_k^{-m}(z)=(z^2-1)^{-\frac{m}{2}}\int_z^\infty\int_\xi^\infty\cdots\int_\xi^\infty Q_k(\xi)(d\xi)^m. \tag{2.16}$$

Proof. See [Hob55, pp. 89-90 and p. 115]. □

In the following we are interested in the associated Legendre functions of the second kind Q_k^m. Therefore, we summarize some properties in the following lemma.

Lemma 2.2.2. *The associated Legendre functions of the second kind have the following properties.*

(i) Let $k, m \in \mathbb{N}$, $m < k$, and $z \in \mathbb{C}$ with $|z| > 1$. Then there holds

$$Q_k^{-m}(z)=(-1)^m\frac{(k-m)!}{(k+m)!}\,Q_k^m(z). \tag{2.17}$$

(ii) Let $z \in \mathbb{C} \setminus [-1,1]$ and $m \in \mathbb{N}_0$. Then there holds

$$Q_k^m(z)=(-1)^m(z^2-1)^{\frac{m}{2}}\frac{m!}{2}\int_{-1}^1\frac{P_k(t)}{(z-t)^{m+1}}\,dt. \tag{2.18}$$

The special case

$$Q_k^0(z)=\int_{-1}^1\frac{P_k(t)}{z-t}\,dt \tag{2.19}$$

is called Neumann's formula.

(iii) Let $k \in \mathbb{N}_0$ and $m \in \mathbb{Z}$. Let $z \in \mathbb{C}$ with $|arg(z-1)| < \pi$ and $w = J^{-1}(z) = z+\sqrt{z^2-1}$. Then there holds

$$Q_k^m(z) = e^{m\pi \mathrm{i}} 2^m \sqrt{\pi}\, \frac{\Gamma(k+m+1)(z^2-1)^{\frac{m}{2}}}{\Gamma\left(k+\frac{3}{2}\right) w^{k+m+1}} \cdot F\left(\frac{m+2}{2}, k+m+1; \frac{2k+3}{2}; \frac{1}{w^2}\right), \tag{2.20}$$

where Γ is the Gamma function and F is the hypergeometric function.

(iv) Let z and w be defined as in (iii). Then there holds

$$Q_k^{-1}(z) = -\frac{\sqrt{\pi}}{2} \frac{\Gamma(k)(z^2-1)^{-\frac{1}{2}}}{\Gamma\left(k+\frac{3}{2}\right) w^k} \cdot F\left(-\frac{1}{2}, k; k+\frac{3}{2}; \frac{1}{w^2}\right), \tag{2.21}$$

$$Q_k^{-2}(z) = \frac{\sqrt{\pi}}{4} \frac{\Gamma(k-1)(z^2-1)^{-1}}{\Gamma\left(k+\frac{3}{2}\right) w^{k-1}} \cdot F\left(-\frac{3}{2}, k-1; k+\frac{3}{2}; \frac{1}{w^2}\right). \tag{2.22}$$

Proof. Part (i) can be found in [GR81, p. 403] or in [Hob55, p. 196]. For part (ii), see [Hob55, p. 116]. Part (iii) is stated in [GR81], and part (iv) is a direct consequence of (iii). □

2.3. Modified Associated Legendre Functions of the Second Kind

We will see in later chapters that the computation of potentials in hp-BEM can be reduced to the evaluation of integrals closely related to the associated Legendre functions of the second kind. We first define these integrals and derive recurrence relations based on the recurrence relation for the Legendre polynomials (2.3). These recurrence relations will be used for the stable and efficient evaluation of the potentials even for high polynomial degrees.
Also, the calculation of Galerkin entries can be traced back to integrals of the functions defined in this section. We will use asymptotic estimates, which we derive in the second subsection, to estimate the magnitude of the Galerkin entries. In the third subsection we examine the recurrence relations derived the first subsection. We show that evaluating the recurrence relations in the forward direction is generally unstable and provide an alternative stable algorithm. In the last subsection we discuss the implementation in detail.

2.3.1. Definition and Recurrence Relations

We start with the definition of the basic integrals $\widetilde{Q}_k^m(z)$ and $R_k^m(z)$.

Definition 2.3.1. (i) For $z \in \mathbb{C} \setminus [-1,1]$ and $k, m \in \mathbb{N}_0$, we define

$$\widetilde{Q}_k^m(z) := \int_{-1}^{1} \frac{P_k(t)}{(z-t)^{m+1}}\, dt\,. \tag{2.23}$$

For $x \in [-1,1]$, we define

$$\widetilde{Q}_k^m(x) := \frac{1}{2} \lim_{\varepsilon \to 0} \left(\widetilde{Q}_k^m(x + \mathrm{i}\varepsilon) + \widetilde{Q}_k^m(x - \mathrm{i}\varepsilon) \right).$$

(ii) For $k \in \mathbb{N}_0$, $m \in \mathbb{N}$ and $z \in \mathbb{C} \setminus (-\infty, 1]$, we define

$$\widetilde{Q}_k^{-m}(z) := \int_{-1}^{1} P_k(t)\,(z-t)^{m-1}\,\log(z-t)\,dt. \tag{2.24}$$

For $x \in (-\infty, 1]$, we define

$$\widetilde{Q}_k^{-m}(x) := \frac{1}{2} \lim_{\varepsilon \to 0} \left(\widetilde{Q}_k^{-m}(x + \mathrm{i}\varepsilon) + \widetilde{Q}_k^{-m}(x - \mathrm{i}\varepsilon) \right).$$

(iii) For $z \in \mathbb{C} \setminus [-1,1]$, $m \in \mathbb{N}_0$ and $k \in \mathbb{N}$, we define

$$R_k^m(z) := \int_{-1}^{1} \frac{N_k(t)}{(z-t)^{m+1}}\,dt. \tag{2.25}$$

For $x \in [-1,1]$, we define

$$R_k^m(x) := \frac{1}{2} \lim_{\varepsilon \to 0} \left(R_k^m(x + \mathrm{i}\varepsilon) + R_k^m(x - \mathrm{i}\varepsilon) \right).$$

(iv) For $k \in \mathbb{N}$, $m \in \mathbb{N}$ and $z \in \mathbb{C} \setminus (-\infty, 1]$, we define

$$R_k^{-m}(z) := \int_{-1}^{1} N_k(t)\,(z-t)^{m-1}\,\log(z-t)\,dt. \tag{2.26}$$

For $x \in (-\infty, 1]$, we define

$$R_k^{-m}(x) := \frac{1}{2} \lim_{\varepsilon \to 0} \left(R_k^{-m}(x + \mathrm{i}\varepsilon) + R_k^{-m}(x - \mathrm{i}\varepsilon) \right).$$

Remark 2.3.2. (i) For the evaluation of the potentials $\widetilde{\mathcal{V}}$, $\widetilde{\mathcal{K}}$, $\widetilde{\mathcal{W}}$, and $\widetilde{\mathcal{K}}'$ defined in (1.3) - (1.6), the functions $\widetilde{Q}_k^{-1}$, $\widetilde{Q}_k^0$, R_k^0, and R_k^1 are required (see Section 3.3). For the calculation of Galerkin entries we also need $\widetilde{Q}_k^{-2}$.

(ii) The potentials for the Lamé equation can also be written as linear combinations of the integrals introduced in Definition 2.3.1 [Ban12].

(iii) For $m > 0$, there holds

$$(m+1)\,\widetilde{Q}_k^{m+1}(z) = -\frac{d}{dz}\widetilde{Q}_k^m(z),$$

$$(m+1)\,\widetilde{Q}_k^{-(m+1)}(z) = \frac{d}{dz}\widetilde{Q}_k^{-(m+2)}(z) + \underbrace{\int_{-1}^{1} P_k(t)(z-t)^m\,dt}_{r_{k,m}}.$$

Due to the orthogonality of the Legendre polynomials, the term $r_{k,m}$ vanishes for $k \geq m$.

(iv) From the definition of $\widetilde{Q}_k^m(x)$ ($m \in \mathbb{N}_0$), we obtain for $x \in (-1,1)$

$$\begin{aligned}
\widetilde{Q}_k^m(x) &= \frac{1}{2} \lim_{\varepsilon \to 0} \left(\widetilde{Q}_k^m(x + \mathrm{i}\varepsilon) + \widetilde{Q}_k^m(x - \mathrm{i}\varepsilon) \right) \\
&= \frac{1}{2} \int_{-1}^{1} P_k(t) \lim_{\varepsilon \to 0} \left[\frac{1}{(x - t + \mathrm{i}\varepsilon)^{m+1}} + \frac{1}{(x - t + \mathrm{i}\varepsilon)^{m+1}} \right] \, dt \\
&= \frac{1}{2} \int_{-1}^{1} P_k(t) \lim_{\varepsilon \to 0} \left[\frac{(x - t + \mathrm{i}\varepsilon)^{m+1} + (x - t - \mathrm{i}\varepsilon)^{m+1}}{(x - t + \mathrm{i}\varepsilon)^{2(m+1)}} \right] \, dt \\
&= \frac{1}{2} \int_{-1}^{1} P_k(t) \, \frac{2(x-t)^{m+1}}{(x-t)^{2(m+1)}} \, dt \\
&= \int_{-1}^{1} \frac{P_k(t)}{|x-t|^{m+1}} \, dt.
\end{aligned}$$

For $x = \pm 1$, the functions $\widetilde{Q}_k^m$ have singularities. For the implementation, we set $\widetilde{Q}_k^m(\pm 1) = 0$. The integrals R_k^m are treated in the same way as $\widetilde{Q}_k^m$.

(v) To analyze the behavior of $\widetilde{Q}_k^{-m}(x)$ ($m \in \mathbb{N}$) for $x \in (-1,1)$, we set $(x + \mathrm{i}\varepsilon - t) = r\, e^{\mathrm{i}\varphi}$. Then there holds $(x - \mathrm{i}\varepsilon - t) = r\, e^{\mathrm{i}(-\varphi)}$, where r and φ depend on t and ε. Furthermore, we obtain

$$\begin{aligned}
\widetilde{Q}_k^{-m}(x + \mathrm{i}\varepsilon) &= \int_{-1}^{1} P_k(t) \, r^{m-1} (\cos((m-1)\varphi) + \mathrm{i}\sin((m-1)\varphi)) \, [\log(r) + \mathrm{i}\varphi] \, dt \\
&= \int_{-1}^{1} P_k(t) \, r^{m-1} \big(\log(r) \cos[(m-1)\varphi] - \varphi \sin[(m-1)\varphi] \big) \, dt \\
&\quad + \mathrm{i} \int_{-1}^{1} P_k(t) \, r^{m-1} \big(\varphi \cos[(m-1)\varphi] + \log(r) \sin[(m-1)\varphi] \big] \, dt.
\end{aligned}$$

Substituting φ by $-\varphi$ gives

$$\operatorname{Re} \widetilde{Q}_k^{-m}(x + \mathrm{i}\varepsilon) = \operatorname{Re} \widetilde{Q}_k^{-m}(x - \mathrm{i}\varepsilon) \quad \text{and} \quad \operatorname{Im} \widetilde{Q}_k^{-m}(x + \mathrm{i}\varepsilon) = -\operatorname{Im} \widetilde{Q}_k^{-m}(x - \mathrm{i}\varepsilon).$$

Therefore, from the definition of $\widetilde{Q}_k^{-m}(x)$ for $x \in (-\infty, 1)$, it follows that the real part of $\widetilde{Q}_k^{-m}(z)$ is continuous for all $z \in \mathbb{C}$ and $\operatorname{Im} \widetilde{Q}_k^{-m}(x) = 0$ for $x \in (-\infty, 1)$.
For $x \in (-\infty, 1)$, with $\varphi = \varphi(\varepsilon, t) = \arg(x - t + \mathrm{i}\varepsilon)$ there holds

$$\begin{aligned}
\operatorname{Re} \widetilde{Q}_k^{-m}(x) &= \lim_{\varepsilon \to 0} \int_{-1}^{1} P_k(t) \, |x - t + \mathrm{i}\varepsilon|^{m-1} \\
&\qquad \big(\log |x - t + \mathrm{i}\varepsilon| \cos[(m-1)\varphi] - \varphi \sin[(m-1)\varphi] \big) \, dt \\
&= \int_{-1}^{1} P_k(t) \, |x - t|^{m-1} \big(\log |x - t| \cos[(m-1)\pi] - \pi \sin[(m-1)\pi] \big) \, dt \\
&= \int_{-1}^{1} P_k(t) \, |x - t|^{m-1} \log |x - t| (-1)^{m-1} dt \\
&= \int_{-1}^{1} P_k(t) \, (x - t)^{m-1} \log |x - t| dt.
\end{aligned}$$

Thus, the definition of $\widetilde{Q}_k^{-m}$ for $x \in (-\infty, -1)$ is equivalent to considering $\log|x-t|$ in the integrals.

For $x \in [-1,1]$, we have to take the Cauchy principal value to evaluate the integrals. Figure 2.2 shows surface plots for some of the integrals.

(vi) From the relation (2.9) between the Legendre polynomials and the Lobatto shape functions, it follows directly that, for $m \in \mathbb{Z}$, there holds

$$R_1^m = \frac{\widetilde{Q}_0^m - \widetilde{Q}_1^m}{2}, \qquad R_2^m = \frac{\widetilde{Q}_0^m + \widetilde{Q}_1^m}{2}, \quad \text{and} \qquad R_k^m = \frac{\widetilde{Q}_{k-1}^m - \widetilde{Q}_{k-3}^m}{2k-3}, k \geq 3.$$

There exist three-term recurrence relations for the integrals $\widetilde{Q}_k^m$ and R_k^m. We start with $\widetilde{Q}_k^m$.

Proposition 2.3.3. *Let $m \in \mathbb{Z}$, $k \in \mathbb{N}$ with $k \geq |m|$ if $m < 0$, and $z \in \mathbb{C} \setminus \{-1, 1\}$. Then there holds*

$$(k-m+1)\widetilde{Q}_{k+1}^m(z) = (2k+1)\, z\, \widetilde{Q}_k^m(z) - (k+m)\, \widetilde{Q}_{k-1}^m(z). \tag{2.27}$$

Proof. We start by proving the statement for $\widetilde{Q}_k^{-m}$, $m \in \mathbb{N}$. For $k \in \mathbb{N}$, $k \geq m$, and $z \in \mathbb{C}$, there holds

$$\begin{aligned}
\widetilde{Q}_{k+1}^{-m}(z) &= \int_{-1}^{1} P_{k+1}(t)\, (z-t)^{m-1} \log(z-t)\, dt \\
&= \frac{2k+1}{k+1} \int_{-1}^{1} t P_k(t)\, (z-t)^{m-1} \log(z-t)\, dt \\
&\quad - \frac{k}{k+1} \int_{-1}^{1} P_{k-1}(t)\, (z-t)^{m-1} \log(z-t)\, dt \\
&= \frac{2k+1}{k+1} \int_{-1}^{1} (t-z)\, P_k(t)\, (z-t)^{m-1} \log(z-t)\, dt \\
&\quad + z\frac{2k+1}{k+1} \int_{-1}^{1} P_k(t)\, (z-t)^{m-1} \log(z-t)\, dt - \frac{k}{k+1} \widetilde{Q}_{k-1}^{-m}(z) \\
&= -\frac{2k+1}{k+1} \int_{-1}^{1} P_k(t)\, (z-t)^{m} \log(z-t)\, dt + \frac{2k+1}{k+1}\, z\, \widetilde{Q}_k^{-m}(z) - \frac{k}{k+1} \widetilde{Q}_{k-1}^{-m}(z).
\end{aligned}$$

Using integration by parts and the formula for the antiderivative of the Legendre polynomials (2.5), we obtain

$$\begin{aligned}
&\frac{2k+1}{k+1} \int_{-1}^{1} P_k(t)\, (z-t)^{m} \log(z-t)\, dt \\
&\qquad = \frac{2k+1}{k+1} \Bigg(\left[\int_{-1}^{t} P_k(s)\, ds\ (z-t)^m \log(z-t) \right]_{t=-1}^{1} \\
&\qquad\qquad + m \int_{-1}^{1} \frac{P_{k+1}(t) - P_{k-1}(t)}{2k+1} (z-t)^{m-1} \log(z-t)\, dt \\
&\qquad\qquad - \int_{-1}^{1} \frac{P_{k+1}(t) - P_{k-1}(t)}{2k+1} (z-t)^{m-1}\, dt \Bigg).
\end{aligned}$$

The first term equals zero due to the property $N_k(\pm 1) = 0$. Because the Legendre polynomials are orthogonal with respect to the L^2 scalar product on $(-1, 1)$, for $m \leq k$, the last term vanishes. Plugging in the definition of $\widetilde{Q}_k^{-m}$ in the second term yields

$$\frac{2k+1}{k+1} \int_{-1}^{1} (t-z) P_k(t)\, (z-t)^{m-1} \log(z-t)\, dt = \frac{m}{2k+1} \left[\widetilde{Q}_{k+1}^{-m} - \widetilde{Q}_{k-1}^{-m} \right].$$

Overall, we obtain

$$\widetilde{Q}_{k+1}^{-m}(z) = \frac{-m}{k+1} \left[\widetilde{Q}_{k+1}^{-m} - \widetilde{Q}_{k-1}^{-m} \right] + z \frac{2k+1}{k+1}\, z\, \widetilde{Q}_k^{-m}(z) - \frac{k}{k+1} \widetilde{Q}_{k-1}^{-m}(z),$$

which is equivalent to

$$\frac{k+m+1}{k+1} \widetilde{Q}_{k+1}^{-m}(z) = \frac{2k+1}{k+1}\, z\, \widetilde{Q}_k^{-m}(z) - \frac{k-m}{k+1} \widetilde{Q}_{k-1}^{-m}(z)$$

and

$$(k+m+1) \widetilde{Q}_{k+1}^{-m}(z) = (2k+1)\, z\, \widetilde{Q}_k^{-m}(z) - (k-m)\, \widetilde{Q}_{k-1}^{-m}(z).$$

For $\widetilde{Q}_k^m$, $m \in \mathbb{N}_0$, we again use the recurrence relation for the Legendre polynomials and obtain

$$\begin{aligned}
\widetilde{Q}_{k+1}^m(z) &= \int_{-1}^{1} \frac{P_{k+1}(t)}{(z-t)^{m+1}}\, dt = \frac{2k+1}{k+1} \int_{-1}^{1} t \frac{P_k(t)}{(z-t)^{m+1}}\, dt - \frac{k}{k+1} \int_{-1}^{1} \frac{P_{k-1}(t)}{(z-t)^{m+1}}\, dt \\
&= \frac{2k+1}{k+1} \left(- \int_{-1}^{1} \frac{P_k(t)}{(z-t)^m}\, dt + z \int_{-1}^{1} \frac{P_k(t)}{(z-t)^{m+1}}\, dt \right) + \frac{k}{k+1} \widetilde{Q}_{k-1}^m \\
&= \frac{2k+1}{k+1} \left(- \int_{-1}^{1} \frac{P_k(t)}{(z-t)^m}\, dt + z \widetilde{Q}_k^m(z) \right) + \frac{k}{k+1} \widetilde{Q}_{k-1}^m(z).
\end{aligned}$$

Again, we use integration by parts for the first integral and plug in (2.5):

$$\begin{aligned}
\int_{-1}^{1} \frac{P_k(t)}{(z-t)^m}\, dt &= \left[\int_{-1}^{t} P_k(s)\, ds\ \frac{1}{(z-t)^m} \right]_{t=-1}^{1} - \int_{-1}^{1} \frac{P_{k+1}(t) - P_{k-1}(t)}{2k+1} \cdot \frac{m}{(z-t)^{m+1}}\, dt \\
&= -\frac{m}{2k+1} \left[\widetilde{Q}_{k+1}^m - \widetilde{Q}_{k-1}^m \right].
\end{aligned}$$

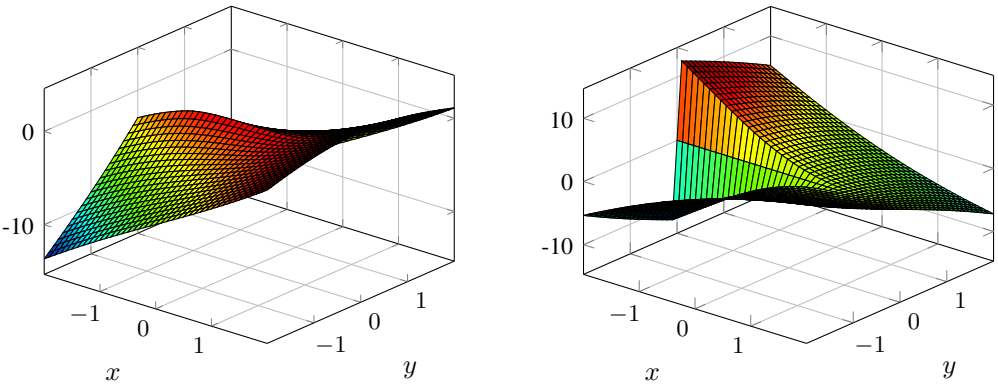
Plugging this in yields

$$\widetilde{Q}_{k+1}^m(z) = -\frac{m}{k+1} \left[\widetilde{Q}_{k+1}^m - \widetilde{Q}_{k-1}^m \right] + \frac{2k+1}{k+1}\, z\, \widetilde{Q}_k^m(z) + \frac{k}{k+1} \widetilde{Q}_{k-1}^m(z),$$

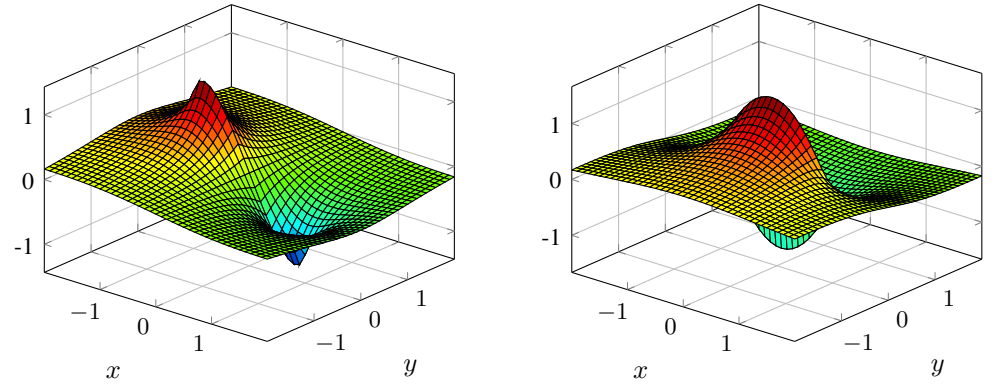
which is equivalent to

$$(k-m+1) \widetilde{Q}_{k+1}^m(z) = (2k+1)\, z\, \widetilde{Q}_k^m(z) - (k+m)\, \widetilde{Q}_{k-1}^m(z).$$

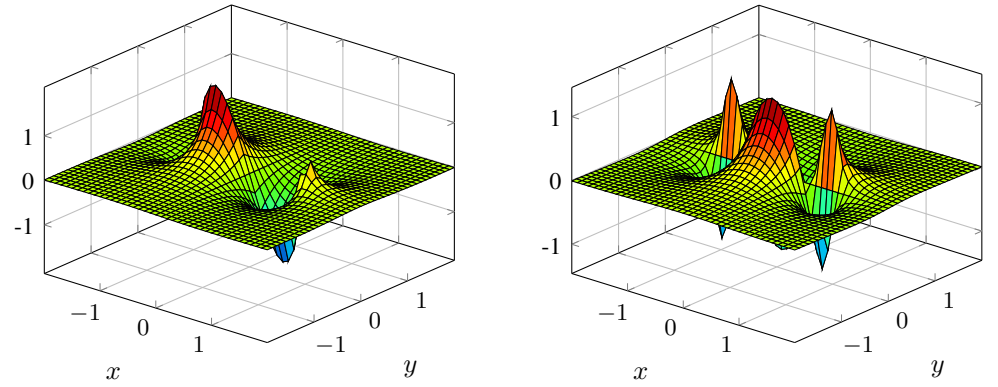
This completes the proof. □



(a) Real part (left) and imaginary part (right) of $\widetilde{Q}_0^{-2}(z)$



(b) Real part (left) and imaginary part (right) of $\widetilde{Q}_1^{-1}(z)$



(c) Real part (left) and imaginary part (right) of $\widetilde{Q}_2^0(z)$

Figure 2.2.: Surface plots of the real and imaginary parts of $\widetilde{Q}_0^{-2}$, $\widetilde{Q}_1^{-1}$, and $\widetilde{Q}_2^0$.

For non-negative orders, i.e. $\widetilde{Q}_k^m(z)$ with $m \geq 0$, we only need the initial values for $k = 0$ and $k = 1$. For negative orders, we need the initial values $\widetilde{Q}_{m-1}^{-m}(z)$ and $\widetilde{Q}_m^{-m}(z)$. The values of $\widetilde{Q}_k^{-m}(z)$ for $k = 0, \ldots, m-2$ have to be computed separately. Initial values for $\widetilde{Q}_k^m$, $m \in \{-2, -1, 0, 1\}$, are stated in the following proposition.

Proposition 2.3.4. *Let*

$$W_1 = \begin{cases} \log(z+1), & z \in \mathbb{C} \setminus (-\infty, 1] \\ \log|z+1|, & z \in (-\infty, 1) \setminus \{-1\} \end{cases}$$

and

$$W_2 = \begin{cases} \log(z-1), & z \in \mathbb{C} \setminus (-\infty, 1] \\ \log|z-1|, & z \in (-\infty, 1) \setminus \{-1\}. \end{cases}$$

For the initial values for $\widetilde{Q}_k^{-2}(z)$, there holds

$$\widetilde{Q}_0^{-2}(z) = \begin{cases} \frac{(z+1)^2}{2} W_1 - \frac{(z-1)^2}{2} W_2 - z, & z \in \mathbb{C} \setminus \{-1, 1\} \\ 2\log(2) - 1, & z = \pm 1, \end{cases}$$

$$\widetilde{Q}_1^{-2}(z) = \begin{cases} \frac{z^3 - 3z - 2}{6} W_1 - \frac{z^3 + 3z + 2}{6} W_2 - \frac{z^2}{3} + \frac{2}{9} & z \in \mathbb{C} \setminus \{-1, 1\} \\ -\frac{2}{3}\log(2) - \frac{1}{9}, & z = \pm 1, \end{cases}$$

$$\widetilde{Q}_2^{-2}(z) = \begin{cases} \frac{(z^2-1)^2}{8} W_1 - \frac{(z^2-1)^2}{8} W_2 - \frac{z^3}{4} + \frac{5}{12} z & z \in \mathbb{C} \setminus \{-1, 1\} \\ \pm\frac{1}{6}, & z = \pm 1. \end{cases}$$

The initial values for $\widetilde{Q}_k^{-1}(z)$ are given by

$$\widetilde{Q}_0^{-1}(z) = \begin{cases} (z+1) W_1 - (z-1) W_2 - 2, & z \in \mathbb{C} \setminus \{-1, 1\} \\ 2\log(2) - 2, & z = \pm 1, \end{cases}$$

$$\widetilde{Q}_1^{-1}(z) = \begin{cases} \frac{z^2 - 1}{2} (W_1 - W_2) - z, & z \in \mathbb{C} \setminus \{-1, 1\} \\ \mp 1, & z = \pm 1. \end{cases}$$

For $\widetilde{Q}_k^0(z)$, we obtain

$$\widetilde{Q}_0^0(z) = \begin{cases} (W_1 - W_2), & z \in \mathbb{C} \setminus \{-1, 1\} \\ 0, & z = \pm 1, \end{cases}$$

$$\widetilde{Q}_1^0(z) = \begin{cases} -z\,(W_1 - W_2) - 2, & z \in \mathbb{C} \setminus \{-1, 1\} \\ -2, & z = \pm 1. \end{cases}$$

The initial values for $\widetilde{Q}_k^1$ *are*

$$\widetilde{Q}_0^1(z) = \begin{cases} \dfrac{-2}{z^2-1}, & z \in \mathbb{C} \setminus \{-1,1\} \\ 0, & z = \pm 1, \end{cases}$$

$$\widetilde{Q}_1^1(z) = \begin{cases} \dfrac{2z}{z^2-1} - (W_1 - W_2), & z \in \mathbb{C} \setminus \{-1,1\} \\ 0, & z = \pm 1. \end{cases}$$

The recurrence relations for the integrals R_k^m, $m \in \mathbb{N}_0$, are stated in the following proposition.

Proposition 2.3.5. *Let* $k \in \mathbb{N}$, $m \in \mathbb{Z}$ *with* $k \geq |m| + 2$ *if* $m < 0$, *and* $k \geq 2$ *if* $m \geq 0$. *Let* $z \in \mathbb{C} \setminus \{-1,1\}$. *Then there holds*

$$(k-m+2)R_{k+3}^m(z) = (2k+1)\, z\, R_{k+2}^m(z) - (k+m-1)\, R_{k+1}^m(z). \tag{2.28}$$

Proof. The statement can be proven by using the relation of the Lobatto shape functions and the Legendre polynomials given in Equation (2.9) and the recurrence relations for $\widetilde{Q}_k^m$ (see Proposition 2.3.3). □

The initial values for R_k^m can be calculated using the values of $\widetilde{Q}_k^m$; see Remark 2.3.2 (vi). By plugging in the initial values for $\widetilde{Q}_k^m(z)$, explicit formulas for the initial values of R_k^m can be derived (see Appendix A).

2.3.2. Asymptotic Estimates for $\widetilde{Q}_k^m$

In Chapter 5 we derive a compression scheme for the single layer Galerkin matrix. Parts of the compression scheme are based on the idea that many matrix entries are close to zero and thus can be neglected. As mentioned before, the computation of matrix entries can be reduced to integrals of $\widetilde{Q}_k^{-1}$. Therefore, in this section we derive asymptotic estimates for the integrals $\widetilde{Q}_k^{-1}(z)$ for $k \to \infty$. These estimates will be used to obtain an a priori upper bound for the magnitude of Galerkin entries. Using this bound, we are then able to determine a priori which matrix entries can be neglected.
First, we establish a connection to the associated Legendre functions of the second kind.

Lemma 2.3.6. *For* $m \in \mathbb{N}_0$, $k \in \mathbb{N}$, *and* $z \in \mathbb{C} \setminus [-1,1]$, *there holds*

$$\widetilde{Q}_k^{-1}(z) = -\sqrt{z^2-1}\, Q_k^{-1}(z), \tag{2.29}$$

$$\widetilde{Q}_k^m(z) = \frac{1}{(z^2-1)^{\frac{m}{2}}\, m!}\, Q_k^m(z). \tag{2.30}$$

Proof. According to Neumann's formula (2.19), there holds

$$\frac{d^m}{dz^m}\, Q_k(z) = \frac{d^m}{dz^m} \int_{-1}^{1} \frac{P_k(t)}{(z-t)}\, dt = m! \int_{-1}^{1} \frac{P_k(t)}{(z-t)^{m+1}}\, dt = m!\, \widetilde{Q}_k^m.$$

With Equation (2.15), we obtain

$$\widetilde{Q}_k^m = \frac{1}{m!}\frac{d^m}{dz^m}\, Q_k(z) = \frac{1}{(z^2-1)^{\frac{m}{2}}\, m!} Q_k^m,$$

which proves (2.30). For order $m = -1$, we use Equation (2.16) and obtain

$$\begin{aligned} Q_k^{-1}(z) &= (z^2-1)^{-1/2} \int_z^\infty Q_k(\xi)\, d\xi \\ &= (z^2-1)^{-1/2} \int_z^\infty \int_{-1}^1 \frac{P_k(t)}{\xi - t}\, dt\, d\xi \\ &= (z^2-1)^{-1/2} \lim_{\psi\to\infty} \left[\int_{-1}^1 P_k(t) \log(\xi - t)\, dt \right]_{\xi=z}^{\psi} \\ &= (z^2-1)^{-1/2} \left(\lim_{\psi\to\infty} \int_{-1}^1 P_k(t) \log(\psi - t)\, dt - \widetilde{Q}_k^{-1}(z) \right). \end{aligned} \tag{2.31}$$

Now we use the Taylor expansion of $\log(\psi - t)$ around $t = 0$ under the assumption that $\psi > 1$,

$$\log(\psi - t) = \log(\psi) + \sum_{\nu=1}^{\infty} (-1)^{\nu-1} \frac{t^\nu}{\nu!\, \psi^\nu}.$$

Using the orthogonality of the Legendre polynomials, we obtain for $k \geq 1$

$$\begin{aligned} \lim_{\psi\to\infty} \int_{-1}^1 P_k(t) \log(\psi - t)\, dt &= \lim_{\psi\to\infty} \int_{-1}^1 P_k(t) \left(\log(\psi) + \sum_{\nu=1}^{\infty} (-1)^{\nu-1} \frac{t^\nu}{\nu!\, \psi^\nu} \right) dt \\ &= \lim_{\psi\to\infty} \int_{-1}^1 P_k(t) \sum_{\nu=1}^{\infty} (-1)^{\nu-1} \frac{t^\nu}{\nu!\, \psi^\nu}\, dt \\ &= \int_{-1}^1 P_k(t) \left(\sum_{\nu=1}^{\infty} (-1)^{\nu-1} \lim_{\psi\to\infty} \frac{t^\nu}{\nu!\, \psi^\nu} \right) dt \\ &= 0. \end{aligned}$$

Combining this result with (2.31) proves (2.29).

□

Remark 2.3.7. (i) One can show inductively that for $m \in \mathbb{N}$, $k \geq m$, and $z \in C \setminus [-1, 1]$, there holds

$$\widetilde{Q}_k^{-m}(z) = (-1)^m m! (z^2-1)^{m/2} Q_k^{-m}(z). \tag{2.32}$$

However, we only need the result for $m = 1$ for our application to hp-BEM, and therefore, we do not proove the general result here.

(ii) An alternative way to prove the statement in part (i) is to show that $\widetilde{Q}_k^{-m}(z)$ and $Q_k^{-m}(z)$ fulfill the same recurrence relation and have the same initial values,
i.e. $\widetilde{Q}_k^{-m}(z) = (-1)^m m! (z^2-1)^{m/2} Q_k^{-m}(z)$ for $k = 1, m+1$. In order to do this, one has to use the relation (2.17) and (2.30) and write $Q_k^{-m}(z)$ in terms of $\widetilde{Q}_k^m(z)$.

Next we derive asymptotic expansions for $Q_k^m(z)$ as k tends to infinity.

Lemma 2.3.8. *Let $z \in \mathbb{C}$ with $|z| > 1$, $m \in \mathbb{Z}$, and $k \in \mathbb{N}$. Furthermore, let $w = J^{-1}(z) = z + \sqrt{z^2-1}$ and $\widetilde{w} = z - \sqrt{z^2-1}$, where we choose the complex square root as stated in Remark 2.0.7. Then there holds for fixed m and $k \to \infty$*

$$Q_k^m(z) \sim (-1)^m \sqrt{\frac{\pi}{2}}\, k^{m-1/2} (z^2-1)^{-1/4} \widetilde{w}^{k+1/2}, \tag{2.33}$$

Furthermore, for the special case $m = -1$, there holds

$$Q_k^{-1}(z) \sim -\frac{\sqrt{\pi}}{2}\, k^{-3/2} (z^2-1)^{-1/2} w^{-(k)} \left(1 - \frac{1}{2w^2} - \frac{1}{9w^4} - \cdots\right). \tag{2.34}$$

Proof. In this proof m is fixed, and the symbol $\sim$ denotes the asymptotic behavior as k tends to infinity. From [EMOT53, p. 136], we get

$$Q_k^m(z) = (-1)^m \sqrt{\pi}\, 2^m \frac{\Gamma(k+m+1)}{\Gamma(k+\frac{3}{2})} \frac{(z^2-1)^{m/2}}{w^{k+m+1}} \cdot F(m+\frac{1}{2}, k+m+1; \frac{2k+3}{2}; \frac{1}{w^2}),$$

$$Q_k^m(z) = (-1)^m \sqrt{\frac{\pi}{2}} \frac{\Gamma(k+m+1)}{\Gamma(k+\frac{3}{2})} \frac{\widetilde{w}^{k+1/2}}{(z^2-1)^{1/4}} \cdot F(m+\frac{1}{2}, m-\frac{1}{2}; \frac{2k+3}{2}; \frac{-\widetilde{w}}{2\sqrt{z^2-1}}),$$

where F is the hypergeometric function. For a definition of the hypergeometric function see e.g. [EMOT53, Chapter II]. For the Gamma functions, there holds

$$\frac{\Gamma(k+m+1)}{\Gamma(k+\frac{3}{2})} = (k+1)\cdots(k+m)\frac{\Gamma(k+1)}{\Gamma(k+\frac{3}{2})} \sim k^{m-1/2},$$

where we used

$$\frac{e^{-1/2}}{\sqrt{|\mu+1|}} \le \frac{\Gamma(\mu+1/2)}{\Gamma(\mu+1)} \le \frac{e}{\sqrt{|\mu+1|}}, \quad \mu \in \mathbb{C}, \mathrm{Re}(\mu) > 0,$$

which is proven in [AR91, Lemma 4].

For the hypergeometric function, there hold the expansions [GR81, p 448]

$$F(m+\frac{1}{2}, k+m+1; \frac{2k+3}{2}; \frac{1}{w^2}) = 1 + \sum_{j=1}^{\infty} \left(\prod_{\nu=0}^{j-1} \frac{(2m+1+\nu)(k+m+1+\nu)}{(2k+3+\nu)}\right) \frac{w^{-2j}}{j!}$$

and

$$F(m+\frac{1}{2}, m-\frac{1}{2}; \frac{2k+3}{2}; \frac{-\widetilde{w}}{2\sqrt{z^2-1}}) = 1 + \sum_{j=1}^{\infty} \left(\prod_{\nu=0}^{j-1} \frac{(2m+1+\nu)(2m-1+\nu)}{2\,(2k+3+\nu)}\right) \frac{1}{j!} \left(\frac{-\widetilde{w}}{2\sqrt{z^2-1}}\right)^j.$$

Considering the first few summands for the special case $m = -1$ in the first expansion gives the estimates for Q_k^{-1}. For fixed m, j, and z, the products in the second expansion converge to zero as k goes to infinity, which means that the sum vanishes. This proves the result (2.33). □

Now we can use Lemma 2.3.8 to derive an asymptotic estimate for the integrals $\widetilde{Q}_k^m$, $m \geq -2$.

Proposition 2.3.9. *Let $z \in \mathbb{C}$ with $|z| > 1$ and $k \in \mathbb{N}$.*

(i) For $\widetilde{w} = z - \sqrt{z^2-1}$, there holds for $k \to \infty$ and fixed m

$$\widetilde{Q}_k^m(z) \sim \frac{(-1)^m}{m!}\sqrt{\frac{\pi}{2}}\, k^{m-1/2}(z^2-1)^{-(2m+1)/4}\ \widetilde{w}^{k+1/2}, \quad m \in \mathbb{N}_0, \tag{2.35}$$

$$\widetilde{Q}_k^{-m}(z) \sim m!(-1)^m\sqrt{\frac{\pi}{2}}\, k^{m-1/2}(z^2-1)^{(2m-1)/4}\ \widetilde{w}^{k+1/2}, \quad m \in \{1,2\}. \tag{2.36}$$

(ii) Let $z \in \mathbb{C}$ with $|z| > 1$ and $k \in \mathbb{N}$. Furthermore, let $w = J^{-1}(z) = z + \sqrt{z^2-1}$. Then there holds for $k \to \infty$

$$\widetilde{Q}_k^{-1}(z) \sim 2\sqrt{\pi}\, k^{-3/2}\, w^{-k}\ \left(1 - \frac{1}{2w^2} - \frac{1}{9w^4} - \cdots\right). \tag{2.37}$$

Proof. Follows from Lemmas 2.3.6 and 2.3.8. □

Since we will need the asymptotic estimate (2.37) to create an upper bound for the matrix entries of the Galerkin matrix for the single layer potential, we examine this estimate numerically. We are interested in an upper bound for $|\widetilde{Q}_k^{-1}(z)|$ for $z \in \mathbb{C}$. Since the summands $1/(2|w|^2), 1/(9|w|^4), \ldots$ in (2.37) are positive, an upper bound for $\widetilde{Q}_k^{-1}$ is given by

$$\overline{Q}_k^{-1}(z) := 4\sqrt{\pi}\, k^{-3/2}\, |w|^{-k}\ \left(1 - \frac{1}{2|w|^2} - \frac{1}{9|w|^4}\right), z \in \mathbb{C} \setminus \{-1,1\}, \tag{2.38}$$

where $w = J^{-1}(z) = z + \sqrt{z^2-1}$. However, for $|z|$ greater than but close to one, the upper bound $\overline{Q}_k^{-1}(z)$ is too large. For $|z| \leq 1$, the estimate has no theoretical proof. Numerical experiments yield for the values of $\widetilde{Q}_k^{-1}$ at the points ± 1

$$\widetilde{Q}_k^{-1}(1) \sim -\frac{32}{(4\,k+1)^2} =: \overline{Q}_k^{-1}(1), \tag{2.39}$$

$$\widetilde{Q}_k^{-1}(-1) \sim (-1)^{k-1}\frac{32}{(4k+1)^2} =: \overline{Q}_k^{-1}(-1). \tag{2.40}$$

We compare $\widetilde{Q}_k^{-1}(z)$ and $\overline{Q}_k^{-1}(z)$ for $z \in \mathbb{C}$ with $\mathrm{Re}(z) \in [-1.5, 1.5]$ and $\mathrm{Im}(z) \in [-1.5, 1.5]$. Figure 2.3 shows a contour plot of the quotient

$$q_k(z) := \frac{|\overline{Q}_k^{-1}(z)|}{|\widetilde{Q}_k^{-1}(z)|}$$

for $k = 16$ (left) and $k = 128$ (right). It can be seen that $|\overline{Q}_k^{-1}(z)|$ is an upper bound in the whole domain under consideration. However, for z close to one and between -1 and 1, we overestimate the actual value by a factor of more than four.

Next we examine the asymptotic behavior of $q_k(z)$ for $k \to \infty$ for six different points $z \in \mathbb{C}$.

In Figure 2.4 it can be seen that asymptotically we obtain a constant quotient in all cases. However, for z close to 1, the quotient is large meaning we overestimate the actual value of $\widetilde{Q}_k^{-1}(z)$. The figure also shows the minimal quotient for $z \in \mathbb{C}$ with $\mathrm{Re}(z) \in [-1.5, 1.5]$ and $\mathrm{Im}(z) \in [-1.5, 1.5]$. This minimal quotient asymptotically approaches 1 from above, which means that we could not choose a smaller upper bound.
Overall, our numerical experiments together with the theoretical estimate (2.37) show that $\overline{Q}_k^{-1}(z)$ is a reliable upper bound for $|\widetilde{Q}_k^{-1}(z)|$ for $z \in \mathbb{C}$.

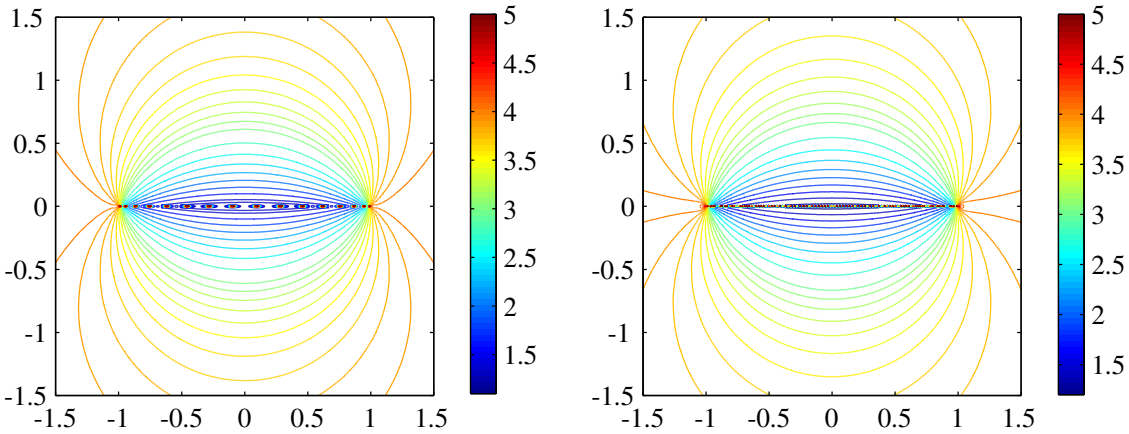


Figure 2.3.: Contour plot of the quotient $q_k(z)$ for $k = 16$ (left) and $k = 128$ (right).

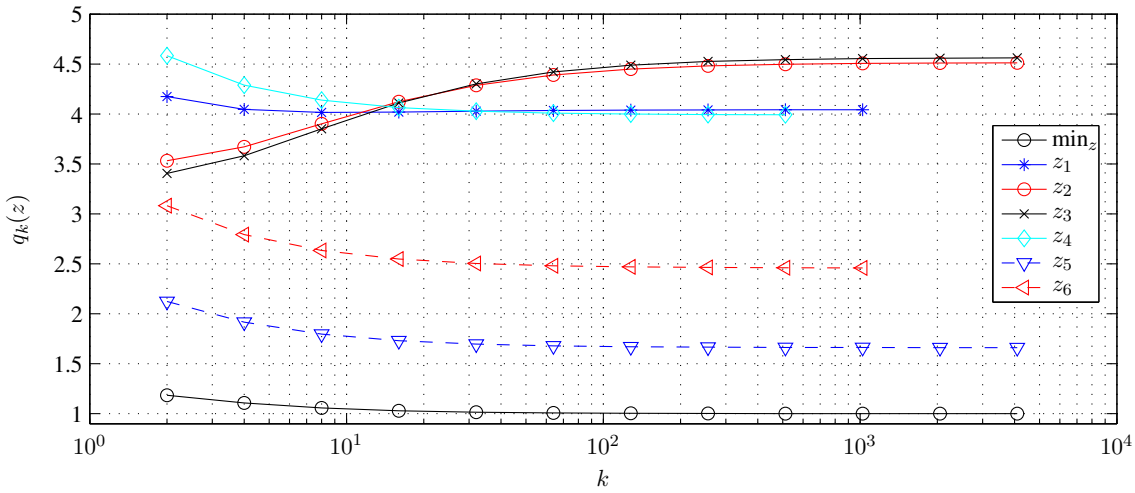


Figure 2.4.: Quotient $q_k(z)$ over k for $z_1 = 1.1$, $z_2 = 1.01$, $z_3 = 1.005 + 1.005i$, $z_4 = 1.7 + 0.2i$, $z_5 = 0.7 - 0.05i$, $z_6 = 0.5 + 0.3i$, and the minimal quotient $\min\{q_k(z) : \mathrm{Re}(z), \mathrm{Im}(z) \in [-1.5, 1.5]\}$.

2.3.3. Minimal Solutions

In Section 2.3.1 we define modified associated Legendre functions and derive recurrence relations that can be used to evaluate them. However, evaluating recurrence relations can be numerically unstable. To characterize and analyze this instability, we introduce the term of minimal and dominant solutions of a three-term recurrence relation. Consider the three-term recurrence relation

$$y_{k+1} + a_k y_k + b_k y_{k-1} = 0, \quad k = 0, 1, 2, \ldots, \tag{2.41}$$

$a_k, b_k \in \mathbb{C}$, $b_k \neq 0$. This recurrence relation has two linearly independent sets of solutions which we call the minimal and dominant solutions.

Definition 2.3.10. (i) A solution $\{f_k\}$ of the three-term recurrence relation (2.41) is called minimal if

$$\lim_{k\to\infty} \frac{f_k}{g_k} = 0 \quad (g_k \neq 0) \tag{2.42}$$

for any solution $\{g_k\}$ that is linearly independent of $\{f_k\}$. All solutions of (2.41) that are not minimal are called dominant.

(ii) The polynomial $t^2 + at + b$ with $a = \lim_{k\to\infty} a_k$ and $b = \lim_{k\to\infty} b_k$ is called the characteristic polynomial of (2.41).

We show that $\widetilde{Q}_k^m$ and R_k^m are minimal solutions of their recurrence relation. First, we need some auxiliary results.

Lemma 2.3.11. *(i) ([Gau67, Theorem 2.1]) If the characteristic polynomial of (2.41) has two distinct roots t_1 and t_2 with $|t_1| > |t_2|$, then for any nontrivial solution $\{y_k\}$ of (2.41), there holds*

$$\lim_{k\to\infty} \frac{y_{k+1}}{y_k} = t_1 \qquad \text{or} \qquad \lim_{k\to\infty} \frac{y_{k+1}}{y_k} = t_2.$$

(ii) ([Gau67, Theorem 2.2]) Under the assumptions from (i), there exist two linearly independent solutions $\{y_{k,1}\}$ and $\{y_{k,2}\}$ of (2.41) with

$$\lim_{k\to\infty} \frac{y_{k+1,r}}{y_{k,r}} = t_r, \quad r = 1, 2.$$

(iii) ([Gau67, p. 34]) The solution $\{y_{k,2}\}$ from part (ii) is the minimal solution of (2.41).

Proposition 2.3.12. *For $z \in \mathbb{C}$ with $|z| > 1$, the integrals $\widetilde{Q}_k^m$, $m \in \mathbb{N}_0 \cup \{-1, -2\}$, and R_k^m, $m \in \mathbb{N}_0 \cup \{-1, -2\}$, are minimal solutions of their respective recurrence relations.*

Proof. Throughout the proof we assume $|z| > 1$. We start with the proof for $\widetilde{Q}_k^m$. For the recurrence relation (2.27), there holds

$$\lim_{k\to\infty} a_k = \lim_{k\to\infty} \frac{2k+1}{k-m+1}\, z = 2z$$

and

$$\lim_{k\to\infty} b_k = \lim_{k\to\infty} \frac{k+m}{k-m+1} = 1.$$

Thus, the characteristic polynomial is $t^2 + 2zt + 1$ and has the roots $t_1 = z + \sqrt{z^2-1}$ and $t_2 = z - \sqrt{z^2-1}$. For $\mathrm{Re}(z) > 0$ or $\mathrm{Re}(z) = 0$ and $\mathrm{Im}(z) > 0$, there holds $|t_2| < 1 < |t_1|$. Now we use the asymptotic estimate (2.35), which yields for $m \in \mathbb{N}_0$

$$\lim_{k\to\infty} \frac{\widetilde{Q}_{k+1}^m(z)}{\widetilde{Q}_k^m(z)} = \lim_{k\to\infty} \widetilde{w} \left(\frac{k+1}{k}\right)^{m-1/2} = \widetilde{w} = t_2.$$

For negative orders, i.e. for $\widetilde{Q}_k^{-m}$, $m \in \{1, 2\}$, we use (2.36) and obtain the same result. Thus, the limit coincides with the smaller square root of the characteristic polynomial. This shows that $\widetilde{Q}_k^m(z)$ is the minimal solution.

For $w \in \mathbb{C}$, $\mathrm{Re}(w) < 0$, or $\mathrm{Re}(w) = 0$ and $\mathrm{Im}(w) < 0$, there holds $|t_1| = |w + \sqrt{w^2-1}| < 1 < |w - \sqrt{w^2-1}| = |t_2|$. Furthermore, there holds $Q_k^m(-z) = (-1)^{k-1} Q_k^m(z)$, and thus $\widetilde{Q}_k^m(-z) = (-1)^{k-1}\widetilde{Q}_k^m(z)$ for $m \in \mathbb{N}_0 \cup \{-1, -2\}$. Therefore, for $w = -z$ (with z defined as above), we have

$$\begin{aligned}\lim_{k\to\infty} \frac{\widetilde{Q}_{k+1}^m(w)}{\widetilde{Q}_k^m(w)} &= -\lim_{k\to\infty} \frac{\widetilde{Q}_{k+1}^m(z)}{\widetilde{Q}_k^m(z)} \\ &= -(z - \sqrt{z^2-1}) = (-z) + \sqrt{z^2-1} = w + \sqrt{w^2-1} = t_1,\end{aligned}$$

which shows that $\widetilde{Q}_k^m(w)$ is the minimal solution.

Next we prove the statement for R_k^m. For $m \in \mathbb{N}_) \cup \{-1, -2\}$ there holds

$$R_k^m(z) = \frac{\widetilde{Q}_{k-1}^m(z) - \widetilde{Q}_{k-3}^m(z)}{2k-3},$$

and thus

$$\frac{R_{k+1}^m(z)}{R_k^m(z)} = \frac{\widetilde{Q}_k^m(z) - \widetilde{Q}_{k-2}^m(z)}{\widetilde{Q}_{k-1}^m(z) - \widetilde{Q}_{k-3}^m(z)} \cdot \frac{2k-3}{2k-1}.$$

Plugging in the asymptotic estimate (2.35) and canceling yields

$$\frac{R_{k+1}^m(z)}{R_k^m(z)} \sim \widetilde{w} \left(\frac{k+1}{k}\right)^m \frac{2k-3}{2k-1}$$

and $\lim_{k\to\infty} R_{k+1}^m(z)/R_k^m(z) = \widetilde{w}$.

Since the roots of the characteristic polynomial for the recurrence relation for R_k^m are again $t_1 = z + \sqrt{z^2-1}$ and $t_2 = z - \sqrt{z^2-1}$, we have shown that $R_k^m(z)$ is the minimal solution for $z \in \mathbb{C}$ with $\mathrm{Re}(z) > 0$ or $\mathrm{Re}(z) = 0$ and $\mathrm{Im}(z) > 0$. For all other z, we consider $R_k^m(-z)$, use the property $\widetilde{Q}_k^m(-z) = (-1)^{k-1}\widetilde{Q}_k^m(z)$, and obtain the result in the same way as for $\widetilde{Q}_k^m$. □

Remark 2.3.13. (i) In the proof of Proposition 2.3.12 we use the asymptotic estimates (2.35) and (2.36). These estimates are only true for $|z| > 1$. Therefore, we can only show that $\widetilde{Q}_k^m(z)$ and $R_k^m(z)$ are minimal solutions of their recurrence relation for $|z| > 1$. However, there exist asymptotic estimates for $Q_k^m(z)$ that are valid for $z \in \mathbb{C} \setminus [-1, 1],\ |z| < 1$ (see [EMOT53, Chapter III]). Exploiting the connection between Q_k^m and $\widetilde{Q}_k^m$ from Lemma 2.3.6, these estimates can be used to derive asymptotic estimates for $\widetilde{Q}_k^m(z)$ for $z \in \mathbb{C} \setminus [-1, 1],\ |z| < 1$. Finally, if we use the resulting asymptotic estimates for $\widetilde{Q}_k^m$ in the proof of Proposition 2.3.12, we obtain the result for $z \in \mathbb{C} \setminus [-1, 1]$ with $|z| < 1$.

(ii) In fact, it can be shown that for $|z| > 1$, $\widetilde{Q}_k^{-m}(z)$ is a minimal solution for all $m \in \mathbb{N}$; see Remark 2.3.7.

The calculation of the minimal solution with the forward recurrence relation is generally unstable. To show this, we present the following derivation ;see [Gau67]. Let $\{f_k\}$ be the minimal solution of the three-term recurrence relation (2.41). Any solution $\{y_k\}$ that is linearly independent of $\{f_k\}$ can be represented in the form $y_k = Af_k + Bg_k$, and there holds

$$\lim_{k\to\infty} \frac{f_k}{y_k} = \lim_{k\to\infty} \frac{f_k/g_k}{B + A(f_k/g_k)} = 0. \tag{2.43}$$

If we approximate f_k numerically, we start with perturbed initial values $y_0 = f_0 + \varepsilon_0$ and $y_1 = f_1 + \varepsilon_1$. Even if all further calculations are performed without error, we obtain a solution $\{y_k\}$ that is generally linearly independent of $\{f_k\}$. Then, because of (2.43),

$$\lim_{k\to\infty} \left| \frac{y_k - f_k}{f_k} \right| = \infty.$$

However, there exist stable algorithms to calculate minimal solutions, such as Gautschi's continued fraction algorithm [Gau67] and Miller's backward recurrence algorithm [Thi52], to name a few but not all. We refer to [Gau81] and the references therein for further details. Both Gautschi's continued fraction algorithm and Miller's backward algorithm use the idea of starting the recurrence relation with $y_\nu = 0$ for some large value of ν, iterating backwards to the initial values, and then updating the calculated iterates in the forward direction. Miller's backward algorithm produces overflow, especially if z is close to the real axis; see [Gau81]. Therefore, we implement Gautschi's continued fraction algorithm for the calculation of modified associated Legendre functions. For the evaluation of the recurrence relation

$$y_{k+1}(z) = (z - \widetilde{a}_k)y_k(z) - \widetilde{b}_k y_{k-1}(z), \quad k = 1, 2, \dots N, \tag{2.44}$$

with $\widetilde{a}_k, \widetilde{b}_k \in \mathbb{R}$ and $\widetilde{b}_k > 0$, Gautschi's continued fraction algorithm is stated in Algorithm 2.1. If $\{f_n(z)\}$ is the minimal solution of (2.44), then $\lim_{\nu\to\infty} f_n^\nu(z) = f_n(z), \quad n = 1, 2, \dots, N$; see [Gau67, p. 39]. The recurrence relations for $\widetilde{Q}_k^m$ and R_k^m are given in the form (2.41). The following lemma gives a relation between the two recurrence relations (2.41) and (2.44).

Algorithm 2.1 Gautschi's continued fraction algorithm for the computation of the minimal solution of (2.44) for $k = 1, 2, \dots N$.

INPUT: $N, \nu \in \mathbb{N}, \nu \gg N$.

OUTPUT: Minimal solution $f_k, k = 0, \dots, N$ of (2.44).

1: Set $r_\nu^\nu = 0$.
2: **for** $k = \nu, \nu-1, \dots, 1, 0$ **do**
3: Set $r_{k-1}^\nu = \frac{\widetilde{b}_k}{z - \widetilde{a}_k - r_k^\nu}$.
4: **end for**
5: Set $f_{-1}^\nu = 1$.
6: **for** $k = 0, 1, \dots, N$ **do**
7: Set $f_k^\nu = r_{k-1}^\nu f_{k-1}^\nu$.
8: **end for**

Lemma 2.3.14. *If f_k fulfills the recurrence relation (2.41), then*

$$\widetilde{f}_k := \left(\prod_{j=2}^{k-1} a_j \right)^{-1} \cdot f_k =: (A_k)^{-1} \, f_k$$

fulfills the recurrence relation (2.44) with $\widetilde{a}_k = 0$ and $\widetilde{b}_k = \frac{A_{k-1}}{A_{k+1}} b_k = \frac{b_k}{a_{k-1}\, a_k}$.

Proof. There holds

$$A_{k+1}\widetilde{f}_{k+1} = f_{k+1} = a_k z f_k - b_k f_{k-1} = a_k A_k z \widetilde{f}_k - b_k A_{k-1} \widetilde{f}_{k-1},$$

which is equivalent to

$$\widetilde{f}_{k+1} = a_k \frac{A_k}{A_{k+1}} z \widetilde{f}_k - b_k \frac{A_{k-1}}{A_{k+1}} \widetilde{f}_{k-1} = z \widetilde{f}_k - \frac{b_k}{a_k\, a_{k-1}} \widetilde{f}_{k-1}.$$

□

Using this result, we can modify Gautschi's continued fraction algorithm. Before we state the modified version, we give an estimate for the choice of ν that is derived in [Gau81].

Lemma 2.3.15. *Let N the index up to which we evaluate the recurrence relation (2.44). Then for the relative error between the exact solution of (2.44) $\{f_k\}$ and the solution $\{f_k^\nu\}$ of the backward algorithm,*

$$e_k := \left| \frac{f_k - f_k^\nu}{f_k} \right|,$$

there holds

$$\max_{1 \le k \le N} e_k = e_N \sim \left(z + \sqrt{z^2 - 1} \right)^{2(N-\nu-1)}$$

for $\nu > N \to \infty$.

Proof. See [Gau81, p. 552]. □

From this result we can immediately derive an estimate for ν such that the relative error stays below a given tolerance ε:

$$\nu \geq N + \frac{\log(1/\varepsilon)}{2\log|z+\sqrt{z^2-1}|} \tag{2.45}$$

Algorithm 2.2 can be used to compute the modified associated Legendre functions $\widetilde{Q}_k^m(z)$ and $R_k^m(z)$.

Algorithm 2.2 Modified Gautschi's continued fraction algorithm that can be used for the computation of $\widetilde{Q}_k^m$ or R_k^m, $k \leq N$.

INPUT: Maximum index $N \in \mathbb{N}$, tolerance $\varepsilon > 0$.

OUTPUT: Minimal solution f_k, $k = 0, \ldots, N$, of (2.41).

1: Compute
$$\nu = \left\lceil N + \frac{\log(1/\varepsilon)}{2\log|z+\sqrt{z^2-1}|} \right\rceil.$$
2: Set $r_\nu^\nu = 0$.
3: **for** $k = \nu, \nu-1, \ldots, k_0$ **do**
4: Set $r_{k-1}^\nu = \frac{b_k}{z - r_k^\nu}$.
5: **end for**
6: **for** $k = 0, \ldots, k_0 - 1$ **do**
7: Compute the initial values f_k^ν as described in Proposition 2.3.4 and Remark 2.3.2 (vi).
8: **end for**
9: **for** $k = k_0, \ldots, N$ **do**
10: Set $f_k^\nu = a_{k-1}\, r_{k-1}^\nu\, f_{k-1}^\nu$.
11: **end for**

From estimate (2.45), it can be seen that ν is constant on ellipses with foci ± 1. The estimate also shows that the computational cost of Gautschi's continued fraction algorithm increases as z approaches the interval $[-1, 1]$. In the following section we will see that we only need to use Gautschi's continued fraction algorithm outside of an ellipse with foci ± 1 with minor axis depending on N. Thus, the computational costs of the algorithm stay limited.

2.3.4. Implementation

In this section we discuss the implementation of the recurrence relations for $\widetilde{Q}_k^m$, $m \in \{-2, -1, 0, 1\}$, and R_k^m, $m \in \{0, 1\}$. We start with the discussion of the stable evaluation of the initial values given in Proposition 2.3.4 and Remark 2.3.2 (vi), respectively.

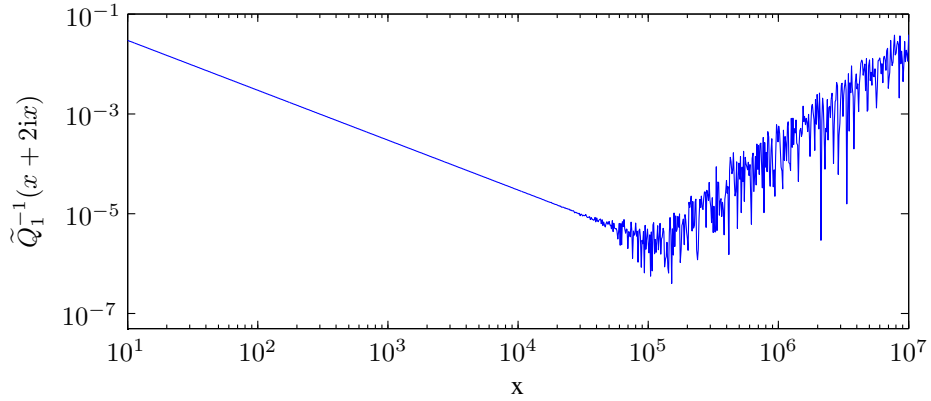


Figure 2.5.: Numerically computed values of $\widetilde{Q}_1^{-1}(x+2\mathrm{i}x)$ over x. The values were computed with a straight forward MATLAB implementation of the formula from Proposition 2.3.4. It can be seen that the computations become unstable for $x \gg 1$.

A first MATLAB implementation of the initial values from Proposition 2.3.4 shows that the formulas presented in this proposition are numerically unstable for $z \in \mathbb{C}$ with $|z| \gg 1$. Figure 2.5 displays the values of $\widetilde{Q}_1^{-1}(z) = \frac{z^2-1}{2}\log\left(\frac{z+1}{z-1}\right) - z$ for $z = x + 2\mathrm{i}x$, which were obtained with the straight forward MATLAB implementation. We see that for large x the calculations become unstable. The reason for this is that for large z, the term $\frac{z^2-1}{2}\log\left(\frac{z+1}{z-1}\right)$ is approximately z, which means that our MATLAB implementation produces inaccurate results due to cancellation. Therefore, for large $|z|$, we use Laurent series expansions of the initial values.

Lemma 2.3.16. *Let $z \in \mathbb{C}$, $|z| > 1$, and $w = J^{-1}(z) = z + \sqrt{z^2-1}$. Then there holds*

$$\log(z+1) = \log\left(\frac{z}{2}\right) + \sum_{k=0}^{\infty} \frac{2}{2k+1} w^{-(2k+1)} - \sum_{k=1}^{\infty} \frac{1}{k} w^{-2k},$$
$$\log(z-1) = \log\left(\frac{z}{2}\right) - \sum_{k=0}^{\infty} \frac{2}{2k+1} w^{-(2k+1)} - \sum_{k=1}^{\infty} \frac{1}{k} w^{-2k},$$

and

$$\frac{1}{z+1} = \sum_{k=1}^{\infty} (-1)^{k+1}\, 2k\, w^{-k}, \qquad \frac{1}{z-1} = \sum_{k=1}^{\infty} 2k\, w^{-k}.$$

Using this result, we can immediately state series expansions for all other initial values.

Proposition 2.3.17. *Let $z \in \mathbb{C}$, $|z| > 1$, and $w = J^{-1}(z)$. For the initial values for $\widetilde{Q}_k^{-1}$, there holds*

$$\widetilde{Q}_0^{-1}(z) = 2\log\left(\frac{w}{2}\right) + \sum_{k=1}^{\infty} \frac{2}{k(4k^2-1)}\, w^{-2k},$$

$$\widetilde{Q}_1^{-1}(z) = \frac{4}{w}\left(-\frac{1}{3} + \sum_{k=1}^{\infty} \frac{1}{(4k^2-1)(2k+3)}\, w^{-2k}\right).$$

The initial values for $\widetilde{Q}_k^0$ are given by

$$\widetilde{Q}_0^0(z) = \sum_{k=0}^{\infty} \frac{4}{2k+1}\, w^{-2k-1}, \qquad \widetilde{Q}_1^0(z) = \sum_{k=1}^{\infty} \frac{8k}{(2k-1)(2k+1)}\, w^{-2k},$$

and for $\widetilde{Q}_k^1$ we have

$$\widetilde{Q}_0^1(z) = \frac{2}{z^2-1}, \qquad \widetilde{Q}_1^1(z) = \sum_{k=1}^{\infty} \frac{16k(k+1)}{(2k+1)} w^{-2k-1}.$$

The initial values for R_k^0 are

$$R_1^0(z) = -\sum_{k=1}^{\infty} \frac{4k\, w^{-2k}}{(4k^2-1)} + \sum_{k=0}^{\infty} \frac{2\, w^{-2k-1}}{2k+1},$$

$$R_2^0(z) = \sum_{k=1}^{\infty} \frac{4k\, w^{-2k}}{(4k^2-1)} + \sum_{k=0}^{\infty} \frac{2\, w^{-2k-1}}{2k+1},$$

$$R_3^0(z) = \sum_{k=0}^{\infty} \frac{4}{(4k^2-1)(2k+3)} w^{-2k},$$

$$R_4^0(z) = \sum_{k=1}^{\infty} \frac{8k}{(4k^2-1)(4k^2-9)} w^{-2k},$$

and for the initial values of R_k^1, there holds

$$R_1^1(z) = \sum_{k=1}^{\infty} 4k\, w^{-2k} - \sum_{k=1}^{\infty} \frac{2-2(2k+1)^2}{2k+1} w^{-2k-1},$$

$$R_2^1(z) = \sum_{k=1}^{\infty} 4k\, w^{-2k} - \sum_{k=1}^{\infty} \frac{2-2(2k+1)^2}{2k+1} w^{-2k-1},$$

$$R_3^1(z) = -\sum_{k=1}^{\infty} \frac{8k}{(4k^2-1)} w^{-2k},$$

$$R_4^1(z) = -\sum_{k=1}^{\infty} \frac{16k(k+1)}{(4k^2-1)(2k+3)} w^{-2k-1}.$$

Note that we have series expansions for $w = J^{-1}(z)$ and not for z. Furthermore, the series expansions are only valid for $|z| > 1$, and the series converge more slowly as $|z|$ approaches one. Therefore, the use of these expansions is only effective if $|z|$ is large enough. For our implementation we use the series expansion if $|w| > 1.5$. The case $|w| \leq 1.5$ will be discussed later. Since we cannot evaluate infinite sums with the computer, we need to find an $N = N(|z|)$, up to which the sums are calculated such the results are accurate up to machine precision. We only need to take into account summands larger than 10^{-16} times the largest summand, since

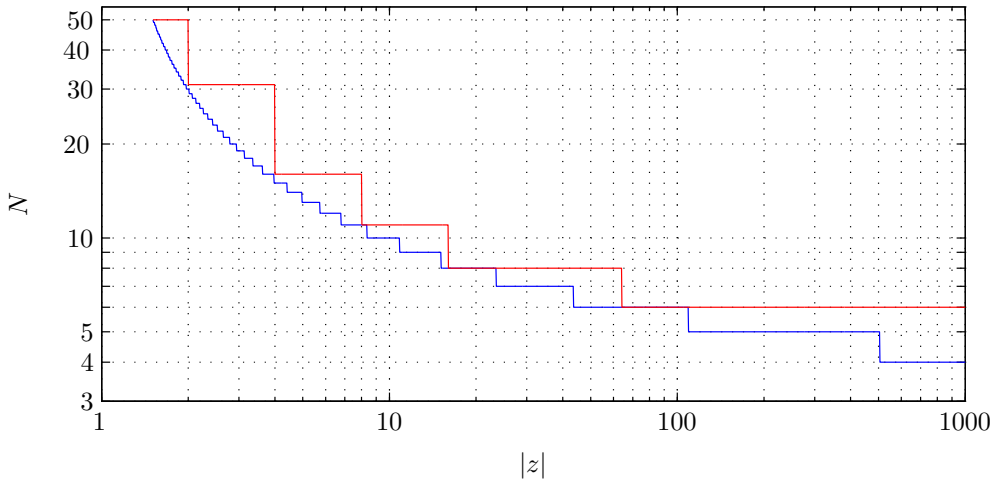


Figure 2.6.: Exact values of N such that (2.46) holds (blue) and estimated, piecewise constant upper bound (red).

numbers in double precision only have 16 significant digits. For the function $\widetilde{Q}_1^1(z)$, we obtain for the quotient of the Nth summand a_N and the first summand a_1

$$\frac{a_N}{a_1} = \frac{3\ N\ (N+1)}{2\ (2N+2)} |z|^{-2N+2} < 10^{-16}. \tag{2.46}$$

Solving this for N leads to a non linear equation. In order to avoid solving this nonlinear equation, we define intervals for $|z|$ and choose N constant on these intervals. Figure 2.6 shows the exact values of N together with the estimates ones. For the other initial values, we determine the upper limit of the sums similarly.

For $|w| \leq 1.5$, i.e. in the near field, it is not efficient to use the Laurent series expansions. For $|z| \leq 1$, the expansions are not even valid. Thus, for the near field, we use analytic formulas (see Appendix A). However, we have to modify some calculations in order to avoid numerical instabilities. To see this, we consider the function $\widetilde{Q}_0^0(z) = \log\left(\frac{z+1}{z-1}\right)$. For $z := u + \mathrm{i}v \in \mathbb{C}$, there holds

$$\begin{aligned}
\log(z+1) - \log(z-1) &= \frac{1}{2}\log\left(\frac{(u+1)^2+v^2}{(u-1)^2+v^2}\right) + \mathrm{i}\left(\arg(z+1) - \arg(z-1)\right) \\
&= \frac{1}{2}\log\left(\frac{(u+1)^2+v^2}{(u-1)^2+v^2}\right) + \mathrm{i}\big(\mathrm{atan2}(v,u+1) - \mathrm{atan2}(v,u-1)\big) \\
&=: W(u,v) + \mathrm{i}\gamma(u,v).
\end{aligned}$$

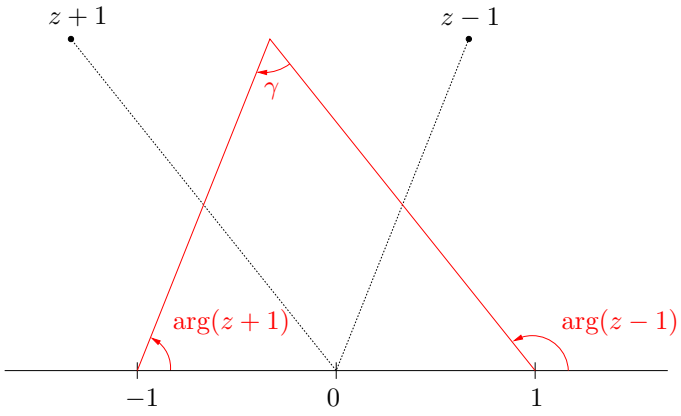


Figure 2.7.: Geometric interpretation of $\gamma(u, v)$. Here $z = u + \mathrm{i}v$.

where the function $\mathrm{atan2}$ is given by

$$\mathrm{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right), & x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi, & x < 0, y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi, & x < 0, y < 0, \\ \frac{\pi}{2}, & x = 0, y > 0, \\ -\frac{\pi}{2}, & x = 0, y < 0, \\ \text{undefined}, & x = y = 0, \end{cases}$$

and implemented in the standard C math library. Figure 2.7 shows the geometric interpretation of $\gamma(u, v)$. We see that γ is close to zero for $|z| > 1$ with $\mathrm{Im}(z)$ close to zero. For $|z| = 1$, we have $\gamma = \pi/2$, and for $|z| < 1$ but close to the real axis, γ is close to π. Thus, in each neighborhood of $z = +1$ and $z = -1$, the values of γ range from 0 to π and for $z \in \mathbb{R}$, γ has a jump at $+1$ and -1. This leads to instabilities in the computation of γ. To overcome these instabilities, we use a series representation of γ in polar coordinates.

Proposition 2.3.18. *For all $z = u + \mathrm{i}v \in \mathbb{C}$ with $r := |z + 1| < 1$ and $\phi = \arg(z + 1)$, there holds*

$$\gamma = \phi - \sigma\pi + \sum_{k=1}^{\infty} \frac{r^k}{k \cdot 2^k} \sin(k\,\phi), \tag{2.47}$$

where $\sigma = 1$ if $v >= 0$ and $\sigma = -1$ otherwise.
For all $z = u + \mathrm{i}v \in \mathbb{C}$ with $r := |z - 1| < 1$ and $\phi = \arg(z - 1)$, there holds

$$\gamma = -\phi - \sum_{k=1}^{\infty} \frac{r^k}{k \cdot 2^k} \sin(k(\phi + \pi)). \tag{2.48}$$

Proof. We start with the proof of the second part. Let $z = 1 + re^{\mathrm{i}\phi}$. This is equivalent to $\phi = \arg(z-1)$ and $r = |z-1|$. Then there holds

$$\mathrm{Im}(\log(z-1)) = \mathrm{Im}(\log(re^{\mathrm{i}\phi})) = \mathrm{Im}(\log(r) + \mathrm{i}\phi) = \phi.$$

Furthermore, using the identity

$$\log(2+z) = \log(2) - \sum_{k=1}^{\infty} \frac{(-z)^k}{k\ 2^k}, \quad \forall z \in \mathbb{C}, |z| < 2,$$

we have

$$\begin{aligned}\mathrm{Im}(\log(z+1)) = \mathrm{Im}(\log(2 + re^{\mathrm{i}\phi})) &= -\mathrm{Im}(\sum_{k=1}^{\infty} \frac{r^k e^{\mathrm{i}k\pi} e^{\mathrm{i}k\phi}}{k\ 2^k}) \\ &= -\sum_{k=1}^{\infty} \frac{r^k\, \mathrm{Im}(e^{\mathrm{i}k(\phi+\pi)})}{k\ 2^k} = -\sum_{k=1}^{\infty} \frac{r^k}{k\ 2^k} \sin(k(\phi+\pi)).\end{aligned}$$

Altogether we obtain

$$\gamma = \mathrm{Im}(\log(z+1) - \log(z-1)) = -\phi - \sum_{k=1}^{\infty} \frac{r^k}{k\ 2^k} \sin(k(\phi+\pi)).$$

For the first part we set $z = u + \mathrm{i}\ v = -1 + re^{\mathrm{i}\phi}$, which corresponds to $r = |z+1|$ and $\phi = \arg(z+1)$. Now we obtain

$$\mathrm{Im}(\log(z+1)) = \arg(z+1) = \phi.$$

For the other term we use the identity

$$\log(x-2) = \log(2) - \mathrm{i}\ \mathrm{csgn}(\mathrm{i}(x-2))\pi - \sum_{k=1}^{\infty} \frac{x^k}{k\ 2^k}, \quad \forall x \in \mathbb{C}, |x| < 1,$$

where

$$\mathrm{csgn}(x) = \begin{cases} 1, & \mathrm{Re}(x) > 0 \text{ or } (\mathrm{Re}(x) = 0 \text{ and } \mathrm{Im}(x) > 0), \\ -1, & \mathrm{Re}(x) < 0 \text{ or } (\mathrm{Re}(x) = 0 \text{ and } \mathrm{Im}(x) < 0). \end{cases}$$

We obtain

$$\begin{aligned}\mathrm{Im}(\log(z-1)) = \mathrm{Im}(\log(-2 + re^{\mathrm{i}\phi})) &= -\mathrm{csgn}(\mathrm{i}(z-1))\pi - \mathrm{Im}\left(\sum_{k=0}^{\infty} \frac{(re^{\mathrm{i}\phi})^k}{k\ 2^k}\right) \\ &= -\mathrm{csgn}(\mathrm{i}(z-1))\pi - \sum_{k=0}^{\infty} \frac{r^k}{k\ 2^k} \sin(k\phi)\end{aligned}$$

and

$$\begin{aligned}\operatorname{csgn}(\mathrm{i}(z-1)) &= \operatorname{csgn}(\mathrm{i}(u-1+\mathrm{i}v)) = \operatorname{csgn}(-v+\mathrm{i}(u-1)) \\ &= \begin{cases} 1, & -v>0 \text{ or } (v=0 \text{ and } u-1>0) \\ -1, & -v<0 \text{ or } (v=0 \text{ and } u-1<0) \end{cases} \\ &= \left.\begin{cases} 1, & v<0 \text{ or } (v=0 \text{ and } u>1) \\ -1, & v>0 \text{ or } (v=0 \text{ and } u<1) \end{cases}\right\} = \begin{cases} 1, & v<0 \\ -1, & v\geq 0. \end{cases}\end{aligned}$$

We get the last identity, because $|z+1|<1 \Rightarrow u<1$.
Finally, there holds

$$\operatorname{Im}(\log(z+1)) - \operatorname{Im}(\log(z-1)) = \phi - \sigma\pi + \sum_{k=0}^{\infty} \frac{r^k}{k\ 2^k}\sin(k\phi),$$

with $\sigma = 1$ for $v>0$ and $\sigma=-1$ for $v\leq 0$. □

For our implementation we use these series expansions for $|z\pm 1|<0.3$. Again, we need an estimate for the index up to which we evaluate the infinite sums in (2.47) and (2.48). Comparing the Nth summand with the first one, we obtain for $r<0.3$

$$\frac{(r/2)^N(1/N)}{r/2}\left|\frac{\sin(N\phi)}{\sin(\phi)}\right| \leq 0.15^{N-1}\frac{1}{N}\cdot N < 10^{-16} \iff 0.83N > 16.83 \iff N > 21.$$

Thus, for our implementation, we evaluate the sums in (2.47) and (2.48) up to $N=22$.
Figure 2.8 depicts an overview of the different methods we use to calculate the initial values. For the evaluation of the analytic formulas for the initial values, we separate complex and real parts whenever possible. A detailed description of how the initial values are calculated with the analytic formulas is given in Appendix A.

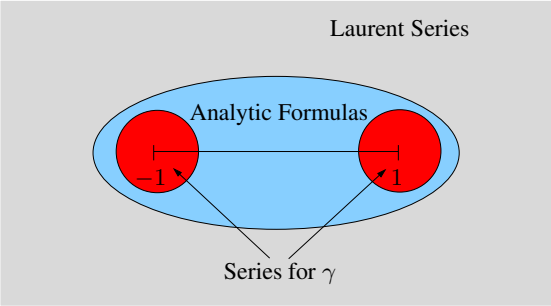


Figure 2.8.: Overview for the calculation of the initial values. In the far field (gray) we use Laurent series; in the near field (blue) we use the analytic formulas, where in neighborhoods of ± 1 we use series expansions of γ (red).

Next we discuss the evaluation of the three-term recurrence relations. First, we observe that the three-term recurrence relations for $\widetilde{Q}_k^m$ and R_k^m can be written in the general form

$$f_{k+s+1}(z) = \frac{2k+1}{k-a}\, z\, f_{k+s}(z) - \frac{k+(a-1)}{k-a} f_{k+s-1}(z), \quad k = k_0, \ldots, N. \tag{2.49}$$

The values of a, k_0 and s for the cases of interest are shown in Table 2.1.

Table 2.1.: Values of the parameters a, k_0, and s in (2.49) for $\widetilde{Q}_k^m$, $m \in \{-2,-1,0,1\}$ and R_k^m, $m \in \{0,1\}$.

	$\widetilde{Q}^{-2}$	$\widetilde{Q}^{-1}$	$\widetilde{Q}^{0}$	$\widetilde{Q}^{1}$	R^0	R^1
k_0	2	1	1	1	1	1
s	0	0	0	0	2	2
a	-3	-2	-1	0	-2	-1

In Section 2.3.3 it is shown that the calculation of minimal solutions using the forward recurrence relation is generally unstable. However, the following numerical experiments show that the calculation is exact up to machine precision inside of an ellipse with foci $\{\pm 1\}$ and minor axis $\rho_{min}(N)$. Here, N is the index up to which we evaluate the recurrence relation. We determine $\rho_{min}(N)$ using numerical experiments in the following way. We evaluate the forward recurrence relation on an elliptical grid and compare the solution to a reference solution that is obtained using the high precision libraries `MPFR` [FHL+07] and `MPC` [EGTZ12]. The elliptical grid is given by

$$\mathcal{G}_E = \left\{ w_{j,k} := J(z_{j,k}),\ z_{j,k} = r_j e^{\mathrm{i}\alpha_k}, j = 1, \ldots, 200, k = 1, \ldots, 100 \right\},$$

where the points $r_j \in [1, r_{max}]$ are chosen to be logarithmic, i.e. $r_j = 1 + 10^{a_j}$, $a_j \in [-4, 0.5]$ equidistant, and $\alpha_k \in [-0.99\ \pi, 0.99\ \pi]$ are equidistant. Now for each maximal index $N \in [3, 1000] \cap \mathbb{N}$, we evaluate the forward recurrence relation up to N and calculate the relative error

$$e_N(z) := \max_{k=3,\ldots,N} \frac{|f_k(z) - f_k^{MPC}(z)|}{|f_k^{MPC}(z)|}.$$

Here f_k is one of the functions $\widetilde{Q}_k^m$ ($m \in \{-2,-1,0,1\}$) or R_k^m ($m \in \{0,1\}$), and $f_k^{MPC}(z)$ is the respective reference solution. Then we determine the minimum minor axis $\rho_{min}(N)$ such that the relative error is smaller than a given tolerance tol inside of the ellipse with foci $\{\pm 1\}$ and minor axis $\rho_{min}(N)$.

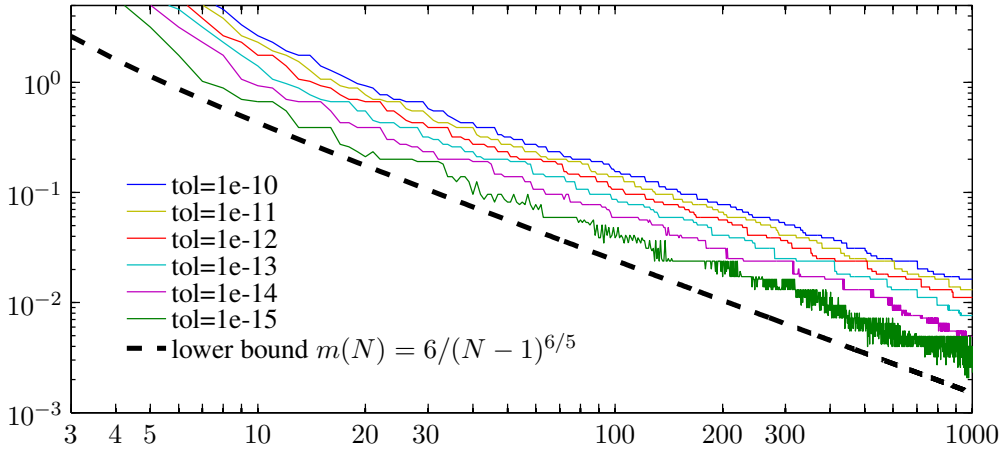


Figure 2.9.: Minor axis $\rho_{min}(N)$ of the ellipse inside of which the relative error for the forward recurrence is smaller than the tolerance *tol* for the function $R_1^k(z)$ over N. The relative error is calculated on the elliptical grid $\mathcal{G}_E$.

Figure 2.9 shows the results for the function $R_k^1(z)$. We see that the function $m(N) := 6/(N-1)^{6/5}$ is a lower bound for the minor axis of the ellipse.

Remark 2.3.19. For the functions $\widetilde{Q}_k^m(z)$, $m \in \{-1, 0, 1\}$, and $R_k^0(z)$, we obtain the same lower bound, $m(N) := 6/(N-1)^{6/5}$. This bound is taken for the implementation.

Outside of the ellipse we use Gautschi's continued fraction algorithm (Algorithm 2.2) and determine the parameter ν using the estimate (2.45). Using the general forward algorithm (2.49), we obtain a general form of Gautschi's continued fraction algorithm for the recurrence relations for $\widetilde{Q}_k^m$, $m \in \{-2, -1, 0, 1\}$, and R_k^m, $m \in \{0, 1\}$; see Algorithm 2.3. Implementing Algorithm 2.3 in this form requires a temporary array `r` of the size ν for the values of r_k^ν . However, ν depends on z. Thus, for each z, one would have to reallocate the array `r`. In order to avoid this, we rewrite the algorithm and obtain Algorithm 2.4. In Algorithm 2.4, we use only one variable for r for $k = \nu, \nu-1, \ldots, N$ and overwrite the values. The values r_k^ν, $k = N-1, \ldots, 1+s$ are stored in the array for f^ν, i.e. we set $f_k^\nu = r_{k-1}$, $k = N, N-1, \ldots, 2+s$. In the forward recurrence we then simply overwrite the values of f_k^ν. Using this algorithm, we do not have to allocate an array for the values of r at all, but only need one extra double variable. For our implementation, we use Algorithm 2.4.

All implementations are collected in a `C` library. For the implementation, we try to avoid complex operations whenever possible, i.e. we calculate real and imaginary parts separately. Furthermore, we provide `mex` files for each function such that the functions can be called from MATLAB. The `mex` function allows for the evaluation of the respective function at multiple

Algorithm 2.3 Modified Gautschi's continued fraction algorithm that can be used for the computation of $\widetilde{Q}_k^m$ or R_k^m, $k \leq N$. The variables a, k_0, and s are defined in Table 2.1.

INPUT: Maximum index $N \in \mathbb{N}$, tolerance $\varepsilon > 0$.

OUTPUT: Minimal solution f_k, $k = 0, \ldots, N$, of (2.49).

Compute ν according to (2.45).
Set $r_\nu^\nu = 0$.
for $k = \nu - s, \ldots, k_0 + 1$ **do**
    Set $r_{k+s-1}^\nu = \frac{k^2-(a-1)^2}{(4k^2-1)(z-r_{k+s}^\nu)}$.
end for
for $k = 0, \ldots, k_0 + s + 1$ **do**
    Compute the initial values f_k^ν as described in Proposition 2.3.4 and Remark 2.3.2 (vi).
end for
for $k = k_0 + 1, \ldots, N - s$ **do**
    Set $f_{k+s}^\nu = \frac{2k-1}{k-a}\, r_{k+s-1}^\nu\, f_{k+s-1}^\nu$.
end for

Algorithm 2.4 Optimized modified Gautschi's continued fraction algorithm that can be used for the computation of $\widetilde{Q}_k^m$ or R_k^m, $k \leq N$. The variables a, k_0, and s are defined in Table 2.1.

INPUT: Maximum index $N \in \mathbb{N}$, tolerance $\varepsilon > 0$.

OUTPUT: Minimal solution f_k, $k = 0, \ldots, N$, of (2.49).

Compute ν according to (2.45).
Set $r = 0$.
for $k = \nu - s, \nu - s - 1, \ldots, N$ **do**
    Set $r = \frac{k^2-(a-1)^2}{(4k^2-1)(z-r)}$.
end for
Set $f_N^\nu = r$.
for $k = N - s, N - s - 1, \ldots, k_0 + s + 1$ **do**
    Set $f_{k+s-1}^\nu = \frac{k^2-(a-1)^2}{(4k^2-1)(z-f_{k+s}^\nu)}$.
end for
for $k = 0, \ldots, k_0 + s + 1$ **do**
    Compute the initial values f_k^ν as described in Proposition 2.3.4 and Remark 2.3.2 (vi).
end for
for $k = k_0 + 1, \ldots, N - s$ **do**
    Set $f_{k+s}^\nu = \frac{2k-1}{k-a}\, f_{k+s}^\nu\, f_{k+s-1}^\nu$.
end for

points that are given in a complex array. In the `mex` functions, we loop over all points and call the respective function for each point. To speed up the calculations, we parallelize the loop using `openMP`. Numerical experiments (see Table 2.2) show that the computation times for the assembly of $(\widetilde{Q}_k^{-1}(z))_{k=0,\dots N}$ are proportional to N^α with $1 \leq \alpha \leq 1.05$. Furthermore, it can be seen that the parallelization scales well.

Table 2.2.: Computation times for the evaluation of $\widetilde{Q}_k^{-1}(z)$ for $k = 0, \dots, N$ and 40000 uniformly distributed points z with $-2 \leq \mathrm{Re}(z), \mathrm{Im}(z) \leq 2$ (near field) and 40000 logarithmically distributed points with $10 \leq \mathrm{Re}(z), \mathrm{Im}(z) \leq 10^5$ (far field), respectively. All computations are done on a desktop computer with an AMD PhenomII X6 processor, 16GB Ram, and the Ubuntu 10.04 operating system

	near field				far field			
# threads	1	2	3	4	1	2	3	4
$N = 32$	0.099	0.050	0.040	0.033	0.063	0.032	0.024	0.021
$N = 64$	0.146	0.073	0.059	0.050	0.111	0.059	0.045	0.039
$N = 128$	0.235	0.119	0.093	0.077	0.234	0.124	0.093	0.079
$N = 256$	0.412	0.212	0.160	0.134	0.438	0.229	0.173	0.149
$N = 512$	0.908	0.425	0.311	0.268	0.876	0.412	0.311	0.261
$N = 1024$	1.886	0.879	0.655	0.574	1.678	0.794	0.573	0.488
$N = 2048$	3.877	1.805	1.458	1.177	3.256	1.490	1.106	0.943

2.4. Integrals of Modified Associated Legendre Functions of the Second Kind

It will be shown in Section 3.4 that the computation of Galerkin entries can be reduced to the evaluation of certain integrals of the functions $\widetilde{Q}_k^m$ and R_k^m. These integrals are introduced in this section. We derive recurrence relations and state initial values. Finally, we describe how the integrals are evaluated accurately.

In the following $a, b \in \mathbb{C}$ are complex numbers, and we denote the line segment between $a+b$ and $a-b$ by

$$T(a,b) := \mathrm{conv}(a-b, a+b) = \{a + bs \,:\, s \in [-1,1]\}.$$

Furthermore, we write $\widehat{T} = [-1,1] = T(0,1)$.

2.4.1. Definition and Recurrence Relations

Definition 2.4.1. Let $a, b \in \mathbb{C}$ such that either $T(a,b) \cap \widehat{T} = \emptyset$ or $T(a,b) \cap \widehat{T} \in \{\pm 1\}$ or $T(a,b) = \widehat{T}$. For $k, \ell, m \in \mathbb{N}_0$, we define

$$I_{k,\ell}^{-1}(a,b) := \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log(a+bs-t)\, dt\, ds, \tag{2.50}$$

$$I_{k,\ell}^{m}(a,b) := \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \frac{1}{(a+bs-t)^{m+1}}\, dt\, ds, \tag{2.51}$$

and for $k \in \mathbb{N}$ and $\ell, m \in \mathbb{N}_0$, we define

$$O_{k,\ell}^{-1}(a,b) := \int_{-1}^{1} P_k(s) \int_{-1}^{1} N_\ell(t) \log(a+bs-t)\, dt\, ds, \tag{2.52}$$

$$O_{k,\ell}^{m}(a,b) := \int_{-1}^{1} P_k(s) \int_{-1}^{1} N_\ell(t) \frac{1}{(a+bs-t)^{m+1}}\, dt\, ds, \tag{2.53}$$

$$U_{k,\ell}^{-1}(a,b) := \int_{-1}^{1} N_k(s) \int_{-1}^{1} P_\ell(t) \log(a+bs-t)\, dt\, ds, \tag{2.54}$$

$$U_{k,\ell}^{m}(a,b) := \int_{-1}^{1} N_k(s) \int_{-1}^{1} P_\ell(t) \frac{1}{(a+bs-t)^{m+1}}\, dt\, ds. \tag{2.55}$$

Remark 2.4.2. For the assembly of Galerkin matrices in our implementation of hp-BEM for the Laplace equation, we need the functions $I_{k,\ell}^{-1}(a,b)$, $O_{k,\ell}^{0}(a,b)$, and $U_{k,\ell}^{0}(a,b)$. Specifically, for each combination of two boundary pieces T_i and T_j, we obtain a $(p_i+1) \times (p_j+1)$ sub-block of the Galerkin matrix, where p_i and p_j are the polynomial degrees of the ansatz space on T_i and T_j, respectively.

In order to compute one sub-block, we have to compute linear combinations of the above double integrals, where $k = p_0, \ldots, p_i + p_1$, $\ell = p_0, \ldots, p_j + p_1$. Here p_0 and p_1 are 0 or 1 depending which double integral is computed.

In Section 3.4 we show that the two elements T_i and T_j can be parametrized, which leads to a reference situation in the complex plane, where $T_i = [-1, 1]$ and $T_j = T(a,b)$, $a, b \in \mathbb{C}$ with $|b| < 1$ (we assume without loss of generality that $|T_i| \leq |T_j|$). Here the complex number a describes the distance of the midpoints of T_i and T_j scaled by the length of T_j, and b describes the length of T_i scaled by the length of T_j and the relative angle of T_j with respect to T_i. Figure 3.4 illustrates the situation.

We summarize some important properties of $I_{k,\ell}^{m}(a,b)$ in the following proposition. The results are proven in detail in [Wei10]. We only give a short outline of the proofs. In order to simplify the notation, we write $I_{k,\ell}^{m}$ instead of $I_{k,\ell}^{m}(a,b)$ whenever it is clear what the arguments are.

Proposition 2.4.3. *Let $a, b \in \mathbb{C}$ with $T(a,b) \cap \widehat{T} = \emptyset$ and $b \neq 0$.*

(i) Let $k, \ell, m \in \mathbb{N}_0$. There holds for $b \in \mathbb{C} \setminus (-\infty, 0]$

$$I_{k,\ell}^{-1}(a,b) = I_{\ell,k}^{-1}\left(-\frac{a}{b}, \frac{1}{b}\right) + 4\delta_{k,0}\delta_{\ell,0}\log(-b), \tag{2.56}$$

$$I_{k,\ell}^{m}(a,b) = \frac{(-1)^{m+1}}{b^{m+1}} I_{\ell,k}^{m}\left(-\frac{a}{b}, \frac{1}{b}\right). \tag{2.57}$$

(ii) For $m \in \mathbb{N}$ and $\ell \in \mathbb{N}_0$, there holds

$$I_{0,\ell}^{-1}(a,b) = \frac{1}{b}\left(\widetilde{Q}_\ell^{-2}(a+b) - \widetilde{Q}_\ell^{-2}(a-b)\right) - 4\delta_{\ell,0}, \tag{2.58}$$

$$I_{0,\ell}^{0}(a,b) = \frac{1}{b}\left(\widetilde{Q}_\ell^{-1}(a+b) - \widetilde{Q}_\ell^{-1}(a-b)\right), \tag{2.59}$$

$$I_{0,\ell}^{m}(a,b) = \frac{1}{m\,b}\left(\widetilde{Q}_\ell^{m-1}(a+b) - \widetilde{Q}_\ell^{m-1}(a-b)\right). \tag{2.60}$$

(iii) For $k, \ell \in \mathbb{N}$ and $m \in \{-1\} \cup \mathbb{N}_0$, there holds

$$b\,(2\ell+1)\left(I_{k+1,\ell}^{m} - I_{k-1,\ell}^{m}\right) = -(2k+1)\left(I_{k,\ell+1}^{m} - I_{k,\ell-1}^{m}\right). \tag{2.61}$$

(iv) For $k, \ell \in \mathbb{N}$ and $m \in \{-1\} \cup \mathbb{N}_0$, there holds

$$\begin{aligned} I_{k+1,\ell}^{m} = {} & \frac{(2k+1)}{b(k+1)(2\ell+1)}\left[(\ell-m+1)I_{k,\ell+1}^{m} + (\ell+m)I_{k,\ell-1}^{m}\right] \\ & - \frac{1}{k+1}\left[k\, I_{k-1,\ell}^{m} - \frac{a(2k+1)}{b}\, I_{k,\ell}^{m}\right] - \frac{4}{3b}\delta_{k,0}\delta_{\ell,1}\delta_{m,-1}. \end{aligned} \tag{2.62}$$

(v) For $k, \ell \in \mathbb{N}$ and $m \in \{-1\} \cup \mathbb{N}_0$, there holds

$$\begin{aligned} & (\ell-k-m+1)I_{k,\ell+1}^{m} \\ & \quad = (2\ell+1)aI_{k,\ell}^{m} - (\ell+k+m)I_{k,\ell-1}^{m} + b(2\ell+1)I_{k+1,\ell}^{m}, \end{aligned} \tag{2.63}$$

$$\begin{aligned} & (\ell+k-m+2)I_{k,\ell+1}^{m} \\ & \quad = (2\ell+1)aI_{k,\ell}^{m} - (\ell-k+m)I_{k,\ell-1}^{m} + b(2\ell+1)I_{k-1,\ell}^{m}, \end{aligned} \tag{2.64}$$

$$\begin{aligned} & (k-\ell-m+1)I_{k+1,\ell}^{m} \\ & \quad = -\frac{(2k+1)a}{b}I_{k,\ell}^{m} - (k+\ell+m)I_{k-1,\ell}^{m} + \frac{2k+1}{b}I_{k,\ell+1}^{m}, \end{aligned} \tag{2.65}$$

$$\begin{aligned} & (k+\ell-m+2)I_{k+1,\ell}^{m} \\ & \quad = -\frac{(2k+1)a}{b}I_{k,\ell}^{m} - (k-\ell+m-1)I_{k-1,\ell}^{m} + \frac{2k+1}{b}I_{k,\ell-1}^{m}. \end{aligned} \tag{2.66}$$

Proof. Part (i) follows directly from switching the order of integration and dividing the term $a + bs - t$ by b. To show part (ii), we have to calculate the inner integrals analytically and then plug in the definition of $\widetilde{Q}_\ell^m$. The first recurrence relation (iii) can be shown using integration

by parts and the property for the antiderivative of the Legendre polynomials (2.5). To prove (iv), we write

$$I^m_{k+1,\ell}(a,b) = \int_{-1}^{1} P_{k+1}(s)\widetilde{Q}^m_\ell(a+bs)\,ds.$$

Plugging in the recurrence relation for the Legendre polynomials and then the recurrence relation for the function $\widetilde{Q}^m_k$, we obtain the result. The recurrence relations in part (v) are linear combinations of the recurrence relations in part (iv). □

The properties of the functions $U^m_{k,\ell}(a,b)$ and $O^m_{k,\ell}(a,b)$ are summarized in the following proposition.

Proposition 2.4.4. *Let $a, b \in \mathbb{C}$ with $T(a,b) \cap \widehat{T} = \emptyset$ and $b \neq 0$.*

(i) Let $\ell, m \in \mathbb{N}_0$ and $k \in \mathbb{N}$. There holds

$$O^{-1}_{k,\ell}(a,b) = U^{-1}_{\ell,k}\left(-\frac{a}{b}, \frac{1}{b}\right) + A_\ell \delta_{k,0}\log(-b), \tag{2.67}$$

$$O^m_{k,\ell}(a,b) = \frac{(-1)^{m+1}}{b^{m+1}}\, U^m_{\ell,k}\left(-\frac{a}{b}, \frac{1}{b}\right), \tag{2.68}$$

with $A_\ell = 2$ if $\ell \in \{1,2\}$, $A_\ell = \frac{4}{3}$ if $\ell = 3$, and $A_\ell = 0$ $\ell \geq 4$.

(ii) For $m \in \mathbb{N}$ and $\ell \in \mathbb{N}$, there holds

$$O^0_{0,\ell}(a,b) = \frac{1}{b}\left(R^{-1}_\ell(a+b) - R^{-1}_\ell(a-b)\right), \tag{2.69}$$

$$O^m_{0,\ell}(a,b) = \frac{1}{m\,b}\left(R^{m-1}_\ell(a+b) - R^{m-1}_\ell(a-b)\right). \tag{2.70}$$

(iii) For $k \in \mathbb{N}_0$ and $m \in \mathbb{Z}$, there holds

$$O^m_{k,1}(a,b) = \frac{I^m_{k,0}(a,b) - I^m_{k,1}(a,b)}{2}, \qquad O^m_{k,2}(a,b) = \frac{I^m_{k,0}(a,b) + I^m_{k,1}(a,b)}{2},$$
$$O^m_{k,\ell}(a,b) = \frac{I^m_{k,\ell-1}(a,b) - I^m_{k,\ell-3}(a,b)}{2\ell-3} \quad (\ell \geq 3).$$

$$U^m_{1,\ell}(a,b) = \frac{I^m_{0,\ell}(a,b) - I^m_{1,\ell}(a,b)}{2}, \qquad U^m_{2,\ell}(a,b) = \frac{I^m_{0,\ell}(a,b) + I^m_{1,\ell}(a,b)}{2},$$
$$U^m_{k,\ell}(a,b) = \frac{I^m_{k-1,\ell}(a,b) - I^m_{k-3,\ell}(a,b)}{2k-3} \quad (k \geq 3).$$

Proof. The proofs of (i) and (ii) are similar to the proof of the previous proposition. Part (iii) follows directly with the relation $N_1(t) = \frac{1}{2}(P_0(t) - P_1(t))$ and $N_2(t) = \frac{1}{2}(P_0(t) + P_1(t))$. For detailed derivations see [Wei10]. □

Remark 2.4.5. (i) Using the relation between $\widetilde{Q}_k^m$ and R_k^m one can derive recurrence relations for $O_{k,\ell}^m$ that are similar to the ones for $I_{k,\ell}^{-1}$ stated in Proposition 2.4.3 (iii)-(v). Using the switching property from Proposition 2.4.4 (i), similar recurrence relations can be derived for $U_{k,\ell}^m$.

(ii) The recurrence relations from Proposition 2.4.3 (iii)-(iv) can be described by Table 2.3. The stars stand for entries that are used to calculate the entry described by o. For example, with the recurrence relation (2.63), the value of $I_{k,\ell+1}^{-1}$ can be computed by linearly combining the values $I_{k,\ell-1}^{-1}$, $I_{k,\ell}^{-1}$, and $I_{k+1,\ell}^{-1}$. For the functions $O_{k,\ell}^m$, one can obtain similar recurrence relations; for $U_{k,\ell}^m(a,b)$ the schemes are rotated by 90 degrees (see part (i) of this remark).

(iii) The functions $I_{k,\ell}^m(a,b)$, $m \in \{-1,0\}$, can be continuously extended for $a,b \in \mathbb{C}$ with $a \pm b = \pm 1$. This follows because initial values for the recurrence relations for $I_{k,\ell}^m(a,b)$, $m \in \{-1,0\}$, are calculated using the functions $\widetilde{Q}_k^{-1}(z)$ and $\widetilde{Q}_k^{-2}(z)$ and these functions can be continuously extended for $z = \pm 1$. For $m \geq 1$, the functions $I_{k,\ell}^m(a,b)$ have singularities for $a,b \in \mathbb{C}$ with $a \pm b = \pm 1$.

(iv) We can extend functions $U_{k,\ell}^0(a,b)$ and $O_{k,\ell}^0(a,b)$ continuously for $a,b \in \mathbb{C}$ with $a \pm b = \pm 1$. For $m \geq 1$, the functions $U_{k,\ell}^m(a,b)$ and $O_{k,\ell}^m(a,b)$ have singularities for $a \pm b = \pm 1$.

2.4.2. Implementation

We derive algorithms to assemble the matrices

$$\begin{aligned}
&\left(I_{k,\ell}^m(a,b)\right)_{k,\ell}, \quad k = 0,\dots,N,\ \ell = 0,\dots,M,\ m \in \{-1,0\}\\
&\left(O_{k,\ell}^0(a,b)\right)_{k,\ell}, \quad k = 0,\dots,N,\ \ell = 1,\dots,M,\\
&\left(U_{k,\ell}^0(a,b)\right)_{k,\ell}, \quad k = 1,\dots,N,\ \ell = 0,\dots,M,
\end{aligned}$$

for $a,b \in \mathbb{C}$ with $T(a,b) \cap \widehat{T} = \emptyset$ and $b \neq 0$, but we include the cases $a \pm b = \pm 1$. Numerical experiments show that the difference schemes derived in the last section (see Proposition 2.4.3

Table 2.3.: Schematic description of the recurrence relations from Proposition 2.4.3.

$I_{k,\ell}^m(a,b)$	(2.61)	(2.62)	(2.63)	(2.64)	(2.65)	(2.66)	Row
Scheme	$\begin{matrix} & * & \\ * & & * \\ & o & \end{matrix}$	$\begin{matrix} & * & \\ * & * & * \\ & o & \end{matrix}$	$\begin{matrix} & & \\ * & * & o \\ & * & \end{matrix}$	$\begin{matrix} & * & \\ * & * & o \\ & & \end{matrix}$	$\begin{matrix} & * & \\ * & * & \\ & o & \end{matrix}$	$\begin{matrix} & * & \\ & * & * \\ & o & \end{matrix}$	k
Column	ℓ	ℓ	ℓ	ℓ	ℓ	ℓ	

and 2.4.4) are highly unstable. Using a decoupling algorithm (see [Mat83]), one can compute the first two rows and columns as well as the diagonal entries of the matrices in a stable way. In some cases the decoupling algorithm can be used to assemble the full matrices in a stable way [Wei10]. However, the problem of finding a stable algorithm to compute all matrix entries for arbitrary $a, b \in \mathbb{C}$ remains unsolved up to now. For the lack of a stable algorithm, we use the unstable difference schemes derived in the previous section.
Algorithm 2.5 is used to calculate the matrix $(I^m_{k,\ell})_{k,\ell}$ for $m \in \{-1, 0\}$. For the calculation of the matrix $(O^0_{k,\ell})_{k,\ell}$, we use Algorithm 2.6, and for the assembly of the matrix $(U^0_{k,\ell})_{k,\ell}$, we use Algorithm 2.7.

Algorithm 2.5 Computation of the matrix $(I^m_{k,\ell})_{k=0,\dots,N,\ell=0,\dots,M}$.

1: Calculate the first row $I^m_{0,\ell}$ for $\ell = 0, \dots, M$ using Proposition 2.4.3 (ii).
2: Calculate the first column $I^m_{k,0}$ for $k = 0, \dots, N$ using Proposition 2.4.3 (ii) together with the switching property Proposition 2.4.3 (i).
3: Calculate the second column:

Calculate $I^m_{1,1}$ using (2.66) rearranged to the form $\begin{matrix} * \\ * \\ * \end{matrix} \quad o$

Calculate $I^m_{k,1}$, $k = 2, \dots N$ using (2.65).

4: Calculate the remaining entries using (2.64).

Algorithm 2.6 Computation of the matrix $(O^0_{k,\ell})_{k=0,\dots,N,\ell=0,\dots,M}$.

1: Calculate the first row $O^0_{0,\ell}$ for $\ell = 1, \dots, M$ using Proposition 2.4.4 (ii).
2: Calculate the remaining part using Proposition 2.4.4 (iii).

Algorithm 2.7 Computation of the matrix $(U^0_{k,\ell})_{k=0,\dots,N,\ell=0,\dots,M}$.

1: Calculate the first column $U^0_{k,0}$ for $k = 1, \dots, N$ using Proposition 2.4.4 (ii) and the switching property (2.68).
2: Calculate the remaining part using Proposition 2.4.4 (iii).

These algorithms are also valid for $a, b \in \mathbb{C}$ with $a \pm b = \pm 1$ (see Remark 2.4.2). In order to still obtain matrices whose entries are accurate up to machine precision, we use the `C` libraries `MPFR` and `MPC` for the calculations. By choosing the number of digits we use for the calculations large enough, we can achieve that the relative rounding errors stay close to machine precision. The straight-forward way is to choose a constant number of digits for all computations. However, the number of digits that is needed such that the relative rounding errors are close to machine precision depends on $a, b \in \mathbb{C}$ and the maximum indices M and N. Thus, choosing a constant

number of digits for all computations might not be valid for all cases while being far too high in some cases. Therefore, we perform numerical experiments to obtain an upper bound for the number of digits needed to compute the functions defined in the previous section such that the values are exact up to fifteen significant digits. The goal is to obtain an upper bound that is as efficient as possible and easy to compute. First, we introduce some restrictions to the values of $a, b \in \mathbb{C}$. We will see in Chapter 3 that these restrictions are reasonable for our application to hp-BEM.

Assumption 2.4.6. For our numerical experiments, we consider $a, b \in \mathbb{C}$, $N, M \in \mathbb{N}$ such that

- $N = M$, i.e. we consider quadratic matrix blocks,
- $|b| \leq 1$ (see Remark 2.4.2),
- exactly one of the following conditions holds:
  1. (near field case) $a + b = \pm 1$ or $a - b = \pm 1$
  2. (far field case) $r(a, b) \geq 1$, where $r(a, b)$ is the minor axis of the largest ellipse with foci ± 1 such that the line segment $T(a, b)$ is completely outside of this ellipse (see Figure 2.10).

Note that the near field case corresponds to the case of neighboring elements in the Galerkin method and the far field case corresponds to the case of disjoint elements. Furthermore, the maximal index N corresponds to the maximum polynomial degree on the two boundary elements under consideration; see Remark 2.4.2.

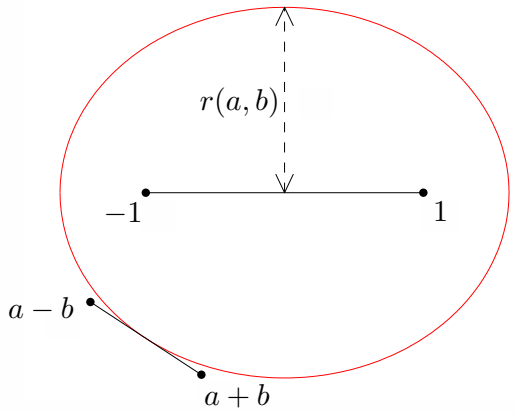


Figure 2.10.: Ellipse distance $r(a, b)$ for $a = -1 - \mathrm{i}$ and $b = 0.4 - 0.25\mathrm{i}$.

We determine an upper bound for the number of digits for the function $I_{k,\ell}^{-1}(a,b)$ and then test this upper bound for the other functions of interest. Given $a, b \in \mathbb{C}$ and a maximum index N, we determine the number of digits D needed for the stable evaluation of $I_{k,\ell}^{-1}(a,b;D)$, $k,\ell = 0,\dots,N$, using Algorithm 2.8. Here, the additional argument D indicates that all computations are performed with a precision of D digits.

Algorithm 2.8 Algorithm for the determination of the number of digits D needed for the stable evaluation of $I_{k,\ell}^{-1}$ for $k,\ell = 0,\dots,N$.

1: Calculate the reference solution $I_{k,\ell}^{-1}(a,b;\widehat{D})$ using $\widehat{D}$ digits with

$$\widehat{D} = \begin{cases} 10000, & \text{if } N \le 100, \\ 20000, & \text{if } 100 < N \le 4000. \end{cases}$$

2: Starting with $D = 64$, set $D = 2 \cdot D$ until the relative error

$$e_D = \frac{|I_{k,\ell}^{-1}(a,b;D) - I_{k,\ell}^{-1}(a,b;\widehat{D})|}{|I_{k,\ell}^{-1}(a,b;\widehat{D})|}$$

is smaller than 10^{-15}.

3: Perform a bisection algorithm with initial values $\overline{D} = D$ and $\underline{D} = D/2$:

4: **while** $\overline{D} - \underline{D} > 1$ **do**

5: Set $D_m = (\overline{D} + \underline{D})/2$ and compute the relative error e_{D_m}.

6: If $e_{D_m} > 10^{-15}$, set $\underline{D} = D_m$, else set $\overline{D} = D_m$.

7: **end while**

We start with numerical experiments for the near field. Because $I_{k,\ell}^{-1}(a,b) = (-1)^{k+1} I_{k,\ell}^{-1}(a,-b)$, the four cases $a \pm b = \pm 1$ can be reduced to $a - b = 1$ and $a + b = -1$. Using the method described above, we determine the number of digits D for different values of $a, b \in \mathbb{C}$ and $N \in \mathbb{N}$. We set $a + b = 1$ and vary the absolute value $|b|$, the angle $\alpha = \arg(b)$, and N. In more detail, we perform test runs for 35 uniform values of $\alpha \in [-0.9\pi, 0.9\pi]$ and all pairs

$$(L,N) \in P_{near} = \{0.5, 0.6, \dots, 1\} \times (\{2^k,\ k = 2,\dots,10\} \cup \{24, 48, 96, 192\}).$$

The choice of $L = |b|$ implies for the application to hp-BEM that the sizes of neighboring elements can not differ by a factor of more than 2, which is a reasonable restriction on the boundary mesh.

Figure 2.11 shows plots of D over α for different values of L and N. We see that for large N, the number of digits D can be modeled by

$$\hat{D}(\alpha, L, N) = A(L,N) \cdot |\alpha|^{B(L,N)} + C(L,N). \tag{2.71}$$

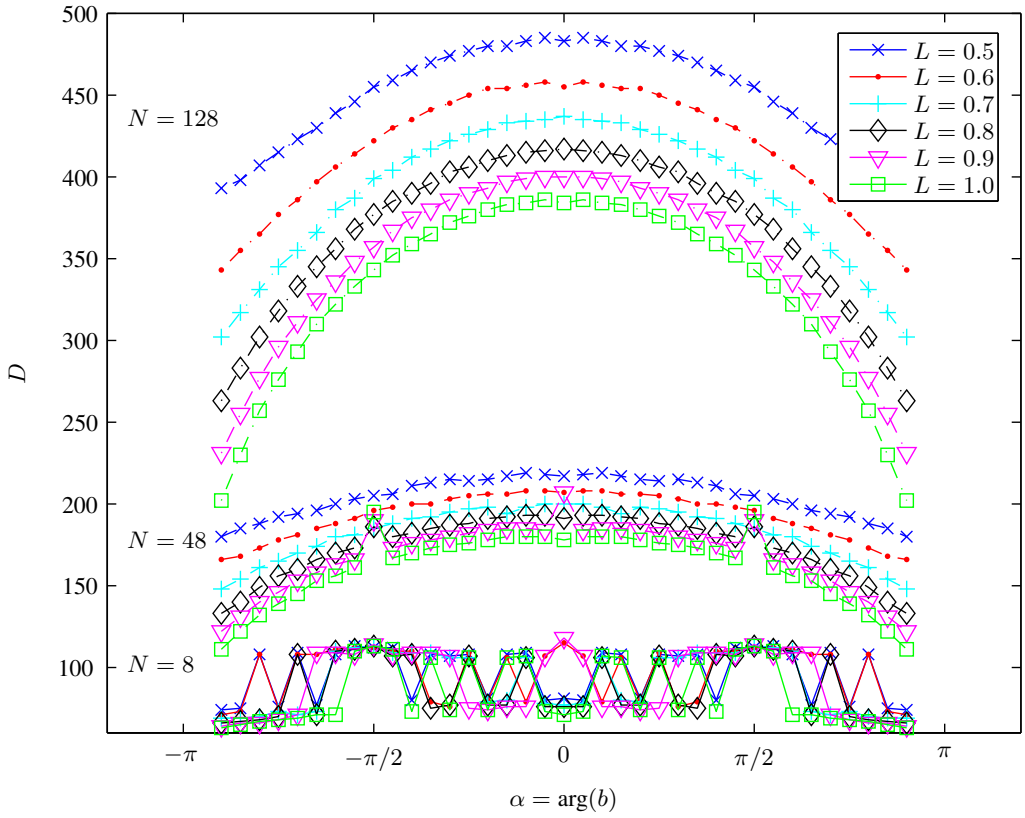


Figure 2.11.: Number of digits D such that the calculation of $I^{-1}_{k,\ell}(a,b)$ $(k,\ell = 0,\ldots,N)$ is exact up to the first fifteen digits over $\alpha = \arg(b)$ for different values of $L = |b|$ and $N = 8$ (sold lines), $N = 48$ (dashed lines) and $N = 128$ (dash-dotted lines). The parameter a is chosen such that $a + b = 1$.

However, for small N this model is not accurate (see e.g. solid lines in Figure 2.11). Furthermore, for some values of L and N, we have to treat the special cases $\alpha = \pm\pi/2$ and $\alpha = 0$, as shown by the peaks in the dashed lines in Figure 2.11. We observe these peaks for $N \leq 64$. First, we determine the coefficients $A(L,N)$, $B(L,N)$ and $C(L,N)$ for $N \geq 24$ and omit the cases $\alpha \in \{\pm\pi/2, 0\}$. In order to fit the model (2.71) to the measurements, we use the MATLAB curve fitting tool `fit`. Internally, MATLAB solves a nonlinear least-squares problem. For each pair (L,N), we obtain values for $A(L,N)$, $B(L,N)$, and $C(L,N)$. Figure 2.12 shows the three coefficients plotted over N for the different values of L. The figure also shows the coefficient of determination R^2 for all curve fits. The fact that all values of R^2 are very close to one shows that the model (2.71) is appropriate for the data. The next step is to determine how $A(L,N)$, $B(L,N)$ and $C(L,N)$ depend on L and N. From Figure 2.12 we see that the values of $B(L,N)$ are almost constant with respect to N. In fact, we find that $B(L,N) = 1.62 \cdot L + 1$ is a good match. For the coefficients $A(L,N)$ and $C(N,L)$, we use the

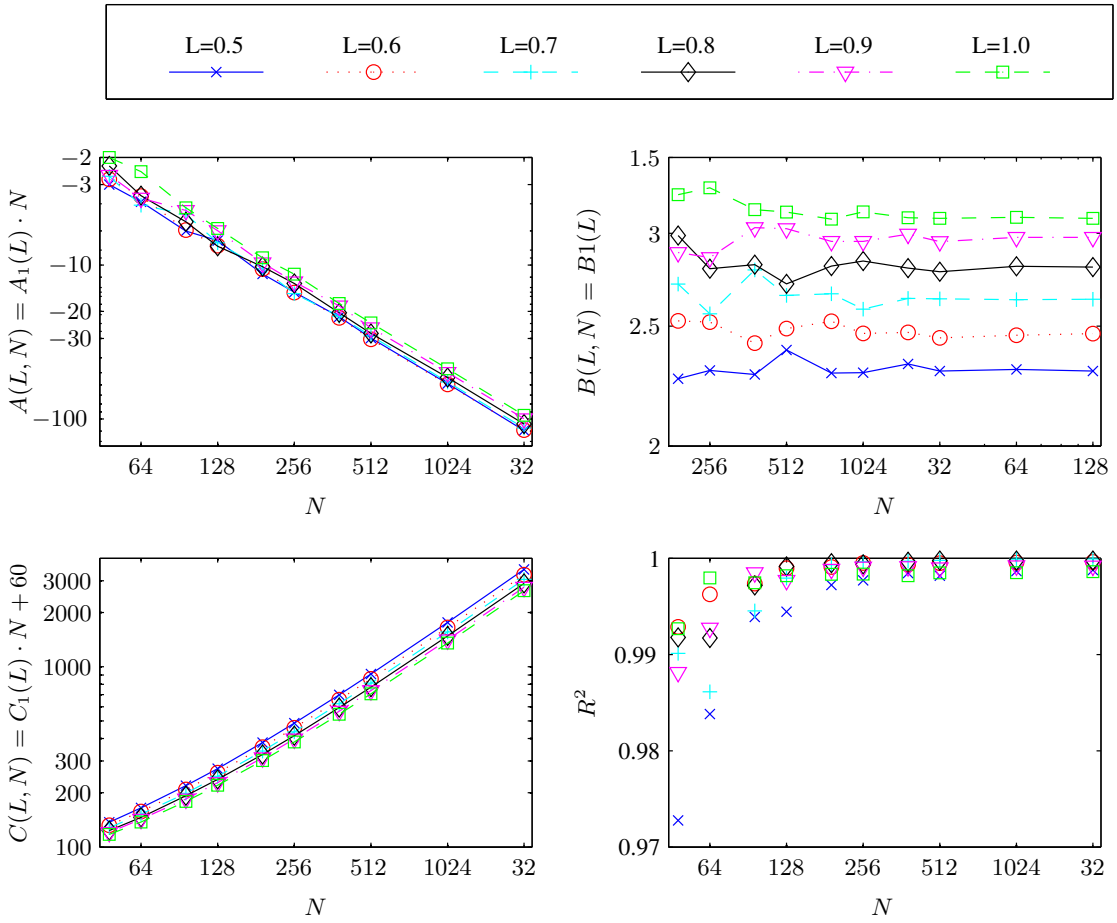


Figure 2.12.: Coefficients $A(L,N)$ (left top), $B(L,N)$ (right top), and $C(L,N)$ (left bottom) over N for different values of L. Right bottom: Coefficient of determination for the nonlinear regression over N for all values of L.

models $A(L,N) = A_1(L)\cdot N$ and $C(L,N) = C_1(L)\cdot N+60$, respectively. Again, we determine the values of $A_1(L)$ and $C_1(L)$ for $L \in \{0.5, 0.6, \ldots, 1\}$ using the MATLAB tool `fit`. For these curve fits, we obtain coefficients of determination $R^2 > 0.999$, which shows that the models for A and C are appropriate. Figure 2.13 shows the coefficients $A_1(L)$ and $C_1(L)$ over L. Performing curve fits for $A_1(L)$ and $C_1(L)$, we obtain the models $A_1(L) = 0.13(L-0.5)^{2.13} - 0.12$ and $C_1(L) = 3.23 \cdot e^{-1.6L} + 1.88$, respectively.

Overall, we obtain the model

$$\widehat{D}(\alpha, L, N) = \left(0.13(L-0.5)^{2.13} - 0.12\right) \cdot N \cdot |\alpha|^{1.62\cdot L+1} + (3.23 \cdot e^{-1.6L} + 1.88) \cdot N + 60,$$

which is valid for for $a+b=1$, $N \geq 24$ and $\alpha \notin \{0, \pm\pi/2\}$. Since we want to ensure that our model computes an upper bound for the number of digits, we add 5%. In order to include the

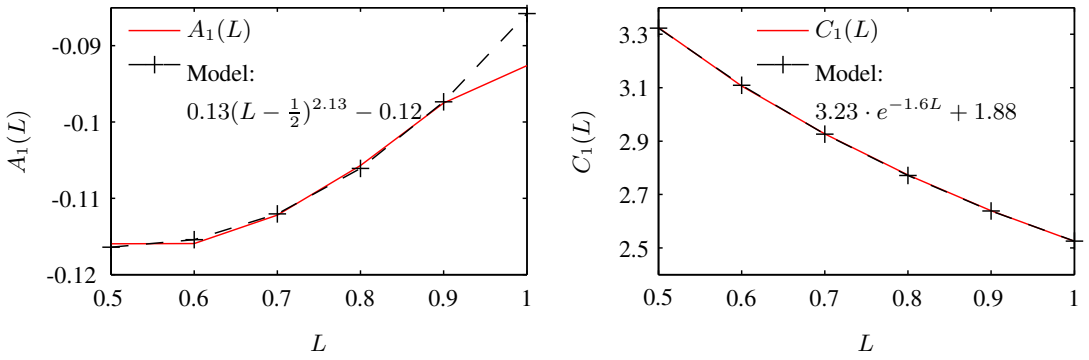


Figure 2.13.: Coefficients $A_1(L)$ and $C_1(L)$ over L (red lines) together with the fitted curve (black dashed line).

cases $N < 24$ and $\alpha \in \{0, \pm\pi/2\}$, we use

$$\widetilde{D}(\alpha, L, N) = \max\{120, 1.05 \cdot \hat{D}(\alpha, N, L)\} + 30\, \mathbb{I}_{\{|\alpha|<0.01 \text{ or } |\alpha\pm\pi/2|<0.01\}},$$

where $\mathbb{I}$ is the indicator function.

We validate our model for all pairs $(L, N) \in P_{near}$ and the cases $a = 1 + b$ and $a = -1 - b$. Furthermore, for each of the functions $I^{-1}_{k,\ell}$, $I^{0}_{k,\ell}$, $O^{0}_{k,\ell}$, we perform 10000 test runs with random combinations of $L \in [0.5, 1]$, $N \in [1, 4096]$ and $\alpha \in [-0.9\ \pi, 0.9\ \pi]$, where we also choose randomly between the cases $a = 1+b$ and $a = -1-b$. Figure 2.14 shows the errors $(\widetilde{D}-D)/D$ for all test runs. The fact that all values are greater than zero shows that the model overestimates the actual number of digits for all test cases. For $N \leq 20$, i.e. the cases for which we set $\widetilde{D} = 120$, the relative errors range from 1% to 200%. However, for these cases, the absolute error is still smaller than 70. For $N > 20$, the relative errors are less than 20% for most cases. For $|\alpha| \geq \pi/2$ and $\alpha = 0$, we obtain larger errors.

For the far field, i.e. the case that $T(a, b)$ is outside of an ellipse with foci ± 1 and minor axis 1, the number of digits D depends on $a, b \in \mathbb{C}$ and $N \in \mathbb{N}$, meaning we have a five dimensional problem. We perform test runs for $I^{-1}_{k,\ell}$ and various values of $a \in \mathbb{C}$ as displayed in Figure 2.15. For each point a, we calculate $D(a, b, N)$ for $b = Le^{\mathrm{i}\alpha}$ for all pairs

$$(L, N) \in P_{far} := \{10^{-6}, 10^{-4}, 10^{-2}, 0.1, 0.5, 1\} \cup \{2^k, k = 1, 2, \ldots, 10\}$$

and 15 uniform values of $\alpha = \arg(b) \in [-0.9\pi, 0.9\pi]$. Furthermore, we perform 10000 test runs for random values of $a, b \in \mathbb{C}$ and $N \in \{2^k, k = 1, 2, \ldots, 10\}$. Here we use the following parameter domains

	$\mathrm{Re}(a)$	$\mathrm{Im}(a)$	$\lvert b\rvert$	$\arg(b)$
$0 \leq N \leq 128$	$[1, 10^6]$	$[1, 10^6]$	$[10^{-6}, 1]$	$[-\pi, \pi]$
$129 \leq N \leq 1024$	$[1, 10^3]$	$[1, 10^3]$	$[10^{-3}, 1]$	$[-\pi, \pi]$

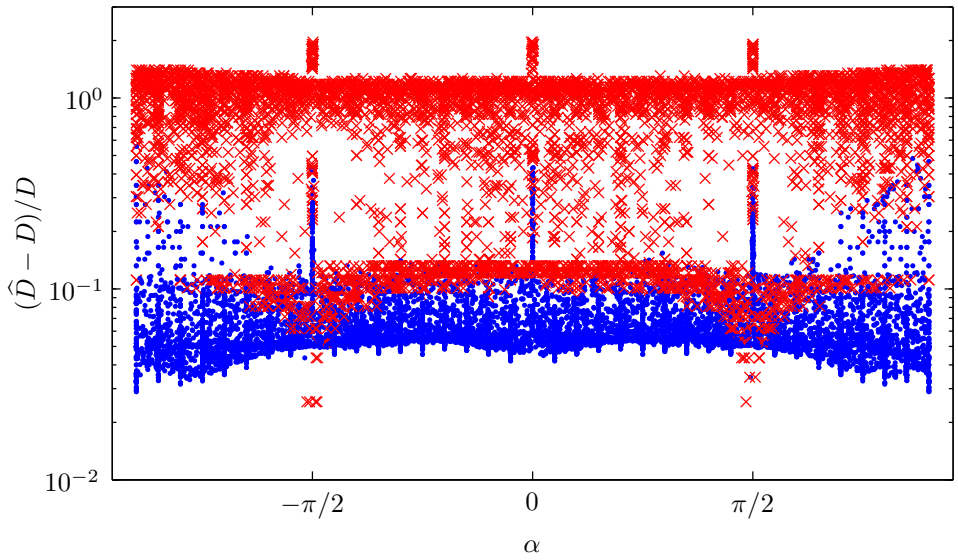


Figure 2.14.: Relative error $(\widetilde{D} - D)/D$ between D and $\widetilde{D}$ over $\alpha = \arg(b)$ for different value for $|b|$ and N. Red crosses: Values for $N \leq 20$. Blue dots: Values for $N > 20$.

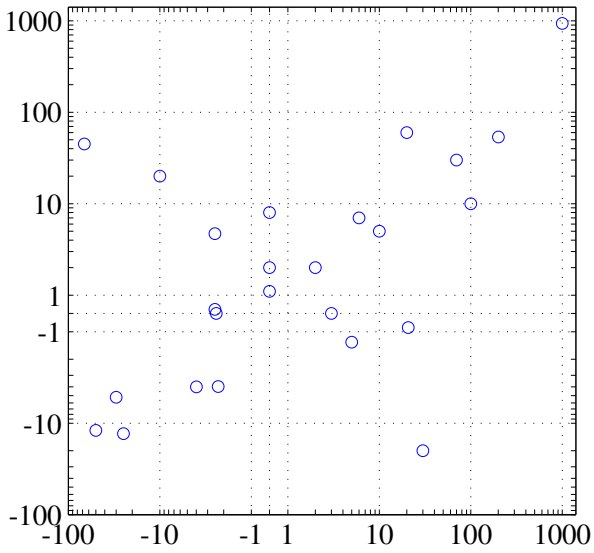


Figure 2.15.: Values of a that were used for the test runs (log scale).

and choose $a, b \in \mathbb{C}$ such that $r(a,b) > 1$. The combination of random test runs with test runs where we set the parameters manually ensures that we cover all extreme cases, e.g. the case $r(a,b) \approx 1$, but also have a good coverage of the full parameter domain. Figure 2.16 shows the number of digits D for all test runs plotted over $r(a,b)/\sqrt{|b|}$. It can be seen that the points behave like the model

$$\widehat{D}(a,b,N) = A(N) \cdot \log\left(\frac{r(a,b)}{\sqrt{|b|}}\right) + B(N).$$

For this model, we only need to find the dependency of the coefficients $A(N)$ and $B(N)$ on N. Thus, we reduced the five dimensional problem to a two dimensional one.
Figure 2.16 shows that the curves change at $D \approx 2000$. Thus, we modify our model to

$$\widehat{D}(a,b,N) = \min\left\{A_1(N) \cdot \log\left(\frac{r(a,b)}{\sqrt{|b|}}\right) + B_1(N), A_2(N) \cdot \log\left(\frac{r(a,b)}{\sqrt{|b|}}\right) + B_2(N)\right\}.$$

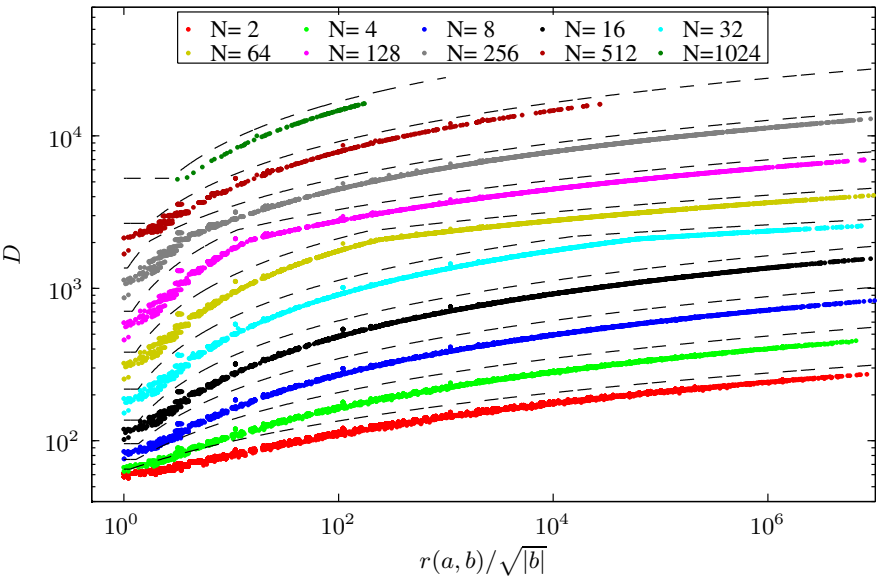


Figure 2.16.: Number of Digits D over $r(a,b)/\sqrt{|b|}$ for different values of $a, b \in \mathbb{C}$ such that $T(a,b)$ is in the far field. The black dashed lines show the model $\widehat{D}(a,b,N)$ for $N \in \{2^k, k = 1, 2, \ldots, 10\}$.

For each $N \in \{2^k, k = 1, \ldots, 8\}$, we determine $A_1(N)$, $B_1(N)$. For $N \in \{2^k, k = 4, \ldots, 10\}$, we determine $A_2(N), B_2(N)$ such that $\hat{D}(a,b,N)$ is an upper bound. Figure 2.17 shows the values of $A_1(N)$,$B_1(N)$ and $A_2(N), B_2(N)$ over N. Using the MATLAB tool `fit`, we obtain the models

$$A_1(N) = 8.76N^{0.92} - 1, \qquad B_1(N) = 1.32N^{1.22} + 60,$$
$$A_2(N) = 3.125N, \qquad B_2(N) = 95N^{0.36} + 850.$$

In all four cases, the coefficients of determination are greater than 0.997, which shows that the models are appropriate. In Figure 2.16, the curves for the model $\widehat{D}$ for $N \in \{2^k, k = 1, \ldots, 10\}$ are displayed (black dashed lines). It can be seen that for these values of N our model is a good upper bound.

In order to ensure that the model also provides an upper bound for general $N \in [0, 2000] \cap \mathbb{N}$, we perform 10000 more test runs, where we choose N randomly in $[0, 2000] \cap \mathbb{N}$ and random values of a, b as described above. Figure 2.18 shows the relative error $(\widehat{D} - D)/D$ over $r(a, b)/\sqrt{|b|}$ for all test runs. It can be seen that the model overestimates the actual number of digits by about 10%-20% for most cases. For small values of $r(a, b)/\sqrt{|b|}$, we obtain smaller errors. The fact that all errors are positive shows that the model $\widehat{D}$ provides an upper bound for all test cases. Random test runs for $I^0_{k,\ell}$, $O^0_{k,\ell}$, and $O^0_{k,\ell}$ show that the model is also appropriate for these functions. The relative errors for these functions behave similarly as for the function $I^{-1}_{k,\ell}$, i.e. all relative errors are between 0.01 and 1.

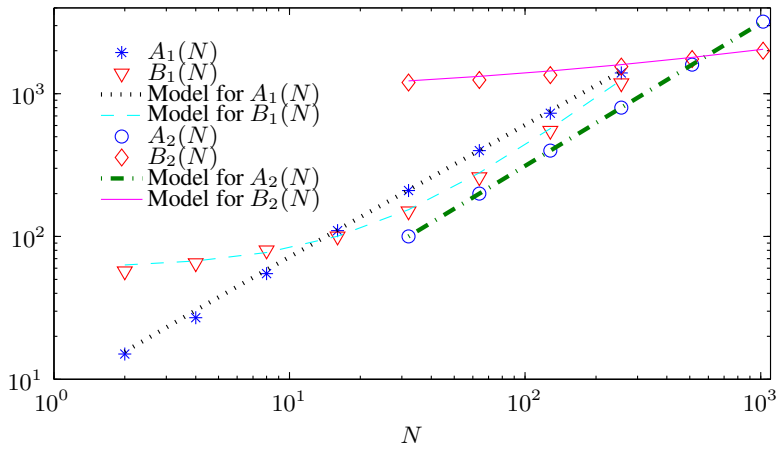


Figure 2.17.: Coefficients $A_1(N)$, $B_1(N)$, $A_2(N)$, $B_2(N)$, and fitted models over N.

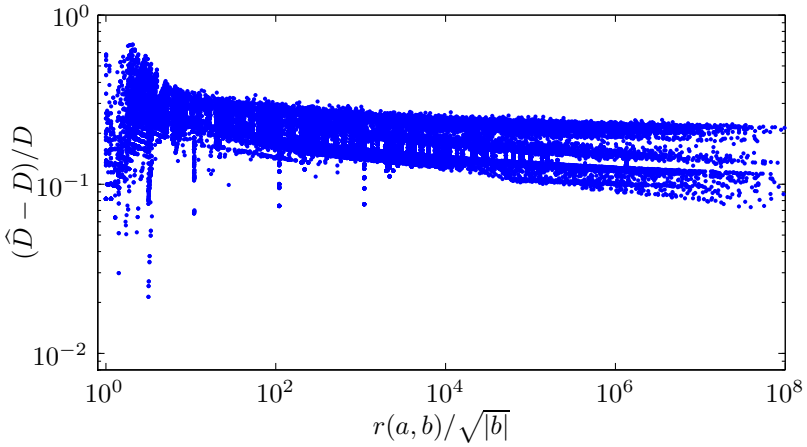


Figure 2.18.: Error $(\widehat{D} - D)/D$ over $r(a, b)/\sqrt{|b|}$ for all test runs.

Remark 2.4.7. An alternative to using recurrence relations for the evaluation of the functions defined in Definition 2.4.1 is using quadrature for the outer integral. By writing

$$I_{k\ell}^{-1}(a,b) = \int_{-1}^{1} P_k(s)\widetilde{Q}_\ell^{-1}(a+bs)\,ds,$$
$$I_{k\ell}^{0}(a,b) = \int_{-1}^{1} P_k(s)\widetilde{Q}_\ell^{0}(a+bs)\,ds,$$
$$O_{k\ell}^{0}(a,b) = \int_{-1}^{1} P_k(s)R_\ell^{0}(a+bs)\,ds,$$

we see that the functions $I_{k,\ell}^{-1}$, $I_{k,\ell}^{0}$, and $O_{k,\ell}^{0}$ can be calculated by integrating the product of a Legendre polynomial and a modified associated Legendre function. It is shown in Section 2.3 that the modified associated Legendre functions can be calculated in a stable and efficient way. In fact, the effort to compute $(\widetilde{Q}_\ell^m(a+bs))_{\ell=0,\dots N}$, $m \in \{-1,0\}$ and $(R_\ell^0(a+bs))_{\ell=1,\dots,N}$, respectively, is $\mathcal{O}(N)$. However, since the integrand is a polynomial of at least degree k, we have to use a quadrature rule that has an order of at least $k/2$. Thus, the overall computational effort to assemble the matrix $(I_{k\ell}^m(a,b))_{k,\ell=1,\dots,N}$ ($m \in \{-1,0\}$) and $(O_{k\ell}^0(a,b))_{k,\ell=1,\dots N}$, respectively, is at least $\mathcal{O}(N^4)$. Numerical experiments (see Figure 2.19) show that the computational effort to compute the matrices using the recurrence relations is less than $\mathcal{O}(N^{2.5})$. However, here the constant is much larger because of the overhead produced by `MPFR` and `MPC`.

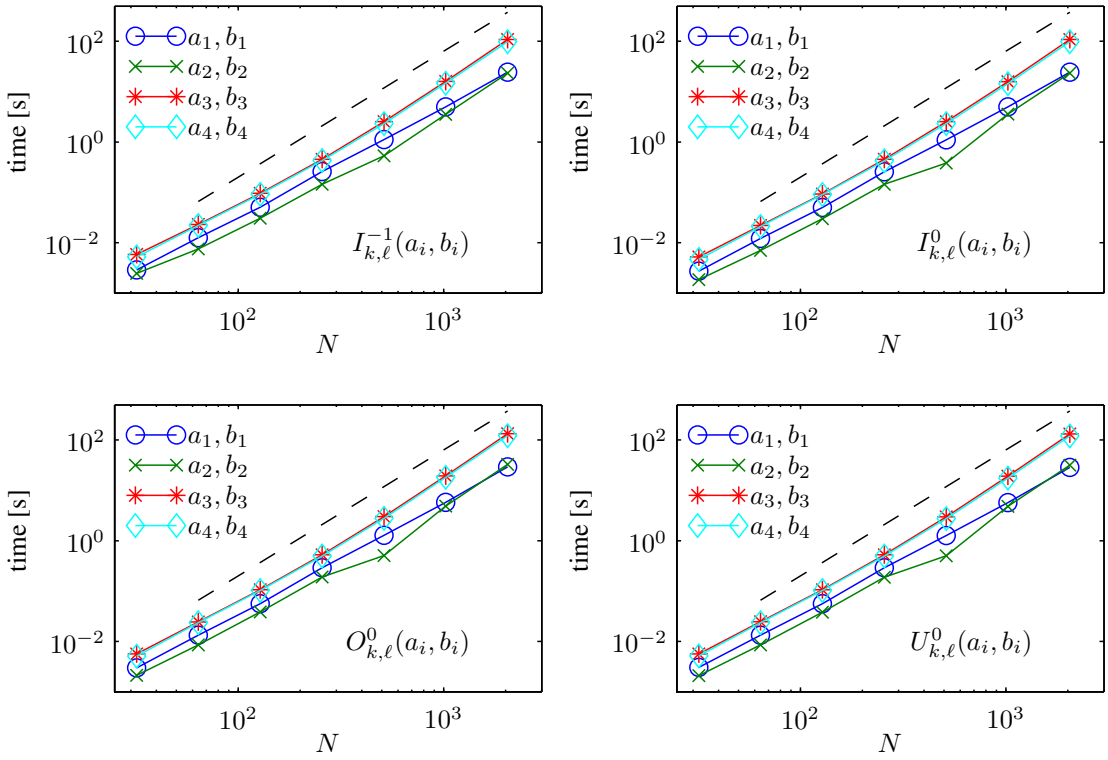


Figure 2.19.: Computation times for the assembly of the matrices $(I^{-1}_{k\ell}(a_i, b_i))_{k,\ell=1,\dots,N}$, $(I^{0}_{k\ell}(a_i, b_i))_{k,\ell=1,\dots,N}$, $(O^{0}_{k\ell}(a_i, b_i))_{k,\ell=1,\dots,N}$, and $(U^{0}_{k\ell}(a_i, b_i))_{k,\ell=1,\dots,N}$ in dependency of N for $a_1 = 1 + \mathrm{i}$, $b_1 = \mathrm{i}$, $a_2 = 2\mathrm{i}$, $b_2 = 0.5$, $a_3 = 3 + 8\mathrm{i}$, $b_3 = 0.01$, $a_4 = 40$, and $b_4 = 0.3 + 0.2\mathrm{i}$. The dashed lines represent $\mathcal{O}(N^{2.5})$. The computations are performed on a desktop computer with an AMD PhenomII X6 Processor, 16GB RAM, and the Ubuntu 10.04 operating system.

3. Implementation of hp-Boundary Element Methods

In this chapter we discuss the implementation of hp-boundary element methods. The goal is to be able to evaluate the potentials introduced in Section 1.2 and to numerically solve the three integral equations introduced in Section 1.2.1 - 1.2.3, namely Symm's integral equation, the hypersingular integral equation, and the mixed integral equation. For the solution of the integral equations, we use a Galerkin method as described in Section 1.2. In this work we focus on the efficient and accurate assembly of the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$, and $\mathbf{W}$. We stress that unlike in most previous works, we do not use discrete spaces consisting of piecewise constant or piecewise linear functions (i.e. lowest order or h-BEM). In this work the emphasis is on the evaluation of potentials and the assembly of Galerkin matrices for ansatz spaces that contain piecewise polynomials of high orders (i.e. p- and hp-BEM).

The stable evaluation of potentials and computation of entries of Galerkin matrices for high orders is achieved by using the results of Chapter 2. Specifically, we use the functions defined in Section 2.3 for the stable evaluation of potentials and the double integrals defined in Section 2.4 for the assembly of Galerkin matrices.

The first section in this chapter gives an overview on existing software packages for boundary element methods. The discrete subspaces we use for the implementation are introduced in the second section. The evaluation of potentials is discussed in the third section, and the fourth section is devoted to assembly of the Galerkin matrices.

All implementations are part of the software package epsBEM [BBF13]. Since this software package is used for the numerical experiments described in the sixth section, we use the fifth section to give an overview of the main features of epsBEM.

In this work we do not discuss methods for the solution of the linear systems arising in hp-BEM or advanced preconditioning techniques for these systems. It turns out that a simple diagonal preconditioning and the solution of the linear systems using the MATLAB backslash operator work well for our numerical experiments, which are presented in the last section.

3.1. Overview of Existing Software Packages

There are many software packages for boundary element methods. In this section we give an overview of some software packages. This overview is by no means complete.

The C++ software **BEM++** ([ŚAB+12]) implements three dimensional boundary element methods for Laplace, Lamé and Helmholtz problems. The arising boundary integrals are computed using quadrature rules from [SaSch04]. For the solution of the linear systems, data compression techniques such as $\mathcal{H}$-Matrices [GBH06] and ACA [Beb00] are supported by interfaces to the library AHMED [Beb08].
BEMLIB is a collection of FORTRAN 77 and MATLAB codes that come with the book [Poz02]. It implements Laplace-, Helmholtz- and Stokes- problems in two and three dimensions.
The **B**oundary **E**lement **T**emplate **L**ibrary (**BETL** [HK12]) is a C++ library implementing three dimensional boundary integral operators for Laplace and Helmholtz problems. It features constant, linear, quadratic, and cubic Lagrangian basis functions for triangular and quadrilateral elements as well as lowest order Raviart-Thomas and Nedelec type basis functions for triangular elements. The arising boundary integrals are computed with quadrature formulas for the inner and outer integral and in some cases semi-analytically. However, **BETL** does not implement boundary element methods.
HILBERT [AEF+11] (**H**ilbert **I**s a **L**ovely **B**oundary **E**lement **R**esearch **T**ool) implements two dimensional boundary element methods for the Laplace operator. The software package consists of both MATLAB and C functions; the core functions are implemented in C and connected to the MATLAB user interface with mex functions. **HILBERT** features functions for the evaluation of potentials and the assembly of Galerkin matrices using lowest order ansatz spaces. Here, the integrals are computed with quadrature for the inner and the outer integral or semi-analytically. Furthermore, several error estimators, for example the $h - h/2$ error estimators that are introduced in [FLP08] and [EFGP12], and h-adaptive algorithms are implemented.
HyENA (**H**yperbolic and **E**lliptic **N**umerical **A**nalysis, [MMRU10]) is a C++ software package for the solution of partial differential equations for which fundamental solutions are known. Since the software is based on fundamental solutions, it can be used for a wide range of operators. It supports two and three dimensional Galerkin and Collocation boundary element methods as well as data compression methods such as $\mathcal{H}$-Matrices and ACA.
The Fortran software package **MAIPROGS** implements both finite element and boundary element methods. For BEM, it supports Laplace, Lamé, and Helmholtz problems. These problems can be solved using hp-BEM. Calculations can be performed with quadruple precision or with the multiple precision library ARPREC [BHLT02]. The results of several numerical experiments, which were performed with **MAIPROGS**, are collected in [Mai12]. They show that the imple-

mentation works for p- and hp-BEM. However, only results for polynomial degrees $p \leq 50$ are shown.

All software packages, except for **MAIPROGS**, implement lowest order BEM and use analytic formulas or quadrature rules for the computation of Galerkin entries. In **MAIPROGS**, p- and hp-BEM is implemented. For the computation of Galerkin entries, quadrature rules or recurrence relations for monomial basis functions are used, which results in a limitation for the polynomial degrees of the ansatz spaces. In the following section we show that the evaluation of potentials and computation of Galerkin entries can be reduced to the evaluation of recurrence relations. Together with the use of multiple precision libraries, we obtain an implementation that produces accurate results even for polynomial degrees $p > 1000$.

3.2. Discrete Spaces for hp-BEM

In Section 1.3 we introduced a triangulation

$$\mathcal{T} := \{T_i,\ i = 1, \dots, \mathcal{N}_e\}$$

of the polygonal boundary $\Gamma = \partial\Omega$ and the discrete spaces $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p})$. In this section we discuss the choice of appropriate bases of these two discrete spaces.
Each element T_i is the affine image of the reference element $\widehat{T} = [-1, 1]$. The mapping γ_i, which is defined by

$$\begin{aligned} &\gamma_i : \widehat{T} \to T_i \in \mathcal{T} \\ &\gamma_i(t) = \frac{A_i + B_i}{2} + \frac{B_i - A_i}{2} t =: m_i + u_i t, \end{aligned} \tag{3.1}$$

maps the reference element $\widehat{T}$ to T_i and can be used to parametrize T_i; see Figure 3.1. In the following, we need some properties of the mapping γ_i. These properties can be derived directly from the definition of the mapping γ_i.

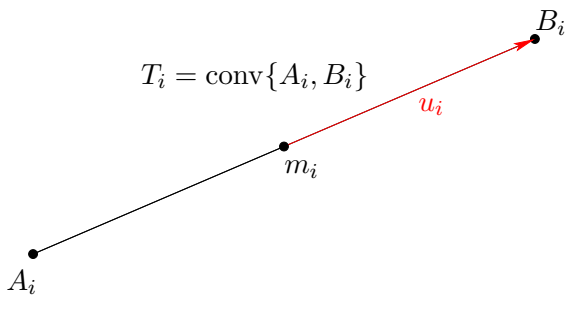


Figure 3.1.: Parametrization of an element $T_i \in \mathcal{T}$

Lemma 3.2.1. *(i) The derivative of γ_i is given by $\gamma_i'(t) = (B_i - A_i)/2$, and there holds $|\gamma_i'(t)| = h_i/2$.*

(ii) The inverse of γ_i is given by

$$\gamma_i^{-1} : T_i \to [-1,1], \qquad \gamma_i^{-1}(x) = \frac{4}{h_i^2} u_i^T (x - m_i),$$

and there holds $\nabla \gamma_i^{-1}(x) = \dfrac{4u_i}{h_i^2}$.

We are now ready to specify the bases of the discrete spaces introduced in Definition 1.3.2. We choose affine images of Legendre polynomials and Lobatto shape functions as element basis functions for the discrete spaces.

Definition 3.2.2. (i) Let $P_k^{(i)} := P_k \circ \gamma_i^{-1}$, $k \in \mathbb{N}_0$, $i \in \{1, \dots, \mathcal{N}_e\}$, and $N_k^{(i)} := N_k \circ \gamma_i^{-1}$, $k \in \mathbb{N}$, $i \in \{1, \dots, \mathcal{N}_e\}$. Here, P_k is the kth Legendre polynomial and N_k is the kth Lobatto shape function; see Lemma 2.1.2 and Definition 2.1.3.

(ii) Let $\mathcal{T}$ be a boundary mesh, $\mathbf{h}$ be a mesh width function, and $\mathbf{p}_0$ and $\mathbf{p}_1$ be polynomial degree functions as defined in Definition 1.3.1. The basis of the space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is given by

$$\bigcup_{T_i \in \mathcal{T}} \left\{ P_0^{(i)}, \dots, P_{p_i}^{(i)} \right\} \tag{3.2}$$

and the basis of $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ is given by

$$\{\mathscr{H}_1, \dots, \mathscr{H}_{\mathcal{N}_v}\} \cup \left(\bigcup_{T_i \in \mathcal{T}} \left\{ N_3^{(i)}, \dots, N_{q_i+1}^{(i)} \right\} \right), \tag{3.3}$$

where $\mathscr{H}_i$ is the hat function for the i-th boundary node.

Remark 3.2.3. (i) In subsequent sections we sometimes write γ_T for the affine mapping from $\widehat{T}$ to $T \in \mathcal{T}$ and $P_k^{(T)} := P_k \circ \gamma_T^{-1}$.

(ii) $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is the space of piecewise polynomials with respect to the triangulation $\mathcal{T}$ and local degrees p_i. For a given triangulation with $\mathcal{N}_e$ elements and polynomial degrees p_i, $i = 1, \dots, \mathcal{N}_e$, the dimension of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is $\dim(S_0) = \sum_{i=1}^{\mathcal{N}_e} (p_i + 1)$.

(iii) The space $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ contains piecewise polynomial but globally continuous functions. The continuity is achieved by combining the functions $N_k^{(i)}$ with $N_k^{(j)}$, $k \in \{1, 2\}$, from neighboring elements T_i and T_j to hat functions $\mathscr{H}_i$.
The dimension of $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ is $\dim(S_1) = \mathcal{N}_v + \sum_{i=1}^{\mathcal{N}_e} (q_i - 1)$.

The choice of appropriate bases for the spaces $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ is important, since the basis affects the condition number of the resulting Galerkin matrix. Figure 3.2 shows the condition number of the single layer Galerkin matrix for different types of element basis functions. Specifically, we use basis functions that are constructed from the following bases on the reference element $[-1, 1]$ using the affine mapping γ_i: Monomials M_k, Legendre polynomials P_k, Chebyshev polynomials of the first kind T_k, and Chebyshev polynomials of the second kind U_k.
Whereas the condition number for monomial basis functions grows exponentially, the condition numbers for Chebyshev and Legendre bases grow algebraically. If we scale the Galerkin matrix by its diagonal entries, the condition number for the resulting Galerkin matrix grows linearly in the polynomial degree p of the ansatz space.

Remark 3.2.4. Heuer analyzes the condition number of the single layer Galerkin matrix in [Heu92] and [Heu96]. It is shown that the condition number of the Galerkin matrix $\mathbf{V}$ for the piecewise Legendre basis is of the order $\mathcal{O}(p^\alpha)$ with $\alpha \approx 2.5$, where p is the polynomial degree of the ansatz space. Furthermore, Heuer shows that the condition number is of the order $\mathcal{O}(p^\alpha)$ with $\alpha \leq 2$ if scaled Legendre polynomials, $\widetilde{P}_k := \sqrt{2k+1}\, P_k$, are used as local basis functions. Our numerical experiments indicate, that the condition number grows only linearly in p if standard Legendre polynomials are used as basis functions and then the resulting Galerkin matrix is scaled by its diagonal.

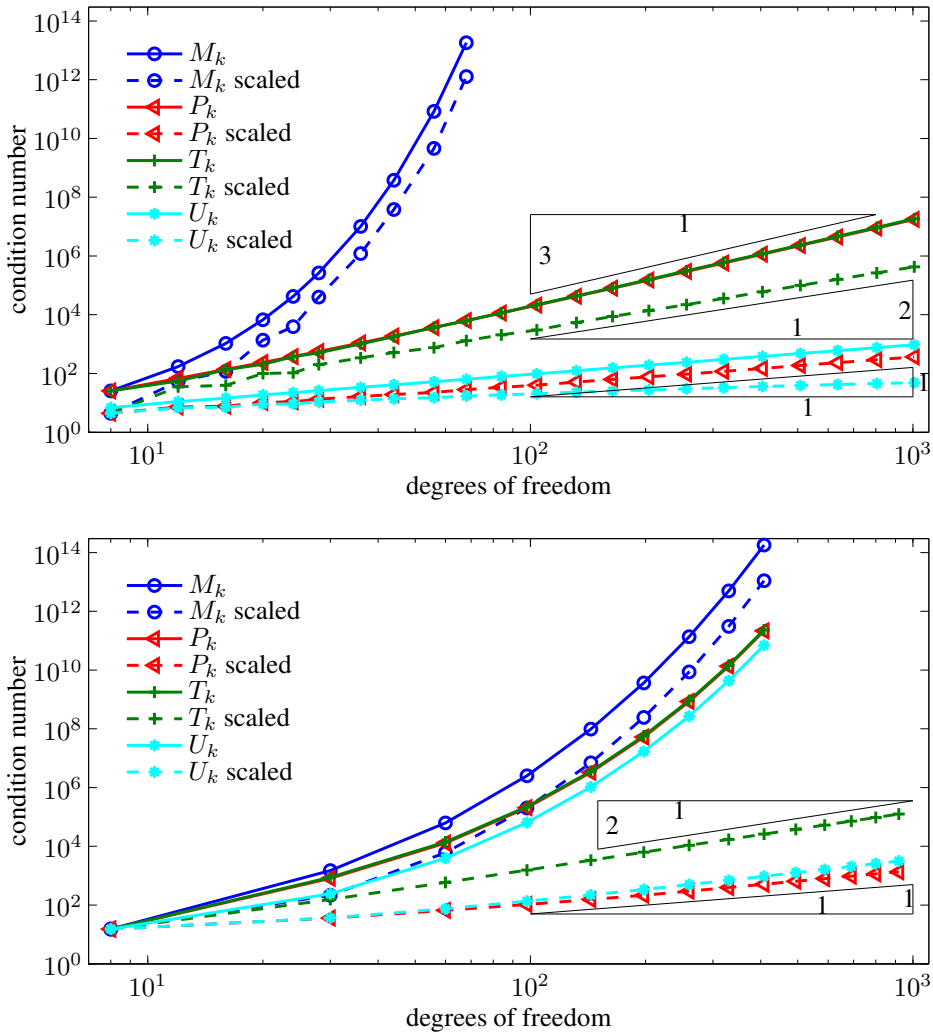


Figure 3.2.: Condition number for the single layer Galerkin matrix for different bases of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. The top picture shows the condition number of the Galerkin matrix for $\Gamma = [-1, 1] \times 0$ and $\mathcal{T} = \{T_1, \ldots, T_4\}$ with $h_1 = \cdots = h_4 = 0.5$ and varying the uniform polynomial degrees $p_1 = \cdots = p_4 = p$. The bottom picture shows the condition number for the matrix with respect to geometric meshes with linearly growing polynomial degree vector on the L-shaped domain as displayed in Figure 1.1(b). Here, "scaled" means that the Galerkin matrices are scaled by their diagonal.

3.3. Evaluation of the Potentials

In this section we present the discretization of the potentials for the discrete spaces $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$. The following theorem shows how the evaluation of the potentials $\widetilde{\mathcal{V}}$, $\widetilde{\mathcal{K}}$, $\widetilde{\mathcal{K}}'$, and $\widetilde{\mathcal{W}}$ can be reduced to the evaluation of the modified associated Legendre functions of the second kind $\widetilde{Q}_k^m$ and R_k^m, which are introduced in Section 2.3.

Theorem 3.3.1. *For a given boundary mesh $\mathcal{T}$, let $T \in \mathcal{T}$, u and m as shown in Figure 3.1, and $w = m - x$ (in order to simplify the notation, we leave out the index i). Furthermore, let*

$$z := \frac{-u^T w - \mathrm{i}\,(u \times w)}{u^T u} \in \mathbb{C}. \tag{3.4}$$

Then for the single layer potential/operator, there holds for $k \in \mathbb{N}_0$

$$\begin{aligned}(\widetilde{\mathcal{V}} P_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T P_k^{(T)}(y) \log|x-y|\, ds_y \\ &= -\frac{|u|}{2\pi} \left(\delta_{k,0} \log|u|^2 + \operatorname{Re} \widetilde{Q}_k^{-1}(z) \right),\end{aligned} \tag{3.5}$$

and for the double layer potential/operator, there holds for $k \in \mathbb{N}$

$$\begin{aligned}(\widetilde{\mathcal{K}} N_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T N_k^{(T)}(y) \frac{\partial}{\partial n_y} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \operatorname{Im} R_k^0(z),\end{aligned} \tag{3.6}$$

with the normal vector n_y (clockwise rotation) with respect to T. Furthermore, for the adjoint double layer potential/operator, there holds for $k \in \mathbb{N}_0$

$$\begin{aligned}(\widetilde{\mathcal{K}}' P_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T P_k^{(T)}(y) \frac{\partial}{\partial n_x} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \left(\frac{u \times n_x}{|u|} \operatorname{Im} \widetilde{Q}_k^0(z) - \frac{u^T n_x}{|u|} \operatorname{Re} \widetilde{Q}_k^0(z) \right)\end{aligned} \tag{3.7}$$

with a normal vector n_x that is normal with respect to x (the direction of x). Finally we obtain for the hypersingular potential/operator for $k \in \mathbb{N}$

$$\begin{aligned}(\widetilde{\mathcal{W}} N_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T N_k^{(T)}(y) \frac{\partial}{\partial n_x} \frac{\partial}{\partial n_y} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \left((u \times n_x) \operatorname{Re} R_k^1(z) + (u^T n_x) \operatorname{Im} R_k^1(z) \right).\end{aligned} \tag{3.8}$$

Proof. Here, we only prove the statement for the single layer potential. The proof of the statements for the other three potentials is very technical, but the ideas are similar to those used for the single layer potential. The full proof is stated in Appendix B.
With the configuration shown in Figure 3.1, for $y \in T$, we have $y = m + tu$, and for $x \in \mathbb{R}^2$,

there holds $x - y = -w - tu$, $t \in [-1, 1]$. Plugging in this expression for $x - y$ yields

$$
\begin{aligned}
&-\frac{1}{2\pi}\int_T P_k^{(T)}(y)\log|x-y|\,ds_y \\
&\quad= -\frac{|u|}{2\pi}\int_{-1}^{1} P_k(t)\log|-w-tu|\,dt = -\frac{|u|}{4\pi}\int_{-1}^{1} P_k(t)\log|-w-tu|^2\,dt \\
&\quad= -\frac{|u|}{4\pi}\int_{-1}^{1} P_k(t)\log\left(u^Tu\left(t^2+2\frac{w^Tu}{u^Tu}t+\frac{w^Tw}{u^Tu}\right)\right)\,dt \\
&\quad= -\frac{|u|}{4\pi}\left[\int_{-1}^{1} P_k(t)\log(u^Tu)\,dt + \int_{-1}^{1} P_k(t)\log\left(t^2+2\frac{w^Tu}{u^Tu}t+\frac{w^Tw}{u^Tu}\right)\,dt\right] \\
&\quad= -\frac{|u|}{4\pi}\left[2\delta_{k,0}\log(u^Tu) + \int_{-1}^{1} P_k(t)\log\left((z_1-t)(z_2-t)\right)\,dt\right],
\end{aligned}
$$

where

$$
\begin{aligned}
z_{1,2} &= -\frac{w^Tu}{u^Tu} \pm \sqrt{\left(\frac{w^Tu}{u^Tu}\right)^2 - \frac{w^Tw}{u^Tu}} \\
&= \frac{-w^Tu \pm \sqrt{(w^Tu)^2 - w^Tw\,u^Tu}}{u^Tu} = \frac{-w^Tu \pm \mathrm{i}(w\times u)}{u^Tu}.
\end{aligned}
$$

For the last equality, we used the Lagrange identity $|a\times b|^2 = a^Ta\,b^Tb - (a^Tb)^2$ for $a, b \in \mathbb{R}^n$, $n \geq 2$. Note that the two roots $z_{1,2}$ are complex conjugates and with

$$
z := z_1 = \frac{-u^Tw - \mathrm{i}\,(u\times w)}{u^Tu} \in \mathbb{C},
$$

we can write $\log\left((z-t)(\bar{z}-t)\right) = 2\,\mathrm{Re}\log(z-t)$. Plugging this in gives

$$
-\frac{1}{2\pi}\int_T P_k^{(T)}(y)\log|x-y|\,ds_y = -\frac{|u|}{2\pi}\left[\delta_{k,0}\log(u^Tu) + \mathrm{Re}\int_{-1}^{1} P_k(t)\log\left(z-t\right)\,dt\right].
$$

□

The results from Theorem 3.3.1 show that the evaluation of all potentials can be reduced to the evaluation of modified associated Legendre functions, which are defined in Section 2.3. This results in short and easy to understand MATLAB codes for evaluation of the potentials. Listing 3.1 shows the implementation of the function `potK` that can be used to evaluate the double layer potential. After having calculated the complex number z from the vertices of the boundary piece given by `vertices` and the evaluation point `x` according to Equation (3.4), we simply call the function `r0`. The function `r0` calculates the integral $(R_k^0(z))_{k=1,\dots,p+1}$ and multiplies the values of $R_k^0(z)$ by the factor $1/(2\pi)$. The function `potK` also works for an N-by-2 matrix `x` of N evaluation points.

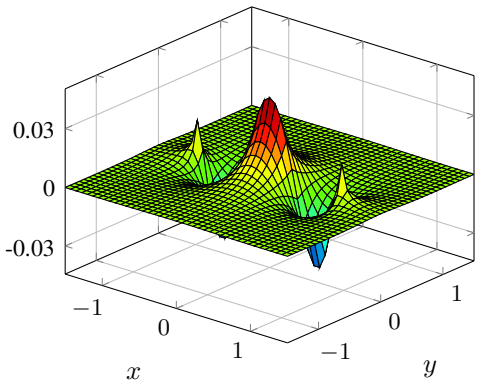
Since the modified associated Legendre functions can be evaluated in a stable and very efficient way using a combination of forward and backward recurrence relations (see Section 2.3.4), the evaluation of the potentials is also fast and efficient. The overhead created by the MATLAB functions is negligible. Figure 3.3 displays examples of all four potentials.

Listing 3.1: Implementation of the function `potK` that computes $((\widetilde{\mathcal{K}}N_\ell^{(T)})(x))_{\ell=1,\dots,p+1}$ with respect to the boundary piece T.

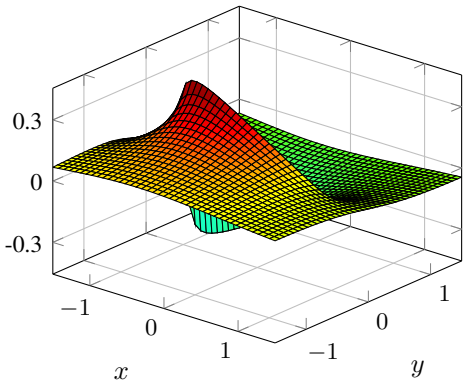
```
function K = potK(vertices,x,p,varargin)
m = (vertices(1,:)+vertices(2,:))/2;
u = (vertices(2,:)-vertices(1,:))/2;
utu = u*u';
w = [m(1)-x(:,1),m(2)-x(:,2)];

z_re = -(u(1).*w(:,1)+u(2).*w(:,2))/utu;
z_im = -(u(1).*w(:,2)-u(2).*w(:,1))/utu;

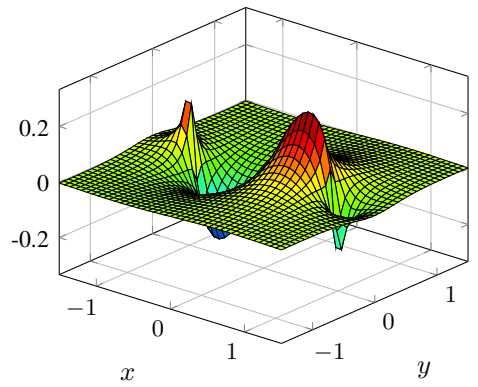
K = imag(r0(complex(z_re,z_im),p,1/(2*pi)).');
```



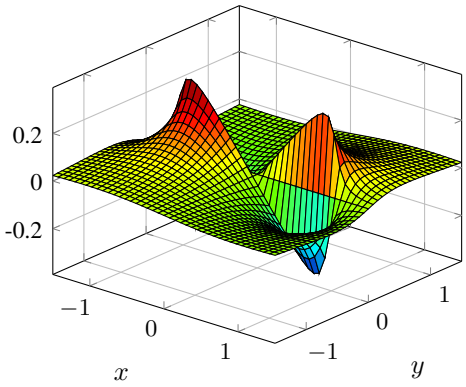
(a) Single Layer Potential $(\widetilde{\mathcal{V}}P_4)(x,y)$



(b) Double Layer Potential $(\widetilde{\mathcal{K}}N_1)(x,y)$



(c) Adjoint Potential $(\widetilde{\mathcal{K}}'P_3)(x,y)$



(d) Hypersinglar Potential $(\widetilde{\mathcal{W}}N_3)(x,y)$

Figure 3.3.: Surface plots of the potentials $\widetilde{\mathcal{V}}$, $\widetilde{\mathcal{K}}$, $\widetilde{\mathcal{K}}'$, and $\widetilde{\mathcal{W}}$ with respect to $T = [-1,1]$.

3.4. Computation of Galerkin Entries

As mentioned before, the computation of Galerkin entries can be reduced to the evaluation of the functions $I^m_{k,\ell}$, $m \in \{-1, 0\}$, $O^0_{k,\ell}$ and $U^0_{k,\ell}$, which are defined in Section 2.4. In the first subsection we derive formulas to compute Galerkin entries. The second subsection gives a detailed description of the implementation.

3.4.1. Theory

First we need some auxiliary results for the parametrization of boundary pieces. For a given triangulation $\mathcal{T}$ of the boundary Γ, we consider two elements $T_i, T_j \in \mathcal{T}$. Without loss of generality, we assume that $|T_i| \leq |T_j|$. From Equation (3.1) we see that for $x \in T_i$, there holds $x = m_i + u_i\, s$, and for $y \in T_j$, we have $y = m_j + u_j\, t$ with $s, t \in [-1, 1]$.
Defining $w := m_j - m_i$, we can write $|x - y| = |u_i\, s - w - u_j\, t|$, which yields

$$|x-y|^2 = |u_i\, s - w - u_j\, t|^2 = |u_j|^2 \left(t^2 - 2\frac{(u_i\, s - w)^T u_j}{u_j^T u_j} t + \frac{(u_i\, s - w)^T (u_i\, s - w)}{u_j^T u_j} \right).$$

The roots of the quadratic polynomial are given by

$$\begin{aligned} z_{1,2}(s) &= \frac{u_j^T (u_i\, s - w) \pm \sqrt{(u_j^T (u_i\, s - w))^2 - |u_j|^2\, |(u_i\, s - w)|^2}}{|u_j|^2} \\ &= \frac{u_j^T (u_i\, s - w) \pm \mathrm{i}\, (u_j \times (u_i\, s - w))}{|u_j|^2} \\ &= s\, \frac{u_j^T u_i \pm \mathrm{i}\, (u_j \times u_i)}{|u_j|^2} - \frac{u_j^T w \pm \mathrm{i}\, (u_j \times w)}{|u_j|^2}. \end{aligned}$$

Note that the two roots are complex conjugates. For the following we choose

$$z(s) = -\frac{u_j^T w + \mathrm{i}\, (\mathrm{u_j} \times \mathrm{w})}{|u_j|^2} + \frac{u_j^T u_i + \mathrm{i}\, (u_j \times u_i)}{|u_j|^2}\, s =: a + b\, s, \quad a, b \in \mathbb{C}, \tag{3.9}$$

and obtain

$$|x - y|^2 = |u_j|^2 \big(z(s) - t\big)\big(\overline{z(s)} - t\big). \tag{3.10}$$

Figure 3.4 shows the parametrization and the geometric interpretation of $a, b \in \mathbb{C}$. The complex numbers $a, b \in \mathbb{C}$ depend on the relative position of the two boundary pieces $T_i, T_j \in \mathcal{T}$. In more detail, the number a can be interpreted as the complex number representing $w = m_i - m_j$ scaled by $|u_j|$, and b is the complex number associated with $|u_i|$ scaled by $|u_j|$.

We start with the Galerkin matrix for the single layer operator. Using the discrete space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ (see (3.2)) as test and ansatz space, we obtain that the Galerkin matrix is a block

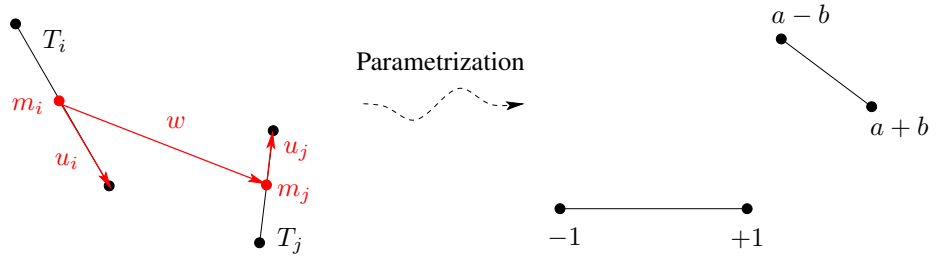


Figure 3.4.: Parametrization of the two boundary pieces $T_i, T_j \in \mathcal{T}$ and geometric interpretation of the complex numbers $a, b \in \mathbb{C}$ from (3.9).

matrix with one $(p_i+1)\times(p_j+1)$ block for each combination of two boundary pieces $T_i, T_j \in \mathcal{T}$, i.e. $\mathbf{V} = (\mathbf{V}^{(i,j)})_{i,j=1,\dots,\mathcal{N}_e}$. The Galerkin entries of one block are given by

$$V_{k,\ell}^{(i,j)} = -\frac{1}{2\pi} \int_{T_i} P_k^{(i)}(x) \int_{T_j} P_\ell^{(j)}(y) \log|x-y| \, ds_y \, ds_x, \quad k = 0, \dots, p_i, \ \ell = 0, \dots, p_j.$$

Theorem 3.4.1. *Let $T_i, T_j \in \mathcal{T}$ and $z(s)$ (resp. a and b) as in (3.9). Then there holds*

$$\begin{aligned} V_{k,\ell}^{(i,j)} &= -\frac{|u_i|\,|u_j|}{2\pi} \left(4\delta_{k,0}\delta_{\ell,0} \log|u_j| + \operatorname{Re} \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \, \log\big(z(s)-t\big) \, dt \, ds \right) \\ &= -\frac{|u_i|\,|u_j|}{2\pi} \left(4\delta_{k,0}\delta_{\ell,0} \log|u_j| + \operatorname{Re} \, I_{k,\ell}^{-1}(a,b) \right). \end{aligned} \tag{3.11}$$

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

Proof. We plug in the parametrizations of T_i and T_j and the result (3.10) to obtain

$$\begin{aligned} &\int_{T_i} P_k^{(i)}(x) \int_{T_j} P_\ell^{(j)}(y) \log|x-y| \, ds_y \, ds_x \\ &\quad = \frac{1}{2} \int_{T_i} P_k^{(i)}(x) \int_{T_j} P_\ell^{(j)}(y) \log|x-y|^2 \, ds_y \, ds_x \\ &\quad = \frac{|u_i|\,|u_j|}{2} \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \left[\log(|u_j|^2) + \log\left[\big(z(s)-t\big)\big(\overline{z(s)}-t\big) \right] \right] \, dt \, ds \\ &\quad = \frac{|u_i|\,|u_j|}{2} \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \left(\log(|u_j|^2) + 2 \operatorname{Re} \log(z(s)-t) \right) \, dt \, ds. \end{aligned}$$

Finally, the L^2 orthogonality of Legendre polynomials yields

$$\int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log(|u_j|^2) \, dt \, ds = 4\delta_{k,0}\delta_{\ell,0} \log(|u_j|^2),$$

which completes the proof. □

For the diagonal blocks, i.e. the case of identical elements $T_i = T_j$, we obtain a direct formula for the Galerkin entries.

Theorem 3.4.2. *For $T_i \in \mathcal{T}$ and $k, \ell \in \mathbb{N}_0$, there holds*

$$\begin{aligned} V_{k,\ell}^{(i,i)} &= -\frac{1}{2\pi} \int_{T_i} P_k^{(i)}(x) \int_{T_i} P_\ell^{(i)}(y) \log|x-y| \, ds_y \, ds_x, \\ &= -\frac{h_i^2}{4\pi} \begin{cases} \log h_i^2 - 3 & \text{, if } k = \ell = 0 \\ \dfrac{4}{(k+\ell+2)(\ell+k)((\ell-k)^2-1)} & \text{, if } k+\ell > 0 \text{ and } k-\ell \text{ is even} \\ 0 & \text{, otherwise.} \end{cases} \end{aligned} \tag{3.12}$$

Proof. Since $T_i = T_j$, there holds $u_i = u_j =: u$ and $w = 0$, which yields $z(s) = s$ and

$$V_{k,\ell}^{(i,i)} = -\frac{|u|^2}{2\pi} \left(4\delta_{k,0}\delta_{\ell,0} \log|u| + \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log|s-t| \, dt \, ds, \right).$$

For $k = \ell = 0$, we have

$$\begin{aligned} V_{0,0}^{(i,i)} &= -\frac{|u|^2}{2\pi} \left(4\log|u| + \int_{-1}^{1} \int_{-1}^{1} \log|s-t| \, dt \, ds, \right) \\ &= -\frac{|u|^2}{2\pi} \left(4\log|u| + 4\log(2) - 6 \right) = -\frac{h_i^2}{4\pi} \left(\log h_i^2 - 3 \right), \end{aligned}$$

where we plugged in $|u| = h_i/2$ in the last step. The case $k, \ell > 0$ is treated in [Sad03, Theorem 2], where the identity

$$\int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log|s-t| \, dt \, ds = \begin{cases} \frac{8}{(k+\ell)(k+\ell+2)((k-\ell)^2-1)}, & k-\ell \text{ even}, \\ 0, & k-\ell \text{ odd}. \end{cases}$$

is shown. □

For the Galerkin matrix $\mathbf{K}$ for the double layer operator, we use the ansatz space $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ and the test space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$. For the Galerkin entries, we have to compute the integrals

$$\widetilde{K}_{k,\ell}^{(i,j)} = -\frac{1}{2\pi} \int_{T_i} P_k^{(i)}(x) \int_{T_j} N_\ell^{(j)}(y) \frac{\partial}{\partial n_y} \log|x-y| \, ds_y \, ds_x, \tag{3.13}$$

for $k = 0, \ldots, p_i, \ \ell = 1, \ldots, q_j + 1$. Since the first $\mathcal{N}_v$ basis functions of $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$ are hat functions, we have to add up the local contributions of $K_{k,1}^{(i,j)}$ and $K_{k,2}^{(i,j)}$ for neighboring elements. Using the parametrization of T_i and T_j, we obtain the following result.

Theorem 3.4.3. *Let $T_i, T_j \in \mathcal{T}$ and $z(s)$ (resp. a, b) as in (3.9). Then there holds*

$$\widetilde{K}_{k,\ell}^{(i,j)} = -|u_i| \operatorname{Im} \int_{-1}^{1} P_k(s) \int_{-1}^{1} \frac{N_\ell(t)}{z(s)-t} \, dt \, ds = -|u_i| \operatorname{Im} O_{k,\ell}^0(a,b), \tag{3.14}$$

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

Proof. We plug in the parametrizations of T_i and T_j and the result (3.10), which yields

$$\begin{aligned}
&\int_{T_i} P_k^{(i)}(x) \int_{T_j} N_\ell^{(j)}(y) \frac{\partial}{\partial n_y} \log|x-y| \, ds_y \, ds_x \\
&\quad = \int_{T_i} P_k^{(i)}(x) \int_{T_j} N_\ell^{(j)}(y) \frac{(x-y)^T u_j^\perp / |u_j|}{|x-y|^2} \, ds_y \, ds_x \\
&\quad = \frac{|u_i|}{|u_j|^2} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \frac{(su_i - w - tu_j)^T u_j^\perp}{(z(s)-t)(\overline{z(s)}-t)} \, dt \, ds \\
&\quad = \frac{|u_i|}{|u_j|^2} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \frac{u_j \times (su_i - w)}{(z(s)-t)(\overline{z(s)}-t)} \, dt \, ds
\end{aligned} \tag{3.15}$$

Next we use the partial fraction expansion

$$\frac{1}{(z-t)(\overline{z}-t)} = \frac{\mathrm{i}}{2\operatorname{Im}(z)} \left(\frac{1}{z-t} - \frac{1}{\overline{z}-t} \right)$$

and with (3.9) we obtain

$$\begin{aligned}
&\frac{|u_i|}{|u_j|^2} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \frac{u_j \times (su_i - w)}{(z(s)-t)(\overline{z(s)}-t)} \, dt \, ds \\
&\quad = \frac{|u_i|}{2} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \, \mathrm{i} \, \frac{u_j \times (su_i - w)}{s(u_j \times u_i) - (u_j \times w)} \left(\frac{1}{z(s)-t} - \frac{1}{\overline{z(s)}-t} \right) \, dt \, ds \\
&\quad = \mathrm{i} \, \frac{|u_i|}{2} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \, \left(\frac{1}{z(s)-t} - \frac{1}{\overline{z(s)}-t} \right) \, dt \, ds \\
&\quad = -|u_i| \operatorname{Im} \int_{-1}^1 P_k(s) \int_{-1}^1 \frac{N_\ell(t)}{z(s)-t} \, dt \, ds.
\end{aligned}$$

□

For identical elements $T_i = T_j$, we derive the following result.

Corollary 3.4.4. *For $T_i \in \mathcal{T}$, $k \in \mathbb{N}_0$, and $\ell \in \mathbb{N}$, there holds*

$$\widetilde{K}_{k,\ell}^{(i,i)} = -\frac{1}{2\pi} \int_{T_i} P_k^{(i)}(x) \int_{T_i} N_\ell^{(i)}(y) \frac{\partial}{\partial n_y} \log|x-y| \, ds_y \, ds_x = 0. \tag{3.16}$$

Proof. For identical elements there holds $u_i = u_j =: u$ and $w = 0$. Thus, Equation (3.15) turns into

$$\begin{aligned}
&\int_{T_i} P_k^{(i)}(x) \int_{T_i} N_\ell^{(i)}(y) \frac{\partial}{\partial n_y} \log|x-y| \, ds_y \, ds_x \\
&\quad = \frac{1}{|u|} \int_{-1}^1 P_k(s) \int_{-1}^1 N_\ell(t) \frac{u \times (su)}{(z(s)-t)(\overline{z(s)}-t)} \, dt \, ds = 0.
\end{aligned}$$

□

For the calculation of the Galerkin matrix $\mathbf{W}$ for the hypersingular operator, we use Nédélec's formula

$$\langle \mathcal{W}u, v\rangle = \left\langle \mathcal{V}\frac{d}{ds}u, \frac{d}{ds}v \right\rangle \quad \forall u, v \in H^1(\Gamma). \tag{3.17}$$

Thus, for the Galerkin matrix for the hypersingular operator, we have to compute integrals of the form

$$\widetilde{W}_{k,\ell}^{(i,j)} = \left\langle \mathcal{W}N_k^{(i)}, N_\ell^{(j)} \right\rangle = \left\langle \mathcal{V}\frac{d}{ds}N_k^{(i)}, \frac{d}{ds}N_\ell^{(j)} \right\rangle. \tag{3.18}$$

For $T_i \in \mathcal{T}$ and $x \in T_i$, the arc length derivative of the transformed Lobatto shape functions is given by

$$\frac{d}{ds}N_k^{(i)}(x) = \frac{d}{ds}\left(N_k(\gamma_i^{-1}(x))\right) = N_k'(\gamma_i^{-1}(x)) \cdot \frac{d}{ds}\gamma_i^{-1}(x) = \frac{2}{h_i}N_k'(\gamma_i^{-1}(x)),$$

(see Lemma 3.2.1), where

$$\begin{aligned} N_1'(t) &= \frac{1}{2}, \quad N_2'(t) = -\frac{1}{2}, \\ N_k'(t) &= P_{k-2}(t), \quad k \geq 3, \end{aligned}$$

for $t \in [-1, 1]$. This yields

$$\widetilde{W}_{k,\ell}^{(i,j)} = \left\langle \mathcal{V}\frac{d}{ds}N_k^{(i)}, \frac{d}{ds}N_\ell^{(j)} \right\rangle = \frac{4}{h_i\, h_j} \cdot \begin{cases} \frac{(-1)^{k+\ell}}{4}V_{0,0}^{(i,j)} & , k, \ell \in \{1,2\} \\ \frac{(-1)^{k-1}}{2}V_{0,\ell-2}^{(i,j)} & , k \in \{1,2\}, \ell \geq 3 \\ \frac{(-1)^{\ell-1}}{2}V_{k-2,0}^{(i,j)} & , \ell \in \{1,2\}, k \geq 3 \\ V_{k-2,\ell-2}^{(i,j)} & , k, \ell \geq 3, \end{cases} \tag{3.19}$$

where $V_{k,\ell}^{(i,j)}$ are entries of the single layer Galerkin matrix. Again, we have to combine $\widetilde{W}_{k,1}^{(i,j)}$ and $\widetilde{W}_{k,2}^{(i,j)}$ from neighboring elements in order to obtain the entries for the hat functions. In order to compute the Galerkin matrix for the hypersingular operator, we compute the Galerkin matrix for the single layer operator and scale and combine its entries according to equation (3.19).

Remark 3.4.5. (i) From Theorems 3.4.1 and 3.4.3, it can be seen that the computation of sub-blocks of the Galerkin entries $(V_{k,\ell}^{(i,j)})$ and $(K_{k,\ell}^{(i,j)})$, respectively, can be reduced to the evaluation of the functions $I_{k,\ell}^{-1}(a,b)$ and $O_{k,\ell}^{0}(a,b)$.

(ii) By switching the order of integration, we can always achieve that $|b| < 1$ (see Assumption 2.4.6). For the single layer Galerkin matrix, we use the switching property (2.57) for the function $I_{k,\ell}^{-1}(a,b)$. For the double layer Galerkin matrix, we use (2.68). Thus, for the assembly of the double layer Galerkin matrix, we need the functions $O_{k,\ell}^{0}(a,b)$ and $U_{k,\ell}^{0}(a,b)$.

3.4.2. Implementation

We use the formulas derived in the last section to assemble the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$, and $\mathbf{W}$. We assemble the matrices block-wise, i.e. for each combination of two boundary pieces $T_i, T_j \in \mathcal{T}$, we compute the corresponding matrix block according to Equations (3.11), (3.12) for $\mathbf{V}$ and Equation (3.14) for $\mathbf{K}$. In order to compute a matrix block $\mathbf{W}^{(i,j)}$ for $\mathbf{W}$, we compute the corresponding block $\mathbf{V}^{(i,j)}$ and use Equation (3.19). Listing 3.2 shows a simple MATLAB implementation for the computation of one sub-block of the Galerkin matrix $\mathbf{V}$ for the single layer operator.

Listing 3.2: MATLAB implementation for the assembly of a sub-block $\mathbf{V}^{(i,j)}$ of the Galerkin matrix $\mathbf{V}$ corresponding to two boundary pieces T_i and T_j.

```
1  function val = galVex(vert_i,vert_j,p_i, p_j)
2
3  ui = (vert_i(2,:)-vert_i(1,:))/2;
4  uj = (vert_j(2,:)-vert_j(1,:))/2;
5  w  = (vert_j(1,:)+vert_j(2,:))/2-(vert_i(1,:)+vert_i(2,:))/2;
6  utui = ui*ui';
7  utuj = uj*uj';
8
9  if(utui>utuj) % switch ui and uj s.t. |b|<1
10   a = complex( (ui(1)* w(1) + ui(2)* w(2))/utui, ...
11                (ui(1)* w(2) - ui(2)* w(1))/utui);
12   b = complex( (ui(1)*uj(1) + ui(2)*uj(2))/utui, ...
13               +(ui(1)*uj(2) - ui(2)*uj(1))/utui);
14   val = real(Im1(a,b,p_j, p_i)).';
15   val(1,1) = val(1,1) + 2 * log(utui);
16 else
17   a = complex(-(uj(1)* w(1) + uj(2)* w(2))/utuj, ...
18               -(uj(1)* w(2) - uj(2)* w(1))/utuj);
19   b = complex( (uj(1)*ui(1) + uj(2)*ui(2))/utuj, ...
20               +(uj(1)*ui(2) - uj(2)*ui(1))/utuj);
21   val = real(Im1(a,b,p_i, p_j));
22   val(1,1) = val(1,1) + 2 * log(utuj);
23 end
24 val = -sqrt(utui*utuj)/(2*pi)*val(1:p_i+1, 1:p_j+1);
```

The inputs for the function are the 2-by-2 matrices `vert_i` and `vert_j`, in which the vertices of T_i and T_j are stored, and the polynomial degrees `p_i` and `p_j` of the ansatz space on T_i and T_j, respectively. In the function we calculate the u_i, u_j and w as shown in Figure 3.1 (lines 3-7).

Then we use these vectors to calculate the complex numbers $a, b \in \mathbb{C}$. Finally, we compute $I^{-1}_{k,\ell}(a,b)$ for $k = 0, \ldots, p_i$, and $\ell = 0, \ldots p_j$ and build the matrix block $\mathbf{V}^{(i,j)}$ according to Equation (3.11). In order to ensure that the number of digits for the evaluation of $I^{-1}_{k,\ell}(a,b)$ is calculated correctly, we have to require $|b| < 1$ (see Section 2.4.2). To achieve this, we switch the order of integration, i.e. we switch u_i and u_j and set $w = -w$ (lines 9-15).
For the assembly of the full Galerkin matrix $\mathbf{V}$, we have to loop over all combinations of boundary pieces $T_i, T_j \in \mathcal{T}$ and compute the corresponding matrix block. Since the matrix $\mathbf{V}$ is symmetric, it suffices to loop over all boundary pieces T_i, T_j with $i \geq j$. Furthermore, for identical elements we use the formula (3.12).
In order to ensure the stability of the computations and to decrease the computational time, we implement all functions for the assembly of the Galerkin matrices in `C`, where we use `MPC` for the calculation of the complex numbers $a, b \in \mathbb{C}$. Furthermore, we parallelize the double loop over the boundary pieces using `openMP`. Here, we achieve a good load balancing for the parallelization with N_T threads with Algorithm 3.1.

Algorithm 3.1 Assembly of the block Galerkin matrix $\mathbf{V} = (\mathbf{V}^{(i,j)})$.

1: $\mathcal{N}_e$ is the number of elements of the boundary mesh $\mathcal{T}$.
2: N_T is the total number of threads.
3: Set $cnt = 0$.
4: **for** $i = 1, \ldots, \mathcal{N}_e$ **do**
5: **if** $k + 1 == N_T - (i \bmod N_T)$ **then**
6: Thread k computes the diagonal block $\mathbf{V}^{(i,i)}$.
7: **end if**
8: **for** $j = i + 2, \ldots, \mathcal{N}_e$ **do**
9: **if** $(cnt \bmod N_T) == k$ **then**
10: Thread k computes the far field block $\mathbf{V}^{(i,j)}$.
11: **end if**
12: cnt++;
13: **end for**
14: **end for**

Figure 3.5 shows an example of a 5×5 block matrix distributed among three threads. Thread 0 computes one diagonal and four other matrix blocks. Threads 1 and 2 compute two diagonal and three off diagonal blocks each.

For the assembly of the matrices $\mathbf{K}$ and $\mathbf{W}$, we proceed similarly. We compute sub-blocks for different combinations of $T_i, T_j \in \mathcal{T}$ and write the blocks into the overall Galerkin matrices. However, here we have to add up entries from neighboring elements to obtain the matrix entries

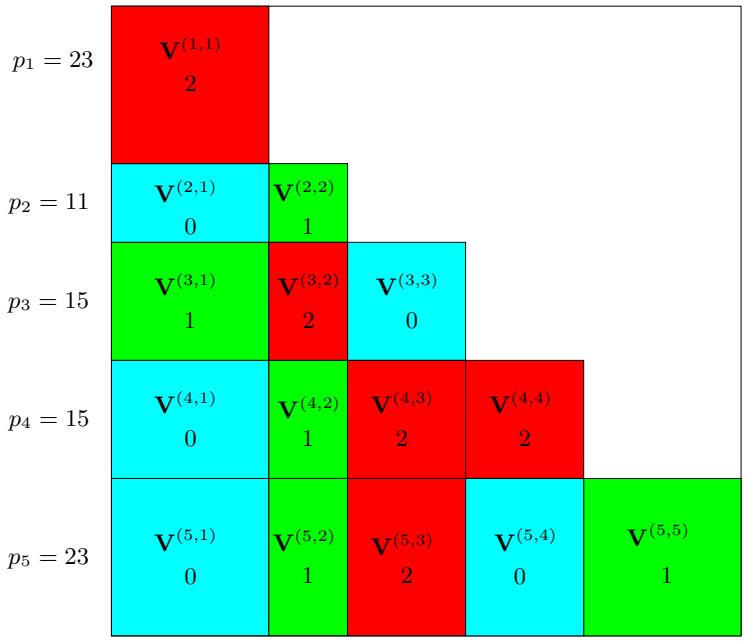


Figure 3.5.: Distribution of matrix blocks $\mathbf{V}^{(i,j)}$ among three threads. Here, the boundary is discretized into five elements $T_1, \dots, T_5$ and the polynomial degrees are as shown in the graphic. Blocks calculated by thread 0 are blue, blocks calculated by thread 1 are green, and blocks calculated by thread 2 are red.

that correspond to hat functions. For the parallelized implementation, these additions have to be atomic operations.

Table 3.1 shows times for the assembly of the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$, and $\mathbf{W}$ for different ansatz spaces S_0 and S_1. All computations are performed for the boundary Γ of the L-shaped domain $[-1,1]\times[-1,1]\setminus[-1,0]\times[-1,0]$. We assemble the Galerkin matrices for the following test cases:

Case 1 Uniform grid with $\mathcal{N}_e = 16$ and uniform polynomial degree vector with $p = 135$.

Case 2 Uniform grid with $\mathcal{N}_e = 128$ and uniform polynomial degree vector with $p = 7$.

Case 3 hp-geometric grid (see Figure 1.1(b)) with $\mathcal{N}_e = 54$. The polynomial degree vector grows linearly from $p = 0$ near the re-entrant corner to $p = 23$.

It can be seen that the computation times for the assembly of $\mathbf{V}$ and $\mathbf{W}$ are almost the same. The assembly of of $\mathbf{K}$ is slower. Furthermore, the parallelization for all three matrices scales well.

Table 3.1.: Computation times in seconds for the assembly of the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$, and $\mathbf{W}$ for the three test cases and different numbers of threads. The computations are performed on a desktop computer with an AMD Phenom II X6 processor (6 cores), 16GB RAM, and the Ubuntu 10.04 operating system.

	Case 1			Case 2			Case 3		
# threads	$\mathbf{V}$	$\mathbf{K}$	$\mathbf{W}$	$\mathbf{V}$	$\mathbf{K}$	$\mathbf{W}$	$\mathbf{V}$	$\mathbf{K}$	$\mathbf{W}$
1	8.15	18.62	8.18	9.03	17.11	9.01	1.10	1.92	1.11
2	4.33	9.79	4.28	4.71	8.84	4.70	0.60	1.00	0.58
3	3.11	6.76	3.11	3.21	6.09	3.22	0.40	0.69	0.40
4	2.66	5.90	2.66	2.70	5.39	2.72	0.33	0.58	0.33
5	2.30	4.67	2.29	2.18	4.14	2.19	0.27	0.47	0.27
6	1.84	3.96	1.85	1.83	3.52	1.88	0.25	0.40	0.24

3.5. The epsBEM Software Package

All implementations discussed in Chapters 2 and 3 are included in the software package epsBEM.

epsBEM [BBF13] is a software package for hp-BEM for the Laplace and the Lamé operator in two dimensions developed by A. Bantle, S. Funken, and the author. The goal of the software package is to provide a stable and efficient implementation of hp-BEM while being easy enough for students and researchers alike to use. Thus, the software is written in a combination of `C` and MATLAB. The functions that are critical with respect to performance and stability are written in `C` and optimized and parallelized. The top layer functions, which are of interest for the user of the epsBEM package, are written in MATLAB and contain short and straight-forward code as shown for example in Listings 3.1 and 3.2. Figure 3.6 shows an overview of the functionality of the epsBEM package.

The foundation of the package is made up of two `C` libraries. In the first library, functions for the evaluation of the modified associated Legendre functions of the second kind, which are described in Section 2.3 (see especially Section 2.3.4), are collected. The second library contains functions for the evaluation of the double integrals introduced in Section 2.4. For polynomial degrees smaller than two, the library provides functions to compute the double integral semi-analytically, i.e. the inner integrals, which are modified associated Legendre functions, are computed analytically and the outer integrals are evaluated using an adapted Gauss quadrature rule. For larger polynomial degrees, we use `MPFR` and `MPC` for the computations as described in Section 2.4.2. Both libraries are connected to MATLAB with `mex` functions.

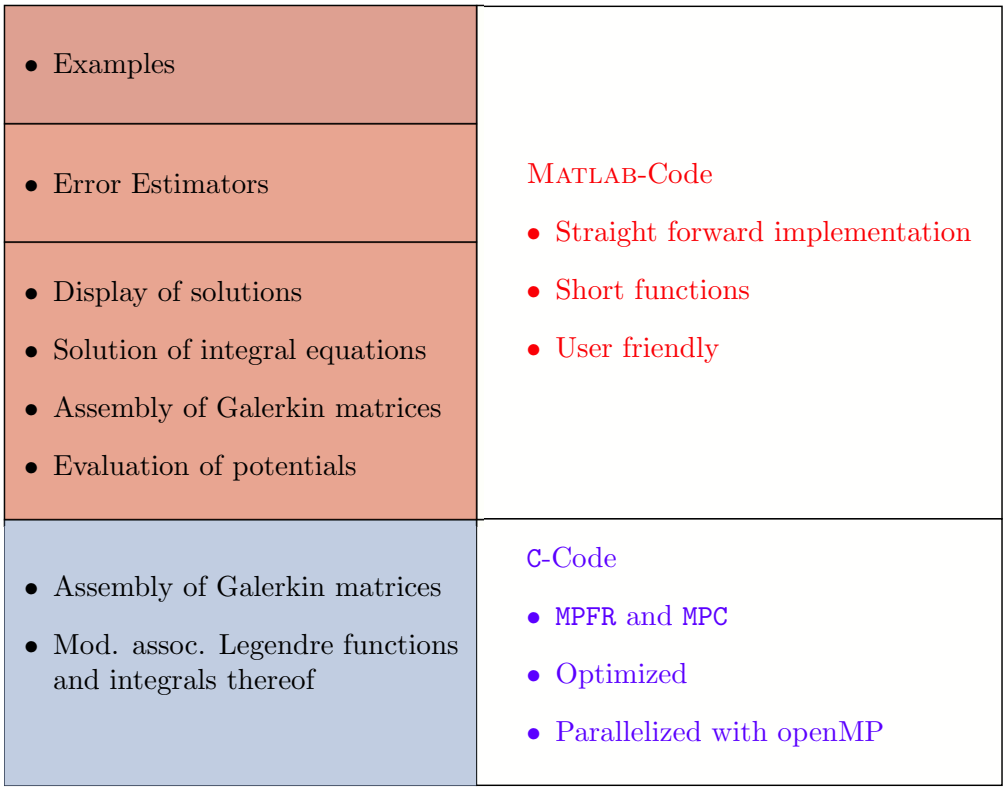


Figure 3.6.: Overview of the functionality of the epsBEM software package.

Based on these two C libraries, functions for the evaluation of all four potentials for the Laplace and the Lamé operator are implemented in MATLAB. For the Laplace operator, see the descriptions in Section 3.3; for the Lamé operator, see [Ban12]. Functions for the computation of Galerkin entries are implemented in both C and MATLAB. Whereas the MATLAB implementations are as short and straight-forward as possible, the C functions for the assembly of Galerkin methods are optimized and parallelized with openMP (see Algorithm 3.1).
The epsBEM package provides functions for the solution of Symm's, the hypersingular, and the mixed integral equation for both Laplace and Lamé operators. The Galerkin matrices are preconditioned with a diagonal preconditioner and the linear systems of equations are solved with the MATLAB backslash operator. Furthermore, several error estimators, namely the estimators introduced in [FLP08] and [EFGP12] as well as the error estimators introduced in Chapter 4, are implemented. The epsBEM package also provides functions for the uniform and h- and hp-adaptive mesh refinement. Here, we implement the adaptive algorithms described in Section 4.3. Finally, functions for displaying solutions and several examples are included in the epsBEM software package.

3.6. Numerical Experiments

We perform numerical experiments to show the stability and efficiency of our implementation of hp-BEM as described in Section 3.3 and 3.4. For the numerical experiments, we use the functions of the epsBEM software package. For the assembly of the Galerkin matrices, we use six threads and the linear systems of equations are solved using the MATLAB backslash operator also using six threads for the computations. All computations are performed on a desktop computer with an AMD Phenom II X6 processor (6 cores), 16GB RAM, and the Ubuntu 10.04 operating system.

The first problem we consider is Symm's integral equation

$$(\mathcal{V}\phi)(x,y) = \frac{\log(2)}{2}$$

on the slit $\Gamma = [-1,1] \times \{0\}$. The exact solution is given by $\phi(x,y) = (1-x^2)^{-1/2}$ and the energy is given by $\|\phi\|_{\mathcal{V}}^2 = \langle \mathcal{V}\phi, \phi \rangle = \pi \frac{\log(2)}{2}$.

Remark 3.6.1. For Symm's integral equation $\mathcal{V}\phi = f$ on the slit $\Gamma = [-1,1]$ with polynomial right-hand side, the exact solution ϕ can be computed analytically. In [Rea79], the result

$$\int_{-1}^{1} \log|x-t| T_n(t) \sqrt{1-t^2}\, dt = \begin{cases} -\pi \log(2), & n = 0 \\ \frac{\pi}{n} T_n(x), & n > 0 \end{cases} \tag{3.20}$$

is stated and proven. Here, T_n are the Chebyshev polynomials of the first kind. Thus, if we plug in a Chebyshev expansion $f(x) = \sum_{k=0}^{\infty} a_k T_k(x)$ for the right-hand side of Symm's integral equation $\mathcal{V}\phi = f$, we obtain the exact solution ϕ directly.

We run different refinement algorithms, namely uniform h- and p-methods as well as geometric hp-methods with two different mesh parameters $\rho = 0.5$ and $\rho = 0.17$. The choice of $\rho = 0.17$ is motivated by [GB86], where it is shown that for a linear polynomial degree distribution, as the one shown in Section 1.4, the optimal mesh parameter is $\rho = (\sqrt{2}-1)^2 \approx 0.17$.
Figure 3.7 (top picture) shows the Galerkin errors and computation times over the degrees of freedom $\mathcal{N}$. In Table 3.2 the computation times for the assembly and solution of the linear system are shown. are shown.
Since the exact solution ϕ has a singularity of order $1/2$, we expect a convergence rate of $1/2$ for the uniform h-version and a convergence rate of 1 for the uniform p-version (see Remark 1.5.4). The numerical experiments confirm these convergence rates. Furthermore, it can be seen that geometric hp-methods yield exponential convergence, where the geometric hp-method with $\rho = 0.17$ converges considerably faster as compared to the one with $\rho = 0.5$. In the bottom picture of Figure 3.7, we display the computation times over the degrees of freedom $\mathcal{N}$. The computation times are the sum of the times for the assembly and the solution of the linear system.

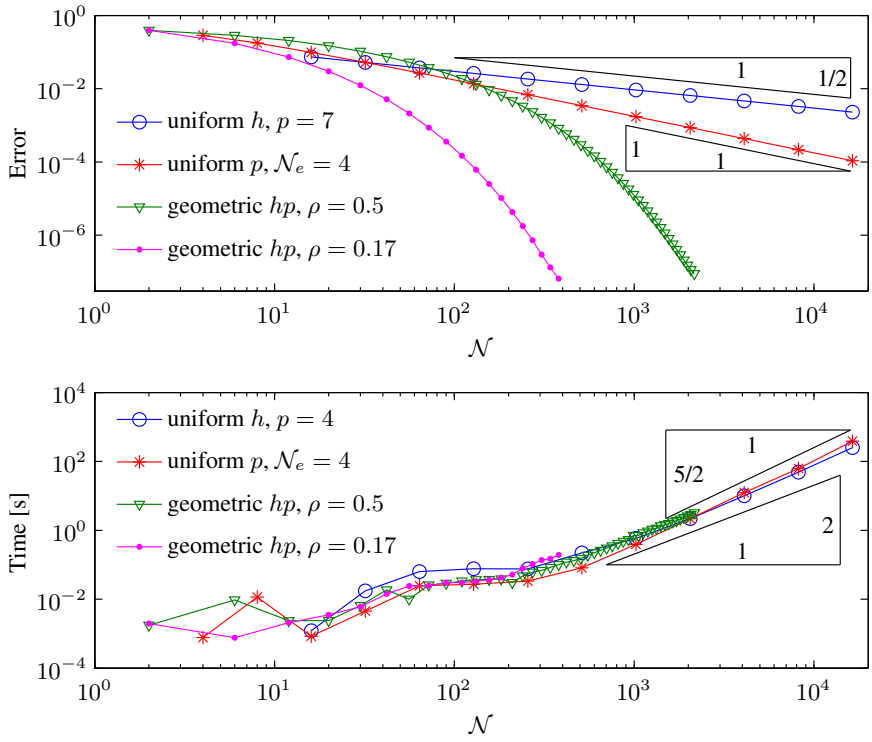


Figure 3.7.: Top: Error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over the number of degrees of freedom $\mathcal{N}$ for Symm's integral equation $\mathcal{V}\phi = \log(2)/2$ on the slit for different mesh refinement strategies. Bottom: Computation times (assembly of Galerkin matrix and right-hand side and solution of the linear system) over $\mathcal{N}$.

Table 3.2.: Computation times for the assembly of the the Galerkin matrix and the right-hand side and for the solution of the linear system for Symm's integral equation $\mathcal{V}\phi = \log(2)/2$ on the slit for different $\mathcal{N}$ for a uniform h- and p-method.

	uniform-h		uniform-p	
$\mathcal{N}$	assemble	solve	assemble	solve
1024	0.49	0.10	0.20	0.19
2048	1.84	0.34	0.98	1.20
4096	7.79	2.23	6.23	6.14
8192	35.62	13.03	42.45	19.59
16384	168.38	82.32	285.96	102.24

It can be seen that the computation times behave almost quadratically with respect to $\mathcal{N}$. For the geometric hp-method with $\rho = 0.17$, the finest problem, which produces a Galerkin solution that has an error of less than 10^{-7}, is assembled and solved in less than one second. Table 3.2 shows the computation times for the assembly and the solution of the linear system separately for uniform h- and p-methods. We see that the time for the solution of the linear system is considerably faster than the assembly in both cases. Here, the main effort for the assembly of the linear system is needed for the assembly of the Galerkin matrix $\mathbf{V}$. The time for the computation of the right-hand side is negligible.

The next benchmark is performed for the L-shape domain and the function u that is given in polar coordinates by

$$u(r, \alpha) = r^{2/3} \cos(\frac{2}{3}\alpha). \tag{3.21}$$

Using $\psi = \gamma_0 u$, we obtain a benchmark problem for Symm's integral equation, i.e. we consider the equation

$$\mathcal{V}\phi = \left(\mathcal{K} + \frac{1}{2}\right)\psi.$$

For the computation of the Galerkin error by $\|\phi - \phi_h^p\|_{\mathcal{V}}^2 = \|\phi\|_{\mathcal{V}}^2 - \|\phi_h^p\|_{\mathcal{V}}^2$, we use the extrapolated energy $\|\phi\|_{\mathcal{V}}^2 = 0.20205809888221625$. We run a uniform h-method with constant polynomial degrees $p = 3$, a uniform p-method on a uniform mesh with 8 elements and geometric hp-methods with mesh parameters $\rho = 0.5$ and $\rho = 0.17$. In Figure 3.8 (top picture), the Galerkin errors over the dimension $\mathcal{N}$ of the discrete ansatz space are displayed. It can be seen that the uniform methods converge algebraically, where the uniform p-method has twice the rate of convergence of the uniform h-method. Both geometric hp-methods yield exponential convergence, where the method with $\rho = 0.17$ converges considerably faster. The plot of the computation times, which are the sum of the times for the assembly and the solution of the linear system, is shown in the bottom picture of Figure 3.8. It can be seen that the times for the uniform h-method are proportional to $\mathcal{N}^2$, for the uniform p-method the times grow faster but are still smaller than the times for the uniform h-method. In Table 3.3, the times for the assembly and the solution of the linear system are listed separately for both uniform methods. It can be seen that the time for the solution of the linear system grows almost like $\mathcal{N}^3$ but is smaller than the time for the assembly by a factor of more than ten.

Our third benchmark problem is the hypersingular integral equation

$$\mathcal{W}\psi = 1$$

that has the exact solution $\psi(x, y) = 2\sqrt{1 - x^2}$. The exact energy is given by $\|\psi\|^2 = \pi$. For this problem we run a uniform h-method with constant polynomial degrees $p = 4$, a uniform

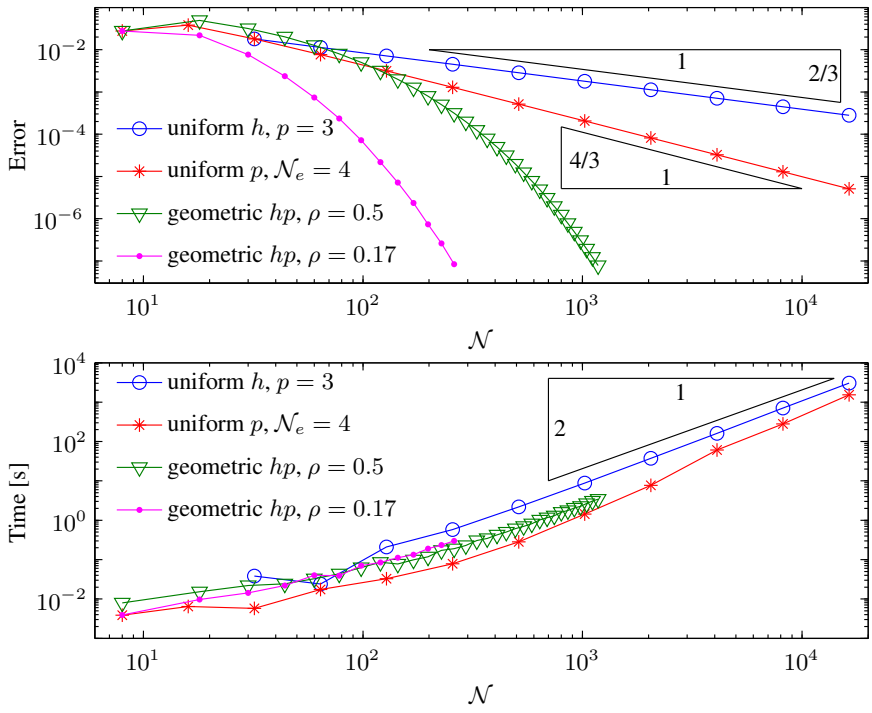


Figure 3.8.: Top: Error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over the number of degrees of freedom $\mathcal{N}$ for Symm's integral equation $\mathcal{V}\phi = (\mathcal{K} + 1/2)\psi$ for $\psi = \gamma_0 u$ with u from (3.21) on the boundary of the L-shape for different mesh refinement strategies. Bottom: Computation times (assembly of Galerkin matrix and right hand side and solution of the linear system) over $\mathcal{N}$.

Table 3.3.: Computation times for the assembly of the the Galerkin matrix and the right-hand side and for the solution of the linear system for Symm's integral equation $\mathcal{V}\phi = (\mathcal{K} + 1/2)\psi$ for $\psi = \gamma_0 u$ with u from (3.21) on the boundary of the L-shape for different $\mathcal{N}$ for a uniform h- and p-method.

	uniform-h		uniform-p	
$\mathcal{N}$	assemble	solve	assemble	solve
1024	8.76	0.06	1.39	0.06
2048	37.25	0.29	7.27	0.47
4096	158.59	2.08	38.15	22.78
8192	694.16	12.78	215.38	64.50
16384	2952.69	81.79	1311.84	219.89

p-method for a uniform mesh with 4 elements, and geometric hp-methods with mesh parameters $\rho = 0.5$ and $\rho = 0.17$. The Galerkin errors and computation times over $\mathcal{N}$ are shown in the top picture of Figure 3.9 and computation times for the assembly and solution of the linear system are shown in Table 3.4.

While both uniform methods yield algebraic convergence, the uniform p-method having twice the convergence rate of the uniform h-method, both geometric hp-methods result in exponential convergence of the Galerkin errors. For $\rho = 0.17$, the Galerkin error is smaller than 10^{-7} for less than 500 degrees of freedom. Looking at the computation times (Figure 3.9, bottom picture), which are again the times for assembling and solving the linear system, we see that the times for the uniform h-method are proportional to $\mathcal{N}^2$, and the times for the uniform p-method are proportional to $\mathcal{N}^{5/2}$. Again, the assembly of the Galerkin matrix consumes the main part of the computation. The times for the solution of the linear systems are faster by a factor of more than four; see Table 3.4.

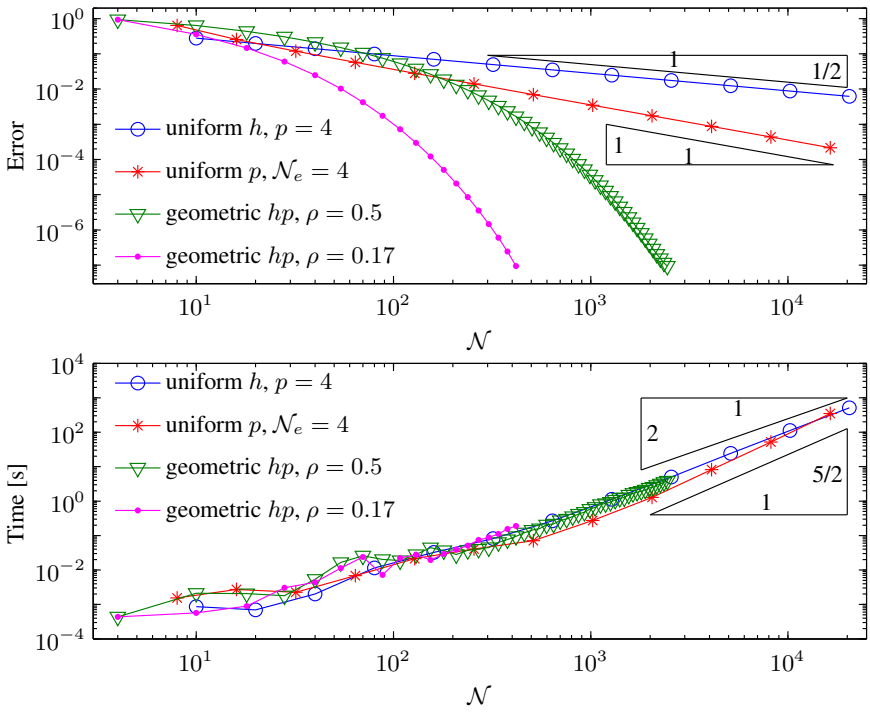


Figure 3.9.: Top: Error $\|\psi - \psi_h^p\|_{\mathcal{W}}$ over the number of degrees of freedom $\mathcal{N}$ for the hypersingular integral equation $\mathcal{W}\psi = 1$ on the slit for different mesh refinement strategies. Bottom: Computation times (assembly of Galerkin matrix and right-hand side and solution of the linear system) over $\mathcal{N}$.

Table 3.4.: Computation times for the assembly of the the Galerkin matrix and the right-hand side and for the solution of the linear system for the hypersingular integral equation $\mathcal{W}\psi = 1$ on the slit for different $\mathcal{N}$ for a uniform h- and p-method.

	uniform h			uniform p	
$\mathcal{N}$	assemble	solve	$\mathcal{N}$	assemble	solve
1280	1.06	0.06	1024	0.21	0.06
2560	4.68	0.32	2048	1.00	0.27
5120	22.05	2.14	4096	6.44	1.72
10240	99.97	12.67	8192	41.58	10.13
20480	434.97	77.12	16384	284.17	63.83

Remark 3.6.2. (i) Overall, the numerical experiments show that our implementation of hp-BEM, especially the assembly of the Galerkin matrices, produces accurate results even for very high polynomial degrees. Note that for the uniform p-methods, we performed computations for $p = 4095$ for the slit examples and $p = 1023$ in the case of the L-shape benchmark. Our implementation also works on geometric meshes. Here, the Galerkin error can be reduced to values smaller than 10^{-7}. This is the smallest possible error in double precision, because we take the square root in the computation of the error.

(ii) The use of `MPFR` and `MPC` for the assembly of the Galerkin matrix results in computation times for the assembly of the Galerkin matrices that are much slower than the times for the solution of the linear systems.

(iii) The times for the assembly of the Galerkin matrices scale almost optimally in the sense that they are proportional to $\mathcal{N}^\alpha$ with α between 2 and 2.5. Since for the assembly of a full $\mathcal{N} \times \mathcal{N}$ matrix we have to compute $\mathcal{N}^2$ matrix entries, our experiments show that the times to compute one matrix entry are almost independent of $\mathbf{h}$ and $\mathbf{p}$. For uniform p-methods the computation time grows faster than $\mathcal{N}^2$. This results from the use of more digits for the assembly of the Galerkin matrix for larger polynomial degrees (see Sections 2.4.2 and 3.4). For comparison, if for the computation of Galerkin entries we used a naive Gauss quadrature rule for the outer integrals and the stable recurrence relations derived in Section 2.3.1 for the inner integrals, the computational cost for the assembly of the Galerkin matrix would be of the order $\mathcal{O}(p^4)$ (see Remark 2.4.7 for more details). Thus, especially for large polynomial degrees, our implementation can still be considered efficient.

(iv) One way to increase the efficiency of the computations would be to implement data compression techniques, which are discussed in Chapter 5.

4. Error Estimators and hp-Adaptive Algorithms

In Chapter 1.5 we present several a priori error estimates for hp-boundary element methods. It is shown that uniform h- or p-methods result in algebraic convergence. In order to achieve faster convergence one has to use boundary meshes that are adapted to the smoothness of the solution, i.e. the meshes need to be finer where the solution has singularities. Using a suitable combination of special types of meshes, namely geometric meshes, and polynomial degree vectors, exponential convergence of refinement algorithms can be achieved (see Section 1.4 and Proposition 1.5.5).

For most realistic problems, where the solution is not known, adaptive algorithms need to be constructed. For the construction of adaptive algorithms, error indictors, which give an estimate for the error on each element of the boundary mesh, are required. There exist different error estimators for h- and hp-BEM, for example residual type error estimators (see e.g. [CS95, CFS96]), two-level estimators based on decompositions of enriched ansatz spaces (see e.g. [HMS01, Heu02]), and $h - h/2$ error estimators (see e.g. [FLP08, EFGP12]). The idea of $h - h/2$ estimators is to obtain an estimate of the error by comparing two solutions that were computed with respect to two different meshes.
Adaptive mesh refinement algorithms for h-BEM are presented analyzed in e.g. [CS95, CFS96] for residual error estimators. For adaptive algorithms using $h - h/2$ error estimators, convergence is shown in [FLOP10, AFLP12]. Recently, it was shown that there exist h-adaptive algorithms that yield quasi-optimal algebraic convergence [FKMP13].
In this chapter we examine if and how $h - h/2$ error estimators can be generalized to error estimators that can be used in hp-BEM. We generalize the theory whenever appropriate estimates exist. However, for the generalization of $h - h/2$ error estimators, we have to use weighted L^2 norms and weighted L^2 projections as well as fractional order Sobolev norms and can therefore not use the standard inverse and approximation estimates. Thus, for cases where theoretical estimates have not yet been proven, we use numerical methods, namely eigenvalue problems, to examine the validity of the estimates involved. This chapter should be seen as a first survey of how $h - h/2$ error estimators could be generalized and as a basis for further research, especially from the theoretical point of view.

In the first section of this chapter we introduce the already existing $h - h/2$ error estimators for Symm's integral equation for lowest order BEM. Sections 2 presents and analyzes possible generalizations. In the third section we present two hp-adaptive algorithms and prove that they produce a sequence of numerical solutions that converges to the exact solution. In the fourth and last section we discuss the implementation of the introduced error estimators and present the results of numerical experiments that show that in fact the numerical solutions constructed by both hp-adaptive algorithms converge exponentially to the exact solution.
Throughout this chapter we use the notation from Section 3.2, i.e. $\mathcal{T}$ is a mesh on Γ, $\mathbf{h}$ and $\mathbf{p}$ are mesh width and polynomial degree functions, and $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ denotes the space of piecewise polynomials with respect to $\mathcal{T}$ as defined in Definition 3.2.2. Furthermore, for $u \in \widetilde{H}^{-1/2}(\Gamma)$, $|\!|\!| u |\!|\!| := \|u\|_{\mathcal{V}}$ denotes the energy norm with respect to the single layer operator. Note that $|\!|\!| \cdot |\!|\!|_{\mathcal{V}}$ is an equivalent norm on $\widetilde{H}^{-1/2}(\Gamma)$.

4.1. $h - h/2$ Error Estimators for Lowest Order BEM

The $h - h/2$ error estimators for lowest order BEM are defined as follows.

Definition 4.1.1. ([FLP08]) Let ϕ_h^p be the solution of Symm's integral equation in the discrete space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. Let $\widehat{\mathcal{T}}$ be the boundary mesh that is obtained by refining each element $T \in \mathcal{T}$ uniformly, and let $\mathbf{h}/2$ be the mesh width function on the new mesh. Lastly, let $\phi_{h/2}^p \in S_0(\widehat{\mathcal{T}}, \mathbf{h}/2, \mathbf{p})$ be the solution in the fine space and $\Pi_{h,p} : \widetilde{H}^{-1/2}(\Gamma) \to S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ be the element-wise L^2 projection onto $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$.
The $h - h/2$ error estimators are defined by

$$\eta_0 = |\!|\!| \phi_{h/2}^p - \phi_h^p |\!|\!|, \tag{4.1a}$$

$$\widetilde{\eta}_0 = |\!|\!| (1 - \Pi_{h,p}) \phi_{h/2}^p |\!|\!|, \tag{4.1b}$$

$$\mu_0 = \|\mathbf{h}^{1/2} (\phi_{h/2}^p - \phi_h^p)\|_{L^2(\Gamma)}, \tag{4.1c}$$

$$\widetilde{\mu}_0 = \|\mathbf{h}^{1/2} (1 - \Pi_{h,p}) \phi_{h/2}^p\|_{L^2(\Gamma)}. \tag{4.1d}$$

In [FLP08] it is proven that the estimators from (4.1) are efficient and reliable as long as $\mathbf{p}$ is held constant. First the equivalence of the Galerkin error and the estimator η_0 is shown. The equivalence of the other estimators follows from the inequalities

$$\widetilde{\mu}_0 \overset{(A)}{\leq} \mu_0 \overset{(B)}{\leq} C_1 \eta_0, \tag{4.2a}$$

$$\eta_0 \overset{(C)}{\leq} \widetilde{\eta}_0 \overset{(D)}{\leq} C_2 \widetilde{\mu}_0, \tag{4.2b}$$

where the constants C_1 and C_2 are independent of the mesh width $\mathbf{h}$. Since some of the proofs can be generalized for hp-BEM, we state them here. To show the equivalence of η_0 with the

Galerkin error, we need the saturation assumption

$$|||\phi - \phi_{h/2}^p||| \leq C_{sat}|||\phi - \phi_h^p|||, \tag{4.3}$$

where C_{sat} is independent of $\mathbf{h}$.

Lemma 4.1.2. ([FLP08]) *Under the saturation assumption (4.3), there holds with* $C = (1 - C_{sat})^{-1/2}$

$$\eta_0 \leq |||\phi - \phi_h^p||| \leq C\eta_0.$$

Proof. Due to Galerkin orthogonality, there holds

$$|||\phi - \phi_h^p|||^2 = |||\phi - \phi_{h/2}^p|||^2 + |||\phi_{h/2}^p - \phi_h^p|||^2 = |||\phi - \phi_{h/2}^p|||^2 + \eta_0^2, \tag{4.4}$$

which yields $\eta_0^2 \leq |||\phi - \phi_h^p|||^2$. Using the saturation assumption (4.3) in (4.4), we obtain

$$|||\phi - \phi_h^p|||^2 \leq C_{sat}^2|||\phi - \phi_h^p|||^2 + \eta_0^2,$$

which is equivalent to

$$|||\phi - \phi_h^p|||^2 \leq (1 - C_{sat})^{-1/2}\eta_0.$$

□

Remark 4.1.3. For finite element methods the saturation assumption (4.9) is proven under some conditions on the local mesh refinement in [DN02]. However, for BEM there is no proof of the saturation assumption. In [CP12] an adaptive algorithm is presented that checks the saturation assumption in each refinement step and enforces it if necessary.

The following result follows from the fact that the L^2 projection $\Pi_{h,p}$ is the orthogonal projection onto $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ with respect to the L^2 scalar product. We state the proof, because it can be generalized directly for hp-BEM.

Lemma 4.1.4. ([FLP08]) *There holds* $\eta_0 \leq \widetilde{\eta}_0$ *and* $\widetilde{\mu}_0 \leq \mu_0$.

Proof. Let $G_h : \widetilde{H}^{-1/2}(\Gamma) \to S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ be the Galerkin projection onto $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. Since $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}) \subseteq S_0(\widehat{\mathcal{T}}, \mathbf{h}/2, \mathbf{p})$, there holds $\phi_h^p = G_h\phi = G_h\phi_{h/2}^p$. Together with the best approximation property of the Galerkin projection, this implies

$$\eta_0 = |||\phi_{h/2}^p - \phi_h^p||| = |||(1 - G_h)\phi_{h/2}^p||| \leq |||(1 - \Pi_{h,p})\phi_{h/2}^p||| = \widetilde{\eta}_0.$$

Because $\Pi_{h,p}$ is the best approximation with respect to the L^2 norm, there holds for all elements $T_i \in \mathcal{T}$

$$\|(1 - \Pi_{h,p})\phi_{h/2}^p\|_{L^2(T_i)}^2 = \min_{\psi_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})} \|\phi_{h/2}^p - \psi_h^p\|_{L^2(T_i)}^2 \leq \|\phi_{h/2}^p - \phi_h^p\|_{L^2(T_i)}^2,$$

and thus

$$\widetilde{\mu}_0^2 = \sum_{i=1}^{\mathcal{N}_e} h_i \|(1 - \Pi_{h,p})\phi_{h/2}^p\|_{L^2(T_i)}^2 \leq \sum_{i=1}^{\mathcal{N}_e} h_i \|\phi_{h/2}^p - \phi_h^p)\|_{L^2(T_i)}^2 = \mu_0^2.$$

□

In order to show Inequality (4.2) (D), we need an approximation result, which we state without proof.

Lemma 4.1.5. ([CP06]) *Let $\Pi_{h,p}$ be the L^2 projection onto $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. Then, for any $v \in L^2(\Gamma)$, there holds*

$$|||v - \Pi_{h,p}v||| \leq C\|\mathbf{h}^{1/2}(v - \Pi_{h,p}v)\|_{L^2(\Gamma)} \leq C\|\mathbf{h}^{1/2}v\|_{L^2(\Gamma)} \tag{4.5}$$

with a constant C that depends on Γ but not on the triangulation $\mathcal{T}$.

Inequality (D) follows directly from (4.5). Inequality (4.2) (B) follows from the following inverse estimate.

Lemma 4.1.6. ([GHS05]) *For any function $\phi_h \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, there holds*

$$\|\mathbf{h}^{1/2}\phi_h^p\|_{L^2(\Gamma)} \leq C(p)|||\phi_h^p|||, \tag{4.6}$$

where the constant $C(p)$ is independent of $\mathbf{h}$ but depends on p.

4.2. Error Estimators for hp-BEM

Next we investigate how the $h-h/2$ error estimators introduced in Section 4.1 can be generalized for p- and hp-BEM. The constants in the approximation estimate (4.5) and the inverse estimate (4.6) are independent of $\mathbf{h}$ but not independent of $\mathbf{p}$. Therefore, the error estimators introduced in Definition 4.1.1 are not useful for p-BEM.
In [Geo07] a more general inverse inequality, which can be used for hp-BEM, is proven.

Lemma 4.2.1. ([Geo07, Theorem 3.9]) *For any function $\phi_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, there holds*

$$\left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}\phi_h^p \right\|_{L^2(\Gamma)} \leq C|||\phi_h^p|||, \tag{4.7}$$

where the constant C is independent of $\mathbf{h}$ and $\mathbf{p}$.

This motivates the definition of the following error estimators.

Definition 4.2.2. For a given mesh $\mathcal{T}$ with mesh width function $\mathbf{h}$ and polynomial degree function $\mathbf{p}$, let $S_{h,p} = S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}) \subset H^{-1/2}(\Gamma)$ and $\widehat{S}_{h,p} = S_0(\widehat{\mathcal{T}}, \widehat{\mathbf{h}}, \widehat{\mathbf{p}}) \subset H^{-1/2}(\Gamma)$ be another discrete space with $S_{h,p} \subset \widehat{S}_{h,p}$. Let $\phi_h^p \in S_{h,p}$ and $\widehat{\phi}_h^p \in \widehat{S}_{h,p}$ be the solution of Symm's integral equation in the respective discrete spaces. Finally, let $\Pi_{h,p} : H^{-1/2}(\Gamma) \to S_{h,p}$ be the element-wise L^2 projection onto $S_{h,p}$.
We define the error estimators

$$\eta = |||\widehat{\phi}_h^p - \phi_h^p|||, \tag{4.8a}$$

$$\widetilde{\eta} = |||(1 - \Pi_{h,p})\widehat{\phi}_h^p|||, \tag{4.8b}$$

$$\mu = \left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}(\widehat{\phi}_h^p - \phi_h^p) \right\|_{L^2(\Gamma)}, \tag{4.8c}$$

$$\widetilde{\mu} = \left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}(1 - \Pi_{h,p})\widehat{\phi}_h^p \right\|_{L^2(\Gamma)}. \tag{4.8d}$$

Remark 4.2.3. There are several options for the choice of the fine space $\widehat{S}_{h,p}$.

(i) For $\widehat{S}_{h,p} = S_{h/2,p} = S_0(\widehat{\mathcal{T}}, \mathbf{h}/2, \mathbf{p})$, we obtain a generalization of the $h - h/2$ error estimators introduced in Definition 4.1.1. In the following we denote these estimators by η_h, $\widetilde{\eta}_h$, μ_h, and $\widetilde{\mu}_h$.

(ii) Another option is $\widehat{S}_{h,p} = S_{h,p^*} = S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}^*)$, where $\mathbf{p}^* > \mathbf{p}$, and we call the resulting error estimators $p - p^*$ estimators. The $p - p^*$ estimators are denoted by η_p, $\widetilde{\eta}_p$, μ_p, and $\widetilde{\mu}_p$. Usually one chooses $\mathbf{p}^* = 2\,\mathbf{p} + 1$ in order to ensure that $\mathbf{p}^*/\mathbf{p} \geq c > 1$.

(iii) The error estimators that result from choosing $\widehat{S}_{h,p} = S_{h/2,p^*} = S_0(\widehat{\mathcal{T}}, \mathbf{h}/2, \mathbf{p}^*)$ are called $h - h/2$, $p - p^*$ estimators and denoted by η_{hp}, $\widetilde{\eta}_{hp}$, μ_{hp}, and $\widetilde{\mu}_{hp}$.

Remark 4.2.4. (i) A scaling with $\mathbf{h}$ and $\mathbf{p}$ similar to the one we use in the definition of μ and $\widetilde{\mu}$ was used before for residual type error estimators for hp-BEM, see [CFS96].

(ii) Instead of dividing by $\mathbf{p}+1$ in the definition of μ and $\widetilde{\mu}$, we could also divide by $\max\{\mathbf{p}, 1\}$. We can not divide by $\mathbf{p}$ because the polynomial degrees of the ansatz space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ may be zero.

(iii) All four error estimators are computable quantities. However, for the construction of adaptive algorithms, we need local refinement indicators. Whereas the energy norm can not be split into local contributions of each boundary element, the L^2 norm can be split, i.e. there holds

$$\mu^2 = \sum_{T \in \mathcal{T}} \mu(T)^2, \quad \mu(T)^2 = \frac{h_T}{(p_T+1)^2} \|\widehat{\phi}_h^p - \phi_h^p\|_{L^2(T)},$$

$$\widetilde{\mu}^2 = \sum_{T \in \mathcal{T}} \widetilde{\mu}(T)^2, \quad \widetilde{\mu}(T)^2 = \frac{h_T}{(p_T+1)^2} \|(1 - \Pi_{h,p})\widehat{\phi}_h^p\|_{L^2(T)}.$$

Here, h_T is the mesh width of an element $T \in \mathcal{T}$ and p_T is the local polynomial degree of the ansatz space on T. For the computation of the estimator μ, we need two discrete solutions, ϕ_h^p and $\widehat{\phi}_h^p$. For the computation of $\widetilde{\mu}$, only one discrete solution, namely $\widehat{\phi}_h^p$, is needed. Therefore, we prefer $\widetilde{\mu}$ for the construction of adaptive algorithms.

The equivalence of η and the Galerkin error can be established using the saturation assumption.

Lemma 4.2.5. *Let the saturation assumption*

$$|||\phi - \widehat{\phi}_h^p||| \leq C_{sat} |||\phi - \phi_h^p|||, \tag{4.9}$$

be satisfied, where C_{sat} is independent of $\mathbf{h}$ and $\mathbf{p}$. Then there holds

$$\eta \leq |||\phi - \phi_h^p||| \leq C\eta$$

with $C = (1 - C_{sat})^{-1/2}$.

Proof. The proof is similar to the proof of 4.1.2 and therefore is omitted. □

As for the error estimators for lowest order BEM, we consider the inequalities

$$\widetilde{\mu} \overset{(A)}{\leq} \mu \overset{(B)}{\leq} C_1\eta, \tag{4.10a}$$

$$\eta \overset{(C)}{\leq} \widetilde{\eta} \overset{(D)}{\leq} C_2\widetilde{\mu}. \tag{4.10b}$$

If we can show that all inequalities hold with constants independent of $\mathbf{h}$ and $\mathbf{p}$, then the equivalence of the four error estimators follows.
The results from Lemma 4.1.4, i.e. inequalities (4.10) (A) and (C), can be shown for the estimators from Definition 4.2.2.

Lemma 4.2.6. *There holds $\widetilde{\mu} \leq \mu$ and $\eta \leq \widetilde{\eta}$.*

Proof. The projection Π_{hp} is the best approximation with respect to the L^2 norm. Therefore, the proof is similar to the proof of Lemma 4.1.4. □

Using the inverse estimate from Lemma 4.2.1, we obtain (4.10) (B). The following result, which was derived in collaboration with M. Karkulik, gives an approximation estimate that enables us to analyze (4.10) (D). For the proof, we use the definition of $H^{1/2}(\Gamma)$ as the interpolation space between $L^2(\Gamma)$ and $H^1(\Gamma)$ (see Remark 1.1.7). This allows us to use the properties of these spaces stated in Remark 1.1.7.

Proposition 4.2.7. *Let $\Pi_{h,p}$ be the L^2 projection from $H^{-1/2}(\Gamma)$ to $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. There holds for all $u \in \widetilde{H}^{-1/2}(\Gamma)$*

$$\|u - \Pi_{h,p}u\|_{\widetilde{H}^{-1/2}(\Gamma)} \leq C \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (u - \Pi_{h,p}u) \right\|_{L^2(\Gamma)} \tag{4.11}$$

with a constant C that is independent of $\mathbf{h}$ and $\mathbf{p}$.
In particular, there holds

$$\widetilde{\eta} \leq C \left\| (\mathbf{p}+1)^{1/2} \right\|_\infty \widetilde{\mu},$$

where the constant C does not depend on $\mathbf{h}$ or $\mathbf{p}$.

Proof. We construct the element-wise projection Π^1 in the following way. If $p_i = 0$, then $(\Pi^1 v)|_{T_i}$ is the L^2 projection into the space of piecewise constants on T_i. If $p_i \geq 1$, then $(\Pi^1 v)|_{T_i}$ is the projection π_p from [Sch98, Theorem 3.17] where $p < p_i$. Using the definition of the dual norm, we obtain

$$\begin{aligned}
\|u - \Pi_{h,p} u\|_{\widetilde{H}^{-1/2}(\Gamma)} &= \sup_{v \in H^{1/2}(\Gamma)} \frac{\langle u - \Pi_{h,p} u, v \rangle}{\|v\|_{H^{1/2}(\Gamma)}} \\
&= \sup_{v \in H^{1/2}(\Gamma)} \frac{\langle u - \Pi_{h,p} u, v - \Pi^1 v \rangle}{\|v\|_{H^{1/2}(\Gamma)}} \\
&= \sup_{v \in H^{1/2}(\Gamma)} \frac{\left\langle \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(u - \Pi_{h,p} u), \frac{(\mathbf{p}+1)^{1/2}}{\mathbf{h}^{1/2}}(v - \Pi^1 v) \right\rangle}{\|v\|_{H^{1/2}(\Gamma)}} \\
&\lesssim \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(u - \Pi_{h,p} u) \right\|_{L^2(\Gamma)} \sup_{v \in H^{1/2}(\Gamma)} \frac{\left\| \frac{(\mathbf{p}+1)^{1/2}}{\mathbf{h}^{1/2}}(v - \Pi^1 v) \right\|_{L^2(\Gamma)}}{\|v\|_{H^{1/2}(\Gamma)}},
\end{aligned}$$

where the last inequality is the Cauchy-Schwarz inequality. The next step is to show that there holds

$$\left\| \frac{(\mathbf{p}+1)^{1/2}}{\mathbf{h}^{1/2}}(v - \Pi^1 v) \right\|_{L^2(\Gamma)} \leq C \|v\|_{H^{1/2}(\Gamma)}$$

with a constant C that is independent of $\mathbf{h}$ and $\mathbf{p}$.
First, we consider all elements $T_i \in \mathcal{T}$ where $p_i \geq 1$. We use the approximation result from [Sch98, Eqn. (3.3.30)], which yields

$$\left\| v - \Pi^1 v \right\|_{L^2(T_i)} \leq \frac{h_i}{p_i + 1} \|v\|_{H^1(T_i)}.$$

For all elements $T_i \in \mathcal{T}$ where $p_i = 0$, there holds the standard approximation estimate

$$\left\| v - \Pi^1 v \right\|_{L^2(T_i)} \leq h_i \|v\|_{H^1(T_i)}.$$

Thus, the operator $E_\alpha : H^\alpha(T) \to L^2(T), v \mapsto v - \Pi^1 v$ has operator norm $\|E_1\| \leq h_i/(p_i+1)$ for $\alpha = 1$ and $\|E_0\| \leq 1$ for $\alpha = 0$. By interpolation (see Remark 1.1.7), we get $\|E_\alpha\| \leq h_i^\alpha/(p_i+1)^\alpha$ for $\alpha \in [0,1]$. In particular, we obtain

$$\left\| v - \Pi^1 v \right\|_{L^2(T_i)} \leq \frac{h_i^{1/2}}{(p_i+1)^{1/2}} \|v\|_{H^{1/2}(T_i)}.$$

Now we can derive an estimate for the norm on Γ. There holds

$$\left\| \frac{(\mathbf{p}+1)^{1/2}}{\mathbf{h}^{1/2}} (v - \Pi^1 v) \right\|^2_{L^2(\Gamma)} = \sum_{T_i \in \mathcal{T}} \frac{p_i + 1}{h_i} \left\| v - \Pi^1 v \right\|^2_{L^2(T_i)} \lesssim \sum_{T_i \in \mathcal{T}} \|v\|^2_{H^{1/2}(T_i)}.$$

Finally, there holds for all $v \in H^{1/2}(\Gamma)$

$$\sum_{T_i \in \mathcal{T}} \|v\|^2_{H^{1/2}(T_i)} \leq C \|v\|^2_{H^{1/2}(\Gamma)}$$

with a constant C; see [CP06, Equation (2.6)]. This concludes the proof. □

Proposition 4.2.7 shows that the the inverse estimate (4.7) does not match the approximation estimate (4.11), and we can not show the equivalence of all error estimators from (4.1). In particular, the constant C_2 in (4.10) depends on $\mathbf{p}$.
Therefore, we introduce weighted error estimators, i.e. we use weighted projections and weighted L^2 norms.

Remark 4.2.8. The introduction of a weight function is motivated from one dimensional p- and hp-FEM. Let $\omega_0(x) = (1+x)(1-x)$ for $x \in [-1,1]$. Furthermore, let $\mathcal{P}^p$ be the space of all polynomials on $[-1,1]$ of degree p. In [Sch98, Section 3.6], the inverse inequality

$$\left\| v' \sqrt{\omega_0} \right\|^2_{L^2(-1,1)} \leq p(p+1) \|v\|^2_{L^2(-1,1)}$$

is proven for any polynomial $v \in \mathcal{P}^p$. Furthermore, it is proven ([Sch98, Section 3.3.1]) that there is an interpolation operator $\Pi : H^1(-1,1) \to \mathcal{P}^p$ such that there holds

$$\left\| \frac{v - \Pi v}{\sqrt{\omega_0}} \right\|^2_{L^2(-1,1)} \leq \frac{1}{p(p+1)} \|(v - \Pi v)'\|^2_{L^2(-1,1)},$$

for all $v \in H^1(-1,1)$. Both estimates are estimates between L^2 and H^1. The inverse inequality (4.2) (B) and the approximation result (D) in (4.2) are inequalities between $H^{-1/2}$ and L^2. This leads to the assumption that the appropriate choice of the weight function for our case is

$$\omega(x) = (1 - x^2)^{1/2}. \tag{4.12}$$

Definition 4.2.9. (i) Let $\omega : [-1,1] \to \mathbb{R}$ be a weight function on $(-1,1)$ and $\omega^{(T)} := \omega \circ \gamma_T^{-1}$, where $\gamma_T : [-1,1] \to T$ is the affine mapping defined in (3.1). In the following we also write $\omega^{(i)} := \omega^{(T_i)}$ for $T_i \in \mathcal{T}$.

(ii) We define the weighted L^2 scalar product

$$(f,g)_{L^2(T),\omega^{(T)}} = \int_T f(x)\, g(x)\, \omega^{(T)}(x)\, ds_x,$$

$$(f,g)_{L^2(\Gamma),\omega} = \sum_{T \in \mathcal{T}} (f,g)_{L^2(T),\omega^{(T)}},$$

and the weighted L^2 norm

$$\|f\|^2_{L^2(\Gamma),\omega} := \sum_{T\in\mathcal{T}} \left\| f \cdot \sqrt{\omega^{(T)}} \right\|^2_{L^2(T)},$$

where $\mathcal{T}$ is a mesh on Γ.

(iii) Let $\Pi^{\omega}_{h,p} : H^{-1/2}(\Gamma) \to S_0(\mathcal{T},\mathbf{h},\mathbf{p})$ be the $\omega^{(i)}$-weighted element-wise L^2 projection onto $S_0(\mathcal{T},\mathbf{h},\mathbf{p})$.

Definition 4.2.10. For a given mesh $\mathcal{T}$ with mesh width function $\mathbf{h}$ and polynomial degree function $\mathbf{p}$, let $S_{h,p} = S_0(\mathcal{T},\mathbf{h},\mathbf{p}) \subset H^{-1/2}(\Gamma)$ and $\widehat{S}_{h,p} = S_0(\widehat{\mathcal{T}},\widehat{\mathbf{h}},\widehat{\mathbf{p}}) \subset H^{-1/2}(\Gamma)$ be another discrete space with $S_{h,p} \subset \widehat{S}_{h,p}$. Let $\phi^p_h \in S_{h,p}$ and $\widehat{\phi}^p_h \in \widehat{S}_{h,p}$ be the solution of Symm's integral equation in the respective discrete spaces. We define the error estimators

$$\eta^{\omega} = |\!|\!| \widehat{\phi}^p_h - \phi^p_h |\!|\!|, \tag{4.13a}$$

$$\widetilde{\eta}^{\omega} = |\!|\!| (1-\Pi^{\omega}_{h,p})\widehat{\phi}^p_h |\!|\!|, \tag{4.13b}$$

$$\mu^{\omega} = \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (\widehat{\phi}^p_h - \phi^p_h) \right\|_{L^2(\Gamma),\omega}, \tag{4.13c}$$

$$\widetilde{\mu}^{\omega} = \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (1-\Pi^{\omega}_{h,p})\widehat{\phi}^p_h \right\|_{L^2(\Gamma),\omega}. \tag{4.13d}$$

Since there holds $\eta^{\omega} = \eta$, the equivalence of η^{ω} and the Galerkin error follows under the saturation assumption (4.9); see Lemma 4.2.5. The weighted L^2 projection $\Pi^{\omega}_{h,p}$ is the best approximation with respect to the weighted L^2 norm, and thus the results from Lemma 4.2.6 can be generalized directly. That is,

$$\eta^{\omega} \leq \widetilde{\eta}^{\omega} \quad \text{and} \quad \mu^{\omega} \leq \widetilde{\mu}^{\omega}.$$

However, there is only little theory for inverse and approximation estimates involving weighted L^2 norms and weighted projections. In order to justify the choice of the weighted error estimators and develop an understanding of the size of the constants in the inequalities, we test the inequalities (B) and (D) from (4.10) numerically using eigenvalue problems. We start with the inverse inequality (B), i.e. we examine the inequalities

$$\left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1} \phi^p_h \right\|_{L^2(\Gamma)} \leq C_{inv}(h,p) |\!|\!| \phi^p_h |\!|\!| \tag{4.14}$$

and

$$\left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} \phi^p_h \right\|_{L^2(\Gamma),\omega} \leq C^{\omega}_{inv}(h,p) |\!|\!| \phi^p_h |\!|\!| \tag{4.15}$$

for the weight function $\omega(x) = \sqrt{1-x^2}$ and for all $\phi_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ with $\phi_h^p \neq 0$.
If $\{\varphi_1, \ldots, \varphi_{\mathcal{N}}\}$ is a basis of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, then ϕ_h^p can be written as $\phi_h^p = \sum_{i=1}^{\mathcal{N}} \phi_i \varphi_i$. Using $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{\mathcal{N}})^T \neq \mathbf{0}$, we obtain

$$|\!|\!| \phi_h^p |\!|\!|^2 = \boldsymbol{\phi}^T \mathbf{V} \boldsymbol{\phi},$$

where $\mathbf{V}$ is the single layer Galerkin matrix. The left-hand side of (4.14) can be written as

$$\left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1} \phi_h^p \right\|_{L^2(\Gamma)}^2 = \boldsymbol{\phi}^T \mathbf{M} \boldsymbol{\phi},$$

where the matrix $\mathbf{M} = (M_{i,j})$ is given by

$$M_{k,\ell} = \left(\frac{\mathbf{h}^{1/2}}{\mathbf{p}+1} \varphi_k, \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1} \varphi_\ell \right)_{L^2(\Gamma)}.$$

Since the basis functions φ_i are local, $\mathbf{M}$ is a block diagonal matrix with one block for each element T_i of the boundary mesh. The entries of the diagonal blocks $\mathbf{M}^{(i,i)}$ are given by

$$\begin{aligned} M_{k,\ell}^{(i,i)} &= \frac{h_i}{(p_i+1)^2} \int_{T_i} P_k^{(i)}(x) P_\ell^{(i)}(x) \, ds_x \\ &= \frac{h_i^2}{2(p_i+1)^2} \int_{-1}^{1} P_k(t) P_\ell(t) \, dt = \delta_{k,\ell} \frac{h_i^2}{(p_i+1)} \frac{1}{2k+1}. \end{aligned} \tag{4.16}$$

For the last equation, we use the L^2 orthogonality of Legendre polynomials, and $\delta_{k,\ell}$ is the Kronecker Delta. Furthermore, the right-hand side of (4.15) can be written as

$$\left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} \phi_h^p \right\|_{L^2(\Gamma),\omega}^2 = \boldsymbol{\phi}^T \widetilde{\mathbf{M}} \boldsymbol{\phi},$$

where the matrix $\widetilde{\mathbf{M}}$ is a block diagonal matrix with one block $\widetilde{\mathbf{M}}^{(i,i)}$ for each element $T_i \in \mathcal{T}$. The elements of $\widetilde{\mathbf{M}}^{(i,i)}$ are given by

$$\widetilde{M}_{k,\ell}^{(i,i)} = \frac{h_i}{(p_i+1)} \int_{T_i} P_k^{(i)}(x) P_\ell^{(i)}(x) \, \omega^{(i)}(x) \, ds_x \tag{4.17}$$

$$= \frac{h_i^2}{2(p_i+1)} \int_{-1}^{1} P_k(t) P_\ell(t) \, \omega(t) \, dt. \tag{4.18}$$

Note that the integral in (4.18) can be evaluated exactly using a Gauss quadrature rule, where the nodes are the roots of the Chebyshev polynomials of the second kind U_k. The quadrature nodes and weights are stated in e.g. [AS84, p. 383].
Using this, we can rewrite (4.14) as

$$\boldsymbol{\phi}^T \mathbf{M} \boldsymbol{\phi} \leq C_{inv}^2 \boldsymbol{\phi}^T \mathbf{V} \boldsymbol{\phi}. \tag{4.19}$$

Substituting $\boldsymbol{\psi} = \mathbf{V}^{1/2}\boldsymbol{\phi}$ (with $\mathbf{V}$ nonsingular and symmetric) yields

$$\frac{\boldsymbol{\psi}^T \mathbf{V}^{-T/2} \mathbf{M} \mathbf{V}^{-1/2} \boldsymbol{\psi}}{\boldsymbol{\psi}^T \boldsymbol{\psi}} \leq C_{inv}^2 \tag{4.20}$$

for all $\boldsymbol{\psi} \in \mathbb{R}^{\mathcal{N}}$ with $\boldsymbol{\psi} \neq \mathbf{0}$. Since for a symmetric matrix $\mathbf{A}$, the Rayleigh quotient $R(\mathbf{A}, \mathbf{x}) = \mathbf{x}^T A \mathbf{x} \,/\, \mathbf{x}^T \mathbf{x}$ is always smaller than the largest eigenvalue of $\mathbf{A}$, Equation (4.20) is equivalent to C_{inv}^2 being the largest eigenvalue of the symmetric matrix $\mathbf{V}^{-T/2} \mathbf{M} \mathbf{V}^{1/2}$. Therefore, we consider the eigenvalue problem $\mathbf{V}^{-T/2} \mathbf{M} \mathbf{V}^{1/2} \mathbf{x} = \lambda \mathbf{x}$, which is equivalent to the generalized eigenvalue problem

$$\mathbf{M}\mathbf{x} = \lambda \mathbf{V}\mathbf{x}. \tag{4.21}$$

Computing the maximum eigenvalue of this problem gives C_{inv}^2.
Proceeding in the same way for (4.15) shows that $(C_{inv}^{\omega})^2$ can be computed as the maximum eigenvalue of

$$\widetilde{\mathbf{M}}\mathbf{x} = \lambda \mathbf{V}\mathbf{x}. \tag{4.22}$$

In order to analyze the dependency of the constants C_{inv} and C_{inv}^{ω} on the mesh width $\mathbf{h}$ and the polynomial degree $\mathbf{p}$, we solve the generalized eigenvalue problems (4.21) and (4.22) for different boundaries Γ, as shown in Figure 4.1. We increase the dimension $\mathcal{N}$ of the ansatz

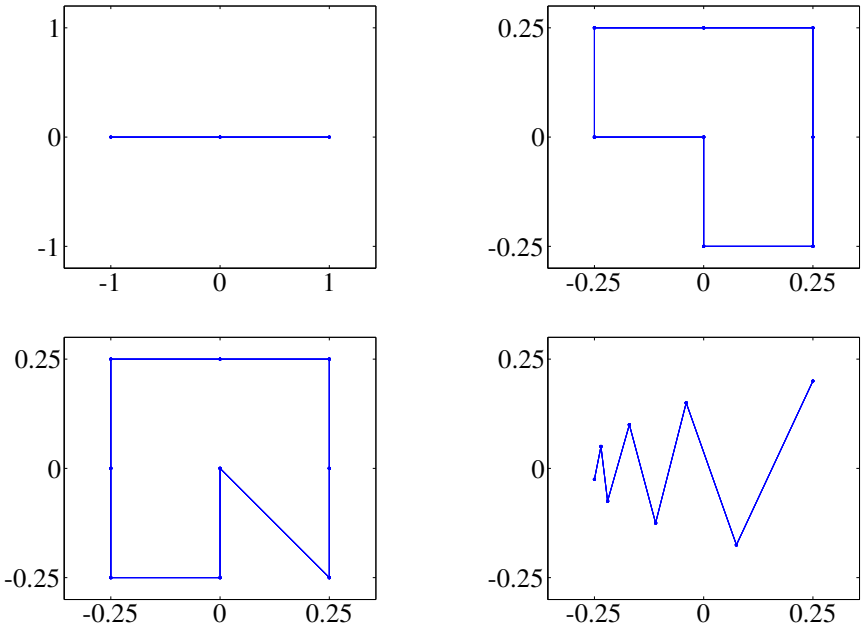


Figure 4.1.: Boundaries used for benchmarks. Top left: Slit. Top right: L-shape. Bottom left: Z-shape. Bottom right: Zigzag

space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ either by a uniform h-method or by a uniform p-method. For the slit and the L-shape we also use geometric hp-methods (see Figure 1.4). Figure 4.2 shows the values of C_{inv} from (4.14) (left) and C^{ω}_{inv} from (4.15) (right) over the dimension $\mathcal{N}$ of the discrete space. The fact that the values of C_{inv} are bounded from above matches the theoretical result from Lemma 4.2.1. The values of C^{ω}_{inv} are also bounded. Thus, at least for the cases tested, the constant C^{ω}_{inv} is independent of $\mathbf{h}$ and $\mathbf{p}$. This strongly indicates that the choice of the weight function gives the correct inverse estimate.

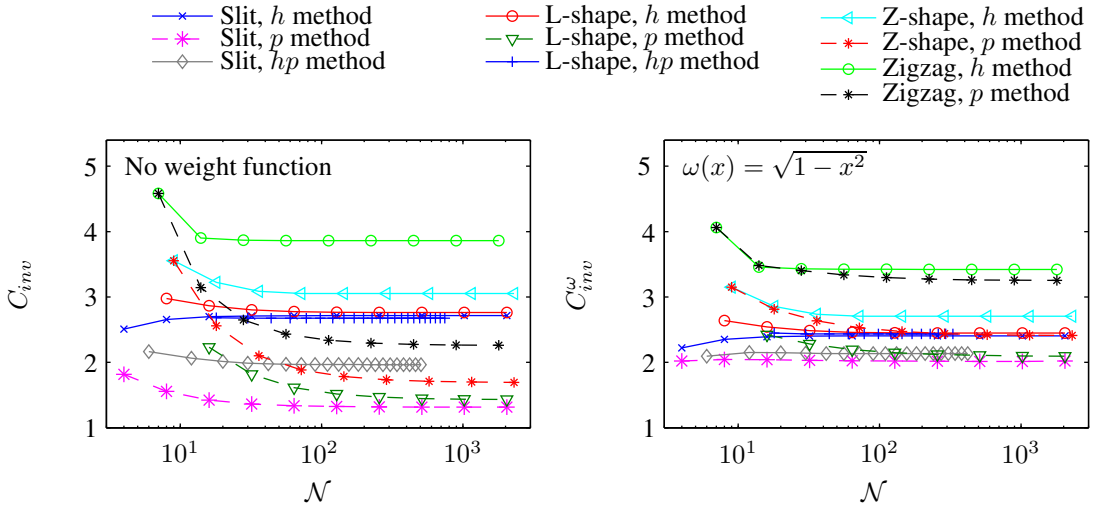


Figure 4.2.: Values of C_{inv} from (4.14) (left) and C^{ω}_{inv} from (4.15) (right) over the dimension of the discrete space for the boundaries from Figure 4.1. We increase $\mathcal{N}$ by uniform h- and p-methods. For the slit and the L-shape, we also test geometric hp-grids.

The next step is to test the approximation estimate. We analyze the inequalities

$$\|\widehat{\phi}^p_h - \phi^p_h\| \leq C_{apx,1} \left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}(1-\Pi_{h,p})\widehat{\phi}^p_h \right\|_{L^2(\Gamma)}, \tag{4.23}$$

$$\|(1-\Pi_{h,p})\widehat{\phi}^p_h\| \leq C_{apx,2} \left\| \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}(1-\Pi_{h,p})\widehat{\phi}^p_h \right\|_{L^2(\Gamma)}, \tag{4.24}$$

$$\|\widehat{\phi}^p_h - \phi^p_h\| \leq C^{\omega}_{apx,1} \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(1-\Pi^{\omega}_{h,p})\widehat{\phi}^p_h \right\|_{L^2(\Gamma),\omega}, \tag{4.25}$$

$$\|(1-\Pi^{\omega}_{h,p})\widehat{\phi}^p_h\| \leq C^{\omega}_{apx,2} \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(1-\Pi^{\omega}_{h,p})\widehat{\phi}^p_h \right\|_{L^2(\Gamma),\omega}. \tag{4.26}$$

Again, we determine the constants using eigenvalue problems similar to (4.19) - (4.21).
We start by deriving eigenvalue problems for (4.23) and (4.24). Assume that we have an orthog-

onal basis B of $\widehat{S}_{h,p}$ such that

$$B = B_L \cup B_Q = \{\varphi_1, \dots, \varphi_{\mathcal{N}}\} \cup \{\varphi_{\mathcal{N}+1}, \dots, \varphi_{\widehat{\mathcal{N}}}\},$$

where B_L is a basis of $S_{h,p}$. The construction of such a basis is discussed later in this section. If we consider the Galerkin matrix $\widehat{\mathbf{V}}$ with respect to the basis B, we can write

$$\widehat{\mathbf{V}} = \begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \\ \mathbf{V}_{QL} & \mathbf{V}_{QQ} \end{pmatrix}, \tag{4.27}$$

where $\mathbf{V}_{LL}$ contains all matrix entries that correspond to combinations of two basis functions in $B_L \times B_L$, $\mathbf{V}_{LQ}$ and $\mathbf{V}_{QL}$ contain matrix entries that correspond to combinations of basis functions in $B_L \times B_Q$ and $B_Q \times B_L$, respectively, and $\mathbf{V}_{QQ}$ contains entries corresponding to combinations of two basis functions in $B_Q \times B_Q$. Note that $\mathbf{V}_{LL}$ is the Galerkin matrix with respect the basis B_L of the coarse space $S_{h,p}$.
Let $\widehat{\boldsymbol{\phi}} = (\boldsymbol{\phi}_L, \boldsymbol{\phi}_Q)^T$ be the coefficient vector of $\widehat{\phi}_h^p \in \widehat{S}_{h,p}$ with respect to the basis B and $\boldsymbol{\phi}$ be the coefficient vector of $\phi_h^p \in S_{h,p}$ with respect to the basis B_L. We construct the coefficient vector of the difference $\widehat{\phi}_h^p - \phi_h^p$ with respect to the basis B as follows. The coefficient vector $\widehat{\boldsymbol{\phi}}$ is the solution of

$$\begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \\ \mathbf{V}_{QL} & \mathbf{V}_{QQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix} = \begin{pmatrix} \mathbf{b}_L \\ \mathbf{b}_Q \end{pmatrix}.$$

We restrict the general right-hand side $(\mathbf{b}_L, \mathbf{b}_Q)^T$ to the coarse space and determine the solution of the system

$$\mathbf{V}_{LL}\boldsymbol{\phi} = \mathbf{b}_L = \begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix}.$$

The restriction of the right-hand side can be done by cutting off the entries for basis functions in B_Q, because the bases B_L of $S_{h,p}$ and B of $\widehat{S}_{h,p}$ are hierarchical and orthogonal. We obtain

$$\boldsymbol{\phi} = \mathbf{V}_{LL}^{-1} \begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{LL} & \mathbf{V}_{LL}^{-1}\mathbf{V}_{LQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix},$$

where $\mathbf{I}_{LL} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is the identity matrix. Now we prolongate the coarse solution to the fine space, which corresponds to setting

$$\widetilde{\boldsymbol{\phi}} = \begin{pmatrix} \boldsymbol{\phi} \\ \mathbf{0} \end{pmatrix}.$$

Finally, the difference of a general solution in the fine space and a general solution in the coarse space is given by

$$\begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix} - \begin{pmatrix} \boldsymbol{\phi} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & -\mathbf{V}_{LL}^{-1}\mathbf{V}_{LQ} \\ \mathbf{0} & \mathbf{I}_{QQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix}.$$

Using this difference, we obtain

$$\begin{aligned}\|\widehat{\phi}_h^p - \phi_h^p\|^2 &= \widehat{\boldsymbol{\phi}}^T \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{V}_{LL}^{-1}\mathbf{V}_{LQ} & \mathbf{I}_{QQ} \end{pmatrix} \begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \\ \mathbf{V}_{QL} & \mathbf{V}_{QQ} \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{V}_{LL}^{-1}\mathbf{V}_{LQ} \\ \mathbf{0} & \mathbf{I}_{QQ} \end{pmatrix} \widehat{\boldsymbol{\phi}} \\ &= \widehat{\boldsymbol{\phi}}^T \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{QQ} - \mathbf{V}_{QL}\mathbf{V}_{LL}^{-1}\mathbf{V}_{LQ} \end{pmatrix} \widehat{\boldsymbol{\phi}}. \end{aligned} \tag{4.28}$$

The norm $\|(1 - \Pi_{h,p})\widehat{\phi}_h^p\|$ on the left-hand side of (4.24) has to be computed in a different way. Recall that the basis B of $\widehat{S}_{h,p}$ is orthogonal with respect to the L^2 scalar product and $\Pi_{h,p}$ is the orthogonal projection with respect to the same scalar product. Therefore, if $\widehat{\boldsymbol{\phi}} = (\boldsymbol{\phi}_L, \boldsymbol{\phi}_Q)^T$ is the coefficient vector of $\widehat{\phi}_h^p$ with respect to the basis B, then the coefficient vector of the projection $\Pi_{h,p}\widehat{\phi}_h^p$ prolongated to the space $\widehat{S}_{h,p}$ is given by $\widetilde{\boldsymbol{\phi}} = (\boldsymbol{\phi}_L, \mathbf{0})^T$. Thus, there holds

$$\begin{aligned}\|(1 - \Pi_{h,p})\widehat{\phi}_h^p\|^2 = (\widehat{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}})^T \widehat{\mathbf{V}} (\widehat{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}) &= \begin{pmatrix} \mathbf{0} & \boldsymbol{\phi}_Q \end{pmatrix} \begin{pmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LQ} \\ \mathbf{V}_{QL} & \mathbf{V}_{QQ} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\phi}_Q \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\phi}_L & \boldsymbol{\phi}_Q \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{QQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix}. \end{aligned} \tag{4.29}$$

For the L^2 norms on the right-hand side of (4.23) and (4.24), we define the scaled mass matrix $\widehat{\mathbf{M}} = (M_{i,j})$ with respect to the basis B of $\widehat{S}_{h,p}$ by

$$M_{i,j} = \left(\frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}\varphi_i, \frac{\mathbf{h}^{1/2}}{\mathbf{p}+1}\varphi_j \right)_{L^2(\Gamma)}.$$

Since the basis B is orthogonal, $\widehat{\mathbf{M}}$ is diagonal, and we can write

$$\widehat{\mathbf{M}} = \begin{pmatrix} \mathbf{M}_{LL} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{QQ} \end{pmatrix}.$$

Using this expression or $\widehat{\mathbf{M}}$, we obtain

$$\begin{aligned}\|(1 - \Pi_{h,p})\widehat{\phi}_h^p\|_{L^2(\Gamma)}^2 = (\widehat{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}})^T \widehat{\mathbf{M}} (\widehat{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}) &= \begin{pmatrix} \mathbf{0} & \boldsymbol{\phi}_Q \end{pmatrix} \begin{pmatrix} \mathbf{M}_{LL} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{QQ} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\phi}_Q \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\phi}_L & \boldsymbol{\phi}_Q \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{QQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix}. \end{aligned} \tag{4.30}$$

Using Equations (4.28) and (4.30), Inequality (4.23) can be written as

$$\widehat{\boldsymbol{\phi}}^T \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{QQ} - \mathbf{V}_{QL}\mathbf{V}_{LL}^{-1}\mathbf{V}_{QQ} \end{pmatrix} \widehat{\boldsymbol{\phi}} \leq (C_{apx,1})^2 \begin{pmatrix} \boldsymbol{\phi}_L & \boldsymbol{\phi}_Q \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{QQ} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_L \\ \boldsymbol{\phi}_Q \end{pmatrix},$$

which is equivalent to

$$\boldsymbol{\phi}_Q^T(\mathbf{V}_{QQ} - \mathbf{V}_{QL}\mathbf{V}_{LL}^{-1}\mathbf{V}_{QQ})\boldsymbol{\phi}_Q \leq (C_{apx,1})^2\boldsymbol{\phi}_Q^T\mathbf{M}_{QQ}\boldsymbol{\phi}_Q.$$

Applying the transformations shown in (4.19) - (4.21) yields $(C_{apx,1})^2$ as the maximum eigenvalue of the generalized eigenvalue problem

$$(\mathbf{V}_{QQ} - \mathbf{V}_{QL}\mathbf{V}_{LL}^{-1}\mathbf{V}_{QQ})\mathbf{x} = \lambda\mathbf{M}_{QQ}\mathbf{x}. \tag{4.31}$$

In the same way, we use (4.29) and (4.30) to compute $(C_{apx,2})^2$ from (4.24) as the maximum eigenvalue of the generalized eigenvalue problem

$$\mathbf{V}_{QQ}\mathbf{x} = \lambda\mathbf{M}_{QQ}\mathbf{x}. \tag{4.32}$$

In order to write (4.25) and (4.26) as generalized eigenvalue problems, we use a basis $\widetilde{B}$ of $\widehat{S}_{h,p}$ that is orthogonal with respect to the weighted L^2 scalar product with weight function ω. Again, we construct the basis such that

$$\widetilde{B} = \widetilde{B}_L \cup \widetilde{B}_Q = \{\varphi_1^\omega, \ldots, \varphi_{\mathcal{N}}^\omega\} \cup \{\varphi_{\mathcal{N}+1}^\omega, \ldots, \varphi_{\widehat{\mathcal{N}}}^\omega\},$$

where $\widetilde{B}_L$ is a basis of $S_{h,p}$. We denote the Galerkin matrix with respect to $\widetilde{B}$ by

$$\widehat{\mathbf{V}}^\omega = \begin{pmatrix} \mathbf{V}_{LL}^\omega & \mathbf{V}_{LQ}^\omega \\ \mathbf{V}_{QL}^\omega & \mathbf{V}_{QQ}^\omega \end{pmatrix} \tag{4.33}$$

and define the scaled mass matrix $\widehat{\mathbf{M}}^\omega = (M_{i,j}^\omega)$ by

$$M_{i,j}^\omega = \left(\frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}\varphi_i^\omega, \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}\varphi_j^\omega \right)_{L^2(\Gamma),\omega}.$$

Note that $\widehat{\mathbf{M}}^\omega$ is again diagonal, since the basis functions φ_i^ω are orthogonal with respect to the weighted scalar product. Thus, we can write

$$\widehat{\mathbf{M}}^\omega = \begin{pmatrix} \mathbf{M}_{LL}^\omega & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{QQ}^\omega \end{pmatrix},$$

where $\mathbf{M}_{LL}^\omega$ is the scaled mass matrix for the basis $\widetilde{B}_L$. Proceeding as in the case without weight functions, we obtain that $(C_{apx,1}^\omega)^2$ is the maximum eigenvalue of the generalized eigenvalue problem

$$(\mathbf{V}_{QQ}^\omega - \mathbf{V}_{QL}^\omega(\mathbf{V}_{LL}^\omega)^{-1}\mathbf{V}_{QQ}\omega)\mathbf{x} = \lambda\mathbf{M}_{QQ}^\omega\mathbf{x}, \tag{4.34}$$

and $(C_{apx,2}^\omega)^2$ is the maximum eigenvalue of

$$\mathbf{V}_{QQ}^\omega\mathbf{x} = \lambda\mathbf{M}_{QQ}^\omega\mathbf{x}. \tag{4.35}$$

It remains to discuss the construction of the bases B and $\widetilde{B}$ of $\widehat{S}_{h,p}$. We consider the cases $\widehat{S}_{h,p} = S_{h,p^*}$, $\widehat{S}_{h,p} = S_{h/2,p}$, and $\widehat{S}_{h,p} = S_{h/2,p^*}$, starting with the first case. In Section 3.2 we defined a basis of $S_{h,p} = S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ by

$$B_L = \bigcup_{T_i \in \mathcal{T}} \{P_0^{(i)}, \dots, P_{p_i}^{(i)}\}.$$

Accordingly, a basis of $S_{h,p^*} = S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}^*)$ is given by

$$B = \bigcup_{T_i \in \mathcal{T}} \{P_0^{(i)}, \dots, P_{p_i^*}^{(i)}\}.$$

Since the Legendre polynomials are orthogonal with respect to the standard L^2 scalar product, both B_L and B are orthogonal. Furthermore, there holds

$$B = B_L \cup \bigcup_{T_i \in \mathcal{T}} \{P_{p_i+1}^{(i)}, \dots, P_{p_i^*}^{(i)}\}.$$

For (4.31) and (4.32) we can use the functions of the package epsBEM (see Section 3.5) to compute the Galerkin matrix $\widehat{\mathbf{V}}$ as defined in Section 3.4 and the mass matrix as given in (4.16). For (4.34) and (4.35), we need a basis that is orthogonal with respect to the weighted L^2 scalar product with weight function $\omega(x) = \sqrt{1-x^2}$. The Chebyshev polynomials of the second kind U_k are orthogonal with respect to ω, i.e. there holds

$$(U_k, U_\ell)_{L^2(-1,1),\omega} = \frac{\pi}{2}\delta_{k,\ell}.$$

Therefore, the bases

$$\widetilde{B}_L = \bigcup_{T_i \in \mathcal{T}} \{U_0^{(i)}, \dots, U_{p_i}^{(i)}\} \quad \text{and} \quad \widetilde{B} = \bigcup_{T_i \in \mathcal{T}} \{U_0^{(i)}, \dots, U_{p_i^*}^{(i)}\}$$

are orthogonal with respect to the ω weighted scalar product and hierarchical. The mass matrix $\widehat{\mathbf{M}}^\omega$ is again a block diagonal matrix with diagonal blocks $\widehat{\mathbf{M}}^{\omega,(i,i)}$ given by

$$\begin{aligned} M_{k,\ell}^{\omega,(i,i)} = \frac{h_i}{(p_i+1)} \int_{T_i} U_k^{(i)}(x) U_\ell^{(i)}(x)\, \omega^{(i)}(x)\, ds_x &= \frac{h_i^2}{2(p_i+1)} \int_{-1}^{1} U_k(t) U_\ell(t)\, \omega(t)\, dt \\ &= \frac{h_i^2}{2(p_i+1)} \frac{\pi}{2} \delta_{k,\ell}. \end{aligned}$$

In Chapter 3 we describe the computation of the Galerkin matrix $\widehat{\mathbf{V}}$ with respect to the Legendre basis B. It is shown that $\widehat{\mathbf{V}}$ is a block matrix with one block $\widehat{\mathbf{V}}^{(i,j)}$ for each combination of two elements $T_i, T_j \in \mathcal{T}$. Since the basis functions of $\widetilde{B}$ are local, the Galerkin matrix $\widehat{\mathbf{V}}^\omega$ is also a block matrix with one block $\widehat{\mathbf{V}}^{\omega,(i,j)}$ for each combination of $T_i, T_j \in \mathcal{T}$. In order to compute the blocks $\widehat{\mathbf{V}}^{\omega,(i,j)}$, we construct the transformation between Legendre and Chebyshev

polynomials in the following way. We are interested in computing coefficients $x_{k,\ell}$ such that $U_k(x) = \sum_{\ell=0}^{n} x_{k,\ell} P_\ell(x)$. We require that

$$\left(\sum_{\ell=0}^{n} x_{k,\ell} P_\ell, P_m \right)_{L^2(-1,1)} = (U_k, P_m)_{L^2(-1,1)}, \quad m = 0, \dots, n, \tag{4.36}$$

which is equivalent to

$$x_{k,\ell} = \frac{(U_k, P_\ell)_{L^2(-1,1)}}{(P_\ell, P_\ell)_{L^2(-1,1)}}. \tag{4.37}$$

If we define the matrix $\mathbf{X}^{(i,j)} = (x_{k,\ell})_{k=0,\dots,p_i,\ell=0,\dots,p_j}$, we obtain

$$\widehat{\mathbf{V}}^{\omega,(i,j)} = \mathbf{X}^{(i,j)} \, \mathbf{V}^{(i,j)} \, (\mathbf{X}^{(i,j)})^T,$$

where $\mathbf{V}^{(i,j)}$ is the matrix block corresponding to the local Legendre bases on T_i and T_j, respectively.

Next we consider the case $\widehat{S}_{h,p} = S_{h/2,p} = S_0(\widehat{\mathcal{T}}, \mathbf{h}/2, \mathbf{p})$ and the inequalities (4.23) and (4.24), i.e. the inequalities with the standard L^2 norm and L^2 projection. Recall that the mesh $\widehat{\mathcal{T}}$ is constructed by halving each element $T \in \mathcal{T}$ into two sons T' and T''. Thus, the Legendre basis of $\widehat{S}_{h,p}$ can be written as

$$B = \bigcup_{T \in \mathcal{T}} \left[\{P_0^{(T')}, \dots, P_{p_T}^{(T')}\} \cup \{P_0^{(T'')}, \dots, P_{p_T}^{(T'')}\} \right].$$

In this case, the bases B and B_L are not hierarchical, i.e. we can not write $B = B_L \cup B_Q$. We construct an orthogonal and hierarchical basis in the following way. We define the functions

$$v_k(x) = \begin{cases} -P_k(2x+1), & x \in [-1, 0], \\ +P_k(2x-1), & x \in (0, 1], \end{cases}$$

and consider the set

$$\{P_0, \dots, P_n\} \cup \{v_0, \dots, v_n\},$$

where $n = \max_{T_i \in \mathcal{T}} p_i$. Then we orthogonalize this set with respect to the L^2 scalar product using the Gram Schmidt algorithm and obtain the set $\{\widetilde{v}_0, \dots, \widetilde{v}_{2n+1}\}$. Note that since the Legendre polynomials are already orthogonal with respect to the L^2 scalar product, there holds $\widetilde{v}_k = P_k$, $k = 0, \dots, n$. The orthogonal basis of $\widehat{S}_{h,p}$ is then given by

$$B = \bigcup_{T_i \in \mathcal{T}} \{\widetilde{v}_0^{(i)}, \dots, \widetilde{v}_{2p_i+1}^{(i)}\},$$

where $\widetilde{v}_k^{(i)} = \widetilde{v}_k \circ \gamma_{T_i}$. Now there holds $B = B_L \cup B_Q$ where $B_L = \bigcup_{T_i \in \mathcal{T}} \{P_0^{(i)}, \dots, P_{p_i}^{(i)}\}$ is the Legendre basis of $S_{h,p}$.

Note that the Gram Schmidt algorithm is numerically unstable. Therefore, we implement the algorithm in *Mathematica*, where we can specify the precision with which the calculations are performed and control the exactness of the output. Since the basis functions of B are local, the scaled mass matrix $\widehat{\mathbf{M}}$ is block diagonal with diagonal blocks $\widehat{\mathbf{M}}^{(i,i)}$. Furthermore, the local basis functions are orthogonal, which means that the blocks $\widehat{\mathbf{M}}^{(i,i)}$ are diagonal with entries

$$M_{k,k}^{(i,i)} = \frac{h_i}{(p_i+1)^2} \int_{T_i} \left(\widetilde{v}_k^{(i)}(x)\right)^2 \, ds_x = \frac{h_i^2}{2(p_i+1)^2} \int_{-1}^{1} \left(\widetilde{v}_k(t)\right)^2 \, dt.$$

The Galerkin matrix $\mathbf{V}$ for the Legendre basis on the fine space $S_{h/2,p}$ can be seen as a block matrix with one $(2p_i+2)\times(2p_j+2)$ block $\mathbf{V}^{(i,j)}$ for each combination of $T_i, T_j \in \mathcal{T}$. Note that here we consider blocks with respect to the coarse mesh $\mathcal{T}$. In order to compute the Galerkin matrix $\widehat{\mathbf{V}}$ with respect to the hierarchical basis B, we need transformation matrices as in (4.36) and (4.37). In more detail, we need to compute coefficients $x_{k,\ell}$ and $y_{k,\ell}$ such that

$$\widetilde{v}_k(x) = \begin{cases} \sum_{\ell=0}^{n} x_{k,\ell} P_\ell(2x+1), & x \in [-1,0) \\ \sum_{\ell=0}^{n} y_{k,\ell} P_\ell(2x-1), & x \in [0,1] \end{cases} \tag{4.38}$$

for $k = 0,\dots,2n+2$, where $n = \max_{T_i\in\mathcal{T}} p_i$. Thus, for $m = 0,\dots,n$, we require the conditions

$$\int_{-1}^{0} \sum_{\ell=0}^{n} x_{k,\ell} P_\ell(2x+1)\; P_m(2x+1)\; dx = \int_{-1}^{0} \widetilde{v}_k(x)\; P_m(2x-1)\; dx, \tag{4.39a}$$

$$\int_{0}^{1} \sum_{\ell=0}^{n} y_{k,\ell} P_\ell(2x-1)\; P_m(2x-1)\; dx = \int_{0}^{1} \widetilde{v}_k(x)\; P_m(2x+1)\; dx, \tag{4.39b}$$

which yields

$$x_{k,\ell} = (2\ell+1) \int_{-1}^{0} \widetilde{v}_k(x)\; P_\ell(2x+1)\; dx, \tag{4.40a}$$

$$y_{k,\ell} = (2\ell+1) \int_{-1}^{0} \widetilde{v}_k(x)\; P_\ell(2x-1)\; dx. \tag{4.40b}$$

Defining the matrices $\mathbf{X}^{(i,j)} = (x_{k,\ell})_{k=0,\dots,p_i,\ell=0,\dots,p_j}$ and $\mathbf{Y}^{(i,j)} = (y_{k,\ell})_{k=0,\dots,p_i,\ell=0,\dots,p_j}$, we obtain that the blocks $\widehat{\mathbf{V}}^{(i,j)}$ of the matrix $\widehat{\mathbf{V}}$ can be computed by

$$\widehat{\mathbf{V}}^{(i,j)} = \begin{pmatrix} \mathbf{X}^{(i,j)} & \mathbf{Y}^{(i,j)} \end{pmatrix} \mathbf{V}^{(i,j)} \begin{pmatrix} \mathbf{X}^{(i,j)} & \mathbf{Y}^{(i,j)} \end{pmatrix}^T,$$

where $\mathbf{V}^{(i,j)}$ is the matrix block corresponding to the local Legendre bases on T_i and T_j, respectively. Here, $T_i, T_j \in \mathcal{T}$ for the coarse mesh $\mathcal{T}$.

For the construction of a basis that is orthogonal with respect to the weighted L^2 scalar product, we define the functions

$$u_k(x) = \begin{cases} -U_k(2x+1), & x \in [-1,0], \\ +U_k(2x-1), & x \in (0,1], \end{cases}$$

and consider the set

$$\{U_0, \dots, U_n\} \cup \{u_0, \dots, u_n\}.$$

The functions in this set are orthogonalized with respect to the weighted L^2 scalar product using the Gram Schmidt algorithm, which results in the set $\{\widetilde{u}_0, \dots, \widetilde{u}_{2n+1}\}$. The orthogonal basis of $\widehat{S}_{h,p}$ is now given by

$$\widetilde{B} = \bigcup_{T_i \in \mathcal{T}} \{\widetilde{u}_0^{(i)}, \dots, \widetilde{u}_{2p_i+1}^{(i)}\},$$

where $\widetilde{u}_k^{(i)} = \widetilde{u}_k \circ \gamma_{T_i}$ and there holds $\widetilde{B} = \widetilde{B}_L \cup \widetilde{B}_Q$ where $\widetilde{B}_L = \bigcup_{T_i \in \mathcal{T}} \{U_0^{(i)}, \dots, U_{p_i}^{(i)}\}$ is the Chebyshev basis of $S_{h,p}$. The scaled mass matrix $\widehat{\mathbf{M}}^{\omega}$ for the basis $\widetilde{B}$ is again block diagonal with one diagonal block $\widehat{\mathbf{M}}^{\omega,(i,i)}$ for each element $T_i \in \mathcal{T}$. The entries of $\widehat{\mathbf{M}}^{\omega,(i,i)}$ are given by

$$M_{k,k}^{\omega,(i,i)} = \frac{h_i}{(p_i+1)^2} \int_{T_i} (\widetilde{u}_k^{(T)}(x))^2 \omega^{(T)}(x)\, ds_x = \frac{h_i^2}{2(p_i+1)^2} \int_{-1}^{1} (\widetilde{u}_k(t))^2 \omega(t)\, dt.$$

In order to compute the blocks $\widehat{\mathbf{V}}^{\omega,(i,j)}$ of the Galerkin matrix $\widehat{\mathbf{V}}^{\omega}$, we need a basis transformation similar to the transformation shown in equations (4.38) - (4.40). We define the coefficients

$$x_{k,\ell}^{\omega} = (2\ell+1) \int_{-1}^{0} \widetilde{u}_k(x)\, P_\ell(2x-1)\, dx,$$
$$y_{k,\ell}^{\omega} = (2\ell+1) \int_{-1}^{0} \widetilde{u}_k(x)\, P_\ell(2x+1)\, dx,$$

and set $\mathbf{X}^{\omega,(i,j)} = (x_{k,\ell}^{\omega})_{k=0,\dots,p_i,\ell=0,\dots,p_j}$ and $\mathbf{Y}^{\omega,(i,j)} = (y_{k,\ell}^{\omega})_{k=0,\dots,p_i,\ell=0,\dots,p_j}$. Then, there holds

$$\widehat{\mathbf{V}}^{\omega,(i,j)} = \begin{pmatrix} \mathbf{X}^{\omega,(i,j)} & \mathbf{Y}^{\omega,(i,j)} \end{pmatrix} \mathbf{V}^{(i,j)} \begin{pmatrix} \mathbf{X}^{\omega,(i,j)} & \mathbf{Y}^{\omega,(i,j)} \end{pmatrix}^T.$$

For the case $\widehat{S}_{h,p} = S_{h/2,p^*} = S_0(\widetilde{\mathcal{T}}, \mathbf{h}/2, \mathbf{p}^*)$, the construction of local, orthogonal and hierarchical bases is similar to the construction of such bases for $S_{h/2,p}$. The only difference is that now we construct the bases for the polynomial degrees $p_T^* > p_T$.

We determine the constants from (4.23) - (4.26) for different meshes and polynomial degree vectors on the boundaries shown in Figure 4.1. In more detail, for all four boundaries, we run uniform h- and p-methods. For the slit and the L-shape, we also run geometric hp-methods. Figure 4.3 shows the values of $C_{apx,1}, C_{apx,2}, C_{apx,1}^{\omega}$, and $C_{apx,2}^{\omega}$ over the dimension of the discrete space for $\widehat{S}_{h,p} = S_{h/2,p}$. The two top pictures show that the constants $C_{apx,1}$ and $C_{apx,2}$ are proportional to $p^{1/2}$ but independent of h, which confirms the theoretical result from Proposition 4.2.7. The two bottom pictures show $C_{apx,1}^{\omega}$ and $C_{apx,2}^{\omega}$ over $\mathcal{N}$. It can be seen that for uniform h- and p-methods, the values are bounded. For geometric hp-methods, only the values of $C_{apx,1}^{\omega}$ are bounded, whereas the values of $C_{apx,2}^{\omega}$ grow as $\mathcal{N}$ increases. This indicates that for the cases tested only $C_{apx,1}^{\omega}$ is independent of the mesh width and the polynomial degrees. The results for the case that $\widehat{S}_{h,p} = S_{h,2p+1}$ are shown in Figure 4.4 and look similar to the results of the first

case. Figure 4.5 shows the results for $\widehat{S}_{h,p} = S_{h/2,2p+1}$. It can be seen that the curves for $C_{apx,1}$, $C_{apx,2}$, and $C^{\omega}_{apx,1}$ behave similar to the respective curves in the first two cases. However, the values of $C^{\omega}_{apx,2}$ grow for uniform h-methods and geometric hp-methods.

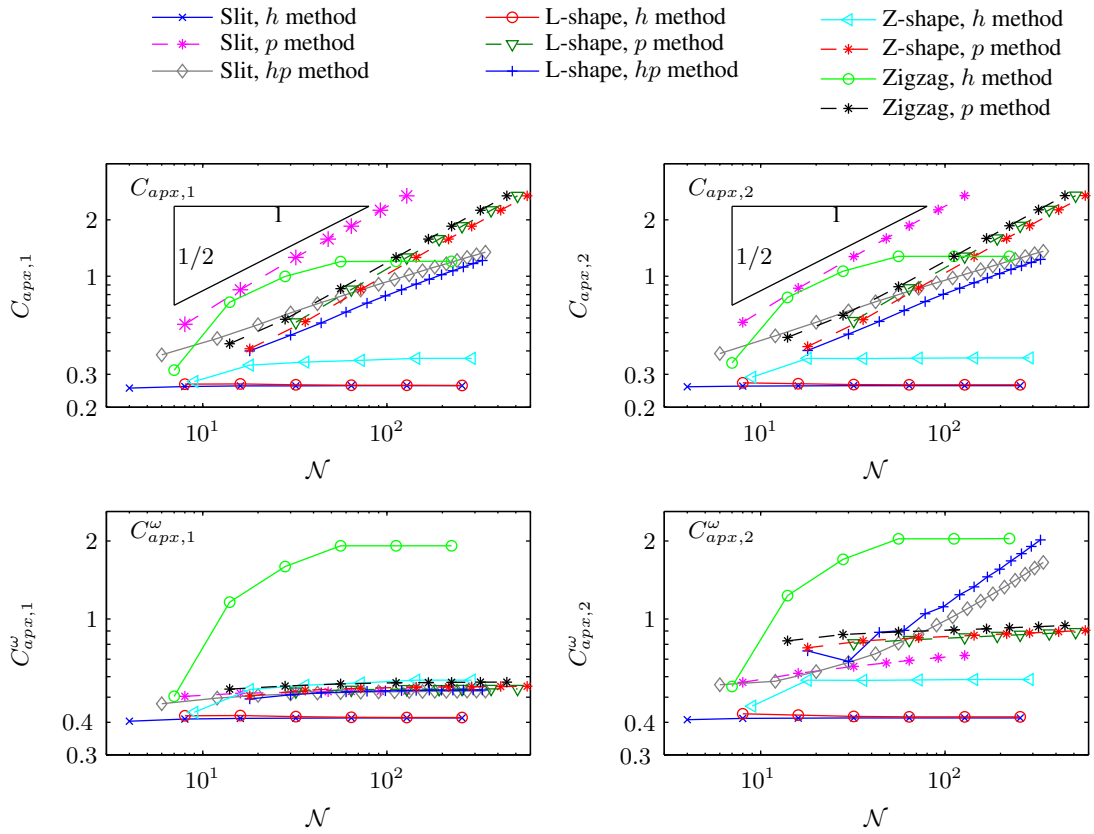


Figure 4.3.: $C_{apx,1}$ (left top), $C_{apx,2}$ (right top), $C^{\omega}_{apx,1}$ (left bottom), $C^{\omega}_{apx,2}$ (right bottom) from the inequalities (4.23) - (4.26) over the dimension $\mathcal{N}$ of the discrete space $S_{h,p}$. The fine space is chosen as $\widehat{S}_{h,p} = S_{h/2,p}$. The constants are computed for the boundaries from Figure 4.1, and $\mathcal{N}$ is increased by uniform h- and p- and geometric hp-methods.

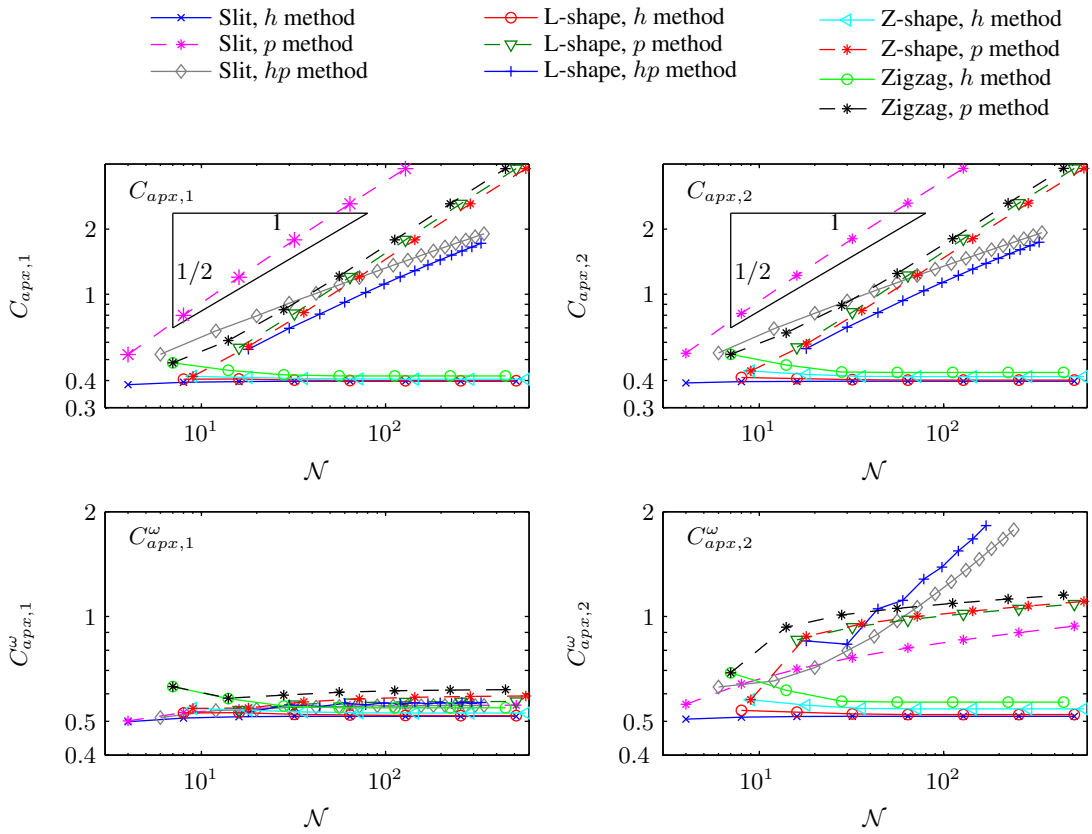


Figure 4.4.: $C_{apx,1}$ (left top), $C_{apx,2}$ (right top), $C^{\omega}_{apx,1}$ (left bottom), $C^{\omega}_{apx,2}$ (right bottom) from the inequalities (4.23) - (4.26) over the dimension $\mathcal{N}$ of the discrete space $S_{h,p}$. The fine space is chosen as $\widehat{S}_{h,p} = S_{h,2p+1}$. The constants are computed for the boundaries from Figure 4.1, and $\mathcal{N}$ is increased by uniform h- and p- and geometric hp-methods.

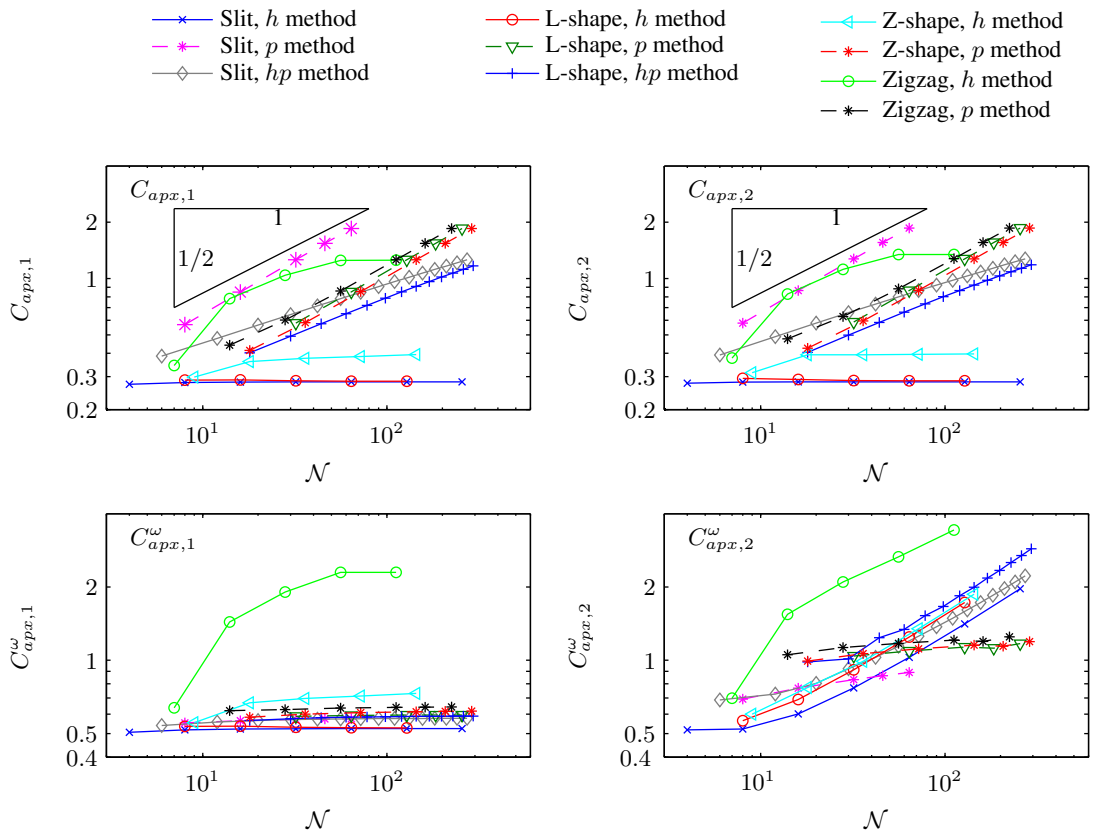


Figure 4.5.: $C_{apx,1}$ (left top), $C_{apx,2}$ (right top), $C^{\omega}_{apx,1}$ (left bottom), $C^{\omega}_{apx,2}$ (right bottom) from the inequalities (4.23) - (4.26) over the dimension $\mathcal{N}$ of the discrete space $S_{h,p}$. The fine space is chosen as $\widehat{S}_{h,p} = S_{h/2,2p+1}$. The constants are computed for the boundaries from Figure 4.1 and $\mathcal{N}$, is increased by uniform h- and p- and geometric hp-methods.

Overall, our numerical experiments, where we determine the constants C_{inv} and C_{inv}^{ω} of the inverse estimates (4.14) and (4.15) as well as the constants $C_{apx,1}$, $C_{apx,2}$, $C_{apx,1}^{\omega}$ and $C_{apx,2}^{\omega}$ of the approximation estimates (4.23) - (4.26) using generalized eigenvalue problems, lead to the following conclusions.

Remark 4.2.11. (i) The results of experiments for the nonweighted estimates, i.e. inequalities (4.14), (4.23), and (4.24), match the theoretical results from Lemma 4.2.1 and Proposition 4.2.7. This shows that the method of computing the constants in the respective inequalities by solving generalized eigenvalue problems is appropriate.

(ii) The constants C_{inv}^{ω} and $C_{apx,1}^{\omega}$ are bounded for growing dimensions $\mathcal{N}$ of the discrete spaces in uniform h- and p-methods and in geometric hp-methods. This indicates that the choice of the weight function and the scaling with respect to h and p in the definition of weighted error estimators is appropriate. In fact, we conclude that the estimators $\widetilde{\mu}_h^{\omega}$, $\widetilde{\mu}_p^{\omega}$ and $\widetilde{\mu}_{hp}^{\omega}$ are equivalent with the estimator η. Additionally, under the saturation assumption, η is equivalent to the Galerkin error, which yields the equivalence of $\widetilde{\mu}_h^{\omega}$, $\widetilde{\mu}_p^{\omega}$ and $\widetilde{\mu}_{hp}^{\omega}$ and the Galerkin error.

(iii) The fact that in some cases the constant $C_{apx,2}^{\omega}$ is not independent of $\mathcal{N}$ indicates that the estimators $\widetilde{\eta}^{\omega}$ and $\widetilde{\mu}^{\omega}$ are not equivalent. Thus, the idea of showing the equivalence of all estimators by proving the six inequalities

$$\begin{aligned} \widetilde{\mu}^{\omega} &\overset{(A)}{\leq} \mu^{\omega} \overset{(B)}{\leq} C_1 \eta^{\omega}, \\ \eta^{\omega} &\overset{(C)}{\leq} \widetilde{\eta}^{\omega} \overset{(D)}{\leq} C_2 \widetilde{\mu}^{\omega}, \end{aligned}$$

as it is done for the error estimators in lowest order BEM might not work. Instead of proving the inequalities (C) and (D) separately, one would have to consider the inequality $\eta^{\omega} \leq C_3 \widetilde{\mu}^{\omega}$.

Remark 4.2.12. In Section 4.1 and this section we only considered error estimators for Symm's integral equation. For the hypersingular integral equation, $h - h/2$ error estimators are introduced in e.g. [EFGP12]. For this case, we have to consider inverse inequalities and approximation estimates between $L^2(\Gamma)$ and $H^{1/2}(\Gamma)$. We expect that for this case, weighted L^2 norms have to be considered where the weight function $\omega(x) = (1 - x^2)^{-1/2}$ has to be chosen (see Remark 4.2.8).

4.3. hp-Adaptive Algorithms

Using the error estimator $\widetilde{\mu}^{\omega}$ introduced in Section 4.2, we construct hp-adaptive algorithms. The construction and analysis of these algorithms was done in collaboration with D. Praetorius. In this section we use the following notation. The mesh in the ℓth step of an adaptive algorithm is denoted by $\mathcal{T}_\ell$. Accordingly, we denote the mesh width function in the ℓth step by $\mathbf{h}_\ell$ and the polynomial degree function by $\mathbf{p}_\ell$. The discrete ansatz space in the ℓth step is given by $S_\ell := S_0(\mathcal{T}_\ell, \mathbf{h}_\ell, \mathbf{p}_\ell)$, and the discrete solution is denoted by $\phi_\ell := \phi_h^p \in S_\ell$. We write $\widehat{S}_\ell := S_0(\mathcal{T}_\ell, \mathbf{h}_\ell, 2\mathbf{p}_\ell + 1)$ and $\widehat{\phi}_\ell = \phi_h^{2p+1}$ or $\widehat{S}_\ell := S_0(\mathcal{T}_\ell, \mathbf{h}_\ell/2, 2\mathbf{p}_\ell + 1)$ and $\widehat{\phi}_\ell = \phi_{h/2}^{2p+1}$, depending on which fine solution we consider for the error estimator. Finally, the element-wise weighted L^2 projection onto S_ℓ is denoted by Π_ℓ^ω.

Both algorithms presented in this section use the Dörfler marking strategy that was introduced in [Dör96] for the construction of an h-adaptive algorithm for lowest order FEM. Given error indicators $\mu(T)$ for all elements $T \in \mathcal{T}$ of a mesh $\mathcal{T}$, one marks all elements $T \in M \subset \mathcal{T}$ where the set M is determined by

$$\theta \sum_{T\in\mathcal{T}} \mu(T)^2 \leq \sum_{T\in M} \mu(T)^2 \tag{4.41}$$

for a given parameter $\theta \in (0, 1)$.

For the creation of an hp-adaptive algorithm, we use the idea that near the singularity of the solution it is better to refine the boundary mesh and on elements where the solution is smooth it is better to increase the polynomial degree of the ansatz space. Furthermore, we assume that the error is larger near singularities of the solution than in areas where the solution is smooth.

Algorithm 4.1 presents a first hp-adaptive algorithm. We use the Dörfler marking (4.41) to single out elements where the error is the largest (see line 4 in Algorithm 4.1). On these elements we use h-refinement. Then we use the Dörfler criterion a second time for all remaining elements (see line 5 in Algorithm 4.1). On all elements that are marked in the second marking, we increase the polynomial degree by one. Finally, we ensure that there holds $\sigma(\mathcal{T}_{\ell+1}) \leq 2\sigma(\mathcal{T}_\ell)$ for the local mesh width ratio of the new mesh and that the polynomial degrees of neighboring elements differ by at most one. For adaptive algorithms for h-BEM, the boundedness of $\sigma(\mathcal{T})$ is necessary in order for the constants C_1 and C_2 in (4.2) not to blow up [FLP08]. The last two steps of Algorithm 4.1 can be seen as a generalization of this property. However, we do not have a proof of the inverse and approximation estimate for our weighted estimators, which means that the last two steps might need adjustment.

Using the concept of estimator reduction from [AFLP12], it can be shown that Algorithm 4.1 converges in the sense that it produces a sequence of estimators $\widetilde{\mu}_\ell$ with $\lim_{\ell\to\infty} \widetilde{\mu}_\ell = 0$. We first need the following result.

Algorithm 4.1 Successive Dörfler marking

Input: Initial mesh $\mathcal{T}_0$ with mesh width function $\mathbf{h}_0$ and polynomial degree function $\mathbf{p}_0$. Adaptivity parameters $\theta_h, \theta_p \in (0,1)$.

1: **for** $\ell = 0, 1, 2, \ldots$ **do**

2: Compute the discrete solution $\widehat{\phi}_\ell = \phi_h^{2p+1} \in S_{h,2p+1} = \widehat{S}_\ell$.

3: Compute the indicators $\widetilde{\mu}_p^\omega(T)$ for all $T \in \mathcal{T}_\ell$ and set $\widetilde{\mu}_\ell^2 = \sum_{T\in\mathcal{T}_\ell} \widetilde{\mu}_p^\omega(T)^2$.

4: Determine the minimal set $M_\ell^h \subset \mathcal{T}_\ell$ such that

$$\theta_h(\widetilde{\mu}_\ell)^2 \leq \sum_{T\in M_\ell^h} \widetilde{\mu}_p^\omega(T)^2.$$

5: Determine the minimal set $M_\ell^p \subset \mathcal{T}_\ell \setminus M_\ell^h$ such that

$$\theta_p\Big(\widetilde{\mu}_\ell^2 - \sum_{T\in M_\ell^h} \widetilde{\mu}_p^\omega(T)^2\Big) \leq \sum_{T\in M_\ell^p} \widetilde{\mu}_p^\omega(T)^2.$$

6: For all elements $T \in M_\ell^p$, increase the polynomial degree p_T by one.

7: Generate new mesh $\mathcal{T}_{\ell+1}$ with $\sigma(\mathcal{T}_{\ell+1}) \leq 2\sigma(\mathcal{T}_\ell)$ by at least halving all elements $T \in M_\ell^h$. The new elements inherit the polynomial degrees from their fathers.

8: Update polynomial degrees such that polynomial degrees on neighboring elements differ by at most one.

9: **end for**

Lemma 4.3.1. ([BV84]) *Let X be a Hilbert space, and X_ℓ be nested subspaces of X, i.e. $X_\ell \subseteq X_{\ell+1}$, for all $\ell \geq 0$. Let $\Pi_\ell : X \to X_\ell$ be the orthogonal projection onto X_ℓ. Then for any $x \in X$ the limit $\lim_{\ell\to\infty} \Pi_\ell x$ exists in X.*

Proposition 4.3.2. *Let the inequalities (4.15) and (4.25) be true. Algorithm 4.1 produces a sequence of estimators $\widetilde{\mu}_\ell$ with $\lim_{\ell\to\infty} \widetilde{\mu}_\ell = 0$. Let ϕ_ℓ be the discrete solution of Symm's integral equation in S_ℓ. Under the assumption that $\widetilde{\mu}_\ell$ is reliable, we obtain that $\lim_{\ell\to\infty} \phi_\ell = 0$, i.e. the convergence of the adaptive algorithm.*

Proof. We apply Lemma 4.3.1 for $X = H^{-1/2}(\Gamma)$. The spaces X_ℓ are the spaces $\widehat{S}_\ell$ in Algorithm 4.1, and the solution $\widehat{\phi}_\ell$ is the Galerkin projection of the exact solution ϕ onto X_ℓ. Recall that the Galerkin projection is the orthogonal projection with respect to the energy norm. Since the sequence of meshes in Algorithm 4.1 is created by local mesh refinement and the polynomial degrees are simply increased, the discrete spaces X_ℓ used in the algorithm are nested. Therefore, the sequence $\widehat{\phi}_\ell$ converges in $H^{-1/2}(\Gamma)$ and is, in particular, a Cauchy sequence, i.e. there holds

$$\lim_{\ell\to\infty} |\!|\!| \widehat{\phi}_{\ell+1} - \widehat{\phi}_\ell |\!|\!| = 0.$$

The next step is to show that

$$\widetilde{\mu}_{\ell+1} \leq (1-\theta/2)^{1/2}\widetilde{\mu}_\ell + C|||\widehat{\phi}_{\ell+1} - \widehat{\phi}_\ell|||, \tag{4.42}$$

where the constant C depends on C^ω_{inv} from (4.15). Since $0 < (1-\theta)^{1/2} < 1$, this implies that $\lim_{\ell\to\infty} \widetilde{\mu}_\ell = 0$ (see [AFLP12, Lemma 2.1]).

We use the notation Π_ℓ for the projection Π^ω_{hp} onto the space $S_{h,p} = S_\ell$ in the ℓth refinement step.

Using the triangle inequality, we obtain

$$\begin{aligned}\widetilde{\mu}_{\ell+1} &= \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell+1} \right\|_{L^2(\Gamma),\omega} \\ &\leq \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell} \right\|_{L^2(\Gamma),\omega} + \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})(\widehat{\phi}_{\ell+1} - \widehat{\phi}_{\ell}) \right\|_{L^2(\Gamma),\omega}.\end{aligned}$$

Since $\widehat{\phi}_{\ell+1} - \widehat{\phi}_\ell$ is in $\widehat{S}_{\ell+1}$ and $\Pi_{\ell+1}$ is the element-wise orthogonal projection onto $S_{\ell+1}$, there holds

$$\begin{aligned}\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})(\widehat{\phi}_{\ell+1} - \widehat{\phi}_{\ell}) \right\|_{L^2(\Gamma),\omega} &\leq \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(\widehat{\phi}_{\ell+1} - \widehat{\phi}_{\ell}) \right\|_{L^2(\Gamma),\omega} \\ &\lesssim \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}^*_{\ell+1}+1)^{1/2}}(\widehat{\phi}_{\ell+1} - \widehat{\phi}_{\ell}) \right\|_{L^2(\Gamma),\omega} \\ &\lesssim C^\omega_{inv}|||\widehat{\phi}_{\ell+1} - \widehat{\phi}_\ell|||,\end{aligned}$$

where for the last step we use the inverse inequality (4.15). Furthermore, there holds

$$\begin{aligned}\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell} \right\|^2_{L^2(\Gamma),\omega} = &\sum_{T_i\in\mathcal{T}_{\ell+1}\setminus\mathcal{T}_\ell} \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell} \right\|^2_{L^2(T_i),\omega} \\ &+ \sum_{T_i\in\mathcal{T}_{\ell+1}\cap\mathcal{T}_\ell} \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell} \right\|^2_{L^2(T_i),\omega}.\end{aligned}$$

The set $\mathcal{T}_\ell\setminus\mathcal{T}_{\ell+1}$ consists of all elements that are refined, i.e. halved, in the ℓth step of the adaptive algorithm. Thus, for all $T \in \mathcal{T}_\ell \setminus \mathcal{T}_{\ell+1}$, there holds $\mathbf{h}_{\ell+1}|_T = \mathbf{h}_\ell|_T/2$ and $\mathbf{p}_\ell|_T \leq \mathbf{p}_{\ell+1}|_T$, which yields for all $T \in \mathcal{T}_\ell \setminus \mathcal{T}_{\ell+1}$

$$\begin{aligned}\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_{\ell} \right\|^2_{L^2(T_i),\omega} &\leq \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell})\widehat{\phi}_{\ell} \right\|^2_{L^2(T_i),\omega} \\ &\leq \frac{1}{2}\left\| \frac{\mathbf{h}_{\ell}^{1/2}}{(\mathbf{p}_{\ell}+1)^{1/2}}(1-\Pi_{\ell})\widehat{\phi}_{\ell} \right\|^2_{L^2(T_i),\omega} = \frac{1}{2}\widetilde{\mu}_\ell(T)^2.\end{aligned}$$

For all elements that are not refined in the ℓth step, i.e. for all $T \in \mathcal{T}_{\ell+1} \cap \mathcal{T}_\ell$, there holds $\mathbf{h}_{\ell+1}|_T = \mathbf{h}_\ell|_T$ and $\mathbf{p}_\ell|_T \leq \mathbf{p}_{\ell+1}|_T$, and thus

$$\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|^2_{L^2(T_i),\omega} \leq \left\| \frac{\mathbf{h}_{\ell}^{1/2}}{(\mathbf{p}_{\ell}+1)^{1/2}}(1-\Pi_{\ell})\widehat{\phi}_\ell \right\|^2_{L^2(T_i),\omega} = \widetilde{\mu}_\ell(T)^2.$$

Altogether, we obtain

$$\begin{aligned}\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|^2_{L^2(\Gamma),\omega} &\leq \frac{1}{2}\sum_{T_i\in\mathcal{T}_{\ell+1}\setminus\mathcal{T}_\ell}\widetilde{\mu}_\ell(T)^2 + \sum_{T_i\in\mathcal{T}_{\ell+1}\cap\mathcal{T}_\ell}\widetilde{\mu}_\ell(T)^2 \\ &= \widetilde{\mu}_\ell^2 - \frac{1}{2}\sum_{T_i\in\mathcal{T}_{\ell+1}\setminus\mathcal{T}_\ell}\widetilde{\mu}_\ell(T)^2.\end{aligned}$$

According to line 8 in Algorithm 4.1, we refine at least all elements in M_ℓ^h, i.e. $M_\ell^h \subseteq \mathcal{T}_{\ell+1}\setminus\mathcal{T}_\ell$. Together with the Dörfler marking in line 5, this yields

$$\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|^2_{L^2(\Gamma),\omega} \leq \widetilde{\mu}_\ell^2 - \frac{1}{2}\sum_{T_i\in M_\ell^h}\widetilde{\mu}_\ell(T)^2 = \left(1-\frac{\theta}{2}\right)\widetilde{\mu}_\ell^2.$$

Now we have shown (4.42), and the proof is complete. □

Another approach for adaptive hp-refinement is presented in Algorithm 4.2. Here, we first determine elements on which the error indicators are the largest using the Dörfler marking (4.41). Then, for all marked elements we determine whether h- or p-refinement is better. To this end we introduce the estimator

$$\widetilde{\mu}_{h1}^\omega = \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(1-\Pi_{h,p^*}^\omega)\phi_{h/2}^{p^*} \right\|_{L^2(\Gamma),\omega}.$$

Comparing the indicators $\widetilde{\mu}_{h1}^\omega(T)$ and $\widetilde{\mu}_{hp}^\omega(T)$ gives an estimate for whether h- or p-refinement is more appropriate for an element T. If for an element T there holds $\widetilde{\mu}_{h1}(T) \ll \widetilde{\mu}_{hp}(T)$, then p-refinement is more appropriate. Therefore, on these elements we increase the polynomial degree, and on all other elements that were marked by the Dörfler criterion we use h-refinement.

Proposition 4.3.3. *Let the inequalities (4.15) and (4.25) be true. Algorithm 4.2 produces a sequence of estimators $\widetilde{\mu}_\ell$ with $\lim_{\ell\to\infty}\widetilde{\mu}_\ell = 0$. Let ϕ_ℓ be the discrete solution of Symm's integral equation in S_ℓ. Under the assumption that $\widetilde{\mu}_\ell$ is reliable, we obtain that $\lim_{\ell\to\infty}\phi_\ell = 0$, i.e. the convergence of the adaptive algorithm.*

Proof. Similar to the proof of Proposition 4.3.2, we can show

$$\widetilde{\mu}_{\ell+1,hp} \leq \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(\Gamma),\omega} + C_{inv}^\omega \,|\!|\!|\widehat{\phi}_{\ell+1}-\widehat{\phi}_\ell|\!|\!|$$

Algorithm 4.2 Nested marking

Input: Initial mesh $\mathcal{T}_0$ with mesh width function $\mathbf{h}_0$ and polynomial degree function $\mathbf{p}_0$. Adaptivity parameters $\theta, \tau \in (0,1)$.

1: **for** $\ell = 0, 1, 2, \ldots$ **do**

2: Refine the mesh $\mathcal{T}_\ell$ uniformly to obtain $\widehat{\mathcal{T}}$.

3: Compute the discrete solution $\widehat{\phi}_\ell = \phi_{h/2}^{2p+1} \in S_{h/2,2p+1} = \widehat{S}_\ell$.

4: Compute the indicators $\widetilde{\mu}_{hp}^{\omega}(T)$ and $\widetilde{\mu}_{h1}^{\omega}(T)$ for all $T \in \mathcal{T}_\ell$ and set

$$\widetilde{\mu}_{\ell,hp}^2 := \sum_{T \in \mathcal{T}_\ell} \widetilde{\mu}_{hp}^{\omega}(T)^2 \quad \text{and} \quad \widetilde{\mu}_{\ell,h1}^2 := \sum_{T \in \mathcal{T}_\ell} \widetilde{\mu}_{h1}^{\omega}(T)^2.$$

5: Determine the minimal set $M_\ell \subset \mathcal{T}_\ell$ such that

$$\theta(\widetilde{\mu}_{\ell,hp})^2 \leq \sum_{T \in M_\ell} \widetilde{\mu}_{hp}^{\omega}(T)^2.$$

6: Determine $M_\ell^p := \left\{T \in M_\ell \,:\, \widetilde{\mu}_{h1}^{\omega}(T)^2 \leq \tau \widetilde{\mu}_{hp}^{\omega}(T)^2\right\}$ and set $M_\ell^h := M_\ell \setminus M_\ell^p$.

7: For all elements $T \in M_\ell^p$, increase the polynomial degree p_T by one.

8: Generate new mesh $\mathcal{T}_{\ell+1}$ with $\sigma(\mathcal{T}_{\ell+1}) \leq 2\sigma(\mathcal{T}_\ell)$ by at least halving all elements $T \in M_\ell^h$. Elements $T' \in \mathcal{T}_{\ell+1}$ inherit the polynomial degrees from their fathers.

9: Update polynomial degrees such that polynomial degrees on neighboring elements differ by at most one.

10: **end for**

with the constant C_{inv}^{ω} from (4.15). It remains to prove

$$\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(\Gamma),\omega} \leq c\, \widetilde{\mu}_{\ell,hp}$$

with a constant $c \in (0,1)$. For all elements T that are not marked in line 5 of Algorithm 4.2, there holds $\mathbf{h}_{\ell+1}|_T \leq \mathbf{h}_\ell|_T$ and $\mathbf{p}_{\ell+1}|_T \leq \mathbf{p}_\ell|_T$, which yields

$$\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \leq \left\| \frac{\mathbf{h}_{\ell}^{1/2}}{(\mathbf{p}_{\ell}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \leq \widetilde{\mu}_{\ell,hp}(T)^2.$$

For elements $T \in M_\ell^h$, there holds $\mathbf{h}_{\ell+1}|_T = \mathbf{h}_\ell|_T/2$ and $\mathbf{p}_{\ell+1}|_T \leq \mathbf{p}_\ell|_T$, and thus

$$\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \leq \frac{1}{2}\left\| \frac{\mathbf{h}_{\ell}^{1/2}}{(\mathbf{p}_{\ell}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \leq \frac{1}{2}\widetilde{\mu}_{\ell,hp}(T)^2.$$

Finally, for $T \in M_\ell^p$, we have $\mathbf{h}_{\ell+1}|_T \leq \mathbf{h}_\ell^T$ and $\mathbf{p}_{\ell+1}|_T = \mathbf{p}_\ell^T + 1$ and obtain

$$\begin{aligned}
\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 &= \left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{h,p}^{\omega})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \\
&\leq \left\| \frac{\mathbf{h}_{\ell}^{1/2}}{(\mathbf{p}_{\ell}+1)^{1/2}}(1-\Pi_{h,p^*}^{\omega})\widehat{\phi}_\ell \right\|_{L^2(T),\omega}^2 \\
&= \widetilde{\mu}_{\ell,h1}^{\omega}(T)^2 \leq \tau \widetilde{\mu}_{\ell,hp}(T)^2,
\end{aligned}$$

where we use line 6 of Algorithm 4.2 in the last step. Combining the last three inequalities, we obtain

$$\begin{aligned}
&\left\| \frac{\mathbf{h}_{\ell+1}^{1/2}}{(\mathbf{p}_{\ell+1}+1)^{1/2}}(1-\Pi_{\ell+1})\widehat{\phi}_\ell \right\|_{L^2(\Gamma),\omega}^2 \\
&\quad\leq \frac{1}{2}\sum_{T\in M_\ell^h} \widetilde{\mu}_{\ell,hp}(T)^2 + \tau \sum_{T\in M_\ell^p} \widetilde{\mu}_{\ell,hp}(T)^2 + \sum_{T\in\mathcal{T}\setminus M_\ell} \widetilde{\mu}_{\ell,hp}(T)^2 \\
&\quad\leq \widetilde{\mu}_\ell - \left(1 - \max\left\{\frac{1}{2},\tau\right\}\right) \sum_{T\in M_\ell} \widetilde{\mu}_{\ell,hp}(T)^2 \\
&\quad\leq \underbrace{\left(1 - \left[1 - \max\left\{\frac{1}{2},\tau\right\}\right]\theta\right)}_{=:c<1} (\widetilde{\mu}_{\ell,hp})^2.
\end{aligned}$$

Overall, we obtain

$$\widetilde{\mu}_{\ell+1,hp} \leq c\widetilde{\mu}_{\ell,hp} + C_{inv}^{\omega}\, |\!|\!| \widehat{\phi}_{\ell+1} - \widehat{\phi}_\ell |\!|\!|$$

with a constant $c \in (0,1)$. Thus, we have a contraction up to a nonnegative sequence that converges to zero, which means that $\lim_{\ell\to\infty} \widetilde{\mu}_{\ell,hp} = 0$. □

Remark 4.3.4. (i) For both algorithms we can prove that a sequence of estimators is created that converges to zero. Together with the proofs and numerical experiments from the last section, which show the reliability of the weighted estimators, this shows that for both algorithms the Galerkin errors converge to zero.

(ii) Algorithm 4.2 uses a more sophisticated refinement strategy than Algorithm 4.1 in the sense that we use two different error estimators to decide whether h- or p-refinement is more appropriate. Therefore, we expect better performance of Algorithm 4.2. The numerical experiments in Section 4.4 confirm this expectation. However, in each step of Algorithm 4.2, we have so solve Symm's integral equation in the space $\widehat{S}_\ell = S_{h/2,2p+1}$, whereas in Algorithm 4.1 we only solve Symm's integral equation in $S_{h,2p+1}$. Note that there holds $\dim S_{h/2,2p+1} = 2\dim S_{h,2p+1} = 4\dim S_{h,p}$. Therefore, the computational effort per step is larger in Algorithm 4.2.

(iii) If we use the estimator $\mu_\ell = \widetilde{\mu}$ without weight function in Algorithms 4.1 and 4.2, we can use the inverse inequality (4.7) and the proof is rigorous. However, since the approximation estimate (4.11) has the wrong scaling in p, the estimator $\widetilde{\mu}$ is reliable only up to a constant depending on p which means that $\lim_{\ell\to\infty}\mu_\ell = 0$ does not imply $\lim_{\ell\to\infty}|||\phi - \phi_\ell||| = 0$. However, numerical experiments show that $\widetilde{\mu}$ is reliable and efficient.

4.4. Implementation and Numerical Experiments

For the numerical experiments we use the functions of the software package epsBEM (see Section 3.5) to assemble the Galerkin matrix and solve the linear systems of equations. Thus, all implementations are done for the piecewise Legendre basis introduced in Section 3.2.
We start with the implementation of the error estimator $\widetilde{\mu}_p^\omega$, which is given by

$$(\widetilde{\mu}_p^\omega)^2 = \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(1-\Pi_{h,p}^\omega)\phi_h^{p^*} \right\|^2_{L^2(\Gamma),\omega} = \sum_{T\in\mathcal{T}} \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}}(1-\Pi_{h,p}^\omega)\phi_h^{p^*} \right\|^2_{L^2(T),\omega}$$

Let $\widetilde{\phi}_h^p = \Pi_{h,p}^\omega \phi_h^{p^*}$ be the element-wise weighted L^2 projection. For each $T\in\mathcal{T}$, the projection is given by

$$\left(\widetilde{\phi}_h^p, P_k^{(T)}\right)_{L^2(T),\omega} = \left(\phi_h^{p^*}, P_k^{(T)}\right)_{L^2(T),\omega}, \quad k = 0,\dots,p_T.$$

Using the local representation of $\widetilde{\phi}_h^p$ and $\phi_h^{p^*}$ on $T\in\mathcal{T}$

$$\widetilde{\phi}_h^p|_T = \sum_{j=0}^{p_T} \widetilde{\phi}_j P_j^{(T)} \quad \text{and} \quad \phi_h^{p^*}|_T = \sum_{j=0}^{p_T^*} \phi_j P_j^{(T)},$$

we obtain

$$\sum_{j=0}^{p_T} \widetilde{\phi}_j \left(P_j^{(T)}, P_k^{(T)}\right)_{L^2(T),\omega} = \sum_{j=0}^{p_T^*} \phi_j \left(P_j^{(T)}, P_k^{(T)}\right)_{L^2(T),\omega}, \quad k = 0, \dots, p_T.$$

If we define the matrices

$$\begin{aligned} \mathbf{X} &= (X_{k,\ell}) \in \mathbb{R}^{p_T \times p_T}, \quad X_{k,\ell} = (P_k^{(T)}, P_\ell^{(T)})_{L^2(T),\omega} \quad \text{and} \\ \widehat{\mathbf{X}} &= (\widehat{X}_{k,\ell}) \in \mathbb{R}^{p_T \times p_T^*}, \quad \widehat{X}_{k,\ell} = (P_k^{(T)}, P_\ell^{(T)})_{L^2(T),\omega}, \end{aligned}$$

then we obtain the local coefficient vector $\widetilde{\boldsymbol{\phi}}|_T$ of the projection $\widetilde{\phi}_h^p|_T$ by solving the linear system

$$\mathbf{X}\widetilde{\boldsymbol{\phi}}|_T = \widehat{\mathbf{X}}\boldsymbol{\phi}|_T,$$

where $\boldsymbol{\phi}|_T$ is the local coefficient vector of $\phi_h^{p^*}|_T$. For the computation of the matrices $\mathbf{X}$ and $\widehat{\mathbf{X}}$, we have to evaluate the integrals

$$(P_j^{(T)}, P_k^{(T)})_{L^2(T),\omega} = \int_T P_j^{(T)}(x) P_k^{(T)}(x) \omega^{(T)}(x)\, ds_x = \frac{h_T}{2} \int_{-1}^{1} P_j(t) P_k(t) \sqrt{1-t^2}\, dt.$$

Here, we use a Gauss quadrature rule where the nodes are chosen as roots of the Chebyshev polynomials of the second kind U_k, which means that the quadrature is exact for the integrals of interest. This is because the polynomials U_k are orthogonal with respect to the weighted L^2 scalar product on $(-1,1)$ with weight function $\omega(x) = \sqrt{1-x^2}$. The weights and nodes for this quadrature are stated in [AS84]. For the computation of the weighted L^2 norm, we have to evaluate the integrals

$$\begin{aligned} \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (1 - \Pi_{h,p}^{\omega}) \phi_h^{p^*} \right\|_{L^2(T),\omega}^2 &= \frac{h_T}{p_T+1} \int_T \left(\phi_h^{p^*}(x) - \widetilde{\phi}_h^p(x) \right)^2 \omega^{(T)}(x)\, ds_x \\ &= \frac{h_T^2}{2(p_T+1)} \int_{-1}^{1} \left(\sum_{j=0}^{p_T} \widetilde{\phi}_j P_j(t) - \sum_{j=0}^{p_T^*} \phi_j P_j(t) \right)^2 \sqrt{1-t^2}\, dt \end{aligned}$$

and again use the Gauss-Chebyshev quadrature.

Listing 4.1 shows the MATLAB implementation of the function that computes the local indicators $\widetilde{\mu}_p^{\omega}(T)$. The inputs are matrices `coordinates` and `elements` describing the boundary mesh $\mathcal{T}$, the polynomial degree vector `p`, the coefficient vector `xs` of the solution $\phi_h^{p^*}$, and the polynomial degree vector `ps` for p^*. First, we compute the weights and nodes for the Gauss-Chebyshev quadrature (line 5). In lines 6-7 we evaluate the Legendre polynomials at the quadrature nodes and compute the transformation matrix $\mathbf{X}$. Then we loop over all elements (lines 11-18). For each element, we first compute the local coefficients of the projection (line 15) and then evaluate the weighted L^2 norm (lines 16-17).

Listing 4.1: Implementation of the error estimator $\widetilde{\mu}_p^{\omega}$

```
function value = estMuTildeP(coordinates, elements, p,xs,ps)
nE = size(elements,1);
h2 = sum((coordinates(elements(:,1),:) - ...
          coordinates(elements(:,2),:)).^2,2);
[wg,xg] = chebquad(max(p+1));
pn = bLegendre(xg,max(ps));
X  = pn'*diag(wg)*pn;

cps = [0; cumsum(ps + 1)];
value = zeros(nE,1);
for i = 1:nE
  idxs = 1:ps(i)+1;
  idx  = 1:p(i)+1;
  inds = (cps(i) + 1):cps(i + 1);
  x = X(idxs,idxs) \(X(idx,idxs)'*xs(inds));
  value(i) = 0.5*h2(i)/(p(i)+1) * ...
             wg'* (pn(:,idxin)*xs(inds) - pn(:,idxs)*x).^2;
end
```

For the computation of the estimator

$$\widetilde{\mu}_{hp}^{\omega} = \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (1 - \Pi_{h,p}^{\omega}) \phi_{h/2}^{p^*} \right\|_{L^2(\Gamma),\omega} = \sum_{T \in \mathcal{T}} \left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (1 - \Pi_{h,p}^{\omega}) \phi_{h/2}^{p^*} \right\|_{L^2(T),\omega}^2,$$

we proceed as follows. First, note that $\phi_{h/2}^{p^*}$ is computed with respect to a uniformly refined mesh $\widehat{\mathcal{T}}$ and that each T in the original mesh $\mathcal{T}$ is split into two sons $T', T'' \in \widetilde{\mathcal{T}}$, i.e. $T = T' \cup T''$. Let $\widetilde{\phi}_h^p = \Pi_{h,p}^{\omega} \phi_{h/2}^{p^*}$ be the element-wise weighted L^2 projection. Furthermore, let

$$\widetilde{\phi}_h^p|_T = \sum_{j=0}^{p_T} \widetilde{\phi}_j P_j^{(T)}, \quad \phi_{h/2}^{p^*}|_{T'} = \sum_{j=0}^{p_T^*} \phi'_j P_j^{(T')}, \quad \text{and} \quad \phi_{h/2}^{p^*}|_{T''} = \sum_{j=0}^{p_T^*} \phi''_j P_j^{(T'')}$$

be the local representations. The local coefficients of the projection can be computed by

$$\left(\widetilde{\phi}_h^p, P_k^{(T)}\right)_{L^2(T),\omega} = \left(\phi_{h/2}^{p^*}, P_k^{(T)}\right)_{L^2(T),\omega} = \left(\phi_{h/2}^{p^*}, P_k^{(T)}\right)_{L^2(T'),\omega} + \left(\phi_{h/2}^{p^*}, P_k^{(T)}\right)_{L^2(T''),\omega}$$

for $k = 0, \ldots, p_T$. Plugging in the representations in the local basis results in a linear system

$$\mathbf{X} \widetilde{\boldsymbol{\phi}}_T = \widehat{\mathbf{X}}' \boldsymbol{\phi}_{T'} + \widehat{\mathbf{X}}'' \boldsymbol{\phi}_{T''},$$

where $\widetilde{\boldsymbol{\phi}}_T$ is the local coefficient vector of $\widetilde{\phi}_h^p$, and $\boldsymbol{\phi}_{T'}$ and $\boldsymbol{\phi}_{T''}$ are the local coefficient vectors of $\phi_{h/2}^{p^*}$ on T' and T'', respectively. The matrices $\mathbf{X} = \left(X_{k,j}\right) \in \mathbb{R}^{p_T \times p_T}$, $\widehat{\mathbf{X}}' = \left(X'_{k,j}\right) \in$

$\mathbb{R}^{p_T \times p_T^*}$, and $\widehat{\mathbf{X}}'' = \left(X''_{k,j}\right) \in \mathbb{R}^{p_T \times p_T^*}$ in this system are given by

$$X_{k,j} = \int_T P_j^{(T)}(x) P_k^{(T)}(x) \omega^{(T)}(x)\, ds_x = \frac{h_T}{2} \int_{-1}^{1} P_j(t) P_k(t) \sqrt{1-t^2}\, dt,$$

$$X'_{k,j} = \int_{T'} P_j^{(T')}(x) P_k^{(T)}(x) \omega^{(T)}(x)\, ds_x = \frac{h_T}{4} \int_{-1}^{1} P_j(t) P_k\left(\frac{t-1}{2}\right) \sqrt{1-\left(\frac{t-1}{2}\right)^2}\, dt,$$

$$X''_{k,j} = \int_{T''} P_j^{(T'')}(x) P_k^{(T)}(x) \omega^{(T)}(x)\, ds_x = \frac{h_T}{4} \int_{-1}^{1} P_j(t) P_k\left(\frac{t+1}{2}\right) \sqrt{1-\left(\frac{t+1}{2}\right)^2}\, dt.$$

For the computation of $X_{k,j}$, we use the Gauss-Chebyshev quadrature as described above. For the computation of $X'_{k,j}$ and $X''_{k,j}$, we construct quadrature nodes and weights such that the integrals are evaluated exactly. This is done by constructing orthogonal polynomials on $(-1,1)$ with respect to the weighted L^2 scalar product with weight functions $\sqrt{1-(\frac{t-1}{2})^2}$ and $\sqrt{1-(\frac{t+1}{2})^2}$, respectively. The nodes, which are the roots of the constructed polynomials, and weights can then be computed using an eigenvalue problem. For a detailed description of the procedure, see [Gau04]. The weighted L^2 norm is computed as follows.

$$\begin{aligned}
&\left\| \frac{\mathbf{h}^{1/2}}{(\mathbf{p}+1)^{1/2}} (1 - \Pi_{h,p}^{\omega}) \phi_{h/2}^{p^*} \right\|^2_{L^2(T),\omega} \\
&\quad = \frac{h_T}{p_T+1} \left[\int_{T'} \left(\phi_{h/2}^{p^*}(x) - \widetilde{\phi}_h^p(x) \right)^2 \omega^{(T)}(x)\, ds_x + \int_{T''} \left(\phi_{h/2}^{p^*}(x) - \widetilde{\phi}_h^p(x) \right)^2 \omega^{(T)}(x)\, ds_x \right] \\
&\quad = \frac{h_T^2}{4(p_T+1)} \Bigg[\int_{-1}^{1} \left(\sum_{j=0}^{p_T} \widetilde{\phi}_j P_j\left(\frac{t-1}{2}\right) - \sum_{j=0}^{p_T^*} \phi_j^L P_j(t) \right)^2 \sqrt{1-\left(\frac{t-1}{2}\right)^2}\, dt \\
&\qquad\qquad + \int_{-1}^{1} \left(\sum_{j=0}^{p_T} \widetilde{\phi}_j P_j\left(\frac{t+1}{2}\right) - \sum_{j=0}^{p_T^*} \phi_j^L P_j(t) \right)^2 \sqrt{1-\left(\frac{t-1}{2}\right)^2}\, dt \Bigg].
\end{aligned}$$

The integrals are again evaluated with the exact quadrature rules.

For the numerical experiments that are presented in the remainder of this section, we need the following result that was derived by S. Funken. For the sake of completeness, we state the proof here.

Lemma 4.4.1. *Symm's integral equation*

$$\mathcal{V}\phi = |x| - \frac{2}{\pi}, \qquad x \in (-1,1)$$

has the solution

$$\phi(x) = \frac{4}{\pi} \frac{\log|2x| + g(x)}{\sqrt{x^2-1}} \quad \textit{with} \quad g(x) = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{T_{2k}}{k(4k^2-1)},$$

where T_k is the kth Chebyshev polynomial of the first kind.

Proof. Let T_k be the kth Chebyshev polynomial of the first kind. We use the expansions

$$|x| = \frac{2}{\pi} + \frac{4}{\pi}\sum_{k=1}^{\infty}(-1)^{k-1}\frac{T_{2k}(x)}{4k^2-1}$$

and

$$\log|x| = -\log(2) + \sum_{k=1}^{\infty}(-1)^{k-1}\frac{T_{2k}(x)}{k}, \quad x \neq 1,$$

from [Riv74]. Then the right-hand side can be written as

$$f(x) = |x| - \frac{2}{\pi} = \frac{4}{\pi}\sum_{k=1}^{\infty}(-1)^{k-1}\frac{T_{2k}(x)}{4k^2-1},$$

and we obtain, using the result from Remark 3.6.1, that the solution $\phi(x)$ is given by

$$\begin{aligned}\phi(x) &= \frac{4}{\pi}\sum_{k=1}^{\infty}(-1)^{k-1}\frac{4k}{4k^2-1}\,\frac{T_{2k}(x)}{\sqrt{1-x^2}}\\ &= \frac{4}{\pi}(1-x^2)^{-1/2}\left[\sum_{k=1}^{\infty}(-1)^{k-1}\frac{T_{2k}(x)}{k} + \sum_{k=1}^{\infty}(-1)^{k-1}\frac{T_{2k}(x)}{k(4k^2-1)}\right]\\ &= \frac{4}{\pi}\frac{\log(2|x|)+g(x)}{\sqrt{1-x^2}}.\end{aligned}$$

□

We test Algorithms 4.1 and 4.2 using the following benchmark problems.

Slit Problem. We consider Symm's integral equation

$$\mathcal{V}\phi = |x| - \frac{2}{\pi} \qquad \text{on } \Gamma = [-1,1].$$

The exact solution of this problem is given in Lemma 4.4.1, and the energy is given by $\|\phi\|_{\mathcal{V}}^2 = \frac{4}{\pi}$ (see Remark 3.6.1). It can be seen that the exact solution has three singularities. Two singularities of order $1/2$ at ± 1 and a logarithmic singularity at the origin.

Z-shape Problem. The second benchmark problem is Symm's integral equation $\mathcal{V}\phi = 1$ on the boundary of the Z-shape (see Figure 4.1). The exact solution of this problem is not known, and the energy is extrapolated to $\|\phi\|_{\mathcal{V}}^2 = 5.101700218060357$.

Figure 4.6 shows the solutions for both the Slit and the Z-shape Problem over the arc length. For the Z-shape, the starting point is chosen to be $(-0.25, -0.25)$ (compare Figure 4.1). It can be seen that the solution for the Z-shape problem has singularities of different orders at each corner of the Z-shape domain. We start by analyzing the choice of parameters θ_h and θ_p in Algorithm 4.1 and θ_h and τ in Algorithm 4.2. To this end, for a given set of parameters we run the respective algorithm, until the energy error is smaller that 10^{-6}. Figure 4.7 shows the number of degrees

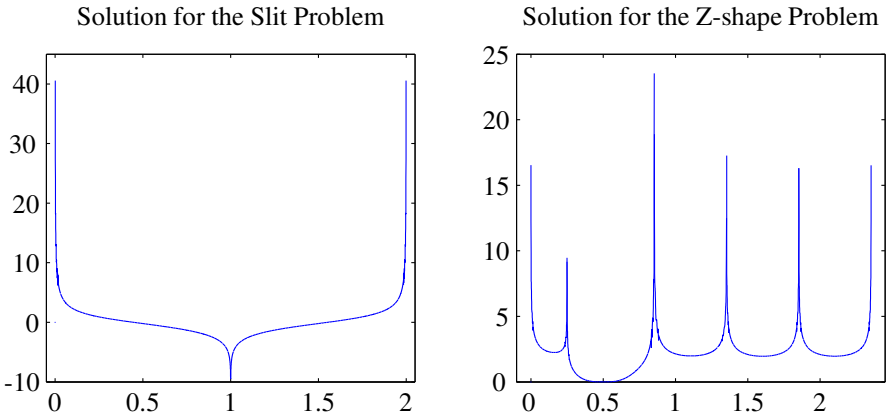


Figure 4.6.: Solutions ϕ for the Slit (left) and the Z-shape Problem (right) over the arc length.

of freedom (color) and the number of iterations (white numbers) needed by either algorithm for both the Slit and the Z-shape Problem. For Algorithm 4.1 (two pictures on the left), we obtain different results for the different problems. For the Slit Problem, we obtain the best results for $\theta_h \in (0.3, 0.5)$ and small θ_p, but for the Z-shape Problem, the fastest convergence is obtained for large θ_h and $\theta_p \geq 0.5$. In contrast to this, for Algorithm 4.2, the parameters that result in the fastest convergence are similar for both problems. It can be seen that the choice of θ primarily influences the number of iterations needed but not the speed of the convergence with respect to degrees of freedom. For τ we have to choose values smaller than 0.15 in order for the algorithm to work properly for the Slit Problem. For the Z-shape, Algorithm 4.2 produces good results if $\tau \leq 0.3$. The dependency on τ can be explained as follows. If for a marked element the quotient $\widetilde{\mu}^{\omega}_{h1}/\widetilde{\mu}^{\omega}_{hp}$ is smaller than τ, we use p-refinement on this element (see Algorithm 4.2 line 6). If we choose τ too large, the quotient $\widetilde{\mu}^{\omega}_{h1}/\widetilde{\mu}^{\omega}_{hp}$ is always smaller that τ for the first iterations of Algorithm 4.2, meaning all marked elements are p-refined. This leads to a long pre-asymptotic range, which results in the slow convergence of the algorithm.

Comparing the two algorithms, we find that for both the Slit and the Z-shape Problems, Algorithm 4.2 performs considerably better. For the Slit Problem, Algorithm 4.1 needs almost 800 degrees of freedom to reduce the energy error below 10^{-6} for the best parameter configuration. For the same problem, Algorithm 4.2 reduces the error below the tolerance using only about 300 degrees of freedom for the optimal choice of the parameters θ and τ. The results for the Z-shape benchmark show the same superiority of Algorithm 4.2. The reason for the significantly better performance of Algorithm 4.2 is that we use a more sophisticated method to distinguish which elements are h-refined and for which elements the polynomial degree is increased. In Algorithm 4.1, we only use two Dörfler criteria without using any information on wether h- or p-refinement is more appropriate for an element. In Algorithm 4.2, we first single out the element on which the error is the largest using a Dörfler criterion. Then for each marked element, we decide which

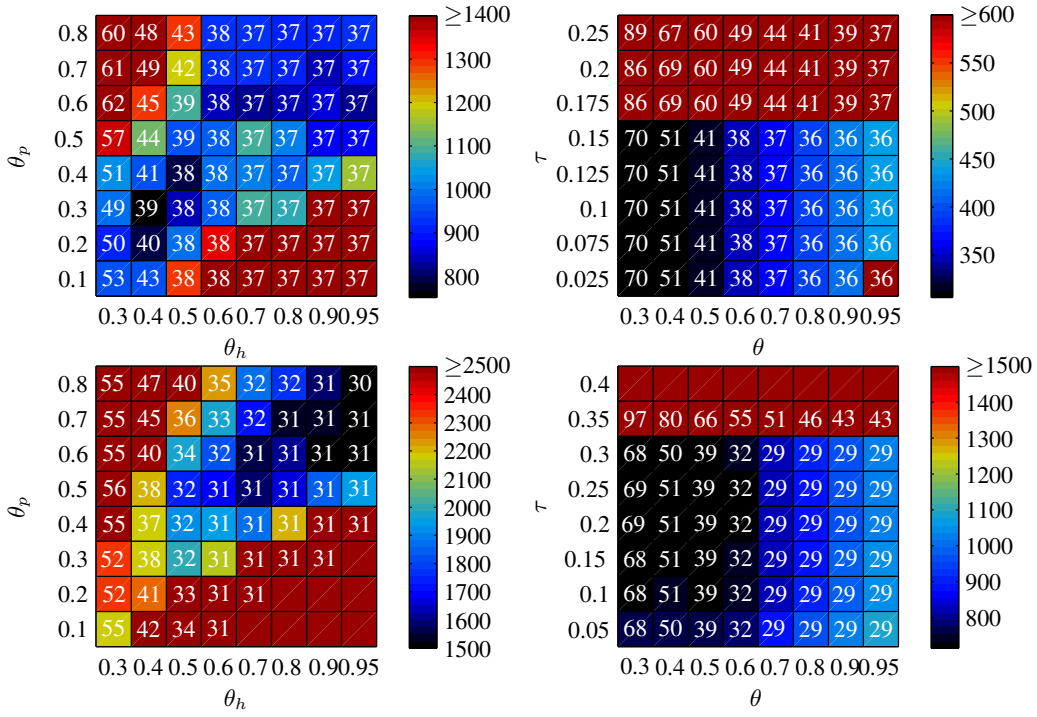


Figure 4.7.: Number of degrees of freedom at which Algorithm 4.1 (left) and 4.2 (right) stops, when we choose the stopping criterion $\|\phi - \phi_h^p\| < 10^{-6}$. The top pictures are for the Slit Problem and the two bottom pictures for the Z-shape Problem. The white numbers indicate how many iterations the algorithms needed. No white number indicates that the algorithm could not reduce the error below the tolerance with less than 5000 degrees of freedom.

refinement type should be used by comparing the two estimators $\widetilde{\mu}_{h,1}^{\omega}$ and $\widetilde{\mu}_{hp}^{\omega}$. This process results in a far better polynomial degree distribution as the following benchmarks show.

We run both algorithms for the Slit Problem, where we choose an initial grid with four elements and polynomial degrees equal to zero. For Algorithm 4.1, we use $\theta_h = 0.4$ and $\theta_p = 0.3$. Algorithm 4.2 is run with $\theta = 0.6$ and $\tau = 0.1$. Figure 4.8 shows the errors and estimators over $\sqrt{\mathcal{N}}$ in a semi logarithmic plot. The curves for both algorithms are almost straight lines. which means that the errors decay like $\exp(-b\sqrt{\mathcal{N}})$ for both algorithms. However, the convergence is faster for Algorithm 4.2.

In Figure 4.9 we display the polynomial degree distribution and the number of refinements that were used to obtain each boundary element for the discrete space after 35 steps for both algorithms. The elements are sorted such that the first element is near the left corner $(-1, 0)$ and the last element is near the right corner $(1, 0)$. Algorithm 4.2 (right picture) produces a boundary mesh that is refined toward all three singularities. It can also be seen that the mesh is refined

more near the stronger singularities at ± 1 as compared to the logarithmic singularity in the origin. The polynomial degree distribution is such that the polynomial degrees are small near the singularities and change linearly from element to element. This matches the a priori convergence results for hp-geometric meshes presented in Section 1.5. The mesh and polynomial degree distribution produced by Algorithm 4.1 are not as structured (see right picture in Figure 4.9). Near the singularity in the origin, the mesh is refined and the polynomial degrees are high, which results in slower convergence.

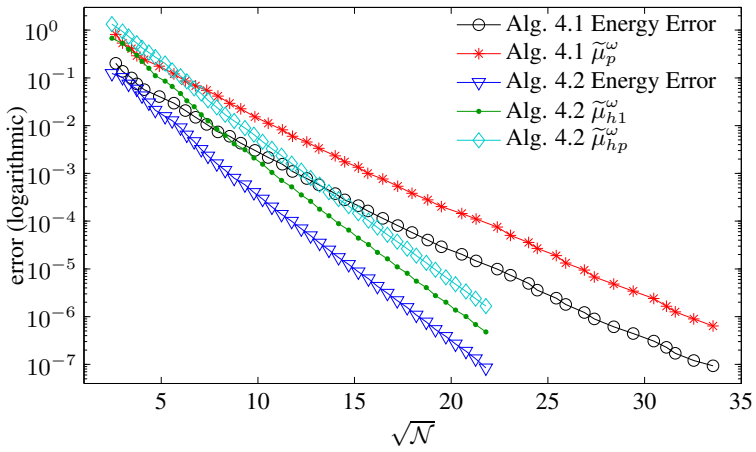


Figure 4.8.: Errors and estimators over $\sqrt{\mathcal{N}}$ for the Slit Problem using Algorithm 4.1 and 4.2. The y-axis is scaled logarithmically.

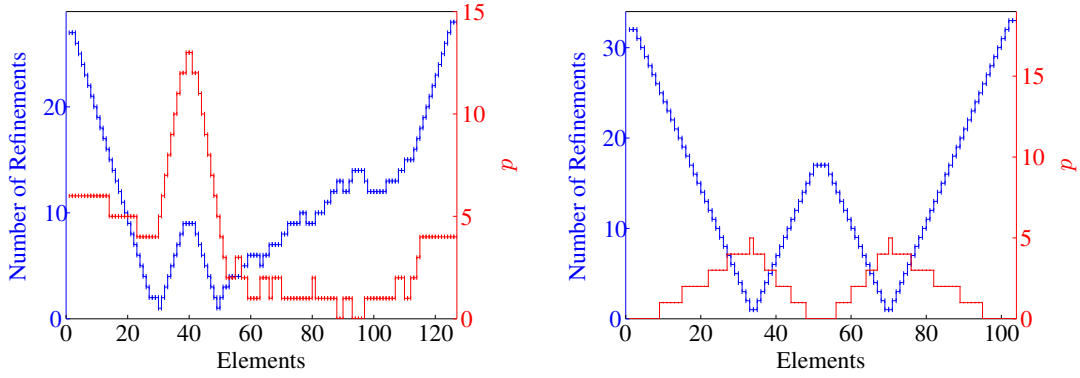


Figure 4.9.: Structure of the boundary mesh and polynomial degree distribution for Algorithm 4.1 (left) and 4.2 (right) after 35 steps for the Slit Problem. The blue curves show the number of refinements for each element (left y-axis in each picture); the red curves show the polynomial degree distribution (right y-axis in each picture).

Both algorithms are also run for the Z-shape Problem with an initial mesh consisting of 16 elements and polynomial degrees 0. For Algorithm 4.1, we choose the parameters $\theta_h = 0.9$ and $\theta_p = 0.6$, and for Algorithm 4.2, we choose $\theta = 0.6$ and $\tau = 0.1$. Error curves for both algorithms are shown in Figure 4.10. Again, both algorithms yield exponential decay of the errors and estimators, and Algorithm 4.2 converges considerably faster with respect to the degrees of freedom. Figure 4.11 shows the structure of the boundary mesh and the polynomial degree distribution after 30 steps for both algorithms. Again, for each element of the boundary mesh, we show how many refinements were made to obtain this element and the polynomial degree. The elements are sorted such that the first element starts at the point $(-0.25, -0.25)$ (see Figure 4.1) and all other elements follow in counterclockwise direction. Algorithm 4.2 (right picture) produces a mesh that is refined towards the singularities, being finer at singularities of higher order (compare Figure 4.6), and the polynomial degrees are smaller near singularities, changing linearly from element to element. The mesh produced by Algorithm 4.1 (left picture) is also refined toward all singularities, but the polynomial degree distribution is not optimal. We see this in the large polynomial degrees near two of the singularities.

The following remark summarizes the results of this section.

Remark 4.4.2. (i) Both hp-adaptive algorithms analyzed in this section result in exponential convergence in the sense that the Galerkin errors for the discrete solutions are proportional to $\exp(-b\sqrt{\mathcal{N}})$.

(ii) Algorithm 4.2 performs better than Algorithm 4.1 in the sense that it reduces the error faster with respect to the degrees of freedom $\mathcal{N}$, i.e. the constant b from (i) is larger for Algorithm 4.2. However, for the error estimators $\widetilde{\mu}^{\omega}_{hp}$ and $\widetilde{\mu}^{\omega}_{h1}$ that are needed in Algorithm 4.2, we have to solve Symm's integral equation in $S_{h/2,2p+1}$. For Algorithm 4.1, we only need the estimator $\widetilde{\mu}^{\omega}_{p}$ which means that we have to compute the solution in $S_{h,p}$. Thus, in each step of Algorithm 4.2, we have to solve a problem that is twice the size of the problem that we have to solve in each step of Algorithm 4.1. Therefore, each step in Algorithm 4.2 is more expensive with respect to the computational effort.

(iii) Algorithm 4.2 seems more robust with respect to the parameters involved. It is shown that for $\tau \leq 0.15$ the algorithm yields fast convergence for both the Slit and the Z-shape Problems. For Algorithm, 4.1 the parameters θ_h and θ_p have to be adjusted to the problem.

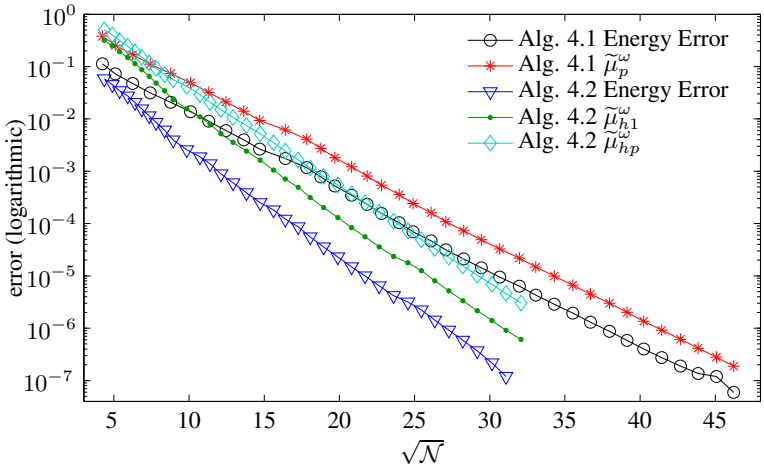


Figure 4.10.: Errors and estimators over $\sqrt{\mathcal{N}}$ for the Z-shape Problem and Algorithm 4.1 and 4.2. The y axis is scaled logarithmically.

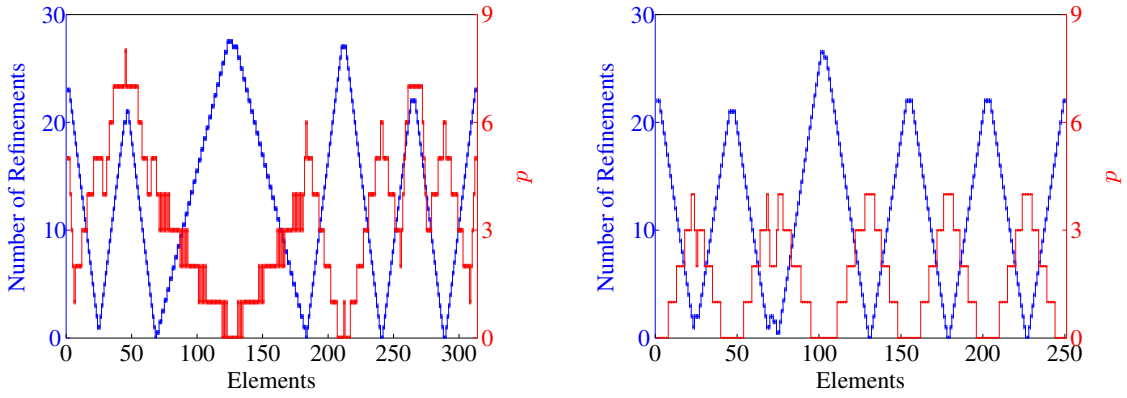


Figure 4.11.: Structure of the boundary mesh and polynomial degree distribution for Algorithm 4.1 (left) and 4.2 (right) after 30 steps for the Z-shape Problem. The blue curves show the number of refinements for each element (left y-axis in each picture), the red curves show the polynomial degree distribution (right y-axis in each picture).

5. Data Compression Techniques for the Single Layer Galerkin Matrix in p-BEM

Unlike in finite element methods where the arising Galerkin matrices are sparse, in boundary element methods one generally has to solve linear systems of equations with densely populated Galerkin matrices. This is a major limitation on the degrees of freedom that can be used for the computations. For example, a full matrix of size 20000×20000 with double precision entries takes roughly 3.2 GB of memory. Furthermore, even if the effort for the computation of one matrix entry is of the order $\mathcal{O}(1)$, the assembly of a full $n \times n$ matrix and the matrix vector multiplication have the complexity $\mathcal{O}(n^2)$. Thus, on desktop computers, computations with larger numbers of degrees of freedom cannot be performed efficiently. Therefore, data compression is crucial for the implementation of fast boundary element methods. The goal of data compression techniques is to reduce the complexity for storage, assembly, matrix vector multiplication, and other matrix operations.
For lowest order boundary element methods, i.e. h-BEM, there exist different data compression techniques. In the first section of this chapter we give an overview of existing compression schemes and explain why these schemes do not work well for p-BEM.

We introduce a data compression scheme for the single layer Galerkin matrix that arises in the p-version of the boundary element method. The compression scheme uses different compression techniques for different sub-blocks of the Galerkin matrix. An overview of the compression scheme is given in the following example.

Example 5.0.3. We consider Symm's integral equation on the slit $\Gamma = [-1, 1] \times \{0\}$, which is discretized uniformly into five boundary elements. We choose the polynomial degree vector $\mathbf{p} = (69, 67, 68, 69, 73)$. In this case, the Galerkin matrix $\mathbf{V}$ is a 5×5 block matrix. Figure 5.1 shows the compressed version of the Galerkin matrix $\mathbf{V}$. The diagonal blocks are split hierarchically into admissible (red) and inadmissible (blue) sub-blocks. The entries of inadmissible sub-blocks are computed directly, and for admissible sub-blocks we create low rank approximations. For the off-diagonal blocks (black), we construct low rank approximations using adaptive cross approximation. The rank of each off-diagonal block in this example is 8. For the far field blocks (green), we neglect small matrix entries and obtain sparse approximations. The green

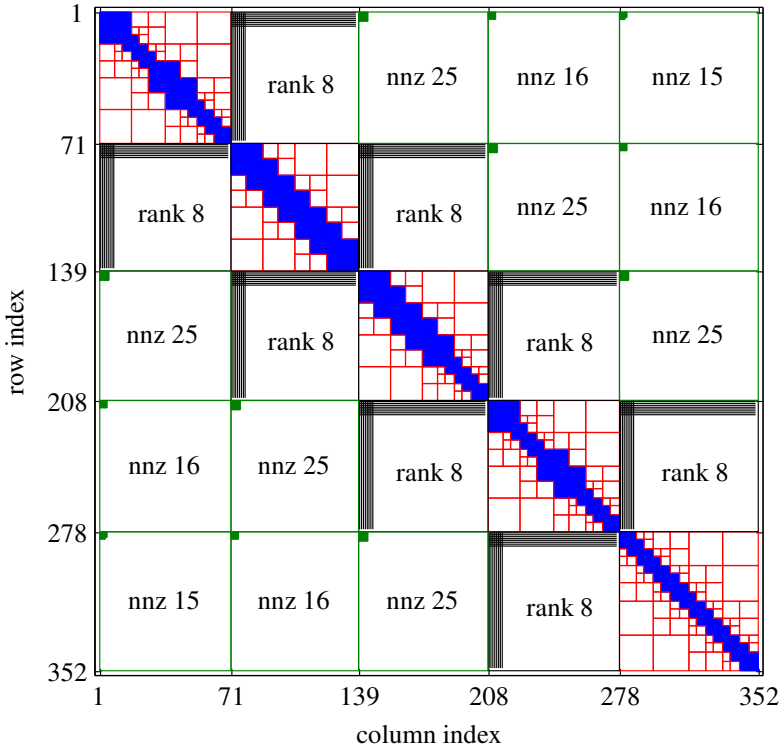


Figure 5.1.: Example of the matrix splitting for the Galerkin matrix $\mathbf{V}$ described in Example 5.0.3.

squares in the upper left corner of each block represent the entries that have to be computed and stored. The number of nonzero entries per block is denoted by *nnz* .

After having described the existing compression schemes in the first section of this chapter, we use the second section to derive an estimate for the effect of perturbations in the Galerkin matrix on the overall error of the Galerkin discretization. The third section discusses the compression of diagonal blocks, i.e. the blocks that correspond to identical elements. In the fourth section we describe the compression of far field blocks, and the fifth section is devoted to the off-diagonal blocks. In the sixth section we present an extension of the data compression format for the single layer Galerkin matrix $\mathbf{V}$ that arises in hp-BEM with quasi-uniform meshes and polynomial degrees. Finally, we discuss some aspects of implementation and present several numerical examples in the last section.

Throughout this chapter we assume that the boundary mesh $\mathcal{T}$ with $\mathcal{N}_e$ elements and the polynomial degrees are quasi-uniform, and we use the notation

$$\underline{h} := \min\{h_i,\ i = 1,\dots,\mathcal{N}_e\}, \quad \overline{h} := \max\{h_i,\ i = 1,\dots,\mathcal{N}_e\},$$
$$\underline{p} := \min\{p_i,\ i = 1,\dots,\mathcal{N}_e\}, \quad \overline{p} := \max\{p_i,\ i = 1,\dots,\mathcal{N}_e\}.$$

Furthermore, we assume that $\underline{p} > 0$ in order to keep the notation simple. Instead, we could write $\max\{\mathbf{p}, 1\}$ instead of $\mathbf{p}$ in the following sections and all estimates would still hold.

5.1. Existing Compression Schemes

In 1987, Greengard and Rokhlin introduced the Fast Multipole Method (FMM) [GR87], which allows for the multiplication of an $N \times N$ matrix with a vector in $\mathcal{O}(N)$ operations for matrices that arise in particle simulations. Around the same time, Hackbusch and Nowak introduced the Panel Clustering Algorithm, which is similar to FMM but designed specifically for matrices arising in lowest order BEM [HN88]. The idea of this algorithm is to group elements of the boundary mesh (called panels) into clusters and use degenerate approximations of the kernel function for pairs of clusters that are far apart from one another. This results in an algorithm that allows for the multiplication of an $N \times N$ boundary element Galerkin matrix with a vector with complexity $\mathcal{O}(N)$. A generalization and abstraction of the ideas of panel clustering are the frameworks of $\mathcal{H}$- and $\mathcal{H}^2$-Matrices introduced in [Hac99] and [HKS00], respectively. $\mathcal{H}$- and $\mathcal{H}^2$-Matrices also provide fast algorithms for addition, multiplication and inversion of matrices. A first approach for the compression of Galerkin matrices arising in p-BEM is presented in [Heu96]. The idea is that many matrix entries are small and thus can be neglected. By doing this, one obtains a sparse approximation of the Galerkin matrix.
Since we use ideas and terminology of the $\mathcal{H}$-Matrix framework for our data compression scheme, we give a short introduction to $\mathcal{H}$-Matrices and mention the main idea and complexity estimates of $\mathcal{H}^2$-Matrices afterwards. This introduction is based on [GBH06], [BGH03], and [Bör10]. Finally, we state and describe two versions of the adaptive cross approximation algorithm.

The idea of $\mathcal{H}$-Matrices is to divide a matrix hierarchically into sub-blocks and then create low rank approximations of certain sub-blocks. Since we only need this special case, we introduce the idea of $\mathcal{H}$-Matrices for the single layer Galerkin matrix of the Laplace problem in 2D. As the ansatz space, we choose the space of piecewise polynomials $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ for a quasi-uniform triangulation $\mathcal{T} = \{T_1, \dots, T_{\mathcal{N}}\}$ of the boundary Γ. The basis of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ is denoted by $\{\varphi_i,\ i = 1, \dots, \mathcal{N}\}$, and we identify each basis function φ_i with a point $x_i \in \Gamma$ that can be chosen as the midpoint of the support of φ_i. The points x_i will be used for the hierarchical splitting. Note that the Galerkin entries can be computed by

$$V_{i,j} = -\frac{1}{2\pi} \int_\Gamma \int_\Gamma \varphi_i(x) \log|x - y| \varphi_j(y)\, ds_y\, ds_x.$$

We start with the hierarchical matrix splitting and introduce some notation.

Definition 5.1.1. Let $\mathcal{I}$ be the index set for the row and column indices of the matrix $\mathbf{V}$, respectively.

(i) Subsets of $\mathcal{I}$ are called clusters and denoted by σ and τ for row and column indices, respectively.

(ii) For a cluster σ, the bounding box $B_\sigma = [a,b] \times [c,d] \subset \mathbb{R}^2$ is the smallest rectangle containing $\bigcup_{i\in\sigma} \operatorname{supp} \varphi_i$, where the functions φ_i are the basis functions of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$.

(iii) The cardinality of a cluster $\sigma \subset \mathcal{I}$ is denoted by $|\sigma|$.

The key structure for the hierarchical splitting is a cluster tree. For simplicity, we define a cluster tree as a binary tree, meaning each node that is not a leaf has two sons. Furthermore, we identify each node with an index set. Formally this is done by defining a cluster tree as a labeled tree (see e.g. [Bör10, Definition 3.4]). Our simplified definition of a cluster tree is inspired by the definition of a cluster tree in [BGH03].

Definition 5.1.2 (Cluster Tree)**.** (i) A binary tree $\mathbb{T}$ is called cluster tree for the index set $\mathcal{I}$ and the leaf size C_{leaf} if

- the root of $\mathbb{T}$ is $\mathcal{I}$,
- each node σ of $\mathbb{T}$ is a subset of $\mathcal{I}$,
- if $\sigma \in \mathbb{T}$ is a leaf, then $|\sigma| \leq C_{leaf}$,
- if $\sigma \in \mathbb{T}$ is not a leaf, then σ has two unique sons $\sigma', \sigma'' \in \mathbb{T}$ with $\sigma' \cap \sigma'' = \emptyset$ and $\sigma' \cup \sigma'' = \sigma$.

(ii) For a node $\sigma \in \mathbb{T}$, the set of all sons of σ is denoted by $\operatorname{sons}(\sigma)$.

(iii) The level of a node $\sigma \in \mathbb{T}$ is defined recursively in the following way:

- The level of the root of $\mathbb{T}$ is $\operatorname{level}(\mathcal{I}) = 0$.
- The level of a son $\sigma' \in \operatorname{sons}(\sigma)$ is defined by $\operatorname{level}(\sigma') = \operatorname{level}(\sigma) + 1$.

(iv) The depth of a cluster tree is defined by

$$\operatorname{depth}(\mathbb{T}) = \max_{\sigma\in\mathbb{T}} \operatorname{level}(\sigma).$$

Algorithm 5.1 can be used to construct a cluster tree. Recall that $x_i \in \Gamma$ is a point associated with the basis function φ_i.

Algorithm 5.1 Construction of a cluster tree

1: **procedure** ClusterTree(cluster σ, cluster tree $\mathbb{T}$, leaf size C_{leaf})
2: **if** $|\sigma| \leq C_{leaf}$ **then**
3: **return** ;
4: **else**
5: Create bounding box B_σ and split B_σ along its longest edge into $B_{\sigma'}$ and $B_{\sigma''}$.
6: Define sons σ' and σ'' by

$$\sigma' = \{i \in \sigma \,:\, x_i \in B_{\sigma'}\}, \quad \sigma'' = \sigma \setminus \sigma'.$$

7: Add σ' to $\mathbb{T}$ and call ClusterTree(σ', $\mathbb{T}$, C_{leaf}).
8: Add σ'' to $\mathbb{T}$ and call ClusterTree(σ'', $\mathbb{T}$, C_{leaf}).
9: **end if**

We use a cluster tree to construct a partitioning of the product index set $\mathbb{P} \subset \mathcal{I} \times \mathcal{I}$. Note that each pair of clusters $(\sigma, \tau) \in \mathbb{P}$ corresponds to a sub-block of $\mathbf{V}$, and thus the partitioning $\mathbb{P}$ is also a partitioning of the Galerkin matrix $\mathbf{V}$ into sub-blocks. The goal is to split the matrix $\mathbf{V}$ into sub-blocks such that on most sub-blocks, called admissible blocks, we can create a low rank approximation by approximating the kernel function $\log|x-y|$, while on a small number of sub-blocks, called inadmissible blocks, we compute the matrix entries directly. To achieve such a subdivision, we introduce an admissibility condition.

Definition 5.1.3. A pair of clusters $(\sigma, \tau) \in \mathcal{I} \times \mathcal{I}$ is called admissible if

$$\min\{\operatorname{diam}(B_\sigma), \operatorname{diam}(B_\tau)\} \leq \eta \operatorname{dist}\{B_\sigma, B_\tau\}, \tag{5.1}$$

where the admissibility constant $\eta > 0$ is fixed.

Algorithm 5.2 can be used to create a block partitioning $\mathbb{P} \subseteq \mathcal{I} \times \mathcal{I}$. The block partitioning consists of admissible and inadmissible pairs of clusters, and we write $\mathbb{P} = \mathbb{P}_{adm} \cup \mathbb{P}_{inadm}$. The function `Splitting` is called with $\sigma = \tau = \mathcal{I}$.
We are now ready to define an $\mathcal{H}$-Matrix.

Definition 5.1.4. Let $k \in \mathbb{N}$ be the rank of admissible blocks. A matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ is called an $\mathcal{H}$-Matrix with respect to a cluster tree $\mathbb{T}$ if, for any admissible block $(\sigma, \tau) \in \mathbb{T} \times \mathbb{T}$, there are matrices $\mathbf{A}^{\sigma\tau} \in \mathbb{R}^{|\sigma| \times k}$ and $\mathbf{B}^{\sigma\tau} \in \mathbb{R}^{|\tau| \times k}$ such that

$$\mathbf{H}|_{\sigma \times \tau} = \mathbf{A}^{\sigma\tau}(\mathbf{B}^{\sigma\tau})^T.$$

Here, $\mathbf{H}_{\sigma \times \tau}$ denotes the sub-block $(H_{i,j})_{i \in \sigma, j \in \tau}$ of the matrix $\mathbf{H}$.

Algorithm 5.2 Block partitioning

```
1: procedure Splitting(cluster σ, cluster τ, admissibility constant η)
2: if (σ, τ) is admissible then
3:    Add (σ, τ) to the set of admissible clusters P_adm;
4:    return ;
5: else if sons(σ) = ∅ and sons(τ) = ∅ then
6:    Add (σ, τ) to the set of inadmissible clusters P_inadm;
7:    return ;
8: else
9:    for all (σ', τ') ∈ sons(σ) × sons(τ) do
10:      Splitting(σ', τ', η);
11:   end for
12: end if
```

The following theorem gives complexity estimates for the storage requirement of an $\mathcal{H}$-Matrix and for the complexity of the multiplication of an $\mathcal{H}$-Matrix with a vector. For a detailed discussion of the estimates with proofs, see e.g. [Hac99] or [GH03].

Proposition 5.1.5. *(i) The storage requirement for an $\mathcal{N} \times \mathcal{N}$ $\mathcal{H}$-Matrix with respect to a cluster tree $\mathbb{T}$ and with rank k on admissible blocks is*

$$N_{St} \leq 2C_{sp}(\operatorname{depth}(\mathbb{T}) + 1) \max\{k, C_{leaf}\}\mathcal{N}.$$

Here, C_{leaf} is the leaf size of the cluster tree and for a partitioning $\mathbb{P} \subset \mathcal{I} \times \mathcal{I}$ the constant C_{sp} is called the sparsity constant and is defined by

$$C_{sp} = \max\left\{\max_{\sigma\in\mathbb{T}}\{\tau \in \mathbb{T} \,:\, (\sigma,\tau) \in \mathbb{P}\}, \max_{\tau\in\mathbb{T}}\{\sigma \in \mathbb{T} \,:\, (\sigma,\tau) \in \mathbb{P}\}\right\}.$$

(ii) For the complexity N_{Mv} of the multiplication of an $\mathcal{N} \times \mathcal{N}$ $\mathcal{H}$-Matrix with a vector in $\mathbb{R}^{\mathcal{N}}$, there holds

$$N_{St} \leq N_{Mv} \leq 2N_{St}.$$

Remark 5.1.6. (i) It is observed a posteriori that the sparsity constant does not blow up when the boundary mesh is refined [GH03].

(ii) For most applications one obtains a balanced cluster tree, which means $\operatorname{depth}(\mathbb{T}) = \mathcal{O}(\log \mathcal{N})$. Then the complexity estimate for the storage requirement of an $\mathcal{H}$-Matrix simplifies to

$$N_{St} \lesssim C_{sp} \max\{k, C_{leaf}\}\mathcal{N} \log(\mathcal{N}).$$

It remains to discuss the construction of the low rank approximation of admissible blocks. There are several methods to obtain low rank approximations. For BEM Galerkin matrices, one can create a degenerate kernel approximation e.g. by interpolation or Taylor expansion of the kernel function; see e.g. [BG04], [GBH06]. Another option is adaptive cross approximation, which is discussed later in this section. Here, we describe the approximation of the kernel function using interpolation. On each admissible sub-block, we approximate matrix entries by interpolating the kernel function $\kappa(x,y) = \log|x-y|$. Since we interpolate the function κ with respect to $x \in \mathbb{R}^2$ on a box $B_\sigma := [a_1,b_1] \times [a_2,b_2]$, we have to use two dimensional interpolation. For Chebyshev nodes $t_{k_d}^{(d)} \in [a_d,b_d]$ $(d=1,2)$, we define the interpolation points $x_k = (t_{k_1}^{(1)}, t_{k_2}^{(2)})$, $k_1,k_2 = 1,\dots,\nu$. Furthermore, we define the two dimensional Lagrange basis polynomials L_k as the product of two one-dimensional Lagrange polynomials $L_{k_1}^{(1)} \cdot L_{k_2}^{(2)}$. Using this, we obtain the interpolation operator

$$I_{x,\sigma}^{\nu}\kappa(x,y) = \sum_{k=1}^{\nu^2} L_k(x)\kappa(x_k,y). \tag{5.2}$$

Plugging in the interpolation polynomial for the kernel function yields

$$\begin{aligned}
&\int_\Gamma \int_\Gamma \varphi_i(x) \log|x-y| \varphi_j(y)\, ds_y\, ds_x \\
&\qquad \approx \int_\Gamma \int_\Gamma \varphi_i(x) \sum_{k=1}^{\nu^2} L_k(x) \log|x_k-y|\, \varphi_j(y)\, ds_y\, ds_x \\
&\qquad = \sum_{k=1}^{\nu^2} \left(\int_\Gamma \varphi_i(x) L_k(x)\, ds_x \right) \left(\int_\Gamma \varphi_j(y) \log|x_k-y|\, ds_y \right) \\
&\qquad = \sum_{k=1}^{\nu^2} \left(\int_\Gamma \varphi_i(x) L_k(x)\, ds_x \right) (\widetilde{\mathcal{V}}\varphi_j)(x_k) \\
&\qquad =: \sum_{k=1}^{\nu^2} a_{i,k} \cdot b_{j,k}.
\end{aligned} \tag{5.3}$$

If we apply this approximation to an admissible sub-block $\mathbf{V}|_{\sigma\times\tau}$, we obtain a low rank approximation of this block that reads

$$\mathbf{V}|_{\sigma\times\tau} = \mathbf{A}\mathbf{B}^T,$$

where $\mathbf{A} = (a_{i,k}) \in \mathbb{R}^{|\sigma|\times\nu^2}$ and $\mathbf{B} = (b_{j,k}) \in \mathbb{R}^{|\tau|\times\nu^2}$. Using the admissibility condition (5.1), it can be shown that for an admissible pair of clusters, the interpolation error decays exponentially with respect to the interpolation order ν. Since we use this error analysis later in this chapter, we state the main results here. First, we need the following lemma.

Lemma 5.1.7. *The kernel function $\kappa(x,y) = \log|x-y|$ is asymptotically smooth, i.e there holds for multi-indices $\alpha,\beta \in \mathbb{N}^2$ that*

$$|\partial_x^\alpha \partial_y^\beta \kappa(x,y)| \le |\alpha+\beta|!\, |x-y|^{-|\alpha|+|\beta|+1}.$$

The next lemma gives an estimate for the interpolation error when the kernel function $\kappa(x,y) = \log|x-y|$ is interpolated with respect to x (or y) on an axis parallel bounding box $B_\sigma \subset \mathbb{R}^2$ (or $B_\tau \subset \mathbb{R}^2$). The ingredients for the proof are the asymptotic smoothness of the kernel function and the Chebyshev interpolation error formula.

Lemma 5.1.8. ([BG04, Theorem 3.2 and Section 4.1]) *Let $B_\sigma, B_\tau \subset \mathbb{R}^2$ be bounding boxes of two clusters $\sigma, \tau \in \mathbb{T}$. Furthermore, let $\kappa(x,y) = \log|x-y|$, $I^\nu_{x,\sigma}\kappa$ be the νth order Chebyshev interpolation polynomial of κ with respect to x on the box B_σ (see Equation 5.2), and $I^\nu_{y,\tau}\kappa$ be the νth order Chebyshev interpolation polynomial of κ with respect to y on the box B_τ. Then, with $D_{\sigma,\tau} = \operatorname{dist}(B_\sigma, B_\tau)$, we obtain the error estimates*

$$\|\kappa - I^\nu_{x,\sigma}\kappa\|_{\infty,B_\sigma} \leq 16\, e\, \frac{(\nu+1)^3}{D_{\sigma,\tau}} \left(1 + \frac{\operatorname{diam}(B_\sigma)}{D_{\sigma,\tau}}\right) \left(1 + 2\frac{D_{\sigma,\tau}}{\operatorname{diam}(B_\sigma)}\right)^{-(\nu+1)}, \tag{5.4a}$$

$$\|\kappa - I^\nu_{y,\tau}\kappa\|_{\infty,B_\tau} \leq 16\, e\, \frac{(\nu+1)^3}{D_{\sigma,\tau}} \left(1 + \frac{\operatorname{diam}(B_\tau)}{D_{\sigma,\tau}}\right) \left(1 + 2\frac{D_{\sigma,\tau}}{\operatorname{diam}(B_\tau)}\right)^{-(\nu+1)}. \tag{5.4b}$$

In [BG04] it is suggested to use

$$\widetilde{\kappa} := \begin{cases} I^\nu_{x,\sigma}\kappa, & \text{if } \operatorname{diam}(B_\sigma) < \operatorname{diam}(B_\tau), \\ I^\nu_{y,\tau}\kappa, & \text{otherwise,} \end{cases} \tag{5.5}$$

for the approximation of the kernel function. This yields

$$\|\kappa - \widetilde{\kappa}\|_{\infty,B_\sigma\times B_\tau} \leq 16\, e\, \frac{(\nu+1)^3}{D_{\sigma,\tau}} \left(1 + \frac{C_{\sigma,\tau}}{D_{\sigma,\tau}}\right) \left(1 + 2\frac{D_{\sigma,\tau}}{C_{\sigma,\tau}}\right)^{-(\nu+1)}, \tag{5.6}$$

where $C_{\sigma,\tau} = \min\{\operatorname{diam}(B_\sigma), \operatorname{diam}(B_\tau)\}$. Plugging in the admissibility condition (5.1) results in the following bound for the interpolation error

$$\|\kappa - \widetilde{\kappa}\|_{\infty,B_\sigma\times B_\tau} \leq 16\, e\, \frac{(\nu+1)^3}{D_{\sigma,\tau}}(1+\eta) \left(1 + \frac{2}{\eta}\right)^{-(\nu+1)}. \tag{5.7}$$

This shows that on admissible pairs of clusters, the interpolation error converges exponentially with respect to the interpolation order ν.

Using Strang's Lemma (Theorem 1.5.2) and inverse estimates, it can be shown that the choice $\nu = \mathcal{O}(\log(\mathcal{N}))$ suffices to keep the consistency error of the $\mathcal{H}$-Matrix approximation of the same order as the discretization error of the Galerkin scheme; see [GHS05] for a proof. Thus, the complexity for the storage of an $\mathcal{N}\times\mathcal{N}$ $\mathcal{H}$-Matrix and the complexity for the matrix vector multiplication with an $\mathcal{N}\times\mathcal{N}$ $\mathcal{H}$-Matrix are of the order $\mathcal{O}(\mathcal{N}\log(\mathcal{N})^3)$.

Remark 5.1.9. ($\mathcal{H}$-Matrices in p-BEM) We emphasize that the $\mathcal{H}$-Matrix approximation does not work well for Galerkin matrices that arise in p-BEM. The splitting of the Galerkin matrix into sub-blocks is a spatial splitting, since we use the distance of bounding boxes for the splitting of clusters in Algorithm 5.1 and also for the admissibility condition (5.1). In p-BEM the number

of basis functions per element, i.e. basis functions that have the same support, grows, while the number of elements remains fixed. Since we can only split clusters whose bounding boxes fulfill the admissibility condition, the leaf size in our cluster tree is a multiple of the polynomial degree of the ansatz space, i.e. this constant C_{leaf} blows up in p-BEM. Additionally, numerical experiments show that the rank of admissible sub-blocks also blows up with growing polynomial degree.

An $\mathcal{H}^2$-Matrix approximation of a BEM Galerkin matrix can be constructed e.g. by interpolating the kernel $\log|x-y|$ with respect to x and y in contrast to only interpolating with respect to one variable in the $\mathcal{H}$-Matrix approach. Let $x_k \in B_\sigma$ and $y_\ell \in B_\tau$, $k,\ell = 1,\dots,\nu$ be interpolation points. Let L_k and L_ℓ be the corresponding Lagrange polynomials. Then there holds

$$\begin{aligned}
&\int_{T_i}\int_{T_j} \varphi_i(x)\log|x-y|\varphi_j(y)\,ds_y\,ds_x \\
&\qquad \approx \int_{T_i}\int_{T_j} \varphi_i(x)\varphi_j(y)\sum_{k=1}^{\nu^2}\sum_{\ell=1}^{\nu^2} L_k(x)L_\ell(y)\log|x_k-y_\ell|\,ds_y\,ds_x \\
&\qquad = \sum_{k,\ell=1}^{\nu^2}\left(\int_{T_i}\varphi_i(x)L_k(x)\,ds_x\right)\log|x_k-y_\ell|\left(\int_{T_j}\varphi_j(y)L_\ell(y)\,ds_y\right) \\
&\qquad =: \sum_{k,\ell=1}^{\nu^2} a_{i,k}\cdot c_{k,\ell}\cdot b_{j,\ell}. \qquad (5.8)
\end{aligned}$$

Thus, an admissible block of a matrix $\mathbf{H}$ can be written as

$$\mathbf{H}_{\sigma\times\tau} = \mathbf{A}^\sigma \mathbf{C}^{\sigma,\tau}\,(\mathbf{B}^\tau)^T,$$

where $\mathbf{A}^\sigma = (a_{i,k}) \in \mathbb{R}^{|\sigma|\times\nu^2}$, $\mathbf{B}^\tau = (b_{j,\ell}) \in \mathbb{R}^{|\tau|\times\nu^2}$, and $\mathbf{C}^{\sigma,\tau} = (c_{k,\ell}) \in \mathbb{R}^{\nu^2\times\nu^2}$. However, the matrices $\mathbf{A}^\sigma$ and $\mathbf{B}^\tau$ are not computed explicitly. For $\sigma \in \mathbb{T}$ and $\sigma' \in \operatorname{sons}(\sigma)$, it can be shown that

$$\mathbf{A}^\sigma|_{\sigma'} := (a_{i,k})_{i\in\sigma',k=1,\dots,\nu} = \mathbf{A}^{\sigma'}\mathbf{T}^{\sigma,\sigma'}$$

with a transfer matrix $\mathbf{T}^{\sigma,\sigma'} \in \mathbb{R}^{|\sigma|\times|\sigma'|}$. For more details, we refer to [Bör10, Section 2.8]. Thus, in order to store an $\mathcal{H}^2$-Matrix, one only has to compute and store the matrices $\mathbf{A}^\sigma$ and $\mathbf{B}^\sigma$ for all leaves σ of the cluster tree $\mathbb{T}$ as well as the transfer matrices $\mathbf{T}^{\sigma,\sigma'}$. This results in a complexity of $\mathcal{O}(\mathcal{N}\,\nu^4)$ for the storage of an $\mathcal{N}\times\mathcal{N}$ $\mathcal{H}^2$-Matrix and the multiplication of this matrix with a vector; see [Bör10, Corollary 3.49]. However, the constants in the complexity estimates are larger as compared to the estimates for $\mathcal{H}$-Matrices. Therefore, only for large problems do $\mathcal{H}^2$-Matrices pay off.

Adaptive cross approximation (ACA) is an algorithm that can be used to create a low rank approximation of a matrix or a sub-block of a matrix by only using the matrix entries themselves.

For the description of ACA, we follow [Beb00] and [Rja02]. We consider a matrix $\mathbf{V} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$, $\mathbf{V} = (V_{i,j})$, and assume that the matrix entries can be expressed by $V_{i,j} = f(x_i, y_j)$ with a function $f : \mathbb{R}^2 \to \mathbb{R}$. The goal is to find a degenerate approximation $f(x, y) = \sum_{k=1}^{\nu} g_k(x)h_k(y)$, which then gives a rank ν approximation of $\mathbf{V}$ by $\mathbf{V} \approx \mathbf{A}\mathbf{B}^T$ with $\mathbf{A} = (A_{i,k}) \in \mathbb{R}^{\mathcal{M}\times\nu}$, $A_{i,k} = g_k(x_i)$, and $\mathbf{B} = (B_{j,k}) \in \mathbb{R}^{\mathcal{N}\times\nu}$, $B_{j,k} = h_k(y_j)$. Algorithm 5.3 gives the standard form of the ACA algorithm with full pivoting. Here, the full matrix $\mathbf{V}$ has to be assembled before the low rank approximation can be constructed. The algorithm then computes a low rank approximation $\mathbf{S} = \mathbf{A}\mathbf{B}^T$ and a residual $\mathbf{R}$ such that there holds $\mathbf{V} = \mathbf{S} + \mathbf{R}$. The construction of the matrices $\mathbf{A}$ and $\mathbf{B}$ requires $\mathcal{O}(\nu\mathcal{M}\mathcal{N})$. The memory requirement is $\mathcal{O}(\mathcal{M}\mathcal{N})$, since the complete matrix $\mathbf{V}$ has to be assembled and stored before it can be compressed.

Algorithm 5.3 Fully pivoted adaptive cross approximation ([Rja02, Algorithm 5]). We use the notation $\mathbf{V}_{\nu,.}$ and $\mathbf{V}_{.,\nu}$ for the νth row and column of the matrix $\mathbf{V}$.

INPUT: Matrix $\mathbf{V}$, tolerance tol.

OUTPUT: Matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{V} \approx \mathbf{A}\mathbf{B}^T$.

1: Set the residual $\mathbf{R}^{(0)} = \mathbf{V}$ and the approximation $\mathbf{S}_0 = 0$.

2: **for** $\nu = 0, 1, 2, \ldots$ **do**

3: **if** $\|\mathbf{R}^{(\nu)}\|_F \leq tol\|\mathbf{V}\|_F$ **then**

4: break;

5: **end if**

6: Find the index of the pivot element, $(i_{\nu+1}, j_{\nu+1}) = \text{argmax}|R^{(\nu)}_{i,j}|$.

7: Compute $\gamma_{\nu+1} = [R^{(\nu)}_{i_{\nu+1},j_{\nu+1}}]^{-1}$.

8: Set $\mathbf{A}_{.,\nu+1} = \gamma_{\nu+1}\mathbf{R}^{(\nu)}_{.,j_{\nu+1}}$ and $\mathbf{B}_{.,\nu+1} = \left[\mathbf{R}^{(\nu)}_{i_{\nu+1},.}\right]^T$.

9: Compute new residual

$$\mathbf{R}^{(\nu+1)} = \mathbf{R}^{(\nu)} - \mathbf{A}_{.,\nu+1}\mathbf{B}^T_{.,\nu+1}.$$

10: Compute new approximation

$$\mathbf{S}^{(\nu+1)} = \mathbf{S}^{(\nu)} + \mathbf{A}_{.,\nu+1}\mathbf{B}^T_{.,\nu+1}.$$

11: **end for**

An improved version is ACA with partial pivoting, which is stated in Algorithm 5.4. The algorithm only uses a few rows and columns of the original matrix $\mathbf{V}$, i.e. the matrix $\mathbf{V}$ never has to be assembled completely. The only requirement is that rows and columns of the matrix $\mathbf{V}$ can be computed. Furthermore, in Algorithm 5.4, line 9 can be omitted and $\|\mathbf{V}\|_F$ in line 10 can

be approximated by a term that only depends on the computed columns of $\mathbf{A}$ and $\mathbf{B}$, respectively; see [Rja02]. Both the storage requirement and the number of operations for the construction of $\mathbf{A}$ and $\mathbf{B}$ are of the order $\mathcal{O}(\nu^2(\mathcal{N}+\mathcal{M}))$. Furthermore, in [Beb00] it is shown that the error $\max_{i,j}|R_{i,j}|$ decays exponentially with respect to the rank ν if there holds $V_{i,j} = f(x_i, y_j)$ for an asymptotically smooth function $f : \mathbb{R}^2 \to \mathbb{R}$ and $(x_i, y_j) \in D_x \times D_y \subset \mathbb{R}^2$ with $\max\{\operatorname{diam}(D_x), \operatorname{diam}(D_y)\} \leq \eta \operatorname{dist}(D_x, D_y)$ for some $\eta > 0$.

Algorithm 5.4 Partially pivoted adaptive cross approximation ([Rja02, Algorithm 6]). We use the notation $\mathbf{V}_{\nu,.}$ and $\mathbf{V}_{.,\nu}$ for the νth row and column of the matrix $\mathbf{V}$.

INPUT: Tolerance tol.

OUTPUT: Matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{V} \approx \mathbf{A}\mathbf{B}^T$.

1: Set approximation $\mathbf{S}_0 = 0$ and initial row $i_1 = 1$;
2: **for** $\nu = 0, 1, 2, \ldots$ **do**
3: Compute row of the residual $\mathbf{R}^{(\nu)}_{i_{\nu+1},.} = \mathbf{V}_{i_{\nu+1},.} - \sum_{\ell=1}^{\nu} \mathbf{A}_{i_{\nu+1},\ell}\mathbf{B}^T_{.,\ell}$.
4: Compute $j_{\nu+1} = \operatorname{argmax}_j |R^{(\nu)}_{i_{\nu+1},j}|$.
5: Compute $\gamma_{\nu+1} = [R^{(\nu)}_{i_{\nu+1},j_{\nu+1}}]^{-1}$.
6: Compute column of the residual $\mathbf{R}^{(\nu)}_{.,j_{\nu+1}} = \mathbf{V}_{.,j_{\nu+1}} - \sum_{\ell=1}^{\nu} \mathbf{A}_{.,\ell}\mathbf{B}^T_{j_{\nu+1},\ell}$.
7: Set new vectors $\mathbf{A}_{.,\nu+1} = \gamma_{\nu+1} R^{(\nu)}_{.,j_{\nu+1}}$ and $\mathbf{B}_{.,\nu+1} = R^{(\nu)}_{i_{\nu+1},.}$.
8: Determine $i_{\nu+2} \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_{\nu+1}\}$. E.g. set [GBH06]

$$i_{\nu+2} = \operatorname{argmax}_{i \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_{\nu+1}\}} |\mathbf{B}_{i,\nu+1}|.$$

9: Compute new approximation

$$\mathbf{S}^{(\nu+1)} = \mathbf{S}^{(\nu)} + \mathbf{A}_{.,\nu+1}[\mathbf{B}_{.,\nu+1}]^T.$$

10: **if** $\|\mathbf{A}_{.,\nu+1}\|_2 \|\mathbf{B}_{.,\nu+1}\|_2 \leq tol \|\mathbf{S}^{(\nu)}\|_F$ **then**
11: break;
12: **end if**
13: **end for**

5.2. Perturbations in the Galerkin Matrix

Before we start with the description of the data compression scheme, we derive an estimate for how perturbations in the Galerkin matrix affect the consistency error in Strang's Lemma 1.5.2. First, we introduce some notation. We consider the discrete variational formulation for Symm's integral equation, which reads

$$a(\phi_h^p, w_h^p) = \langle \mathcal{V}\phi_h^p, w_h^p \rangle = \langle f, w_h^p \rangle \quad \forall w_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}). \tag{5.9}$$

Given a basis $\{\varphi_i,\ i = 1, \ldots, \mathcal{N}\}$ of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, we write $\phi_h^p = \sum_{i=1}^{\mathcal{N}} \phi_i \varphi_i$ and $w_h^p = \sum_{i=1}^{\mathcal{N}} w_i \varphi_i$. These substitutions yield

$$a(\phi_h^p, w_h^p) = \sum_{i,j \in \mathcal{I}} \phi_i V_{i,j} w_j,$$

where $\mathcal{I} = \{1, \ldots, \mathcal{N}\}$ is an index set and $\mathbf{V} = (V_{i,j})_{i,j \in \mathcal{I}}$ is the Galerkin matrix for the single layer operator. However, in practice one always deals with an approximation $\widetilde{\mathbf{V}} \approx \mathbf{V}$ of the Galerkin matrix, e.g. due to rounding errors in the computations, which means that we replace (5.9) by

$$a_h(\phi_h^p, w_h^p) = \sum_{i,j \in \mathcal{I}} \phi_i \widetilde{V}_{i,j} w_j = \langle f, w_h^p \rangle \quad \forall w_h^p \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}). \tag{5.10}$$

The question is how the approximation error affects the overall Galerkin error when approximating the Galerkin matrix $\mathbf{V}$ by $\widetilde{\mathbf{V}}$. The following theorem, which is a generalization of [GHS05, Theorem 5.1], states the main result. Note that here, we use that we have Legendre polynomials as element basis functions and assume $\mathbf{p} > 0$. The estimate in Theorem 5.2.1 would still hold if we wrote $\max\{\mathbf{p}, 1\}$ instead of $\mathbf{p}$.

Theorem 5.2.1. *If for all $i, j \in \mathcal{I}$ there holds for some $\varepsilon > 0$*

$$|V_{i,j} - \widetilde{V}_{i,j}| \leq \varepsilon,$$

then

$$\frac{|a(v,w) - a_h(v,w)|}{||v||_{H^{-1/2}(\Gamma)} ||w||_{H^{-1/2}(\Gamma)}} \leq \varepsilon\ \frac{\overline{p}^4}{\underline{h}^3} \tag{5.11}$$

for all $v, w \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$.

Proof. For this proof we are guided by the proof of Theorem 5.1 in [GHS05].
Let $v, w \in S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ with $v = \sum_{i \in \mathcal{I}} v_i \varphi_i$ and $w = \sum_{j \in \mathcal{I}} w_j \varphi_j$. Then

$$\begin{aligned} |a(v,w) - a_h(v,w)| &= \left| \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} v_i (V_{i,j} - \widetilde{V}_{i,j}) w_j \right| \leq \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} |v_i|\ |V_{i,j} - \widetilde{V}_{i,j}|\ |w_j| \\ &\leq \varepsilon \left(\sum_{i \in \mathcal{I}} |v_i| \right) \left(\sum_{j \in \mathcal{I}} |w_j| \right). \end{aligned}$$

Furthermore, using the Cauchy Schwarz inequality, we obtain

$$\sum_{i\in\mathcal{I}} |v_i| = \sum_{i\in\mathcal{I}} \left(\frac{p_i^{3/2}}{h_i}\right)\left(\frac{h_i}{p_i^{3/2}}\,|v_i|\right) \leq \left(\sum_{i\in\mathcal{I}} \frac{p_i^3}{h_i^2}\right)^{1/2} \left(\sum_{i\in\mathcal{I}} \frac{h_i^2}{p_i^3}|v_i|^2\right)^{1/2}, \tag{5.12}$$

where p_i and h_i are the polynomial degree and the mesh width on the support of φ_i, respectively. We consider the factors in (5.12) separately. First, there holds

$$\sum_{i\in\mathcal{I}} \frac{p_i^3}{h_i^2} = \sum_{T\in\mathcal{T}} \frac{p_T^3}{h_T^2} \sum_{k=0}^{p_T} 1 \leq 2 \sum_{T\in\mathcal{T}} \frac{p_T^4}{h_T^2} \leq 2\,\frac{\overline{p}^4}{\underline{h}^3}. \tag{5.13}$$

Using the properties of the local basis functions we obtain for each $T \in \mathcal{T}$

$$\|v\|_{L^2(T)}^2 = \frac{h_T}{2}\int_{-1}^{1} \left(\sum_{k=0}^{p_T} v_k^{(T)}\; P_k(t)\right)^2 dt = \frac{h_T}{2}\sum_{k=0}^{p_T} \frac{2}{2k+1}\;\left(v_k^{(T)}\right)^2 \geq \frac{1}{3}\frac{h_T}{p_T}\sum_{k=0}^{p_T}\left(v_k^{(T)}\right)^2.$$

Plugging this in yields for the second factor in (5.12)

$$\sum_{i\in\mathcal{I}} \frac{h_i^2}{p_i^3}\,|v_i|^2 = \sum_{T\in\mathcal{T}} \frac{h_T^2}{p_T^3} \sum_{k=0}^{p_T}\;\left(v_k^{(T)}\right)^2 \leq 3\sum_{T\in\mathcal{T}} \frac{h_T}{p_T^2}\,\|v\|_{L^2(T)}^2 = 3\left\|\frac{\mathbf{h}^{1/2}}{\mathbf{p}} v\right\|_{L^2(\Gamma)}^2. \tag{5.14}$$

Plugging (5.13) and (5.14) into (5.12) yields

$$\sum_{i\in\mathcal{I}} |v_i| \lesssim \frac{\overline{p}^2}{\underline{h}^{3/2}} \left\|\frac{\mathbf{h}^{1/2}}{\mathbf{p}} v\right\|_{L^2(\Gamma)}. \tag{5.15}$$

Finally, we use the inverse estimate

$$\left\|\frac{\mathbf{h}^{1/2}}{\mathbf{p}}\,u\right\|_{L^2(\Gamma)} \lesssim \|u\|_{H^{-1/2}(\Gamma)} \tag{5.16}$$

given in [Geo07, Theorem 3.9] and obtain

$$\sum_{i\in\mathcal{I}} |v_i| \lesssim \frac{\overline{p}^2}{\underline{h}^{3/2}} \left\|\frac{\mathbf{h}^{1/2}}{\mathbf{p}} v\right\|_{L^2(\Gamma)} \lesssim \frac{\overline{p}^2}{\underline{h}^{3/2}} \|v\|_{H^{-1/2}(\Gamma)}.$$

Using the same estimate for $\sum_{j\in\mathcal{I}} |w_j|$ completes the proof. □

Theorem 5.2.1 gives an upper bound for the consistency error in Strang's Lemma (1.5.2). We require that this consistency error is smaller than the discretization error of the Galerkin scheme. For hp-methods on quasi-uniform meshes and with quasi-uniform polynomial degrees, the Galerkin error between the exact solution ϕ of Symm's integral equation (1.11) and the discrete solution ϕ_h^p of (5.9) behaves like

$$\|\phi - \phi_h^p\|_{H^{-1/2}(\Gamma)} \leq C\frac{h^s}{p^{2s}},$$

where C is a constant independent of $\mathbf{h}$ and $\mathbf{p}$ and s depends on the regularity of the exact solution ϕ (see Section 1.5). Therefore, for given mesh width and polynomial degree functions $\mathbf{h}$ and $\mathbf{p}$, we require that there holds

$$\frac{|a(v,w) - a_h(v,w)|}{\|v\|_{H^{-1/2}(\Gamma)}\|w\|_{H^{-1/2}(\Gamma)}} \leq \varepsilon \, \frac{\overline{p}^4}{\underline{h}^3} \lesssim \frac{\underline{h}^s}{\overline{p}^{2s}},$$

which is equivalent to requiring for the perturbations in the Galerkin matrix that

$$|V_{i,j} - \widetilde{V}_{i,j}| \leq \varepsilon \lesssim \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}} \tag{5.17}$$

for all $i, j \in \mathcal{I}$.

Remark 5.2.2. If we considered the diagonally scaled Galerkin matrix it could be shown that the right hand side in (5.11) is $\varepsilon \, |\log(\underline{h})| \cdot \overline{p}^3/\underline{h}$. In this case, the estimate (5.17) would turn into $|V_{i,j} - \widetilde{V}_{i,j}| \leq \varepsilon \lesssim |\log(\underline{h})| \cdot \underline{h}^{1+s}/\overline{p}^{3+2s}$.

5.3. Diagonal Blocks

In this section we use the notation $\widetilde{\mathbf{V}} = \mathbf{V}^{(i,i)} = (V_{k,\ell})_{k,\ell=1,\dots,p_i+1}$ for any diagonal block of the Galerkin matrix $\mathbf{V}$. Here, p_i is the polynomial degree of the ansatz space on the element $T_i \in \mathcal{T}$.

For the entries of a diagonal block $\widetilde{\mathbf{V}}$, there exists an explicit formula, as stated and proven in Theorem 3.4.2. We obtain

$$V_{k,\ell} = -\frac{h_i^2}{4\pi} \begin{cases} \log h_i^2 - 3 & \text{, if } k = \ell = 0 \\ \dfrac{4}{((k+\ell+1)^2-1)((k-\ell)^2-1)} & \text{, if } k+\ell > 0 \text{ and } k-\ell \text{ is even} \\ 0 & \text{, otherwise.} \end{cases} \tag{5.18}$$

From this formula, it can be seen that the nonzero matrix entries behave smoothly with respect to the indices k and ℓ if $|k - \ell| > 1$. The idea is to divide the matrix block hierarchically into sub-blocks. On sub-blocks that are separated from the diagonal, we create low rank approximations by interpolating the matrix entries with respect to the indices. On sub-blocks that are not separated from the diagonal, we compute the entries directly using the formula given above.

For the hierarchical splitting, we use the framework of $\mathcal{H}$-Matrices [Hac99] that was introduced in Section 5.1 and adjust it to our setting. The following definition gives an admissibility condition.

Definition 5.3.1. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, we say that a $d_k \times d_\ell$ sub-block $\widetilde{\mathbf{A}}_S := (a_{k,\ell})_{(k,\ell)\in S}$ with $S = \{k_0, \dots, k_0 + d_k\} \times \{\ell_0, \dots, \ell_0 + d_\ell\} =: \sigma \times \tau$ is admissible if

$$\min\{d_k, d_\ell\} < \eta \, \operatorname{dist}(\sigma, \tau), \tag{5.19}$$

where the admissibility constant $\eta > 0$ has to be chosen.

Note that we do not approximate the kernel function as in the standard $\mathcal{H}$-Matrix approach. Instead, we approximate the matrix entries themselves. Therefore, we do not need a spatial splitting using bounding boxes as introduced in Section 5.1, but only a splitting of the index set $\mathcal{I}$. For the construction of a cluster tree, we use a modification of Algorithm 5.1 that is stated in Algorithm 5.5.

Algorithm 5.5 Cluster tree (splitting with respect to indices)

1: **procedure** ClusterTree(cluster σ, cluster tree $\mathbb{T}$, leaf size C_{leaf})
2: **if** $|\sigma| < C_{leaf}$ **then**
3: **return** ;
4: **else**
5: For $\sigma = \{k_0, \dots, k_0 + d\}$, define sons σ' and σ'' by

$$\sigma' = \{k_0, \dots, k_0 + \lfloor d/2 \rfloor\}, \quad \sigma'' = \{k_0 + \lfloor d/2 \rfloor + 1, \dots, k_0 + d\}.$$

6: Add σ' to $\mathbb{T}$ and call ClusterTree(σ', $\mathbb{T}$, C_{leaf}).
7: Add σ'' to $\mathbb{T}$ and call ClusterTree(σ'', $\mathbb{T}$, C_{leaf}).
8: **end if**

Using the admissibility condition (5.19), we can create a hierarchical matrix splitting with Algorithm 5.2. Note that the admissible pairs of clusters correspond to admissible matrix blocks and the inadmissible pairs of clusters correspond to inadmissible matrix blocks. Figure 5.2 shows the hierarchical splitting of a 30×30 matrix.

For each admissible block, we create a low rank approximation using interpolation with respect to the row indices. Therefore, we consider the function

$$f : \mathbb{R}^2 \to \mathbb{R}, \;\; f(x, y) = \frac{1}{(x + y + 2)(x + y)((y - x)^2 - 1)}. \tag{5.20}$$

Note that there holds

$$V_{\ell,k} = V_{k,\ell} = -\frac{h_i^2}{\pi} f(k, \ell)$$

for all k, ℓ with $k + \ell > 0$ and $|k - \ell|$ even. Since the matrix block $\widetilde{\mathbf{V}}$ is symmetric and we only evaluate f at indices $k, \ell \in \mathbb{N}$, we can assume without loss of generality that $x > y \geq 1$. Furthermore, in order to keep the notation simple, we only consider square blocks $S = \{k_0, \dots, k_0 + d\} \times \{\ell_0, \dots, \ell_0 + d\}$. However, the error analysis can be generalized in a direct way to rectangular sub-blocks.

The following error analysis is based on and adapted from the error analysis in [AR91]. Let $t_1, \dots t_\nu \in [-1, 1]$ be Chebyshev interpolation points and L_n, $n = 1, \dots, \nu$, be the corresponding Lagrange basis polynomials. For a given square $S = \{k_0, \dots, k_0 + d\} \times \{\ell_0, \dots, \ell_0 + d\}$,

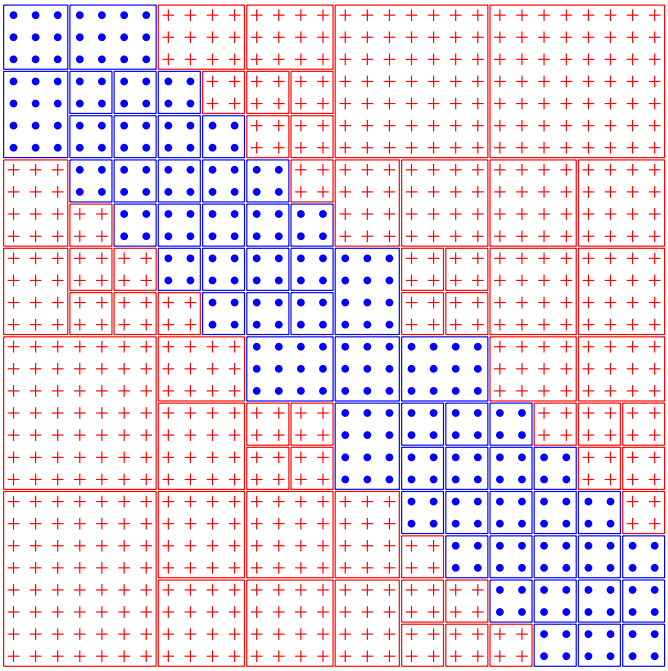


Figure 5.2.: Hierarchical splitting of a 30×30 matrix. The red blocks correspond to admissible blocks, where each red cross stands for one matrix entry. The blue blocks correspond to inadmissible blocks, and each blue dot corresponds to one matrix entry. For the splitting, we used $\eta = 0.5$ and $C_{leaf} = 3$.

the entries of the corresponding sub-block $\widetilde{\mathbf{V}}_S$ are given by

$$V_{k_0+k,\ell_0+\ell} = -\frac{h_i^2}{\pi} f(k_0+k,\ell_0+\ell) \approx -\frac{h_i^2}{\pi} \sum_{n=1}^{\nu} f(k_0+t_n d,\ell_0+\ell) L_n\left(\frac{k}{d}\right)$$
$$= -\frac{h_i^2}{\pi} \sum_{n=1}^{\nu} a_{\ell,n} b_{k,n}. \tag{5.21}$$

Thus, the $d \times d$ block $\widetilde{\mathbf{V}}_S$ can be approximated by two $d \times \nu$ blocks $\mathbf{A} = (a_{\ell,n})$, $\mathbf{B} = (b_{k,n})$, and the rank of the approximation is the order of the interpolation.

In the following we estimate the interpolation error, in order to be able to get an upper bound for the interpolation error that can be made without affecting the overall error of the Galerkin method.

Note that each square $S = \{k_0,\ldots,k_0+d\} \times \{\ell_0,\ldots,\ell_0+d\}$ can be identified with $\widehat{S} =$

$[x_0, x_0+d] \times [y_0, y_0+d] \subset \mathbb{R}^2$ with $x_0 = k_0 \geq y_0 = \ell_0$. Furthermore, $\widehat{S}$ is admissible if

$$d < \eta\,(x_0 - y_0 - d) \quad \iff d \leq \frac{\eta}{1+\eta}(x_0 - y_0). \tag{5.22}$$

We are interested in the interpolation error when we interpolate f with respect to x on an admissible square $\widehat{S} = [x_0, x_0+d] \times [y_0, y_0+d]$. Therefore, we consider the function $F_y := f(\cdot, y)$ and state the following result.

Theorem 5.3.2. *Let $S := [x_0, x_0+d] \times [y_0, y_0+d]$ be an admissible square. i.e.*

$$d < \eta\,(x_0 - y_0 - d)$$

for some given $\eta > 0$ and $d > 0$. For $y \in [y_0, y_0+d]$, let F_y be as defined above and $I_x^\nu F_y$ be the νth order Chebyshev interpolation polynomial on $[x_0, x_0+d]$. Then for the interpolation error, there holds

$$E(F_y, \nu, S) := \sup_{x \in (x_0, x_0+d)} |F_y(x) - I_x^\nu F_y(x)| \leq \frac{8}{d^2}\left(\frac{\eta}{2}\right)^{\nu+2}. \tag{5.23}$$

For the proof of this theorem we need some auxiliary results. The first one is a direct consequence of Cauchy's integral formula and stated in [AR91].

Lemma 5.3.3. ([AR91, Lemma 2]) *Let $z \in \mathbb{C}$, $r \in \mathbb{R}$, $r > 0$, and $D := \{w \in \mathbb{C} : |w - z| \leq r\}$. If $g : D \to \mathbb{C}$ is continuous on D and analytic on $D \setminus \partial D$, then there holds for the kth derivative of g on D*

$$|g^{(k)}(z)| \leq \frac{k!}{r^k} \sup_{\alpha \in [0, 2\pi]} \left|g(z + re^{\mathrm{i}\alpha})\right|.$$

We use this lemma to obtain an upper bound for the kth derivative of the function F_y. Therefore, we extend F_y to $\mathbb{C}$ and derive an upper bound for $F_y(x + re^{\mathrm{i}\alpha})$. Choosing $r := (x-y)/2$ ensures that F_y is continuous on $\{w \in \mathbb{C} : |w - x| \leq r\}$ and analytic in the interior of this disc.

Lemma 5.3.4. *Let $x, y \in \mathbb{R}$ with $x > y > 1$ and $x - y > 2$. Let $r := \frac{x-y}{2}$ and F_y as defined above. Then there holds*

$$\sup_{\alpha \in [0, 2\pi]} \left|F_y(x + re^{\mathrm{i}\alpha})\right| \leq \frac{1}{(x-y)^2}. \tag{5.24}$$

Proof. There holds for any $\alpha \in [0, 2\pi]$

$$\begin{aligned}
|F_y(x + r\,e^{\mathrm{i}\alpha})| &= \left| \frac{1}{(x + y + 2 + r\,e^{\mathrm{i}\alpha})(x + y + r\,e^{\mathrm{i}\alpha})\,[(x - y + r\,e^{\mathrm{i}\alpha})^2 - 1]} \right| \\
&\leq \frac{1}{|x + y + 2 + r\,e^{\mathrm{i}\alpha}|\;|x + y + r\,e^{\mathrm{i}\alpha}|\;|(x - y + r\,e^{\mathrm{i}\alpha})^2 - 1|} \\
&\leq \frac{1}{|x + y + r\,e^{\mathrm{i}\alpha}|^2\;|x - y + r\,e^{\mathrm{i}\alpha}|^2}.
\end{aligned}$$

We consider the two factors in the denominator separately and use $r = (x-y)/2$. We obtain

$$\begin{aligned}
|x+y+r\ e^{\mathrm{i}\alpha}|^2 &= |x+y+r\cos(\alpha)+\mathrm{i}r\sin(\alpha)|^2 \\
&= (x+y+r\cos(\alpha))^2 + r^2\sin^2(\alpha) \\
&= (x+y)^2 + 2(x+y)r\underbrace{\cos(\alpha)}_{\geq -1} + r^2(\cos^2(\alpha)+\sin^2(\alpha)) \\
&\geq (x+y-r)^2 = (x+y-\frac{x}{2}+\frac{y}{2})^2 \\
&= \frac{1}{4}(x+3y)^2 \geq 4
\end{aligned}$$

and

$$|x-y+r\ e^{\mathrm{i}\alpha}|^2 = \left|(x-y)\left(1+\frac{e^{\mathrm{i}\alpha}}{2}\right)\right|^2 = (x-y)^2\left|1+\frac{e^{i\alpha}}{2}\right|^2 \geq \frac{1}{4}(x-y)^2.$$

Overall, we obtain

$$|F_y(x+r\ e^{\mathrm{i}\alpha})| \leq \frac{1}{(x-y)^2},$$

which completes the proof. □

We are now ready to prove Theorem 5.3.2.

Proof of Theorem 5.3.2. Let $x, y \in S$ for an admissible square S. The interpolation error for νth order Chebyshev interpolation of the function F_y on the interval $(x_0, x_0+d) \subset \mathbb{R}$ is given by

$$E(F_y,\nu,S) = \sup_{x\in(x_0,x_0+d)} |F_y - I_x^\nu F_y| = \frac{2\,d^\nu}{\nu!\ 4^\nu} \sup_{x\in(x_0,x_0+d)} |F_y^{(\nu)}(x)|. \tag{5.25}$$

Combining the result (5.24) with Lemma 5.3.3, we obtain

$$|F_y^{(\nu)}(x)| \leq \frac{\nu!\ 2^\nu}{(x-y)^{\nu+2}}.$$

Together with (5.25), we get

$$E(F_y,\nu,S) \leq 2\,\frac{d^\nu}{\nu!\ 4^\nu}\nu!\,2^\nu \sup_{x\in(x_0,x_0+d)} \frac{1}{(x-y)^{\nu+2}} \leq 2\,\frac{d^\nu}{2^\nu} \sup_{(x,y)\in S} \frac{1}{(x-y)^{\nu+2}}.$$

For $(x,y) \in S$ with an admissible square S, there holds

$$(x-y) \geq (x_0 - y_0 - d) \geq \frac{d}{\eta}.$$

Plugging in the previous inequality yields

$$E(F_y,\nu,S) \leq 2\,\frac{d^\nu}{2^\nu} \sup_{(x,y)\in S} \frac{1}{(x-y)^{\nu+2}} \leq \frac{8}{d^2}\left(\frac{\eta}{2}\right)^{\nu+2}.$$

□

The interpolation error corresponds to a perturbation of the Galerkin matrix $\mathbf{V}$. Therefore, the estimate (5.17) also applies for the interpolation error, i.e. we have to require

$$E(F_y, \nu, S) \lesssim \left(\frac{\eta}{2}\right)^{\nu+2} \frac{1}{d^2} \lesssim \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}}. \tag{5.26}$$

Solving the inequality for the interpolation order, which is also the rank of the low rank approximations on admissible sub-blocks, we obtain

$$\nu \gtrsim \frac{2\log(d) - (4+2s)\log(\overline{p}) + (3+s)\log(\underline{h})}{\log(n/2)} - 2,$$

which means that the choice

$$\nu = \mathcal{O}(\log \overline{p})$$

suffices for uniform p-BEM. Thus, using the complexity estimates for $\mathcal{H}$-Matrices from Proposition 5.1.5, we find that the storage requirement for one diagonal block N_{st}^D and the complexity for the multiplication of one diagonal block with a vector N_{Mv}^D are of the order

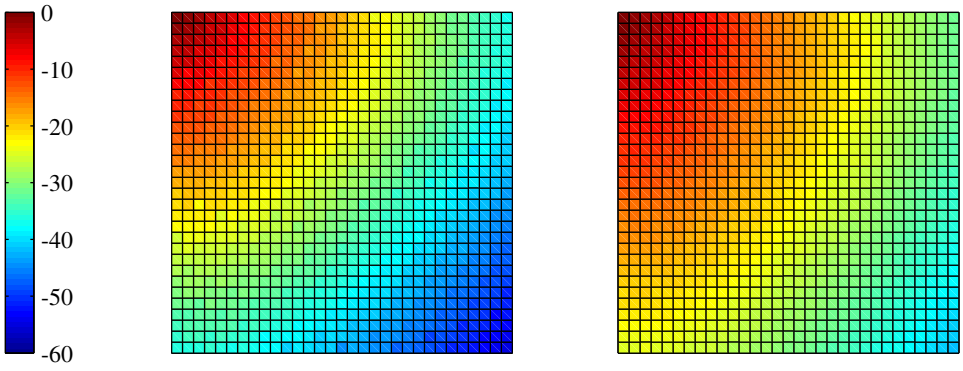
$$N_{st}^D, N_{Mv}^D \in \mathcal{O}(\overline{p}\log(\overline{p})^2). \tag{5.27}$$

5.4. Far Field Blocks

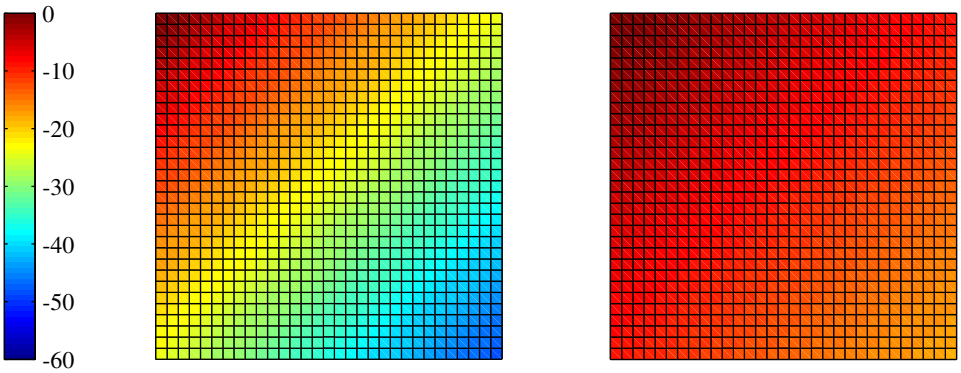
For the far field blocks, i.e. the blocks that correspond to disjoint elements, we use the fact that the matrix entries decay rapidly. Figure 5.3(a) visualizes sub-blocks of the Galerkin matrix $\mathbf{V}$ for different combinations of boundary elements T_i and T_j for $p_i = p_j = 31$. Each square corresponds to one matrix entry and the color corresponds to the logarithm of the absolute value of that matrix entry. Since small perturbations in the Galerkin matrix do not affect the overall error of the Galerkin scheme (see Theorem 5.2.1), matrix entries of small magnitude can simply be neglected. In order to determine a priori which matrix entries are small and can be neglected, i.e. without having to compute them, we need an upper bound for the magnitude of the matrix entries. The following lemma, which is stated and proven in [Heu96, Lemma 1], gives such an upper bound. Since it will help us analyze the quality of this upper bound, we also give an outline of the proof.

Lemma 5.4.1. ([Heu96, Lemma 1]) *There exists a constant C depending on Γ such that*

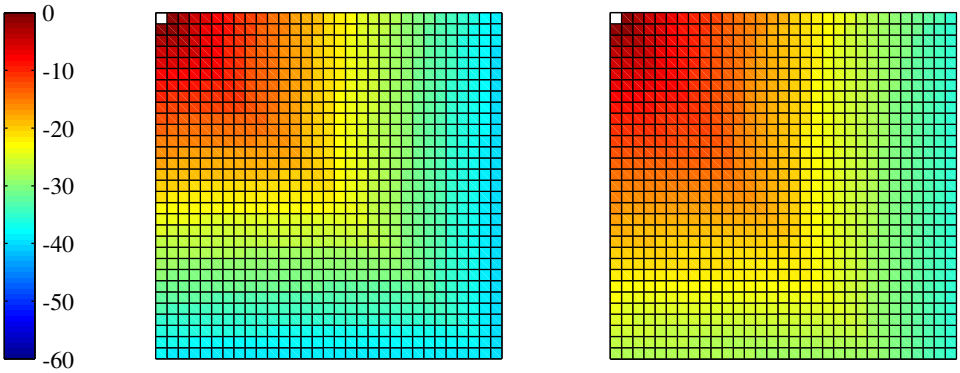
$$|V_{k,\ell}^{(i,j)}| = |\langle \mathcal{V} P_k^{(i)}, P_\ell^{(j)} \rangle| \leq \frac{C}{\pi} \left(\frac{\max\{|T_i|, |T_j|\}}{2\,\mathrm{dist}\{T_i, T_j\}} \right)^{k+\ell}. \tag{5.28}$$



(a) Matrix block for $T_i = \mathrm{conv}((-1,0),(1,0))$, $T_j = \mathrm{conv}((5,5),(6.5,4))$ (left), and $T_j = \mathrm{conv}((3,0),(4,0))$ (right), respectively. The color corresponds to the logarithm of the absolute value of the matrix entries.



(b) Upper bound (5.28) for $T_i = \mathrm{conv}((-1,0),(1,0))$, $T_j = \mathrm{conv}((5,5),(6.5,4))$ (left), and $T_j = \mathrm{conv}((3,0),(4,0))$ (right), respectively. The constant C was chosen such that for $k = \ell = 0$ the upper bound coincides with the exact matrix entry.



(c) Upper bound (5.35) for $T_i = \mathrm{conv}((-1,0),(1,0))$, $T_j = \mathrm{conv}((5,5),(6.5,4))$ (left), and $T_j = \mathrm{conv}((3,0),(4,0))$ (right), respectively.

Figure 5.3.: Matrix blocks $\mathbf{V}^{(i,j)}$ and upper bounds (5.28) and (5.35) for different combinations of $T_i = \mathrm{conv}((-1,0;1,0))$ and T_j for $p = 31$.

Outline of the proof. We do not state the complete proof here, but mention the key ideas. We use the Taylor expansion of $\log|x-y|$ and the orthogonality of Legendre polynomials to obtain

$$\begin{aligned}\langle \mathcal{V}P_k^{(i)}, P_\ell^{(j)}\rangle &= -\frac{1}{2\pi}\int_{T_i}\int_{T_j} P_k^{(i)}(x)P_\ell^{(j)}(y)\log|x-y|\,ds_y\,ds_x \\ &= -\frac{1}{2\pi}\int_{T_i}\int_{T_j} P_k^{(i)}(x)P_\ell^{(j)}(y)R_{k,\ell}(x,y)\,ds_y\,ds_x,\end{aligned}$$

where $R_{k,\ell}(x,y)$ is the remainder of the Taylor expansion. Using an estimate for the partial derivatives of $\log|x-y|$, one obtains an estimate for $|R_{k,\ell}(x,y)|$. Plugging in this estimate and using the Cauchy-Schwarz inequality completes the proof. □

We combine the upper bound (5.28) with the estimate (5.17) for the perturbations in the Galerkin matrix and observe that we can neglect all matrix entries with

$$|V_{k,\ell}^{(i,j)}| \leq \frac{C}{\pi}\left(\frac{\max\{|T_i|,|T_j|\}}{2\operatorname{dist}\{T_i,T_j\}}\right)^{k+\ell} =: \frac{C}{\pi}c^{k+\ell} \lesssim \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}}. \tag{5.29}$$

Solving this inequality for $k+\ell$ yields

$$k+\ell \geq \frac{-(3+s)\log(\underline{h}) + (4+2s)\log(\overline{p}) + \log(C/\pi)}{-\log(c)}. \tag{5.30}$$

Thus, we only need to compute and store $\mathcal{O}(\log(\overline{p})^2)$ matrix entries per far field block. Using the criterion (5.30), we create a first algorithm for the compression of a far field block $\mathbf{V}^{(i,j)}$, which is stated in Algorithm 5.6.

Algorithm 5.6 Far field compression (1)

Input: Boundary elements $T_i, T_j \in \mathcal{T}$

1: Compute
$$c := \frac{\max\{|T_i|,|T_j|\}}{2\operatorname{dist}\{T_i,T_j\}}.$$

2: (**A priori compression.**) Determine ℓ_0 such that Equation (5.30) holds for all $\ell = 0,1,\ldots,\ell_0$ and $k=0$.

3: Compute matrix block $(V_{k,\ell}^{(i,j)})_{k,\ell=0,\ldots,\ell_0}$.

4: (**A posteriori compression.**) Neglect all computed matrix entries, whose absolute value is smaller than $\underline{h}^{3+s}/\overline{p}^{4+2s}$ and store the remaining entries as a sparse matrix.

Note that in step 2 of Algorithm 5.6 we compute a square matrix block as opposed to a triangular one as indicated by Equation (5.30). The reason for this is that we compute the matrix entries using recurrence relations (see Sections 3.4 and 2.4) and our implementation produces rectangular matrix blocks. Furthermore, it suffices to consider the first row, since the bound

(5.30) is symmetric with respect to k and ℓ.

However, the upper bound (5.28) is too coarse, especially for matrix blocks that correspond to elements T_i, T_j with small distance; see Figure 5.3(b). This results from the Taylor expansion of the logarithm being used for the construction of the upper bound (5.28). If the distance of the boundary elements T_i and T_j is small, the Taylor expansion does not give a good approximation of the kernel function $\log|x-y|$ for $x \in T_i$ and $y \in T_j$.

To obtain a better algorithm, we derive another upper bound based on asymptotic estimates. First, recall that the computations of Galerkin entries $V_{k,\ell}^{(i,j)}$ can be reduced to the evaluation of the integral

$$I_{k,\ell}^{-1}(a,b) = \int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log(a+bs-t)\, dt\, ds,$$

where the complex numbers a and b depend of the relative position and size of the boundary elements T_i and T_j. The detailed derivation is shown in Theorem 3.4.1. Furthermore, plugging in the definition of the function $\widetilde{Q}_\ell^{-1}$ (see Definition 2.3.1) and applying the Cauchy Schwarz inequality yields

$$\begin{aligned}
\left|I_{k,\ell}^{-1}(a,b)\right|^2 &= \left|\int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log(a+bs-t)\, dt\, ds\right|^2 \\
&= \left|\int_{-1}^{1} P_k(s) \widetilde{Q}_\ell^{-1}(a+bs)\, ds\right|^2 \\
&\leq \int_{-1}^{1} P_k(s)^2\, ds \int_{-1}^{1} \left|\widetilde{Q}_\ell^{-1}(a+bs)\right|^2\, ds.
\end{aligned}$$

Plugging in the orthogonality property of the Legendre polynomials results in the upper bound

$$\left|I_{k,\ell}^{-1}(a,b)\right|^2 \leq \frac{2}{2k+1} \int_{-1}^{1} \left|\widetilde{Q}_\ell^{-1}(a+bs)\right|^2\, ds, \tag{5.31}$$

where the integral can be computed by a quadrature rule. However, since we apply the Cauchy Schwarz inequality, we lose the effect of the orthogonality of the Legendre polynomials. The estimate decays fast with respect to ℓ, since $\widetilde{Q}_\ell^{-1}(z)$ decays fast with respect to ℓ (see the asymptotic estimate (2.38)). But with respect to k, the upper bound (5.31) decays slowly and thus is not effective. Therefore, we switch the order of integration and obtain for $k+\ell > 0$

$$\begin{aligned}
\left|I_{k,\ell}^{-1}(a,b)\right|^2 &= \left|\int_{-1}^{1} P_k(s) \int_{-1}^{1} P_\ell(t) \log(a+bs-t)\, dt\, ds\right|^2 \\
&= \left|\int_{-1}^{1} P_\ell(t) \int_{-1}^{1} P_k(s) \log\left(-\frac{a}{b} + \frac{1}{b}t - s\right)\, ds\, dt\right|^2 \\
&\leq \frac{2}{2\ell+1} \int_{-1}^{1} \left|\widetilde{Q}_k^{-1}\left(-\frac{a}{b} + \frac{1}{b}t\right)\right|^2\, dt.
\end{aligned} \tag{5.32}$$

Combining the bounds (5.31) and (5.32) and using $z_1(s) = a + bs$ and $z_2(s) = -a/b + 1/b \cdot s$, we obtain

$$\left|I_{k,\ell}^{-1}(a,b)\right|^2 \leq \min\Bigg\{ \frac{2}{2k+1} \int_{-1}^{1} \left|\widetilde{Q}_\ell^{-1}(z_1(s))\right|^2 \, ds, \\ \frac{2}{2\ell+1} \int_{-1}^{1} \left|\widetilde{Q}_k^{-1}(z_2(s))\right|^2 \, ds \Bigg\} =: \left(\widehat{I}_{k,\ell}(a,b)\right)^2. \tag{5.33}$$

However, even though the time for the evaluation of $\widetilde{Q}_\ell^{-1}$ is almost negligible as compared to the time for the computation of Galerkin entries (recall that we use `MPC` for the computation of Galerkin entries), we achieve a further improvement by using the asymptotic estimate (2.38):

$$\left|I_{k,\ell}^{-1}(a,b)\right|^2 \leq \min\Bigg\{ \frac{32\,\pi}{2k+1} \ell^{-3} \int_{-1}^{1} \left|(z_1(s) + \sqrt{z_1(s)^2 - 1}\right|^{-2\ell} ds, \\ \frac{32\,\pi}{2\ell+1} k^{-3} \int_{-1}^{1} \left|(z_2(s) + \sqrt{z_2(s)^2 - 1}\right|^{-2k} ds \Bigg\} =: \left(\overline{I}_{k,\ell}(a,b)\right)^2. \tag{5.34}$$

Plugging the estimates (5.33) and (5.34), respectively, into the formula (3.11) for the evaluation of $V_{k,\ell}^{(i,j)}$ yields for $k, \ell \in \mathbb{N}$

$$|V_{k,\ell}^{(i,j)}| \leq \frac{h_i\, h_j}{8\pi}\, \widehat{I}_{k,\ell}(a,b) \qquad \text{and} \qquad |V_{k,\ell}^{(i,j)}| \leq \frac{h_i\, h_j}{8\pi}\, \overline{I}_{k,\ell}(a,b), \tag{5.35}$$

respectively. Figure 5.3(c) shows the upper bound (5.35) for two combinations of boundary elements. Comparing this estimate with the estimate (5.28) (Figure 5.3(b)) and the exact matrix block (Figure 5.3(a)), it can be seen that the new estimate (5.35) performs considerably better than the estimate (5.28). Using the upper bound (5.35), we modify Algorithm 5.6 and obtain an improved algorithm for the far field compression stated in Algorithm 5.7.

Table 5.1 shows the sizes of a far field block after the a priori and the a posteriori compression. It can be seen that Algorithm 5.7 produces a considerably better a priori compression than Algorithm 5.6. Furthermore, after the a posteriori compression, only a small number of matrix entries remain. For example, for $T_j = \mathrm{conv}((3,0),(4,0))$, only 8 entries of the 32×32 matrix block need to be stored.

Algorithm 5.7 Far field compression (2)

Input: Boundary elements $T_i, T_j \in \mathcal{T}$

1: Compute $a, b \in \mathbb{C}$ according to Equation (3.9).

2: (**A priori compression.**)
Set $k = 0$ and determine ℓ_0 such that there holds (see (5.17))

$$\frac{h_i\, h_j}{8\pi}\, \overline{I}_{k,\ell}(a,b) \leq \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}}, \quad \forall \ell = 1, \ldots, \ell_0.$$

Set $\ell = 0$ and determine k_0 such that there holds (see (5.17))

$$\frac{h_i\, h_j}{8\pi}\, \overline{I}_{k,\ell}(a,b) \leq \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}}, \quad \forall k = 1, \ldots, k_0.$$

3: Compute matrix block $(V_{k,\ell}^{(i,j)})_{k=0,\ldots,k_0,\ell=0,\ldots,\ell_0}$.

4: (**A posteriori compression.**) Neglect all computed matrix entries whose absolute value is smaller than $\underline{h}^{3+s}/\overline{p}^{4+2s}$ and store the remaining entries as a sparse matrix.

Table 5.1.: Sizes of a compressed far field block for $T_i = \mathrm{conv}((-1,0),(1,0))$ and $T_j = \mathrm{conv}(A_j, B_j)$ after the a priori compression in Algorithm 5.6 and 5.7, respectively, and after the a posteriori compression. We choose $p = 31$ and set $h = 1$, which means that $\underline{h}^{7/2}/\overline{p}^{5} \approx 3.5 \cdot 10^{-8}$. Note that in this case the full block has 1024 entries.

A_j	B_j	nnz a priori Alg. 5.6	nnz a priori Alg. 5.7	nnz a posteriori
$(3.0, 0.0)$	$(4.0, 0.0)$	81	42	14
$(2.0, 2.0)$	$(3.0, 3.0)$	121	42	18
$(7.2, 5.1)$	$(6.9, 3.7)$	49	36	10
$(-2.0, 2.0)$	$(0.0, 1.5)$	1024	88	35

5.5. Off-Diagonal Blocks

For the off-diagonal blocks, i.e. matrix blocks $\mathbf{V}^{(i,j)}$ that correspond to neighboring boundary elements T_i, T_j, we do not have a direct formula for the computation of matrix entries as it is the case for the blocks $\mathbf{V}^{(i,i)}$. Thus, we can not use the hierarchical format presented in Section 5.3. Furthermore, the matrix entries do not decay as rapidly as for far field blocks (see Figure 5.4). Therefore, a sparse matrix format as used for the far field compression in Section 5.4 is not appropriate here.

The compression scheme we use for the off-diagonal blocks is adaptive cross approximation. We compare the two different versions of the cross approximation algorithm stated in Section 5.1, namely the fully pivoted ACA (Algorithm 5.3) and the partially pivoted ACA (Algorithm 5.4). However, for both algorithms, we modify the stopping criterion such that it fits the error estimate for perturbations in the Galerkin matrix (5.17). We exit the "for" loop, if

$$\max_{i=0,\dots p_i, j=0,\dots p_j} |R^{(k)}(i,j)| < \varepsilon = \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}} \tag{5.36}$$

for a given tolerance. Recall that there holds

$$\mathbf{V}^{(i,j)} = \mathbf{S}^{(\nu)} + \mathbf{R}^{(\nu)},$$

and therefore the absolute approximation error is

$$\max_{k=0,\dots p_i, \ell=0,\dots p_j} |V^{(i,j)}_{k,\ell} - S^{\nu}_{k,\ell}| = \max_{k=0,\dots p_i, \ell=0,\dots p_j} R^{(\nu)}_{k,\ell}.$$

For the comparison of the two algorithms, we choose $T_i = \operatorname{conv}((-1,0),(1,0))$ and $T_j = \operatorname{conv}((1,0), B)$ with $B = (1 - r\cos(\alpha), r\sin(\alpha))$. Figure 5.4 shows the matrix blocks $\mathbf{V}^{(i,j)}$ for four different combinations of r and α. Here, the color corresponds to the logarithm of the absolute value of each matrix entry. It can be seen that, especially for angles $\alpha \geq \pi/2$, the matrix entries do not behave like a smooth function.

Figure 5.5 shows the rank of the approximation produced by Algorithm 5.3 and 5.4, respectively, for decreasing tolerance ε in (5.36). We run both algorithms for a $p \times p$ matrix block for growing values of p. For each p, we choose $\varepsilon = p^{-4}$. Whereas the ranks produced by the fully pivoted ACA grow like $\log(1/\varepsilon)^2$, the ranks produced by the partially pivoted ACA grow faster in most cases. Only for angles close to π, the partially pivoted ACA creates low rank approximations with ranks that are of the order $\mathcal{O}(\log(1/\varepsilon)^2)$.

Another problem that arises in the partially pivoted ACA (Algorithm 5.4) is that it requires methods for the computation of a single row or column of the matrix that has to be approximated. Since we compute the matrix entries using recurrence relations (see Section 3.4), it is not possible to compute a row or a column of the matrix block $\mathbf{V}^{(i,j)}$ without having computed all previous rows and columns, respectively. Therefore, we choose the fully pivoted algorithm for

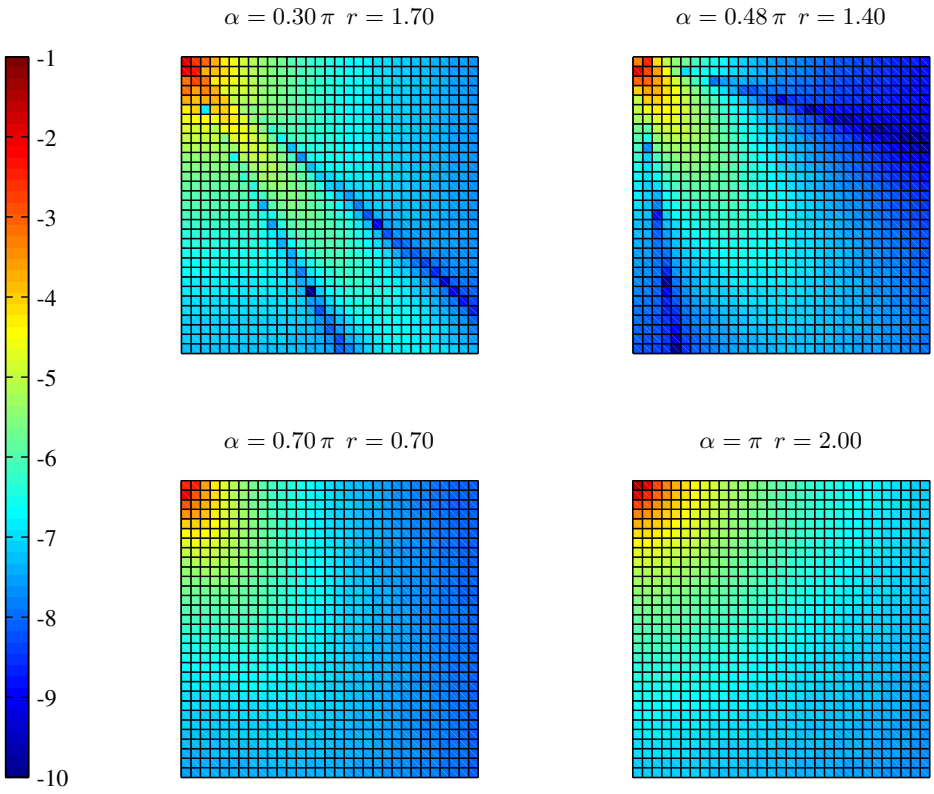
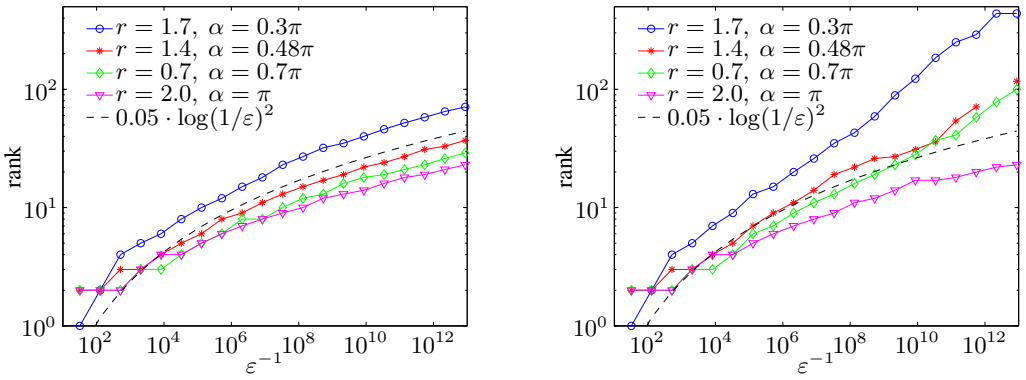


Figure 5.4.: Matrix blocks $\mathbf{V}^{(i,j)}$ for $T_i = \mathrm{conv}((-1,0),(1,0))$ and $T_j = \mathrm{conv}((1,0),B)$ with $B = (1 - r\cos(\alpha), r\sin(\alpha))$ for different $\alpha \in [0,\pi]$ and $r > 0$. The polynomial degrees are chosen as $p_i = p_j = 31$. The color corresponds to the absolute value of the logarithm with base 10 of the matrix entries.



(a) Ranks produced by the fully pivoted ACA.

(b) Ranks produced by the partially pivoted ACA.

Figure 5.5.: Ranks produced by the fully and partially pivoted ACA over the inverse of the tolerance ε from (5.36) for $T_i = \mathrm{conv}((-1,0),(1,0))$ and $T_j = \mathrm{conv}((1,0),B)$ with $B = (1 - r\cos(\alpha), r\sin(\alpha))$ for different $\alpha \in [0,\pi]$ and $r > 0$.

the approximation of off-diagonal blocks. The final algorithm we use is stated in Algorithm 5.8. If we choose the tolerance ε as in (5.17), for the rank ν produced by the fully pivoted ACA, there holds

$$\nu \leq C \log \left(\frac{\overline{p}^{4+2s}}{\underline{h}^{3+s}} \right)^2 . \tag{5.37}$$

This means that for p-BEM with constant $\mathbf{h}$ we have $\nu = \mathcal{O}(\log(\overline{p})^2)$. Overall, we obtain that the storage requirement and the complexity of the matrix vector multiplication are proportional to $p \log(\overline{p})^2$ for a $p \times p$ off-diagonal block.

Algorithm 5.8 Compression of off-diagonal blocks. We use the notation $\mathbf{V}_{\nu,.}$ and $\mathbf{V}_{.,\nu}$ for the νth row and column of the matrix $\mathbf{V}$.

INPUT: Vertices of the boundary elements $T_i, T_j \in \mathcal{T}$, polynomial degrees p_i, p_j, and a tolerance ε.

OUTPUT: Matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{V}^{(i,j)} \approx \mathbf{A}\mathbf{B}^T$.

1: Compute the matrix block $\mathbf{V}^{(i,j)}$ using the methods described in Section 3.4.
2: Set the residual $\mathbf{R}^{(0)} = \mathbf{V}^{(i,j)}$.
3: **for** $\nu = 0, 1, 2, \ldots$ **do**
4: **if** $\max_{i=0,\ldots,p_i, j=0,\ldots,p_j} |R^{(\nu)}_{i,j}| \leq \varepsilon$ **then**
5: break;
6: **end if**
7: Find the index of the pivot element, $(i_{\nu+1}, j_{\nu+1}) = \operatorname{argmax}_{i=0,\ldots,p_i, j=0,\ldots,p_j} |R^{(\nu)}_{i,j}|$.
8: Compute $\gamma_{\nu+1} = [R^{(\nu)}_{i_{\nu+1}, j_{\nu+1}}]^{-1}$.
9: Set $\mathbf{A}_{.,\nu+1} = \gamma_{\nu+1} R^{(\nu)}_{.,j_{\nu+1}}$ and $\mathbf{B}_{.,\nu+1} = \left[R^{(\nu)}_{i_{\nu+1},.} \right]^T$.
10: Compute new residual

$$\mathbf{R}^{(\nu+1)} = \mathbf{R}^{(\nu)} - \mathbf{A}_{.,\nu+1} [\mathbf{B}_{.,\nu+1}]^T .$$

11: **end for**

5.6. Extension of the Compression Scheme

The data compression scheme can be extended to the single layer Galerkin matrix arising in hp-BEM with quasi-uniform meshes and quasi-uniform polynomial degrees. The resulting compression scheme consists of two layers. Each block $\mathbf{V}^{(i,j)}$ corresponding to two boundary elements T_i, T_j is compressed using the techniques described in the previous sections. Then we combine (compressed) sub-blocks using the standard $\mathcal{H}$-Matrix approach applied to the boundary elements. For this, we define the set of the indices of boundary elements, $\mathcal{I}_E := \{1, \ldots, \mathcal{N}_e\}$, and set x_i as the midpoint of the element T_i. Using Algorithm 5.1 we create a cluster tree $\mathbb{T}_E$

for the boundary elements. The clusters, which now contain indices of boundary elements, are denoted by σ_E and τ_E. Furthermore, $\mathbf{V}|_{\sigma_E \times \tau_E}$ is the sub-block of $\mathbf{V}$ that consists of all blocks $\mathbf{V}^{(i,j)}$ for $(i,j) \in \sigma_E \times \tau_E$.

Using the admissibility condition (5.1) and Algorithm 5.2, we create a hierarchical splitting of the Galerkin matrix $\mathbf{V}$ into blocks $\mathbf{V}|_{\sigma_E \times \tau_E}$. Blocks corresponding to inadmissible pairs of clusters (σ_E, τ_E) are built using the corresponding compression scheme from Section 5.3, 5.4, and 5.5, respectively. For sub-blocks corresponding to admissible pairs of clusters, we first neglect small entries as described in Section 5.4. However, we only use the a priori compression (Step 1-3 of Algorithm 5.7). Thus, for a $(p_i+1) \times (p_j+1)$ sub-block $\mathbf{V}^{(i,j)} = (V_{k,\ell}^{(i,j)})$ of a block $\mathbf{V}|_{\sigma_E \times \tau_E}$, we now only have to compute $\widetilde{\mathbf{V}}^{i,j} = (V_{k,\ell}^{(i,j)})_{k=0,\dots,p_i^0,\ell=0,\dots,p_j^0}$, where p_i^0, p_j^0 are of the order $\mathcal{O}(\log(\overline{p}))$. Instead of computing these entries directly, we approximate them by interpolating the kernel function $\log|x-y|$ with respect to x as shown in (5.3). We use this procedure for all matrix entries of all sub-blocks $\mathbf{V}^{(i,j)}$ of a block $\mathbf{V}|_{\sigma_E \times \tau_E}$, where we choose the same interpolation order ν for each sub-block $\mathbf{V}^{(i,j)}$. This results in a low rank approximation of the form $\mathbf{V}|_{\sigma_E \times \tau_E} = \mathbf{A}\mathbf{B}^T$ with

$$\mathbf{A} \in \mathbb{R}^{n \times \nu}, \quad n = \sum_{i \in \sigma_E} p_i^0 = \mathcal{O}(|\sigma| \log(\overline{p})), \tag{5.38a}$$

$$\mathbf{B} \in \mathbb{R}^{m \times \nu}, \quad m = \sum_{j \in \tau_E} p_j^0 = \mathcal{O}(|\tau| \log(\overline{p})). \tag{5.38b}$$

Example 5.6.1 illustrates the compression scheme.

Example 5.6.1. We consider Symm's integral equation on $\Gamma = [-1,1] \times \{0\}$. For the discretization, we use a uniform triangulation $\mathcal{T}$ of Γ with $\mathcal{N}_e = 32$ elements. Furthermore, we use a uniform polynomial degree vector $p_i = p = 18$, $i = 1, \dots \mathcal{N}_e$. The compressed Galerkin matrix is shown on Figure 5.6. The diagonal blocks (size 19×19) are displayed in red and blue, the off-diagonal blocks are displayed in black, and the sparse blocks are green. The admissible blocks with respect to the admissibility condition (5.1) are displayed in magenta. The sizes of the matrices $\mathbf{A}$ and $\mathbf{B}$ in (5.38) are indicated by the magenta bars.

As compared to the compression techniques introduced in Sections 5.3, 5.4, and 5.5, the only new approximation error we introduce is the interpolation error for the interpolation of $\log|x-y|$ on admissible far field blocks. In order to be able to use the results of Section 5.2, we derive an upper bound for the absolute error of the matrix entries. Recall that we use the approximation

$$\widetilde{\kappa} := \begin{cases} I_{x,\sigma}^{\nu} \kappa, & \text{if } \operatorname{diam}(B_\sigma) < \operatorname{diam}(B_\tau), \\ I_{y,\tau}^{\nu} \kappa, & \text{otherwise.} \end{cases} \tag{5.39}$$

Using Equation (5.7), this yields

$$\|\kappa - \widetilde{\kappa}\|_{\infty, B_\sigma \times B_\tau} \leq 16\, e\, \frac{(\nu+1)^3}{D_{\sigma,\tau}} (1+\eta) \left(1 + \frac{2}{\eta}\right)^{-(\nu+1)} =: E(\sigma, \tau, \eta, \nu).$$

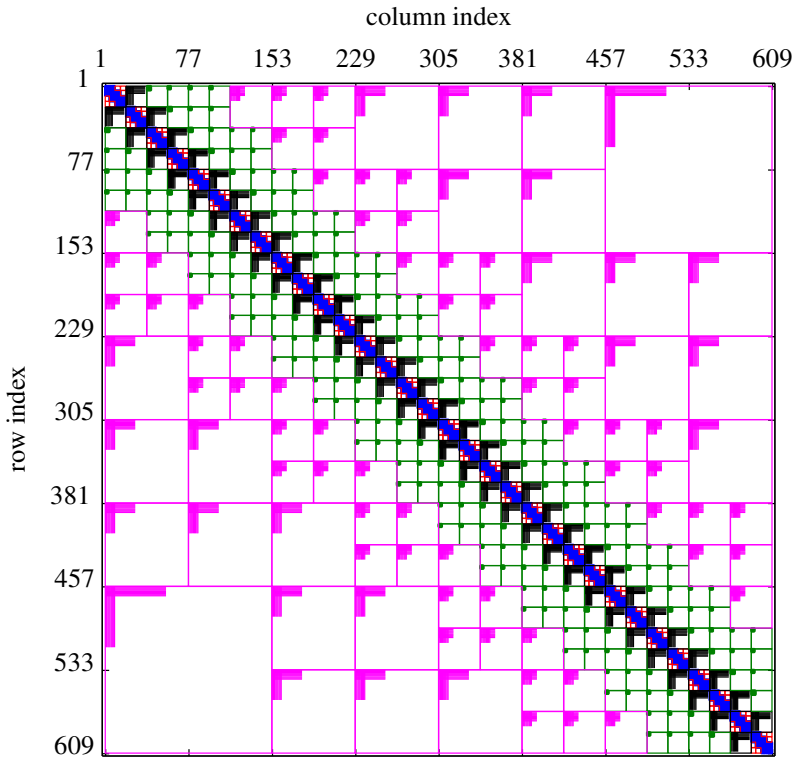


Figure 5.6.: Example of the matrix splitting for the Galerkin matrix $\mathbf{V}$ described in Example 5.6.1.

Let (σ_E, τ_E) be a pair of admissible clusters with respect to the admissibility condition (5.1) for some $\eta > 0$. We denote the basis functions whose indices f belong to σ_E by φ_f and the basis functions whose indices f' belong to τ by $\varphi_{f'}$. Note that there holds $\varphi_f = P_k^{(i)}$ for some boundary element T_i with $i \in \sigma_E$ and $0 \leq k \leq p_i^0$, and similarly $\varphi_{f'} = P_\ell^{(j)}$ for some boundary element T_j, with $j \in \tau_E$ and $0 \leq \ell \leq p_j^0$. For the absolute error in the matrix entries, there holds

$$\begin{aligned} |V_{f,f'} - \widetilde{V}_{f,f'}| &= \left| \int_\Gamma \int_\Gamma \varphi_f(x) \kappa(k,y) \varphi_{f'} \, ds_y \, ds_x - \int_\Gamma \int_\Gamma \varphi_f(x) \widetilde{\kappa}(k,y) \varphi_{f'} \, ds_y \, ds_x \right| \\ &\leq \int_\Gamma \int_\Gamma |\varphi_f(x)| \cdot |\kappa(k,y) - \widetilde{\kappa}(k,y)| \cdot |\varphi_{f'}| \, ds_y \, ds_x \\ &\leq \int_\Gamma |\varphi_f(x)| \, ds_x \int_\Gamma |\varphi_{f'}| \, ds_y \, E(\sigma, \tau, , \eta, \nu). \end{aligned} \tag{5.40}$$

Furthermore, using the Cauchy Schwarz inequality and the orthogonality of the Legendre poly-

nomials, we obtain

$$\begin{aligned}\int_\Gamma |\varphi_f(x)|\, ds_x &= \int_{T_i} |P_k^{(i)}(x)|\, ds_x = \frac{h_i}{2}\int_{-1}^{1} |P_k(t)|\, dt \\ &\leq \frac{h_i}{2}\left(\int_{-1}^{1} 1^2\, dt \cdot \int_{-1}^{1} (P_k(t))^2\, dt\right)^{1/2} \\ &= \frac{h_i}{\sqrt{2k+1}} \leq \frac{\overline{h}}{\sqrt{2}}.\end{aligned}$$

Plugging this into (5.40) yields

$$|V_{f,f'} - \widetilde{V}_{f,f'}| \leq \frac{\overline{h}^2}{2} E(\sigma, \tau, , \eta, \nu).$$

Combining this with estimate (5.17), we obtain that in order to keep the consistency error in the same order as the best approximation error we have to choose the interpolation order ν such that

$$\frac{\overline{h}^2}{2} E(\sigma, \tau, , \eta, \nu) \lesssim \frac{\underline{h}^{3+s}}{\overline{p}^{4+2s}},$$

which is equivalent to

$$\frac{(\nu+1)^3}{D_{\sigma,\tau}}(1+\eta)\left(1+\frac{2}{\eta}\right)^{-(\nu+1)} \lesssim \frac{\underline{h}^{1+s}}{\overline{p}^{4+2s}}.$$

This shows that the interpolation order ν can be chosen as $\mathcal{O}(\log(\mathcal{N}))$, where $\mathcal{N}$ is the dimension of the underlying ansatz space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$. Note that since we have a quasi-uniform grid, there holds $\overline{p}/\underline{h} \lesssim \mathcal{N}$.

Finally, we give estimates for the storage requirement and the complexity of the multiplication of the compressed matrix with a vector.

Proposition 5.6.2. *We consider the discrete space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$ with $\dim(S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})) = \mathcal{N}$. Let $\mathbb{T}_E$ be a cluster tree for the indices of the boundary elements $T_i \in \mathcal{T}$ with leaf size $C_{leaf,E}$. Let $C_{sp,E}$ be the sparsity constant for the block partitioning created with $\mathbb{T}_E$. For the storage requirement N_{st} of a compressed $\mathcal{N}\times\mathcal{N}$ matrix $\widetilde{\mathbf{V}}$ with respect to the discrete space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, there holds*

$$N_{st} \lesssim C_{sp,E}\ \max\{C_{leaf,E}, \log(\overline{p})\}\ \operatorname{depth}(\mathbb{T}_E)\ \mathcal{N}_e\ \log(\mathcal{N})^2 + \mathcal{N}\ \log(\mathcal{N})^2. \tag{5.41}$$

Proof. The proof builds upon the proof of [GBH06, Lemma 7.6] in which the complexity estimates for $\mathcal{H}$-Matrices are shown. The storage requirement for one sub-block corresponding to an admissible pair of clusters (σ_E, τ_E) is bounded by

$$N_{\sigma_E,\tau_E,adm} \leq \nu^2\ \log(\overline{p})\ (|\sigma_E| + |\tau_E|) \lesssim \log(\mathcal{N})^2\ \log(\overline{p})\ (|\sigma| + |\tau|)$$

(see Equation (5.38)). For an inadmissible pair of clusters (σ_E, τ_E), the corresponding sub-block $\mathbf{V}_{\sigma_E \times \tau_E}$ consists of either diagonal, off-diagonal, or sparse blocks $\mathbf{V}^{(i,j)}$ ($i \in \sigma_E$, $j \in \tau_E$). For a diagonal block or an off-diagonal block, the storage requirement is of the order $\mathcal{O}(\overline{p}\ \log(\mathcal{N})^2)$ (see Equations (5.26) and (5.37)). Note that we use $\overline{p}/\underline{h} = \mathcal{O}(\mathcal{N})$ for quasi-uniform meshes with quasi-uniform polynomial degree vectors. Thus, all diagonal and off-diagonal blocks together have a storage requirement that is of the order $\mathcal{O}(\mathcal{N}_e\ \overline{p}\ \log(\mathcal{N})^2) = \mathcal{O}(\mathcal{N}\ \log(\mathcal{N})^2)$. For a sparse block, the storage requirement is of the order $\mathcal{O}(\log(\mathcal{N})^2)$ (see Equation (5.30)). Using these estimates, for the overall storage requirement we obtain

$$\begin{aligned} N_{st} &\lesssim \sum_{(\sigma_E,\tau_E)\in\mathbb{P}_{adm}} \log(\mathcal{N})^2\ \log(\overline{p})\ (|\sigma_E| + |\tau_E|) + \\ &\qquad \sum_{(\sigma_E,\tau_E)\in\mathbb{P}_{inadm}} \log(\mathcal{N})^2 C_{leaf,E}(|\sigma_E| + |\tau_E|) + \mathcal{N}\ \log(\mathcal{N})^2 \\ &\lesssim \max\{C_{leaf,E}, \log(\overline{p})\}\ \log(\mathcal{N})^2 \sum_{(\sigma_E,\tau_E)\in\mathbb{P}} (|\sigma_E| + |\tau_E|) + \mathcal{N}\ \log(\mathcal{N})^2. \end{aligned}$$

Finally, from [GBH06, Lemma 7.4] we get that

$$\sum_{(\sigma_E,\tau_E)\in\mathbb{P}} (|\sigma_E| + |\tau_E|) \leq 2\ C_{sp,E}\ (\mathrm{depth}(\mathbb{T}_E) + 1)\ \mathcal{N}_e,$$

where $C_{sp,E}$ is the sparsity constant defined in Proposition 5.1.5. This yields

$$N_{st} \lesssim C_{sp,E}\ \max\{C_{leaf,E}, \log(\overline{p})\}\ \mathrm{depth}(\mathbb{T}_E)\ \mathcal{N}_e\ \log(\mathcal{N})^2 + \mathcal{N}\ \log(\mathcal{N})^2.$$

□

Remark 5.6.3. (i) For quasi-uniform meshes one obtains balanced cluster trees $\mathbb{T}_E$. In this case there holds $\mathrm{depth}(\mathbb{T}_E) = \mathcal{O}(\log(\mathcal{N}_e))$, and the storage requirement is bounded by

$$N_{st} \lesssim \log(\mathcal{N})^2 \Big(\mathcal{N} + C \log(\overline{p})\ \cdot \mathcal{N}_e \log(\mathcal{N}_e) \Big), \tag{5.42}$$

where C depends on the sparsity constant $C_{sp,E}$ and the leaf size $C_{leaf,E}$.

(ii) If either the mesh size $\mathbf{h}$ or the polynomial degree vector $\mathbf{p}$ are held constant (uniform h-BEM or uniform p-BEM), the storage requirement N_{st} is of the order $\mathcal{O}(\mathcal{N}\ \log(\mathcal{N})^3)$.

(iii) The complexity for the multiplication of the compressed matrix with a vector, N_{mv} is of the same order as the storage requirement of the compressed matrix. This follows from the fact that for each sub-block $\mathbf{V}^{(i,j)}$, the complexities for storage and matrix vector product are of the same magnitude.

5.7. Implementation and Numerical Experiments

5.7.1. Aspects of Implementation

The implementation of the compression scheme is based on the `C` library `Hlib` (Version 1.3). The `HLib` provides all necessary data structures and functions for

- the storage of a boundary grid,
- the assembly and storage of a cluster tree based on bounding boxes,
- the assembly and storage of an $\mathcal{H}$-Matrix for lowest order two and three dimensional BEM, and
- the matrix vector multiplication for an $\mathcal{H}$-Matrix.

Note that the `HLib` provides much more functionality than the items listed here, which are only the data structures and functions we use for our implementation. We adjust the data structures and corresponding functions that we need for our compression scheme. For example, we implement Algorithm 5.5 and introduce data structures for sparse matrices and low rank approximations of clusters of sparse blocks. The data structures that are used in the implementation are listed in Appendix C.
For the solution of the linear system of equations, we use the conjugate gradient (cg) method [HS52], where we use the diagonal preconditioning.

Remark 5.7.1. (i) In [Heu96] it is shown that the condition number of the Galerkin matrix $\mathbf{V}$ with respect to a basis consisting of scaled Legendre polynomials grows at most like p^2, where p is the polynomial degree of the ansatz space. Our numerical experiments (see Figure 3.2) show that the condition number for the diagonally scaled Galerkin matrix is proportional to p. Since we use a cg method with diagonal preconditioning, the number of iterations in the cg algorithm depends on p. In more detail, if $\kappa(\mathbf{V})$ is the condition number of $\mathbf{V}$, then the convergence rate of the cg method is

$$\frac{\sqrt{\kappa(\mathbf{V})}-1}{\sqrt{\kappa(\mathbf{V})}+1}.$$

Assuming that there holds $\kappa(\mathbf{V}) = \mathcal{O}(p)$, we obtain that the convergence rate is of the order

$$\mathcal{O}\left(\frac{\sqrt{p}-1}{\sqrt{p}+1}\right).$$

This will be visible in the computation times presented in this section.

(ii) One possible preconditioner is the additive Schwarz method, which is analyzed for p-BEM in [Heu96], [HST98], and [TS00]. It is shown that the condition number of the preconditioned system is of the order $\mathcal{O}(\log(1+p)^2)$. Another option is to implement an incomplete LU-decomposition in the compressed matrix format and use that as a preconditioner. For standard $\mathcal{H}$-Matrices, this approach is described for example in [BG06].

Since the sub-blocks $\mathbf{V}^{(i,j)}$ of the Galerkin matrix $\mathbf{V}$ arising in the uniform p-method can be assembled independently from each other, the assembly of the Galerkin matrix can be parallelized. For the parallelization, we take into account that the assembly times for different types of sub-blocks vary strongly. Consider the assembly of the Galerkin matrix for the Slit Problem (i.e. $\Gamma = [-1,1] \times \{0\}$) with a boundary mesh that consists of 16 uniform elements and a uniform polynomial degree vector with $p_i = 255$, $i = 1, \ldots, 16$. For this setup, the time for the assembly of all diagonal blocks is $t_{diag} = 0.009$ seconds, the assembly of all far field blocks requires $t_{far} = 0.057$ seconds, and the assembly time for all off-diagonal blocks is $t_{off} = 2.55$ seconds. Note that we expect large computation times for the off-diagonal blocks, since we have to assemble each full matrix block before it can be compressed. In order to achieve a good load balancing for the parallelization with N_T threads, we distribute the matrix blocks manually among the threads. Algorithm 5.9 is used for the assembly of the compressed Galerkin matrix. Note that we exploit the symmetry of the Galerkin matrix.

Algorithm 5.9 Assembly of the block Galerkin matrix $\mathbf{V} = V^{(i,j)}$.

N_T is the total number of threads, and k represents the thread ID.
Set $cnt = 0$.
for $i = 1, \ldots, \mathcal{N}_e$ **do**
  if $k + 1 == N_T - (i \bmod N_T)$ **then**
    Thread k computes the diagonal block $\mathbf{V}^{(i,i)}$.
  end if
  if $(i \bmod N_T) == k$ **and** $i < \mathcal{N}_e$ **then**
    Thread k computes the off-diagonal block $\mathbf{V}^{(i,i+1)}$.
  end if
  for $j = i + 2, \ldots, \mathcal{N}_e$ **do**
    if $((cnt \bmod N_T) == k)$ **then**
      Thread k computes the far field block $\mathbf{V}^{(i,j)}$.
    end if
    cnt++;
  end for
end for

We test the parallelization by considering the boundary of the Z-shape domain that is discretized into $\mathcal{N}_e$ elements (i.e. $\mathcal{T} = \{T_1, \ldots, T_{\mathcal{N}_e}\}$). Furthermore, on each element we use the same polynomial degree p. For the Galerkin matrix with respect to the ansatz space $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, we have to compute $\mathcal{N}_e$ diagonal blocks, $\mathcal{N}_e$ off-diagonal blocks, and $0.5 \cdot (\mathcal{N}_e - 1) \cdot (\mathcal{N}_e - 2) - 1$ far field blocks. Note that the matrix is symmetric, and we only compute the upper right part. Table 5.2 shows the number of diagonal, off-diagonal, and far field blocks that each thread computes for different numbers of elements and threads. It can be seen that the blocks are distributed as evenly as possible among the threads.

		$\mathcal{N}_e = 9$			$\mathcal{N}_e = 36$		
# Threads	Tid	nD	nR	nS	nD	nR	nS
1	0	9	9	27	36	36	594
2	0	4	5	14	18	18	297
	1	5	4	13	18	18	297
4	0	2	3	7	9	9	148
	1	2	2	7	9	9	149
	2	2	2	7	9	9	148
	3	3	2	6	9	9	148

Table 5.2.: Number of diagonal (nD), off-diagonal (nR), and far field blocks (nS) that each thread computes for different numbers of elements ($\mathcal{N}_e$) and threads. Here, Tid stands for the thread ID.

Table 5.3 shows the computation times for the assembly of the compressed Galerkin matrix for different values of $\mathcal{N}_e$ and p and for different numbers of threads. It can be seen that even though the matrix blocks are distributed almost evenly among the threads, the parallelization scales not optimally. The reason for this is that the computation time per sub-block is not constant for different sub-blocks, even for sub-blocks of the same type (e.g. two off-diagonal blocks). For sub-blocks that correspond to elements which are on one line, all computations only involve real numbers, i.e. we only use the library `MPFR`. If two elements are not on one line, the computation of the entries of the corresponding matrix block requires complex arithmetic. Then we have to use functions from the library `MPC`, which are slower than the corresponding functions from the library `MPFR`. Furthermore, the number of digits used for the computations may vary between different sub-blocks, since it depends on the polynomial degree and the distance of the two corresponding boundary elements.

	$\mathcal{N}_e = 9, p = 497$		$\mathcal{N}_e = 18, p = 173$		$\mathcal{N}_e = 144, p = 23$	
# Threads	$t_i\ [s]$	t_i/t_1	$t_i\ [s]$	t_i/t_1	$t_i\ [s]$	t_i/t_1
1	5.90	1.00	0.95	1.00	5.80	1.00
2	3.50	0.59	0.61	0.60	3.00	0.52
4	1.99	0.34	0.39	0.41	1.70	0.29

Table 5.3.: Computation times for the assembly of the compressed Galerkin matrix for the Z-shape with $\mathcal{N}_e$ elements and polynomial degree p for different numbers of elements and values for p. The variable t_i represents the total computation time for i threads. The computations are performed on a desktop computer with an AMD Phenom II X6 processor, 16GB RAM, and the Ubuntu 10.04 operating system.

In order to be able to use the compression schemes within the epsBEM framework, we implement an interface for MATLAB. Here, we use MATLAB's ability to load shared libraries instead of writing `mex` files for the interface. The advantage of this approach is that the compressed matrix can be kept in the memory and the user is able to work with the compressed matrix (e.g. to perform matrix vector multiplications) from within MATLAB.

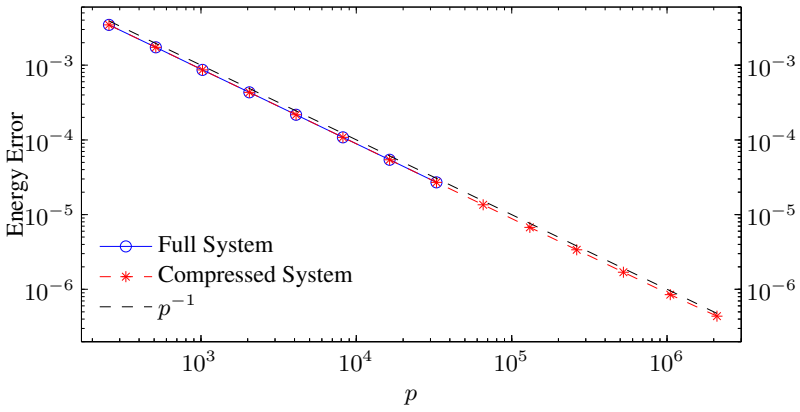
5.7.2. Uniform p-Method

To test the compression scheme for the diagonal blocks, we consider Symm's integral equation $(\mathcal{V}\phi)(x,y) = \log(2)/2$ on the slit $\Gamma = [-1,1] \times \{0\}$. For this problem, the exact solution is $\phi(x,y) = (1-x^2)^{-1/2}$ and the exact energy is $\|\phi\|_{\mathcal{V}}^2 = \log(2)/2 \cdot \pi$ (see Remark 3.6.1). We use a boundary mesh that consists of only one element and perform a uniform p-method, i.e. we increase the polynomial degree. In this case, the Galerkin matrix only consist of one $(p+1)\times(p+1)$ diagonal block, and the right-hand side $\mathbf{b}$ of the linear system can be computed explicitly. Due to the orthogonality of Legendre polynomials, there holds

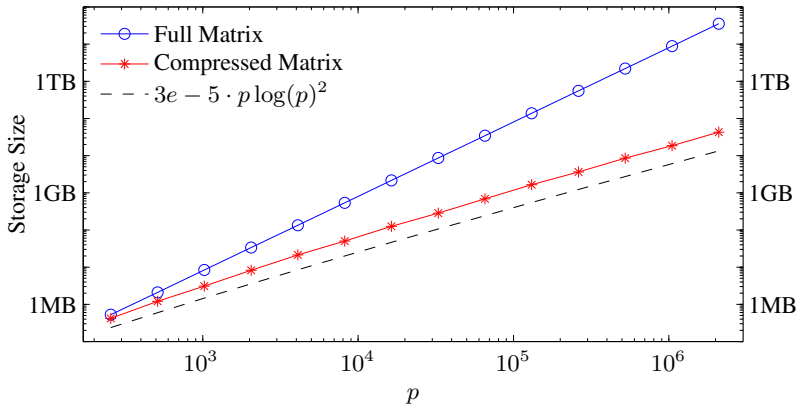
$$b_k = \int_{-1}^{1} \frac{\log(2)}{2} P_k(t)dt = \begin{cases} \log(2), & k = 0 \\ 0, & k > 0. \end{cases}$$

For the compression scheme, which in this case is only the scheme described in Section 5.3, we use the admissibility constant $\eta = 0.5$ and leaf size $C_{leaf} = 32$. Furthermore, we choose the order of the Chebyshev interpolation in (5.21) as $\nu = \lceil 0.6 \log_2(p) \rceil$. The full system is assembled with the corresponding function from the epsBEM package. Figure 5.7(a) shows the energy errors for the full and the compressed system. Recall that energy error can be computed by

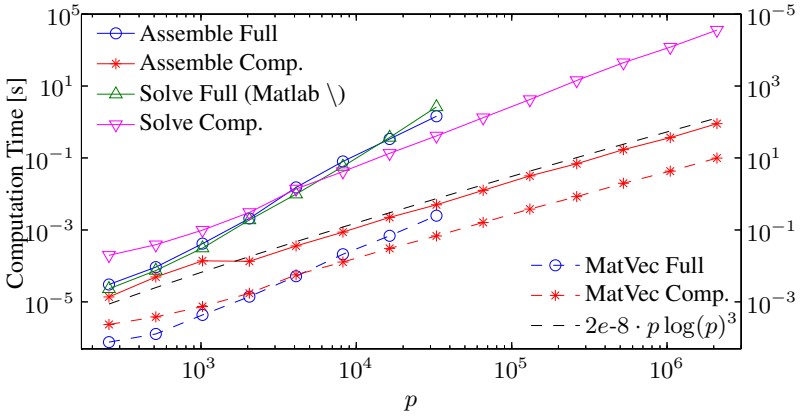
$$\|\phi - \phi_h^p\|_{\mathcal{V}}^2 = \|\phi\|_{\mathcal{V}}^2 - \|\phi_h^p\|_{\mathcal{V}}^2.$$



(a) Energy error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over p for the full and compressed system.



(b) Storage requirement for the full and compressed Galerkin matrix over p. The full Galerkin matrix is only assembled for $p \leq 2^{15}$. All other values are extrapolated assuming that one double precision number uses 8 bytes of memory.



(c) Computation times over p. The computations are performed on a desktop computer with an AMD Phenom II X6 processor, 16GB RAM, and the Ubuntu 10.04 operating system.

Figure 5.7.: Energy error, storage requirement of the Galerkin matrices, and computation times. We solve Symm's integral equation $\mathcal{V}\phi = \log(2)/2$ on the slit $\Gamma = [-1, 1] \times \{0\}$. The boundary mesh consists of one element, and we increase the polynomial degree p of the ansatz space.

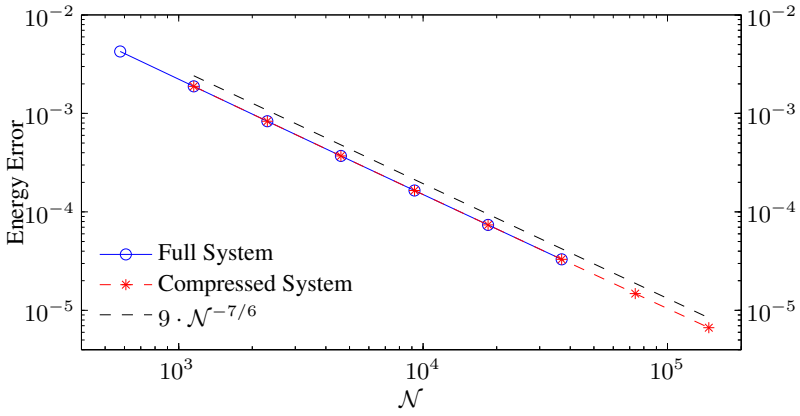
Both errors are of the order p^{-1} as expected, since the solution ϕ is $\phi(x,y) = (1+x)^\alpha(1-x)^\alpha$ with $\alpha = -1/2$ (see Remark 1.5.4). The fact that the error for the compressed system is very close to the error of the full system might lead to the assumption that the interpolation order ν is chosen too large. However, numerical experiments show that the cg algorithm that is used to solve the compressed system no longer converges for $p \geq 2^{16} - 1$ if ν is chosen smaller than $\lceil 0.55 \log_2(p) \rceil$.

The storage sizes for the compressed and full matrix can be seen in Figure 5.7(b). For the compressed system, the storage requirement is proportional to $p \log(p)^2$. This coincides with the theoretical result (5.27). In Figure 5.7(c) computation times over the polynomial degree p are displayed. The time for the assembly of the full matrix grows quadratically; for the compressed system, the assembly time is proportional to $p \log(p)^3$ with a proportionality constant smaller than $2 \cdot 10^{-8}$. The time for the multiplication of the compressed matrix with a vector has the same asymptotic behavior as the assembly time and is about ten times faster than the assembly time. The time for the solution of the compressed system grows faster than the assembly time or the time for the matrix vector multiplication. The reason for this is that we do not have a good preconditioner. Therefore, the number of iterations in the cg method grows with the matrix size (see Remark 5.7.1). For example, for $p = 1023$ the algorithm needs 121 steps, and for $p = 16383$ the algorithm needs 435 steps.

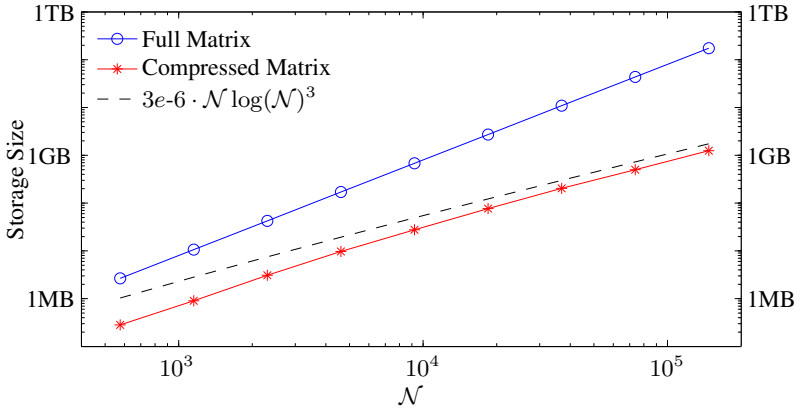
Next we consider Symm's integral equation $\mathcal{V}\phi = 1$ on the boundary of the Z-shape (see Figure 4.1). The exact solution and energy of the solution is not known. Therefore, we use the extrapolated energy $\|\phi\|_{\mathcal{V}}^2 = 5.101700218060357$. The boundary is discretized into 18 boundary elements, and we increase the polynomial degree of the ansatz space. Again, the entries of the right-hand side vector $\mathbf{b}$ of the linear system can be computed explicitly by

$$b_j = \int_{T_i} P_k^{(i)}(t)dt = \begin{cases} h_i, & k = 0 \\ 0, & k > 0. \end{cases}$$

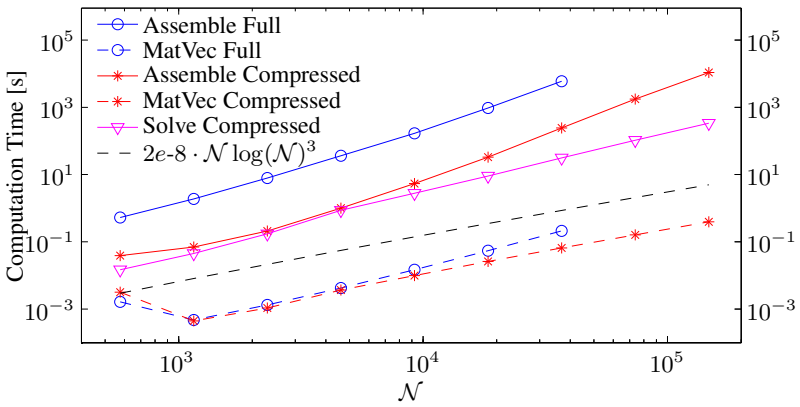
For the compression scheme, we use the admissibility constant $\eta = 0.5$ and leaf size $C_{leaf} = 32$. Furthermore, we choose the order of the Chebyshev interpolation in (5.21) as $\nu = \lceil 0.6 \log_2(p) \rceil$ and use four threads for the computations. The full system is assembled and solved using functions form the epsBEM software package, where the assembly is parallelized with four threads. In Figure 5.8(a) the energy errors for the full and the compressed system can be seen. The errors for the compressed system have the same asymptotic behavior and are of the same magnitude as the errors for the full system. Again, decreasing the interpolation degree on the diagonal blocks results in the solver no longer converging. The storage requirement for the compressed system is proportional to $\mathcal{N} \log(\mathcal{N})^3$ (see Figure 5.8(b)), which confirms the theoretical results. Figure 5.8(c) shows computation times. The computation time for the assembly of the compressed



(a) Energy error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over $\mathcal{N}$ for the full and compressed systems.



(b) Storage requirement for the full and compressed Galerkin matrices over $\mathcal{N}$. The full Galerkin matrix is only assembled for $\mathcal{N} \leq 2^{15}$. All other values are extrapolated assuming that one double precision number uses 8 bytes of memory.



(c) Computation times over the degrees of freedom. The computations are performed on a desktop computer with an AMD Phenom II X6 processor, 16GB RAM, and the Ubuntu 10.04 operating system.

Figure 5.8.: Energy error, storage requirement of the Galerkin matrices, and computation times. We consider Symm's integral equation $\mathcal{V}\phi = 1$ on the boundary of the Z-shape. The boundary mesh consists of 18 elements, and we increase the polynomial degree p of the ansatz space.

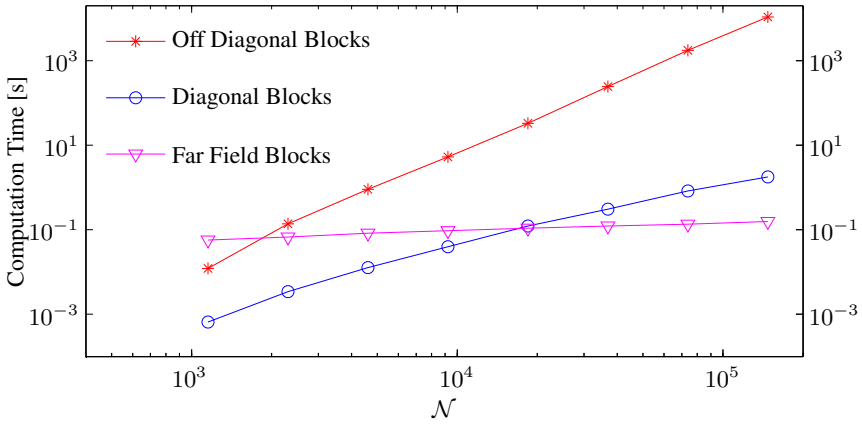


Figure 5.9.: Computation times for all diagonal, off-diagonal, and far field blocks over the number of degrees of freedom $\mathcal{N}$ for the Z-shape with 18 elements and growing polynomial degree p.

matrix has the same asymptotic behavior as the assembly time for the full matrix. The reason for this is that we assemble off-diagonal blocks fully before compressing them. Analyzing the computation times for the assembly of the compressed system more closely (see Figure 5.9), we find that the most computational time is required for the assembly of the off-diagonal blocks. The time for the assembly of all diagonal blocks is proportional to $\mathcal{N}\log(\mathcal{N})^3$, and the time for the assembly of all far field blocks is nearly constant.

5.7.3. Uniform h- and hp-Method

We consider the Z-shape Problem with $\mathcal{N}_e = 9$ elements and an initial polynomial degree distribution given by

$$p = (28, 29, 17, 29, 25, 17, 20, 24, 30)^T .$$

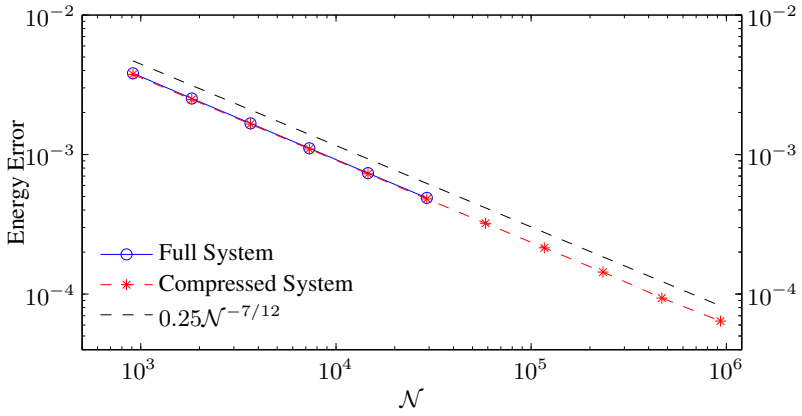
The polynomial degrees are held constant, and we perform a uniform h-method. For the compression scheme, we use $C_{leaf,E} = 4$, $\eta = 0.5$. Diagonal blocks are assembled fully. For the two dimensional interpolation in the far field (see Equation (5.3)), we choose $\nu = \lceil 0.5 \log_2(\mathcal{N}) \rceil$ (recall that all admissible blocks are of the rank ν^2). For the assembly and solution of the full system we use the corresponding functions from the epsBEM package. Figure 5.10 shows energy errors (a), storage sizes (b), and computation times (c) for the full and compressed systems. It is shown that the errors for the compressed system are of the same magnitude and show the same asymptotic behavior as the errors for the full system. Both errors are of the order $\mathcal{O}(\mathcal{N}^{7/12})$, which is half the order we obtained for the uniform p-method for the Z-shape problem (see Figure 5.8(a)). The storage requirement for the compressed system is proportional to $\mathcal{N}\log(\mathcal{N})^3$, which coincides with the theoretical result (see Remark 5.6.3). The times for the assembly and

solution of the compressed system are of the order $\mathcal{O}(\mathcal{N}\log(\mathcal{N})^6)$. The complexity of the matrix vector multiplication with the compressed matrix is of the order $\mathcal{O}(\mathcal{N}\log(\mathcal{N})^3)$, which is what the theoretical estimate (5.41) predicts (see also Remark 5.6.3 iii). As for the uniform p-method, the condition number of the Galerkin matrix grows as $\mathcal{N}$ is increased, and thus the number of iterations in the cg algorithm used for the solution of the compressed system increases. This results in the solution of the linear system having a higher complexity than the matrix vector multiplication.

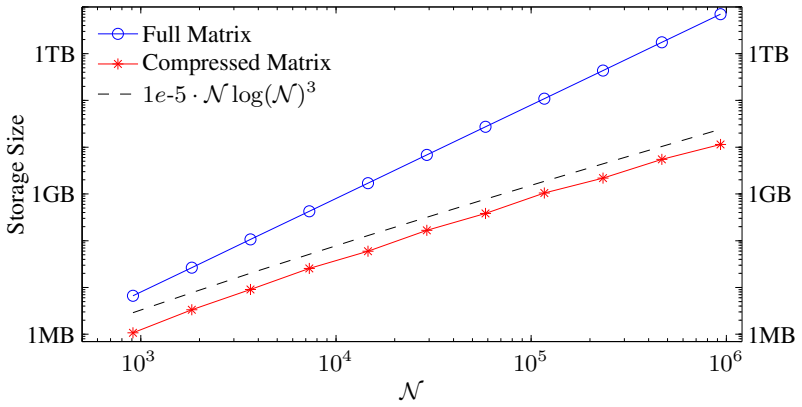
For our last numerical experiment, we consider again Symm's integral equation $(\mathcal{V}\phi)(x,y) = \log(2)/2$ on the rotated slit $\Gamma = \mathrm{conv}(1/\sqrt{2}\,(-1,-1), 1/\sqrt{2}\,(1,1))$. We start with a boundary mesh that consists of 8 elements and constant polynomial degree vector $p_1 = \cdots = p_8 = 7$. In each refinement step we double the polynomial degree on each element and then refine the mesh uniformly. For the compression of the diagonal blocks, we choose $C_{leaf} = 32$, $\eta = 0.5$, and the order for the interpolation scheme is $\nu = \lceil 0.6\,\log_2(p)\rceil$. For the cluster tree with respect to the elements, we choose $C_{leaf,E} = 2$ and $\eta_E = 0.5$, and the interpolation order on admissible far field blocks is chosen to be $\nu_E = \lceil \log_2(0.8\,\mathcal{N})\rceil$.
The results are displayed in Figure 5.11. The energy errors for both the full and compressed systems decay like $\mathcal{N}^{-3/4}$, which matches the theoretical result from Section 1.5; see Figure 5.11(a). The storage requirement for the compressed system and the complexity of the matrix vector multiplication are of the order $\mathcal{N}\log(\mathcal{N})^2$ (see Figure 5.11(b) and (c)), matching the theoretical estimate (5.42). If we set $\overline{p} = \mathcal{N}_e = \sqrt{\mathcal{N}}$, then the estimate (5.42) simplifies to

$$N_{st} \lesssim \mathcal{N}\log(\mathcal{N})^2 + C\sqrt{\mathcal{N}}\log(\mathcal{N})^4,$$

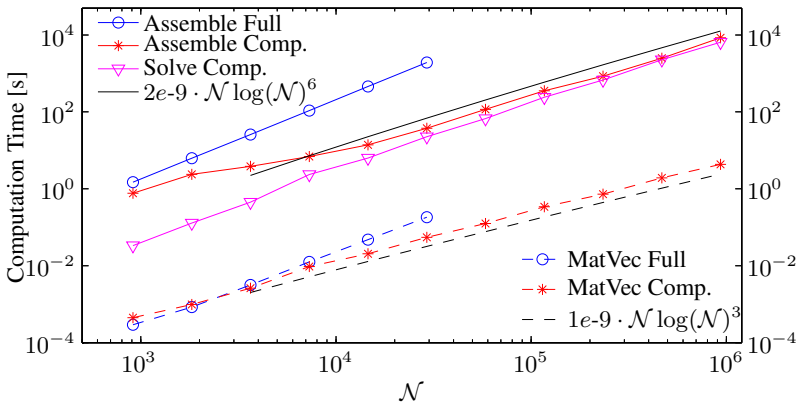
where the constant C depends on the sparsity constant $C_{sp,E}$ and the leaf size $C_{leaf,E}$. The computation time for the assembly of the compressed matrix grows roughly like $\mathcal{N}^{7/4}$, and the time for the solution of the compressed system grows roughly like $\mathcal{N}^{3/4}$.



(a) Energy error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over $\mathcal{N}$ for the full and compressed systems.

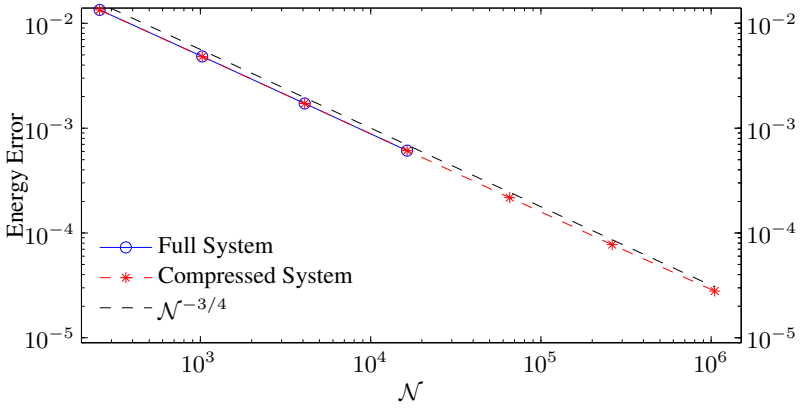


(b) Storage requirement for the full and compressed Galerkin matrices over $\mathcal{N}$. The full Galerkin matrix is only assembled for $\mathcal{N} \leq 2^{15}$. All other values are extrapolated assuming that one double precision number uses 8 bytes of memory.

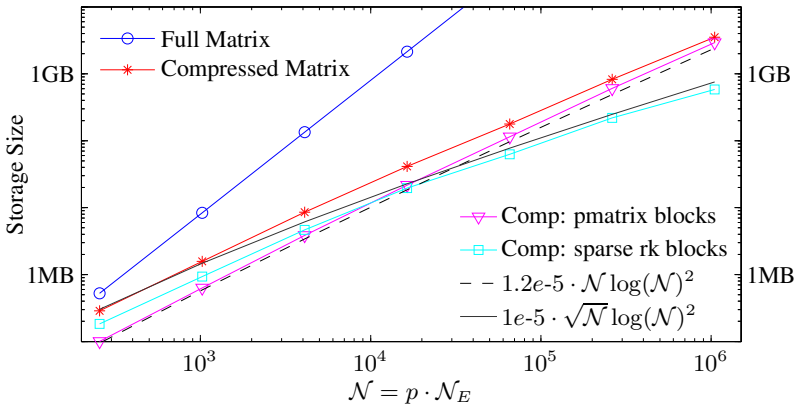


(c) Computation times over $\mathcal{N}$. The computations are performed on a desktop computer with an AMD Phenom II X6 processor, 16GB RAM, and the Ubuntu 10.04 operating system.

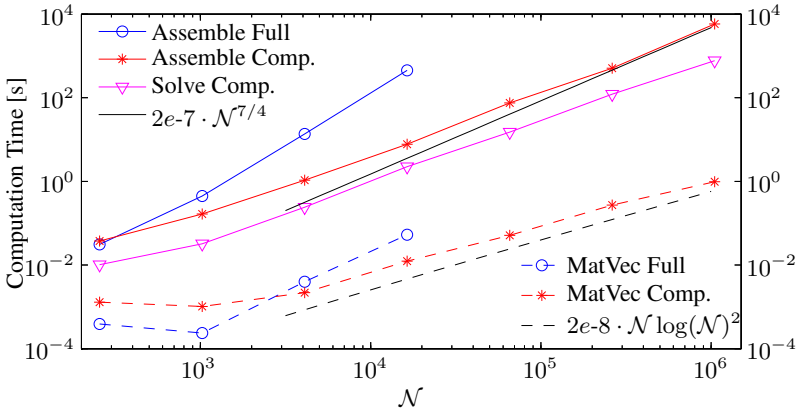
Figure 5.10.: Energy error, storage requirement of the Galerkin matrices, and computation times for a uniform h-method for Symm's integral equation $\mathcal{V}\phi = 1$ on the boundary of the Z-shape. The initial boundary mesh consists of 9 elements, and we use the polynomial degree vector $p = (28, 29, 17, 29, 25, 17, 20, 24, 30)^T$.



(a) Energy error $\|\phi - \phi_h^p\|_{\mathcal{V}}$ over $\mathcal{N}$ for the full and compressed systems.



(b) Storage requirement for the full and compressed Galerkin matrices over $\mathcal{N}$. The full Galerkin matrix is only assembled for $\mathcal{N} \leq 2^{15}$. All other values are extrapolated assuming that one double precision number uses 8 bytes of memory.



(c) Computation times over $\mathcal{N}$. The computations are performed on a desktop computer with an AMD Phenom II X6 processor, 16GB RAM, and the Ubuntu 10.04 operating system.

Figure 5.11.: Energy error, storage requirement of the Galerkin matrices, and computation times. We solve Symm's integral equation $\mathcal{V}\phi = \log(2)/2$ on the slit $\Gamma = [-1, 1] \times \{0\}$. We increase the dimension of the ansatz space by refining the mesh uniformly and doubling the polynomial degree on each boundary element in each step.

Summary and Outlook

In this work we considered three aspects of the hp-boundary element method for the Laplace operator in two dimensions. First, we discussed the efficient implementation of hp-BEM, where we focused on the accurate and efficient evaluation of potentials and computation of Galerkin entries. Second, we examined the generalization of existing a posteriori error estimators and the construction of hp-adaptive algorithms. The last topic we discussed was the analysis and implementation of a data compression scheme for the single layer Galerkin matrix arising in p- and uniform hp-BEM.

The foundation of our stable implementation of hp-BEM is the idea of using Legendre polynomials and their antiderivatives, called Lobatto shape functions, as local bases for the discrete spaces. First, this resulted in Galerkin matrices whose condition numbers grow linearly with respect to the polynomial degree p. Second, we showed that the evaluation of potentials and computation of Galerkin matrices can be done using recurrence relations.
In more detail, we reduced the evaluation of the potentials $\widetilde{\mathcal{V}}$, $\widetilde{\mathcal{K}}$, $\widetilde{\mathcal{K}}'$, and $\widetilde{\mathcal{W}}$ to the evaluation of basic integrals, which we called $\widetilde{Q}_k^m$ and R_k^m. We showed that these integrals are closely related to associated Legendre functions of the second kind. We then derived three-term recurrence relations for the evaluation of these integrals and presented a modified version of Gautschi's continued fraction algorithm for the stable and efficient evaluation of the recurrence relations.
It was also shown that the computation of Galerkin entries can be reduced to basic double integrals, specifically, the integrals of $\widetilde{Q}_k^m$ and R_k^m. For these double integrals, we derived four-term recurrence relations. Since stable backward algorithms for the evaluations of the double integrals are not known for all cases, we evaluated the recurrence relations in the forward direction. In general this approach unstable, so we used multiple precision libraries for the computations. In order to guarantee the accuracy of the computations up to 15 significant digits, we developed an upper bound for the number of digits required for the computations using numerical experiments.
In order to improve the efficiency of our implementation, we used `openMP` for the parallelization of the code. Overall, we obtained an implementation of hp-BEM that produces accurate results even for large polynomial degrees. We presented numerical experiments showing that for hp-methods, the Galerkin errors can be reduced to double machine precision. Furthermore, our implementation produced accurate results for polynomial degrees $p > 1000$. Hereby, the

computational effort was small enough to run the experiments on a desktop computer.
Our implementation of hp-BEM, which is included in the software package epsBEM, is the first published implementation that works for polynomial degrees greater than 50. It opens the door for the practical exploration of many aspects of high order and hp-BEM. We used this implementation as a basis for the construction and analysis of error estimators for hp-BEM and hp-adaptive algorithm as well as for the construction of a data compression scheme for the single layer Galerkin matrix arising in p- and uniform hp-BEM.
This work, where we only considered the Laplace operator in two dimensions, sets the foundation for further research. The approach of reducing the evaluation of potentials and computation of Galerkin entries to basic integrals that are computed using recurrence relations can be extended to other differential operators. In [Ban12] the implementation of hp-BEM for the Lamé operator in two dimensions is discussed. For the Stokes problem in two dimensions, which is closely related to the Lamé problem, our approach can also be generalized. Currently, research is being completed on the extension to Helmholtz problems in two dimensions.
Up until now, it is not clear if and how this approach could be extended to 3D-BEM. A possible approach would be to define orthogonal polynomials on triangles or quadrilaterals and then derive recurrence relations for these polynomials.

The second topic we discussed in this work was the generalization of error estimators introduced for lowest order BEM to estimators that can be used in p- and hp-BEM. We proved that a generalization of $h - h/2$ estimators from [FLP08] using standard L^2 norms and L^2 projections does not result in error estimators that are efficient and reliable in p-BEM. As an alternative, we proposed error estimators that use weighted L^2 norms and weighted L^2 projections. We numerically analyzed the inverse and approximation inequalities needed to prove the efficiency and reliability of the proposed error estimators using eigenvalue problems. Our results indicated that the choice of the weight function is appropriate and the introduced estimators are efficient and reliable. Using one of the introduced estimators, namely the estimator $\widetilde{\mu}^{\omega}$, we constructed two hp-adaptive algorithms. Using the concept of estimator reduction from [FLOP10], we proved that both algorithms produce a sequence of estimators that converges to zero. This means that the adaptive algorithms converge. Numerical experiments showed that this convergence is, in fact, exponential.
Further research will be performed on this topic. A theoretical proof of the inverse and approximation inequalities with weighted L^2 norms and weighted projections is desired. Furthermore, in this work we only considered Symm's integral equation $\mathcal{V}\phi = f$, i.e. estimates between the spaces $H^{-1/2}(\Gamma)$ and $L^2(\Gamma)$. If approximation results for the weighted projections and inverse inequalities between $H^{1/2}(\Gamma)$ and $L^2(\Gamma)$ could be shown, we could generalize the $h - h/2$ estimators for the hypersingular integral equation from [EFGP12]. The approximation estimates are also needed for the introduction of data oscillations, which give an estimate for the approx-

imation errors on the right-hand side of Symm's integral equation $\mathcal{V}\phi = (\mathcal{K} + 1/2)\psi$ and the hypersingular equation $\mathcal{W}\psi = (1/2 - \mathcal{K}')\phi$. A combination of error indicators and data oscillations could be used to steer hp-adaptive algorithms for the two integral equations.

Lastly, we described in detail a new data compression scheme for the single layer Galerkin matrix arising in p- and hp-BEM. The compression scheme uses different compression techniques for different sub-blocks of the Galerkin matrix.
For diagonal blocks, i.e. blocks corresponding to identical elements, we constructed an $\mathcal{H}$-Matrix approximation by splitting the blocks hierarchically into sub-blocks and creating low rank approximations on certain sub-blocks. The low rank approximations were created by interpolating matrix entries with respect to their indices. It was shown that the compressed $p \times p$ blocks have a storage requirement proportional to $p\log(p)^2$ and the complexity of the matrix vector multiplication is of the same order.
For off-diagonal blocks, which correspond to neighboring elements, we used cross approximation to obtain a low rank approximation of the respective block. We performed numerical experiments showing that the ranks needed for the low rank approximation of a $p \times p$ block are of the order $\mathcal{O}(\log(p)^2)$. This means that the storage requirement for one off-diagonal block and the complexity of the matrix vector multiplication are proportional to $p\log(p)^2$.
Blocks corresponding to disjoint elements, called far field blocks, were approximated by sparse matrices. Using asymptotic estimates for Legendre functions, we were able to derive an upper bound for the magnitude of matrix entries. This bound was used to determine a priori which matrix entries are small and thus can be neglected. It was proven that only $\mathcal{O}(\log(p)^2)$ matrix entries of a $p \times p$ matrix block need to be computed and stored.
For the Galerkin matrix arising in quasi uniform hp-BEM, i.e. the case where the we deal with large polynomial degrees and fine meshes, we extended the $\mathcal{H}$-Matrix framework to compute and store low rank approximations of clusters of far field blocks. We showed that our compression scheme has complexities similar to the ones of $\mathcal{H}$-Matrices, i.e. the amount of storage needed for the compressed matrix and the cost of the matrix vector multiplication are proportional to $\mathcal{N}\log(\mathcal{N})^3$.
Whereas the complexities for the assembly of compressed diagonal and far field blocks are of the order $\mathcal{O}(p\log(p)^2)$ and $\mathcal{O}(\log(p)^2)$, respectively, the complexity for the construction of low rank approximations for off-diagonal blocks is quadratic in p. This results from needing to assemble the full matrix block before it can be compressed. Here, further research should be performed. If more theoretical information about the structure of off-diagonal blocks or even an analytic formula for the computation of matrix entries could be obtained, we could use an $\mathcal{H}$-Matrix approach for the compression of an off-diagonal block as opposed to cross approximation for the full block.
Using the relation $\langle \mathcal{W}\psi_1, \psi_2\rangle = \langle \mathcal{V}d\psi_1/ds, d\psi_2/ds\rangle$ ($\psi_1, \psi_2 \in H^{1/2}(\Gamma)$) between the sin-

gle layer and the hypersingular operator, the compression scheme can be generalized for the Galerkin matrix of the hypersingular operator in a direct way.

A. Formulas for the Calculation of Initial Values for $\widetilde{Q}_k^m$ and R_k^m

In this section we give a detailed description on how the initial values for the recurrence relations for $\widetilde{Q}_k^m(z)$ and $R_k^m(z)$ are implemented. We use the notation

$$\begin{aligned}\log(z+1)-\log(z-1) &= \log(|z+1|)-\log(|z-1)+\mathrm{i}\,(\arg(z+1)-\arg(z-1))\\ &=: W+\mathrm{i}\gamma.\end{aligned}$$

The Integrals $\widetilde{Q}_k^{-1}(z)$

For the initial values $\widetilde{Q}_k^{-1}$, $k\in\{0,1,2\}$ there holds for $z\in\mathbb{C}\setminus\mathbb{R}$:

$$\begin{aligned}\widetilde{Q}_0^{-1}(z) &= (z+1)\log(z+1)-(z-1)\log(z-1)-2\\ &= z(\log(z+1)-\log(z-1))+(\log(z+1)+(\log(z-1)))-2\\ &= (u+iv)(W+i\gamma)+\\ &\quad \frac{1}{2}\log(\underbrace{(u+1)^2+v^2}_{=:b1})+\mathrm{i}\,\underbrace{\arg(z+1)}_{=:a1}+\frac{1}{2}\log(\underbrace{(u-1)^2+v^2}_{=:b2})+\mathrm{i}\,\underbrace{\arg(z-1)}_{=:a2}-2\\ &= uW+\frac{1}{2}\log(b1\cdot b2)-v\gamma-2+\mathrm{i}\,[vW+u\gamma+a1+a2]\,,\end{aligned}$$

$$\begin{aligned}\widetilde{Q}_1^{-1}(z) &= \frac{z^2-1}{2}\,(\log(z+1)-\log(z-1))-z\\ &= \left(\frac{u^2-v^2-1}{2}+\mathrm{i}\,uv\right)(W+\mathrm{i}\,\gamma)-z\\ &= \frac{u^2-v^2-1}{2}W-uv\,\gamma-u+\mathrm{i}\left[\frac{u^2-v^2-1}{2}\gamma+\frac{uv}{2}W-v\right],\end{aligned}$$

$$\widetilde{Q}_2^{-1}(z) = z\widetilde{Q}_1^{-1}(z)+\frac{2}{3}.$$

A. Formulas for the Calculation of Initial Values for $\widetilde{Q}_k^m$ and R_k^m

For $u \in \mathbb{R}$, we obtain

$$\begin{aligned}
\widetilde{Q}_0^{-1}(u) &= u \log\left(\frac{|u+1|}{|u-1|}\right) + \log(|u+1| \cdot |u-1|) - 2, \\
\widetilde{Q}_1^{-1}(u) &= \frac{u^2-1}{2} \log\left(\frac{|u+1|}{|u-1|}\right) - u, \\
\widetilde{Q}_2^{-1}(u) &= u\widetilde{Q}_1^{-1}(u) + \frac{2}{3}.
\end{aligned}$$

Note that we take the absolute values in the logarithms such that the values of the integrals are real. The imaginary parts are set to zero.

The Integrals $\widetilde{Q}_k^0(z)$

For $z \in \mathbb{C} \setminus \mathbb{R}$, the initial values of $\widetilde{Q}_k^0(z)$ are given by

$$\begin{aligned}
\widetilde{Q}_0^0(z) &= \log(z+1) - \log(z-1) = W + \mathrm{i}\,\gamma, \\
\widetilde{Q}_1^0(z) &= z\widetilde{Q}_0^0(z) - 2 = u\,W - v\,\gamma - 2 + \mathrm{i}(u\,\gamma + v\,W).
\end{aligned}$$

For $z = u \in \mathbb{R} \setminus \{-1, 1\}$, we calculate

$$\begin{aligned}
\widetilde{Q}_0^0(z) &= \log(|u+1|) - \log(|u-1|) = \log\left(\frac{|u+1|}{|u-1|}\right), \\
\widetilde{Q}_1^0(z) &= z\widetilde{Q}_0^0(z) - 2.
\end{aligned}$$

For $z \in \{-1, 1\}$ the values of the integrals $\widetilde{Q}_0^0(z)$ and $\widetilde{Q}_1^0(z)$ are infinite. Thus, in the `C`-program these values are set to zero.

The Integrals $\widetilde{Q}_k^1(z)$

For $z \in \mathbb{C} \setminus \mathbb{R}$, the initial values of $\widetilde{Q}_k^0(z)$ are given by

$$\begin{aligned}
\widetilde{Q}_0^1(z) &= \frac{2}{z^2-1} \\
&= \frac{2}{u^2-v^2-1+\mathrm{i}\,2uv} = 2\frac{u^2-v^2-1-\mathrm{i}\,2uv}{(u^2-v^2-1)^2+4u^2v^2} \\
&= ... \\
&= 2\frac{u^2-v^2-1}{(u^2+v^2)^2+2(v^2-u^2)+1} + \mathrm{i}\,\frac{-4uv}{(u^2+v^2)^2+2(v^2-u^2)+1} \\
&=: A + \mathrm{i}\,B,
\end{aligned}$$

$$\begin{aligned}
\widetilde{Q}_1^1(z) &= \frac{2z}{z^2-1} - (\log(z+1) - \log(z-1)) \\
&= z\widetilde{Q}_0^1(z) - (W + \mathrm{i}\,\gamma) \\
&= u\,A - v\,B - W + \mathrm{i}\Big(u\,B + v\,A - \gamma\Big),
\end{aligned}$$

For $z = u \in \mathbb{R} \setminus \{-1, 1\}$, we calculate

$$\begin{aligned}
\widetilde{Q}_0^0(z) &= \frac{2}{u^2-1} =: A, \\
\widetilde{Q}_0^1(z) &= \frac{2u}{u^2-1} - \log(|u+1|) + \log(|u-1|) = u \cdot A - \log\left(\frac{|u+1|}{|u-1|}\right).
\end{aligned}$$

For $z \in \{-1, 1\}$, the values of the integral are infinite. In the C-program, we set the values to 0.

A. Formulas for the Calculation of Initial Values for $\widetilde{Q}_k^m$ and R_k^m

The Integrals $R_k^0(z)$

The initial values of $R_k^0(z)$ are for $z \in \mathbb{C}$

$$
\begin{aligned}
R_1^0(z) &= \frac{1}{2}\left(\widetilde{Q}_0^0 - \widetilde{Q}_1^0\right) = \frac{1}{2}\left(\widetilde{Q}_0^0 - z\widetilde{Q}_0^0 + 2\right) = \frac{1}{2}(-(z-1)\widetilde{Q}_0^0 + 2) \\
&= \frac{1}{2}\left(-(u-1+\mathrm{i}v)\left[W + \mathrm{i}\,\gamma\right] + 2\right) \\
&= \frac{1}{2}\left(-(u-1)W + v\,\gamma + 2\right) + \mathrm{i}\frac{1}{2}\left(-v\,W - (u-1)\gamma\right), \\
R_2^0(z) &= \frac{1}{2}\left(\widetilde{Q}_0^0 + \widetilde{Q}_1^0\right) = \frac{1}{2}\left(\widetilde{Q}_0^0 + z\widetilde{Q}_0^0 - 2\right) = \frac{1}{2}((z+1)\widetilde{Q}_0^0 - 2) \\
&= \frac{1}{2}\left((u+1+iv)\left[W + \mathrm{i}\,\gamma\right] - 2\right) \\
&= \frac{1}{2}\left((u+1)W - v\,\gamma - 2\right) + \mathrm{i}\frac{1}{2}\left(v\,W + (u+1)\gamma\right), \\
R_3^0(z) &= \frac{1}{2}\left(\widetilde{Q}_2^0 - \widetilde{Q}_0^0\right) = \frac{z^2-1}{2}\widetilde{Q}_0^0 - z \\
&= \frac{u^2 - v^2 - 1 + \mathrm{i}\,2uv}{2}\left[W + \mathrm{i}\,\gamma\right] - u - \mathrm{i}\,v \\
&= \frac{u^2 - v^2 - 1}{2}W - uv\,\gamma - u + \mathrm{i}\left(\frac{u^2 - v^2 - 1}{2}\gamma + uv\,W - v\right) \\
R_4^0(z) &= zR_3^0(z) + \frac{2}{3} = u \cdot Re(R_3^0(z)) - v \cdot \mathrm{Im}(R_3^0(z)) + \frac{2}{3} + \mathrm{i}\left(u \cdot \mathrm{Im}(R_3^0(z)) + v \cdot \mathrm{Re}(R_3^0(z))\right).
\end{aligned}
$$

For $z = u \in \mathbb{R} \setminus \{-1, 1\}$, we have

$$
\begin{aligned}
R_1^0(u) &= \frac{1}{2}(-(u-1)\widetilde{Q}_0^0(u) + 2) = \frac{1}{2}(-(u-1)\log\left(\frac{|u+1|}{|u-1|}\right) + 2), \\
R_2^0(u) &= \frac{1}{2}((u+1)\widetilde{Q}_0^0(u) - 2) = \frac{1}{2}((u+1)\log\left(\frac{|u+1|}{|u-1|}\right) + 2), \\
R_3^0(u) &= \frac{u^2-1}{2}\widetilde{Q}_0^0(u) - u = \frac{u^2-1}{2}\log\left(\frac{|u+1|}{|u-1|}\right) - u, \\
R_4^0(u) &= u \cdot R_3^0(u)) + \frac{2}{3}.
\end{aligned}
$$

Finally, for $u \in \{-1, 1\}$, we get

$$
\begin{aligned}
R_1^0(u) &= R_2^0(u) = 1, \\
R_3^0(u) &= -u, \\
R_4^0(u) &= -u^2 + \frac{2}{3} = -\frac{1}{3}.
\end{aligned}
$$

The Integrals $R_k^1(z)$

For $z \in \mathbb{C} \setminus \mathbb{R}$, we get the initial values

$$\begin{aligned}
R_1^1(z) &= \frac{1}{2}\left(\widetilde{Q}_0^1 - \widetilde{Q}_1^1\right) = \frac{1}{2}\left(\log(z+1) - \log(z-1)\right) - \frac{1}{z+1} \\
&= \frac{1}{2}\left(W + \mathrm{i}\,\gamma\right) - \frac{u+1-\mathrm{i}\,v}{(u+1)^2+v^2} \\
&= \frac{1}{2}W - \frac{u+1}{(u+1)^2+v^2} + \mathrm{i}\left(\frac{1}{2}\gamma + \frac{v}{(u+1)^2+v^2}\right), \\
R_2^1(z) &= \frac{1}{2}\left(\widetilde{Q}_0^1 + \widetilde{Q}_1^1\right) = -\frac{1}{2}\left(\log(z+1) - \log(z-1)\right) + \frac{1}{z-1} \\
&= -\frac{1}{2}\left(W + \mathrm{i}\,\gamma\right) + \frac{u-1-\mathrm{i}\,v}{(u-1)^2+v^2} \\
&= -\frac{1}{2}W + \frac{u-1}{(u-1)^2+v^2} - \mathrm{i}\left(\frac{1}{2}\gamma + \frac{v}{(u+1)^2+v^2}\right), \\
R_3^1(z) &= \frac{1}{2}\left(\widetilde{Q}_2^1 - \widetilde{Q}_0^1\right) = -z\left(\log(z+1) - \log(z-1)\right) + 2 \\
&= -(u+\mathrm{i}\,v)\left(W + \mathrm{i}\,\gamma\right) + 2 \\
&= -u\,W + v\,\gamma - \mathrm{i}\left(u\,\gamma + v\,W\right), \\
R_4^1(z) &= \frac{1}{2}(1-3z^2)\left(\log(z+1) - \log(z-1)\right) + 3z \\
&= \frac{1}{2}(1-3(u^2-v^2) - \mathrm{i}\,6uv)\left(W + \mathrm{i}\,\gamma\right) + 3u + \mathrm{i}\,3v \\
&= \frac{1-3(u^2-v^2)}{2}W + 3u(1+v\,\gamma) + \mathrm{i}\left(\frac{1-3(u^2-v^2)}{2}\gamma + 3v(1-W)\right).
\end{aligned}$$

For $z \in \mathbb{R} \setminus \{-1, 1\}$, we calculate

$$\begin{aligned}
R_1^1(u) &= \frac{1}{2}\log\left(\frac{|u+1|}{|u-1|}\right) - \frac{1}{u+1}, \\
R_2^1(u) &= -\frac{1}{2}\log\left(\frac{|u+1|}{|u-1|}\right) + \frac{1}{u-1}, \\
R_3^1(u) &= -u\log\left(\frac{|u+1|}{|u-1|}\right) + 2, \\
R_4^1(u) &= \frac{1}{2}(1-3u^2)\log\left(\frac{|u+1|}{|u-1|}\right) + 3u.
\end{aligned}$$

For $u \in \{-1, 1\}$, we take the Cauchy principal values of the integrals, which leads to

$$\begin{aligned}
R_1^1(1) &= -\tfrac{1}{2}, \quad R_1^1(-1) = 0, \\
R_2^1(1) &= 0, \quad R_2^1(-1) = \frac{1}{2}, \\
R_3^1(1) &= 2, \quad R_3^1(-1) = 2, \\
R_4^1(1) &= 3, \quad R_4^1(-1) = -3.
\end{aligned}$$

B. Evaluation of Potentials

Theorem B.0.2. *For a given boundary mesh $\mathcal{T}$, let $T \in \mathcal{T}$, u and m as shown in Figure 3.1, and $w = m - x$ (in order to simplify the notation, we leave out the index i). Furthermore, let*

$$z := \frac{-u^T w - \mathrm{i}\,(u \times w)}{u^T u} \in \mathbb{C}.$$

Then for the single layer potential/operator, there holds for $k \in \mathbb{N}_0$

$$\begin{aligned}(\widetilde{\mathcal{V}} P_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T P_k^{(T)}(y) \log|x-y|\, ds_y \\ &= -\frac{|u|}{2\pi} \left(\delta_{k0} \log|u|^2 + \operatorname{Re} \widetilde{Q}_k^{-1}(z) \right),\end{aligned}$$

and for the double layer potential/operator, there holds for $k \in \mathbb{N}$

$$\begin{aligned}(\widetilde{\mathcal{K}} N_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T N_k^{(T)}(y) \frac{\partial}{\partial n_y} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \operatorname{Im} R_k^0(z),\end{aligned}$$

with the normal vector n_y (clockwise rotation) with respect to T. Furthermore, for the adjoint double layer potential/operator, there holds for $k \in \mathbb{N}_0$

$$\begin{aligned}(\widetilde{\mathcal{V}}' P_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T P_k^{(T)}(y) \frac{\partial}{\partial n_x} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \left(\frac{u \times n_x}{|u|} \operatorname{Im} \widetilde{Q}_k^0(z) - \frac{u^T n_x}{|u|} \operatorname{Re} \widetilde{Q}_k^0(z) \right)\end{aligned}$$

with a normal vector n_x, which is normal with respect to x (the direction of x). Finally we obtain for the hypersingular potential/operator for $k \in \mathbb{N}$

$$\begin{aligned}(\widetilde{\mathcal{W}} N_k^{(T)})(x) &= -\frac{1}{2\pi} \int_T N_k^{(T)}(y) \frac{\partial}{\partial n_x} \frac{\partial}{\partial n_y} \log|x-y|\, ds_y \\ &= -\frac{1}{2\pi} \left((u \times n_x) \operatorname{Re} R_k^1(z) + (u^T n_x) \operatorname{Im} R_k^1(z)\, dt \right).\end{aligned}$$

Proof. We start with the single layer potential. With the configuration shown in Figure 3.1 we have for $y \in T$ $y = m + tu$, and for $x \in \mathbb{R}^2$ there holds $x - y = -w - tu$, $t \in [-1, 1]$. Plugging

this in yields

$$
\begin{aligned}
&-\frac{1}{2\pi}\int_T P_k^{(T)}(y)\log|x-y|\,ds_y \\
&\quad= -\frac{|u|}{2\pi}\int_{-1}^{1} P_k(t)\log|-w-tu|\,dt = -\frac{|u|}{4\pi}\int_{-1}^{1} P_k(t)\log|-w-tu|^2\,dt \\
&\quad= -\frac{|u|}{4\pi}\int_{-1}^{1} P_k(t)\log\left(u^Tu\left(t^2+2\frac{w^Tu}{u^Tu}t+\frac{w^Tw}{u^Tu}\right)\right)\,dt \\
&\quad= -\frac{|u|}{4\pi}\left[\int_{-1}^{1} P_k(t)\log(u^Tu)\,dt+\int_{-1}^{1} P_k(t)\log\left(t^2+2\frac{w^Tu}{u^Tu}t+\frac{w^Tw}{u^Tu}\right)\,dt\right] \\
&\quad= -\frac{|u|}{4\pi}\left[2\delta_{k,0}\log(u^Tu)+\int_{-1}^{1} P_k(t)\log\left((z_1-t)(z_2-t)\right)\,dt\right],
\end{aligned}
$$

where

$$
\begin{aligned}
z_{1,2} &= -\frac{w^Tu}{u^Tu}\pm\sqrt{\left(\frac{w^Tu}{u^Tu}\right)^2-\frac{w^Tw}{u^Tu}} \\
&= \frac{-w^Tu\pm\sqrt{(w^Tu)^2-w^Tw\,u^Tu}}{u^Tu} = \frac{-w^Tu\pm \mathrm{i}(w\times u)}{u^Tu}.
\end{aligned}
$$

For the last equality, we used the Lagrange identity $|a\times b|^2 = a^Ta\,b^Tb-(a^Tb)^2$ for $a,b\in\mathbb{R}^n$, $n\geq 2$. Note that the two roots $z_{1,2}$ are complex conjugates. With

$$
z := z_1 = \frac{-u^Tw-\mathrm{i}\,(u\times w)}{u^Tu}\in\mathbb{C} \tag{B.1}
$$

we can write $\log\left((z-t)(\bar{z}-t)\right) = 2\,\mathrm{Re}\log(z-t)$. Plugging this in gives

$$
-\frac{1}{2\pi}\int_T P_k^{(T)}(y)\log|x-y|\,ds_y = -\frac{|u|}{2\pi}\left[\delta_{k,0}\log(u^Tu)+\mathrm{Re}\int_{-1}^{1} P_k(t)\log\left(z-t\right)\,dt\right].
$$

To proof the assertion for the double layer operator and its adjoint operator simultaneously, we first define a general normal vector $n\in\mathbb{R}^2$ with $|n|=1$. Then we can derive the two identities

$$
\big(\nabla_y\log|x-y|\big)n = -\frac{(x-y)^Tn}{(x-y)^T(x-y)} = -\big(\nabla_x\log|x-y|\big)n \tag{B.2}
$$

and

$$
n = \frac{u^Tn}{u^Tu}\,u-\frac{u\times n}{u^Tu}\,u_\perp.
$$

Furthermore let $f: T\to\mathbb{R}$ be sufficiently smooth. We remark that we will use this general setting for the following proof and later we will replace n by n_y and n_x, respectively, and f by N_k and P_k, respectively, to get the two specific results for both operators. As before, we use z

from (B.1) and the transformation $y(t) = x + w + tu$. We obtain

$$
\begin{aligned}
&\int_T f(\gamma^{-1}(x))\big(\nabla_y \log|x-y|\big)n\,ds_y \\
&\quad= -\int_T f(\gamma^{-1}(x))\frac{(x-y)^T n}{(x-y)^T(x-y)}\,ds_y = |u|\int_{-1}^1 f(t)\frac{(w+tu)^T n}{(w+tu)^T(w+tu)}\,dt \\
&\quad= \frac{1}{|u|}\int_{-1}^1 f(t)\frac{(w+tu)^T((u^T n)u-(u\times n)\,u_\perp)}{(w+tu)^T(w+tu)}\,dt \\
&\quad= \frac{1}{|u|}\Bigg[(u\times n)\int_{-1}^1 f(t)\frac{u\times w}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt \\
&\qquad\qquad + u^T n\int_{-1}^1 f(t)\frac{(w+tu)^Tu}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt\Bigg] \\
&\quad= \frac{1}{|u|}\Bigg[(u\times n)\int_{-1}^1 f(t)\frac{u\times w}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt \\
&\qquad\qquad + u^T n\int_{-1}^1 f(t)\frac{u^Tu(t-\overline{z})+u^Tu\,\overline{z}+w^Tu}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt\Bigg] \\
&\quad= \frac{1}{|u|}\Bigg[\Big((u\times n)(u\times w)+(u^T n)\,(\overline{z}u+w)^Tu\Big)\int_{-1}^1 \frac{f(t)}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt \\
&\qquad\qquad + u^T n\int_{-1}^1 \frac{f(t)}{t-z}\,dt\Bigg] \\
&\quad= \frac{1}{|u|}\Bigg[\Big((u\times n)+\mathrm{i}(u^T n)\Big)\int_{-1}^1 f(t)\frac{(u\times w)}{t^2u^Tu+2t\,w^Tu+w^Tw}\,dt - u^T n\int_{-1}^1 \frac{f(t)}{z-t}\,dt\Bigg].
\end{aligned}
$$

For the fraction in the first integral of the right-hand side, we use partial fraction expansion to get

$$
\frac{u\times w}{t^2u^Tu+2t\,w^Tu+w^Tw} = -\frac{\mathrm{i}}{2}\frac{1}{z-t}+\frac{\mathrm{i}}{2}\frac{1}{\overline{z}-t}.
$$

Note that

$$
\begin{aligned}
-\frac{\mathrm{i}}{2}\frac{1}{z-t} &= -\frac{\operatorname{Im} z}{2(z-t)(\overline{z}-t)}+\mathrm{i}\frac{t-\operatorname{Re} z}{2(z-t)(\overline{z}-t)},\\
\frac{\mathrm{i}}{2}\frac{1}{\overline{z}-t} &= -\frac{\operatorname{Im} z}{2(z-t)(\overline{z}-t)}-\mathrm{i}\frac{t-\operatorname{Re} z}{2(z-t)(\overline{z}-t)},\\
\frac{1}{z-t} &= -\frac{t-\operatorname{Re} z}{(z-t)(\overline{z}-t)}-\mathrm{i}\frac{\operatorname{Im} z}{(z-t)(\overline{z}-t)}.
\end{aligned}
$$

Using this, we write

$$\begin{aligned}\int_T f(\gamma^{-1}(x))\big(\nabla_y \log|x-y|\big)n\,ds_y \\ &= \frac{1}{|u|}\left[\Big((u\times n)+\mathrm{i}(u^T n)\Big)\operatorname{Im}\int_{-1}^{1}\frac{f(t)}{z-t}\,dt - u^T n\int_{-1}^{1}\frac{f(t)}{z-t}\,dt\right] \\ &= \frac{u\times n}{|u|}\operatorname{Im}\int_{-1}^{1}\frac{f(t)}{z-t}\,dt - \frac{u^T n}{|u|}\operatorname{Re}\int_{-1}^{1}\frac{f(t)}{z-t}\,dt.\end{aligned}$$

For $f = N_k$ and the normal vector $n = n_y$ we get the statement for the double layer potential since n_y is normal with respect to T (clockwise rotation) and thus perpendicular to u and $u \times n_y = -|u|$. With $f = P_k$ and $n = n_x$ we get the assertion for the adjoint double layer operator from the above calculations because of Equation (B.2).

Finally, we prove the statement for the hypersingular operator. The second directional derivative of the kernel reads

$$\frac{\partial}{\partial n_x}\frac{(y-x)^T n_y}{(x-y)^T(x-y)} = 2\frac{n_x^T(x-y)\,(x-y)^T n_y}{((x-y)^T(x-y))^2} - \frac{n_x^T n_y}{(x-y)^T(x-y)}.$$

We substitute $y(t) = x + w + tu$. Together with the property $n_y = u_\perp/|u|$, this gives for the first term of the right-hand side

$$\begin{aligned}n_x^T\frac{(w+tu)(w+tu)^T}{((w+tu)^T(w+tu))^2}n_y &= n_x^T\frac{(w+tu)(w+tu)^T}{((w+tu)^T(w+tu))^2}\frac{u_\perp}{|u|} \\ &= -\frac{u\times w}{|u|}\frac{(w+tu)^T n_x}{((w+tu)^T(w+tu))^2} = -\frac{u\times w}{|u|^3}\frac{(w+tu)^T(u^T n_x\,u + (n_x\times u)u_\perp)}{((w+tu)^T(w+tu))^2} \\ &= -\frac{u\times w}{|u|^3}\left(\frac{w^T(u^T n_x\,u + n_x\times u\,u_\perp)}{((w+tu)^T(w+tu))^2} + \frac{tu^T u\,u^T n_x}{((w+tu)^T(w+tu))^2}\right) \\ &= -\frac{u\times w}{|u|^3}\left(\frac{u^T n_x\,w^T u + (n_x\times u)\,(w\times u)}{(u^T u)^2(t-z)^2(t-\overline{z})^2} + \frac{tu^T u\,u^T n_x^T}{(u^T u)^2(t-z)^2(t-\overline{z})^2}\right) \\ &= -\frac{u\times w}{|u|^5}\left(\frac{w^T n_x}{(t-z)^2(t-\overline{z})^2} + \frac{t\,u^T n_x}{(t-z)^2(t-\overline{z})^2}\right).\end{aligned}$$

The second term can be written in the form

$$\frac{n_x^T n_y(w+tu)^T(w+tu)}{((w+tu)^T(w+tu))^2} = \frac{1}{|u|}\frac{n_x^T u_\perp}{(w+tu)^T(w+tu)} = -\frac{u\times n_x}{|u|^3}\frac{1}{(t-z)(t-\overline{z})}.$$

Using partial fraction expansion, we obtain

$$\int_{-1}^{1}\frac{f(t)}{(t-z)(t-\overline{z})}\,dt = \frac{\mathrm{i}}{2\operatorname{Im}(z)}\left(\int_{-1}^{1}\frac{f(t)}{z-t}\,dt - \int_{-1}^{1}\frac{f(t)}{\overline{z}-t}\,dt\right) = -\frac{1}{\operatorname{Im}(z)}\operatorname{Im}\int_{-1}^{1}\frac{f(t)}{z-t}\,dt,$$

$$\int_{-1}^{1}\frac{f(t)}{(t-z)^2(t-\overline{z})^2}\,dt = -\frac{1}{2\operatorname{Im}(z)^3}\operatorname{Im}\int_{-1}^{1}\frac{f(t)}{z-t}\,dt - \frac{1}{2\operatorname{Im}(z)^2}\operatorname{Re}\int_{-1}^{1}\frac{f(t)}{(z-t)^2}\,dt,$$

$$\begin{aligned}\int_{-1}^{1}\frac{f(t)\,t}{(t-z)^2(t-\overline{z})^2}\,dt &= -\frac{\operatorname{Re}(z)}{2\operatorname{Im}(z)^3}\operatorname{Im}\int_{-1}^{1}\frac{1}{z-t}\,dt \\ &\quad - \frac{\operatorname{Re}(z)}{2\operatorname{Im}(z)^2}\operatorname{Re}\int_{-1}^{1}\frac{1}{(z-t)^2}\,dt + \frac{1}{2\operatorname{Im}(z)}\operatorname{Im}\int_{-1}^{1}\frac{1}{(z-t)^2}\,dt.\end{aligned}$$

Thus one can write the hypersingular operator as

$$
\begin{aligned}
&\int_T N_k(\gamma^{-1}(y))\frac{\partial}{\partial n_x}\frac{\partial}{\partial n_y}\log|x-y|\,ds_y \\
&\quad= |u|\int_{-1}^{1} N_k(t)\left(2n_x^T\frac{(w+tu)(w+tu)^T}{((w+tu)^T(w+tu))^2}n_y - \frac{n_x^T n_y}{(w+tu)^T(w+tu)}\right)dt \\
&\quad= \frac{1}{|u|^2}\int_{-1}^{1} N_k(t)\left(-\frac{2}{|u|^2}\frac{(w^T n_x)(u\times w)}{(t-z)^2(t-\overline{z})^2} - \frac{2}{|u|^2}\frac{t(u^T n_x)(u\times w)}{(t-z)^2(t-\overline{z})^2} + \frac{u\times n_x}{(t-z)(t-\overline{z})}\right)dt \\
&\quad= \left(-\frac{u\times n_x}{u\times w} + w^T n_x\frac{|u|^2}{(u\times w)^2} - \frac{u^T n_x u^T w}{(u\times w)^2}\right)\mathrm{Im}\int_{-1}^{1}\frac{N_k(t)}{z-t}\,dt \\
&\qquad - \left(-w^T n_x\frac{|u|^2}{u\times w} + u^T n_x\frac{u^T w}{u\times w}\right)\mathrm{Re}\int_{-1}^{1}\frac{N_k(t)}{(z-t)^2}\,dt + u^T n_x\,\mathrm{Im}\int_{-1}^{1}\frac{N_k(t)}{(z-t)^2}\,dt \\
&\quad= (u\times n_x)\,\mathrm{Re}\int_{-1}^{1}\frac{N_k(t)}{(z-t)^2}\,dt + (u^T n_x)\,\mathrm{Im}\int_{-1}^{1}\frac{N_k(t)}{(z-t)^2}\,dt,
\end{aligned}
$$

where we used the Lagrangian identity $(w^T n_x)(u^T u) - (u^T n_x)(u^T w) = -(u\times w)(n_x\times u)$ in the last step. □

C. Implementation of the Compression Scheme from Chapter 5

For the implementation of the data compression schemes introduced in Chapter 5, we use and extend the `C` library `HLib` (Version 1.3). In the following we list all data structures that we use and indicate how and where they were modified. All modified or introduced lines are marked by (MB) in a comment. For a detailed description of the `HLib` data structures see e.g. [GBH06].

For a cluster tree we use the following data structures. Here additional variables are introduced such that index set for degrees of freedom (i.e. for basis functions) as well as index sets for boundary elements can be handled. An index set for degrees of freedom is handled by the start index `start` and the size `size` in the data structure `_cluster`. For the index set of elements we use `start_e` for the start index and `size_e` for the number of indices. In the structure `clustertree` we introduce the variables `ndof_e` and `nidx_e` that describe the number of elements.

```
1  struct _cluster
2  {
3    int start;      /* First element of associated index set */
4    int size;       /* Size of associated index set */
5    int start_e;    /* First element of the index set for elements (MB)*/
6    int size_e;     /* Size of index set for elements (MB)*/
7    double *bmin;   /* Bounding box */
8    double *bmax;
9    int d;          /* Spatial dimension of the bounding box */
10   int sons;       /* Number of sons */
11   pcluster *son; /* Array of sons */
12   int name;       /* Index for block clusters */
13 };
14
15 struct _clustertree
16 {
17   int ndof;      /* Number of degrees of freedom */
18   int nidx;      /* Maximal number of indices */
19   int ndof_e;      /* Number of degrees of freedom elements (MB)*/
20   int nidx_e;    /* Maximal number of indices elements (MB)*/
```

```
  int clusters; /* Number of clusters */
  int *dof2idx; /* Maps dof to indices */
  int *idx2dof; /* Maps indices to dof (or -1 if no dof) */
  int *dof2idx_e;   /* Maps dof to indices  for elements (MB)*/
  int *idx2dof_e;   /* Maps indices to dof (or -1 if no dof) for elements (MB)*/
  pcluster root;
};
```

For the storage of a rank approximation of a matrix block, we use the structure `rkmatrix`. As compared to the original structure provided by the `HLib`, we introduce two variables `startrow` and `startcol` describing the location of the low rank block in the overall Galerkin matrix.

```
struct _rkmatrix
{
  int k;        /* maximum rank */
  int kt;       /* actual rank */
  int rows;     /* # rows of block */
  int cols;     /* # columns of block */
  int startrow; /* first row index (MB)*/
  int startcol; /* first column index (MB)*/
  double* a;    /* data matrices */
  double* b;
};
```

To store a full matrix block we use

```
struct _fullmatrix
{
  int rows;  /* # rows of block */
  int cols;  /* # columns of block */
  double* e; /* data */
};
```

A diagonal block is stored in the structure `supermatrix`. Note that this is a recursive data structure. The construction of a `supermatrix` is done using an algorithm similar to Algorithm 5.2, using a cluster tree. For the assembly of the diagonal block we only consider the function f from (5.20) that describes the entries diagonal block without the scaling factor $h^2/2\pi$. Thus, we introduce variables `h2` and `fac` for the storage of this scaling factor.

```
struct _supermatrix
{
  int rows;        /* # rows of block */
  int cols;        /* # columns of block */
  int block_rows;  /* rows of sub-supermatrixes */
  int block_cols;  /* columns of sub-supermatrixes */
  double h2;       /* squared mesh width of the correpsonding element (MB)*/
  double fac;      /* scaling factor (MB) */
  psupermatrix* s; /* array of sub supermatrixes */
```

```
  pfullmatrix f;   /* pointer to a full matrix block */
  prkmatrix r;     /* pointer to a low rank matrix block */
};
```

A sparse matrix block is stored in the following data structure. This data structure does not exist in the `HLib`. Note that we use the compressed row storage format here.

```
struct _sparsematrix
{
  int rows;    /* # rows of block */
  int cols;    /* # columns of block */
  int nnz;     /* # non zero entries */
  int p0rows;  /* polynomial degrees after the a priori compression */
  int p0cols;
  /* data vectors for the storage in the compressed row storage format */
  double *data;
  int *colidx;
  int *rowptr;
};
```

For the storage of the whole compressed Galerkin matrix we introduce the following structures. The whole matrix is stored in the structure `pmatrix`. A `pmatrix` consists of matrix blocks, which are stored in the structure `pblock`. Recall that each block corresponds to one combination of boundary elements $T_i, T_j \in \mathcal{T}$. A `plock` has a pointer to either a `sparsematrix` (if the block is a far field block) , a `supermatrix` (if the block is a diagonal block) or to a `rkmatrix` (if the block is an off diagonal block).

```
struct _pblock
{
    int eli;         /* index of the element T_i */
    int elj;         /* index of the element T_j */
    int starti;      /* first row index of the matrix block */
    int startj;      /* first column index of the matrix block */
    int pi;          /* polynomial degree on T_i */
    int pj;          /* polynomial degree on T_j */
    double hi;       /* mesh width on T_i */
    double hj;       /* mesh width on T_j */
    psparsematrix c; /* pointer to a sparsematrix */
    prkmatrix r;     /* pointer to an rkmatrix */
    psupermatrix s;  /* pointer to a supermatrix */
};
struct _pmatrix
{
    int size;           /* matrix size */
    int nE;             /* number of boundary elements */
    ppblock *blocks;    /* array of matrix block (nE*nE) */
};
```

For uniform hp-methods, we introduce the structure `hprkmatrix` for the storage of low rank approximations far field blocks. This data structure does not exist in the original `HLib` but is an extension of the structure `rkmatrix`.

```
struct _hprkmatrix
{
  int k;          /* maximum rank */
  int kt;         /* actual rank */
  int rows;       /* # rows of block */
  int cols;       /* # columns of block */
  int nzrows;     /* # non zero rows of the block */
  int nzcols;     /* # non zero columns of the block */
  int startRows; /* first row index of the block */
  int startCols; /* first column index of the block */
  int nErows;     /* # row elements in block */
  int nEcols;     /* # column elements in block */
  int* prows;     /* pol.deg. vector for row elements */
  int* pcols;     /* pol.deg. vector for col elements */
  int p0rows;     /* pol.deg for rows after a priori compression */
  int p0cols;     /* pol.deg for columns after a priori compression */
  double* a;      /* data matrices */
  double* b;
};
```

Finally, we generalize the structure `supermatrix` and introduce the structure `hpsupermatrix`. The main difference is that the new structure has a pointer to an `hprkmatrix` instead of an `rkmatrix`.

```
struct _hpsupermatrix
{
  int size;
  int nE;           /* number of elements for supermatrix */
  int rows;         /* # rows of block */
  int cols;         /* # columns of block */
  int block_rows;   /* rows of sub-supermatrixes */
  int block_cols;   /* columns of sub-supermatrixes */

  phpsupermatrix *s; /* array of sub hpsupermatrixes */
  phprkmatrix r;     /* pointer to an hoprkmatrix */
  ppmatrix b;        /* pointer to a pmatrix */
};
```

Bibliography

[AEF+11] M. Aurada, M. Ebner, M. Feischl, S. Ferraz-Leite, P. Goldenits, M. Karkulik, M. Mayr, and D. Praetorius. HILBERT-a Matlab implementation of adaptive 2d-BEM. Asc report 24/2011, TU Vienna, 2011.

[AF03] R.A. Adams and J.J.F. Fournier. *Sobolev Spaces*. Academic Press, 2003.

[AFLP12] M. Aurada, S. Ferraz-Leite, and D. Praetorius. Estimator reduction and convergence of adaptive BEM. *Appl. Numer. Math.*, 62:787–801, 2012.

[AR91] B. Alpert and V. Rokhlin. A fast algorithm for the evaluation of Legendre expansions. *Research Report Yale University*, pages 1–29, 1991.

[AS84] M. Abramowitz and I.A. Stegun. *Pocketbook of Mathematical Functions*. Verlag Harri Deutsch, 1984.

[BÖ6] S. Börm. $\mathcal{H}^2$-Matrix arithmetics in linear complexity. *Computing*, 77(1):1–28, 2006.

[Ban12] A. Bantle. Efficient implementation of the hp-boundary element method for the Navier-Lamé equation. Master Thesis, University of Ulm, Germany, 2012.

[BBF13] A. Bantle, M. Bantle, and S. Funken. epsBEM, efficient p-stable Matlab implementation of 2d BEM for Laplace and Lamé problems. Technical Report, University of Ulm, Germany, 2013.

[Beb00] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86:565–589, 2000.

[Beb08] M. Bebendorf. Hierarchical Matrices: A means to efficiently solve elliptic boundary value problems. *Lecture Notes in Computational Science and Engineering (LNCSE), Springer-Verlag*, 63, 2008.

[BG04] S. Börm and L. Grasedyck. Low-rank approximation of integral operators by interpolation. *Computing*, 72(3-4):325–332, 2004.

[BG06] S. Le Borne and L. Grasedyck. $\mathcal{H}$-Matrix preconditioners in convection-dominated problems. *Siam J. Matrix Anal. Appl.*, 27(4):1172–1183, 2006.

[BGH03] S. Boerm, L. Grasedyck, and W. Hackbusch. An introduction to hierarchical matrices with applications. *Engineering analysis with boundary elements*, 27(5):405—422, 2003.

[BHLT02] D.H. Bailey, Y. Hida, X.S. Li, and B. Thompson. ARPREC: An arbitrary precision computation package. Lbnl-53651, Lawrence Berkeley National Laboratory, 2002.

[Bör10] S. Börm. *Efficient numerical methods for non-local operators. $\mathcal{H}^2$-Matrix compression, algorithms and analysis*. EMS Tracts in Mathematics, 2010.

[BS91] I. Babuska and B. Szabó. *Finite Element Analysis*. John Wiley and Sons, Inc., 1991.

[BV84] I. Babuska and M. Vogelius. Feedback and adaptive finite element solution of one-dimensional boundary value problems. *Numer. Math.*, 44(1):75–102, 1984.

[CFS96] C. Carstensen, S. Funken, and E.P. Stephan. A posteriori error estimates for hp-boundary element methods. *Appl. Anal.*, 61(3-4):233–253, 1996.

[CMPS04] C. Carstensen, M. Maischak, D. Praetorius, and E.P. Stephan. Residual-based a posteriori error estimate for hypersingular equation on surfaces. *Numer. Math.*, 97(3):397–425, 2004.

[CMS01] C. Carstensen, M. Maischak, and E.P. Stephan. A posteriori error estimate and h-adaptive algorithm on surfaces for Symm's integral equation. *Numer. Math.*, 90:197–213, 2001.

[CNV12] C. Canuto, R.H. Nochetto, and M. Verani. Contraction and optimality properties of adaptive Legendre-Galerkin methods: the 1-dimensional case. *arXiv:1206.5524v1 [math.NA]*, pages 1–26, 2012.

[CP06] C. Carstensen and D. Praetorius. Averaging techniques for the effective numerical solution of Symm's integral equation of the first kind. *SIAM J. Sci. Comput.*, 27(4):1226–1260, 2006.

[CP12] C. Carstensen and D. Praetorius. Convergence of adaptive boundary element methods. *J. Integral Equations Appl.*, 24:1–23, 2012.

[CS95] C. Carstensen and E.P. Stephan. A posteriori error estimates for boundary element methods. *Mathematics of Computation*, 64(210):483–500, 1995.

[Dij08] W. Dijkstra. Condition numbers in the boundary element method: shape and solvability. *Dissertation Technische Universiteit Eindhoven*, pages 1–181, 2008.

[DN02] W. Dörfler and R.H. Nochetto. Small data oscillation implies the saturation assumption. *Numer. Math.*, 91(1):1–12, 2002.

[Dör96] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM J. Numer. Anal.*, 33(3):1106–1124, 1996.

[EFFP08] C. Erath, S. Ferraz-Leite, S. Funken, and D. Praetorius. Energy norm based a posteriori error estimation for boundary element methods in two dimensions. *Appl. Numer. Math.*, 59:2713-2734, 2009.

[EFGP12] C. Erath, S. Funken, P. Goldenits, and D. Praetorius. Simple error estimators for the Galerkin BEM for some hypersingular integral equation in 2d. *accepted for publication in Appl. Anal.*, 2012.

[EGTZ12] A. Enge, M. Gastineau, P. Théveny, and P. Zimmermann. *mpc — A library for multiprecision complex arithmetic with exact rounding*. INRIA, 1.0 edition, July 2012. `http://mpc.multiprecision.org/`.

[EMOT53] A. Erdélyi, W. Magnus, F. Oberhettinger, and F.G. Tricomi. Higher transcendental functions vol. 1. *McGraw-Hill, New York*, Jan 1953.

[FHL+07] L. Fousse, G. Hanrote, V. Lefèvre, P. Pélissier, and P. Zimmermann. Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2), 2007.

[FKMP13] M. Feischl, M. Karkulik, J.M. Melenk, and D. Praetorius. Quasi-optimal convergence rate for an adaptive boundary element method. *accepted for publication in SIAM J. Numer. Anal.*, 2013.

[FLOP10] S. Ferraz-Leite, C. Ortner, and D. Praetorius. Convergence of simple adaptive Galerkin schemes based on $h - h/2$ error estimators. *Numer. Math.*, 115:291–316, 2010.

[FLP08] S. Ferraz-Leite and D. Praetorius. Simple a posteriori error estimators for the h-version of the boundary element method. *Computing*, 83:135–162, 2008.

[Gau67] W. Gautschi. Computational aspects of three-term recurrence relations. *Siam Review*, 9(1), 1967.

[Gau81] W. Gautschi. Minimal solutions of three-term recurrence relations and orthogonal polynomials. *Mathematics of Computation*, 36(154):1–8, 1981.

[Gau04] W. Gautschi. *Orthogonal polynomials, Computation and approximation*. Oxford Science Publications, 2004.

[GB86] W. Gui and I. Babuska. The h, p and $h-p$ versions of the finite element method in 1 dimension. *Numer. Math.*, 49:613–657, 1986.

[GBH06] L. Grasedyck, S. Boerm, and W. Hackbusch. Hierarchical matrices. *Lecture Note (MPI Leipzig)*, 21:1–173, Jun 2006.

[Geo07] E.H. Georgoulis. Inverse-type estimates on hp-finite elements spaces and applications. *Mathematics of Computation*, 77(261):201–219, 2007.

[GH03] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-Matrices. *Computing*, 70:295–334, 2003.

[GuHe06] B. Guo and N. Heuer. The optimal convergence of the h-p version of the boundary element method with quasiuniform meshes for elliptic problems on polygonal domains. *Advances in Computational Mathematics*, 24(822):353–374, 2006.

[GHS05] I.G. Graham, W. Hackbusch, and S.A. Sauter. Finite elements on degenerate meshes: inverse-type inequalities and applications. *IMA J. Numer. Anal.*, 25(2):379–407, 2005.

[GR81] I.S. Gradstein and I.M. Ryshik. *Summen- Produkt- und Integral-Tafeln*. Verlag Harri Deutsch Thun Frankfurt/M., 1981.

[GR87] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[Hac99] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-Matrices. Part i: Introduction to $\mathcal{H}$-Matrices. *Computing*, 62:89–108, 1999.

[Heu92] N. Heuer. *hp-Versionen der Randelementmethode*. PhD thesis, 1992.

[Heu96] N. Heuer. Efficient algorithms for the p-version of the boundary element method. *Journal of Integral Equations and Applications*, 8(3):1–24, 1996.

[Heu02] N. Heuer. An hp-adaptive refinement strategy for hypersingular operators on surfaces. *Numer. Methods Partial for Differential Eq.*, 18(3):396–419, 2002.

[HK12] R. Hiptmair and L. Kielhorn. BETL — a generic boundary element template library. Technical report, Seminar for Applied Mathematics, ETH Zürich, 2012.

[HKS00] W. Hackbusch, B. Khoromskij, and S.A. Sauter. On $\mathcal{H}^2$-Matrices. *Lectures on Applied Mathematics, Springer Verlag Berlin*, 2000.

[HN88] W. Hackbusch and Z.P. Nowak. O cloznosti metoda panelej (on complexity of the panel method, russian). *In Vychislitelnye protsessy i systemy 6, Nauka, Moscow*, pages 233–244, 1988.

[HMS01] N. Heuer, M.E. Mellado, and E.P. Stephan. hp-adaptive two-level methods for boundary integral equations on curves. *Computing*, 67(4):305–334, 2001.

[Hob55] E.W. Hobson. *The theory of spherical and ellipsoidal harmonics*. Chelsea Pub. Co., 1955.

[HS52] M.R. Hestens and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

[HS96] N. Heuer and E.P. Stephan. The hp-version of the boundary element method on polygons. *Journal of Integral Equations and Applications*, 8(2), 1996.

[HST98] N. Heuer, E.P. Stephan, and T. Tran. Multilevel additive Schwarz method for the h-p version of the Galerkin boundary element method. *Mathematics of Computation*, 67(222):501–518, 1998.

[Jas63] M.A. Jaswon. Integral equation methods in potential theory. I. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 275:23–32, 1963.

[Kar12] M. Karkulik. *Zur Konvergenz und Quasioptimalität adaptiver Randelementmethoden*. PhD thesis, TU Wien, Austria, 2012.

[Mac67] T.M. MacRobert. *Spherical harmonics. An elementary treatise on harmonic functions with applications*. Pergamon Press, 1967.

[Mai96] M. Maischak. *hp-Methoden für Randintegralgleichungen bei 3D-Problemen, Theorie und Implementierung*. PhD thesis, Universität Hannover, Germany, 1996.

[Mai12] M. Maischak. B.O.N.E. book of numerical experiments. Technical Rseport, Institut für Angewandte Mathematik, Universitat Hannover SISCM, Brunel University, UK, 2012.

[Mat83] R.M.M. Mattheij. On decoupling of linear recursions. *Bull. Austral. Math. Soc.*, 1983.

[McL00] W. McLean. *Strongly elliptic systems and boundary integral equations*. Camebridge University Press, 2000.

[MMRU10] M. Messner, M. Messner, F. Rammerstorfer, and P. Urthaler. Hyperbolic and elliptic numerical analysis BEM library (hyENA). Preprint TU Graz, Austria, 2010.

[Poz02] C. Pozrikidis. A practical guide to boundary-element methods with the software library BEMlib. *Chapman&Hall/CRC*, 2002.

[PS09] G. Polya and G. Szegö. Über den transfiniten Durchmesser (Kapazitätskonstante) von ebenen und räumlichen Punktmengen. *Journal für die reine und angewandte Mathematik (Crelle's Journal).*, 1931(165):4–49, 2009.

[Rea79] J.B. Reade. Asymptotic behaviour of eigen-values of certain integral equations. *Proceedings of the Edinburgh Mathematical Society*, 22:137–144, 1979.

[Riv74] T.J. Rivlin. *The Chebyshev polynomials*. Wiley, New York, 1974.

[Rja02] S. Rjasanow. Adaptive cross approximation of dense matrices. pages 1–12, Apr 2002.

[ŚAB+12] W. Śmigaj, S. Arridge, T. Betcke, J. Phillips, and M. Schweiger. Solving boundary integral problems with BEM++. *submitted to ACM Trans. Math. Software*, 2012.

[Sad03] S.Y. Sadov. Coupling of the Legendre polynomials with kernels $|x-y|$ and $\ln|x-y|$. *preprint arXiv:math/0310063v1*, pages 1–23, 2003.

[Sch98] C. Schwab. *p- and hp- Finite Element Methods. Theory and Applications in Solid and Fluid Mechanics*. Oxford University Press, 1998.

[SlSp88] I.H. Sloan and A. Spence. The Galerkin method for integral equations of the first kind with logarithmic kernel: Theory. *IMA J. Numer. Anal.*, 8:105–122, 1988.

[StSu91] E.P. Stephan and M. Suri. The $h-p$ version of the boundary element method on polygonal domains with quasiuniform meshes. *ESAIM: Mathematical Modeling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 25(6):783–807, 1991.

[SaSch97] S.A. Sauter and C. Schwab. Quadrature for hp-Galerkin BEM in $\mathbb{R}^3$. *Numer. Math.*, 78:211–258, 1997.

[SaSch04] S. Sauter and C. Schwab. *Randelementmethoden*. Vieweg+Teubner Verlag, 2004.

[Ste96] E.P. Stephan. The h-p boundary element method for solving 2- and 3-dimensional problems. *Comput. Methods Appl. Mech. Engrg.*, 133:183–208, 1996.

[Ste03] O. Steinbach. *Numerische Näherungsverfahren für elliptische Randwertprobleme. Finite Elemente und Randelemente*. B.G. Teubner, Stuttgart, Leipzig, Wiesbaden, 2003.

[SW92] C. Schwab and W.L. Wendland. On numerical cubatures of singular surface integrals in boundary element methods. *Numer. Math.*, 62:343–369, 1992.

[Sym63] G.T. Symm. Integral equation methods in potential theory. II. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 275:33–46, 1963.

[Thi52] H.P. Thielman. *British Association for the Advancement of Science, Mathematical Tables, Vol. X. Bessel Functions: Functions of Positive Integer Order, Part II*. Cambridge Univ. Press, 1952.

[TS00] T. Tran and E.P. Stephan. Additive Schwarz algorithms for the p version of the Galerkin boundary element method. *Numer. Math.*, 85:433–468, 2000.

[Wei10] H.F. Weidle. Rekursionen zur stabilen Berechnung von speziellen Integralen in der Randelementmethode. Bachelor Thesis, University of Ulm, Germany, 2010.

List of Symbols

General Notation

$\lvert\cdot\rvert$	absolute value or cardinality of a set
d/ds	arc length derivative
$\mathcal{O}(f)$	Landau symbol, f is an asymptotic upper bound
$\lesssim$	smaller than or equal to up to constant
$\gtrsim$	greater than or equal to up to constant
$\mathrm{Re}(z)$, $\mathrm{Im}(z)$	real and imaginary part of a complex number
$\mathrm{conv}(X)$	convex hull of the elements in the set X
span(X)	space that contains al linear combinations of elements in the set X
$\mathrm{dist}\{\cdot,\cdot\}$	Euclidean distance
$\delta_{k,\ell}$	Kronecker symbol

Function Spaces, Scalar Products, and Norms

$C^k(\Omega)$	space of k times continuously differentiable functions, 1
$C_0^k(\Omega)$	space of functions in $C^k(\Omega)$ with compact support in Ω, 1
$C^\infty(\Omega)$	space of infinitely times differentiable functions, 1
$C_0^\infty(\Omega)$	space of functions in $C^\infty(\Omega)$ with compact support in Ω, 1
$L^2(\Omega)$	Lebesgue space of square integrable functions, 1
$L^2_{loc}(\Omega)$	Lebesgue space of locally square integrable functions, 2
$L^\infty(\Omega)$	space of essentially bounded functions, 2
$H^k(\Omega)$	Sobolev space of order k on Ω ($k \geq 0$), 2
$H^k_{loc}(\Omega)$	space of local $H^k(\Omega)$ functions, 2
$H^k_0(\Omega)$	closure of $C_0^\infty(\Omega)$ in $H^s(\Omega)$ ($k \geq 0$), 3
$\widetilde{H}^k(\Omega)$	closure of $C_0^\infty(\Omega)$ in $H^s(\mathbb{R}^n)$ ($k \geq 0$), 3
$H^{-k}(\Omega)$	dual space of $\widetilde{H}^k(\Omega)$ ($k \geq 0$), 3
$\widetilde{H}^{-k}(\Omega)$	dual space of $H^k(\Omega)$ ($k \geq 0$), 3
$H^k(\Gamma)$	Sobolev space of order k on the boundary Γ, 3
$H^k_*(\Gamma)$	space of functions in $H^k(\Gamma)$ with integral mean zero, 4

$(\cdot,\cdot)_{L^2(\Omega)}$	scalar product in $L^2(\Omega)$, 1
$\langle\cdot,\cdot\rangle_{X\times Y}$	extended L^2 scalar product, 3
$(\cdot,\cdot)_{H^k(\Omega)}$	scalar product in $H^k(\Omega)$, 2
$\|\cdot\|_{L^2(\Omega)}$	norm induced by the scalar product in $L^2(\Omega)$, 2
$\|\cdot\|_{H^k(\Omega)}$, $\|\cdot\|_{H^k(\Gamma)}$	norm on $H^k(\Omega)$ and $H^k(\Omega)$, 2
$\|\cdot\|_{H^\sigma(\Omega)}$	Sobolev-Slobodeckij semi norm ($\sigma \in (0,1)$), 3
γ_0^{int}, γ_0^{ext}, γ_0	trace operators, 5
γ_1^{int}, γ_1^{ext}, γ_1	co-normal derivatives, 5

Boundary Integral Equations

$\widetilde{V}$, V	single layer potential and operator, 6,7
$\widetilde{K}$, K	double layer potential and operator, 6,7
$\widetilde{K}'$, K'	adjoint double layer potential and operator, 6,8
$\widetilde{W}$, W	hypersingular potential and operator, 6,8
$\widetilde{N}$	Newton potential, 6
$\mathbf{V}$, $\mathbf{K}$, $\mathbf{W}$	Galerkin matrix for single layer, double layer, and hypersingular operator, 14
$\mathbf{M}$	mass matrix
$\|\!\|\!\|\cdot\|\!\|\!\|_V$, $\|\!\|\!\|\cdot\|\!\|\!\|_W$	energy norm, leave out index if operator is clear

Boundary Mesh and Ansatz Spaces

$\mathcal{T}$	boundary mesh on Γ, 12
$\mathcal{N}_e$, $\mathcal{N}_v$	number of elements and nodes in $\mathcal{T}$, 12
T, T_i	element of $\mathcal{T}$,12
$\mathbf{h}$, $\underline{h}$, $\overline{h}$	mesh width function, minimum and maximum mesh width, 13
h_i, h_T	mesh width of T_i and T, respectively, 13
$\mathbf{p}$, $\mathbf{p}_0$, $\mathbf{p}_1$	polynomial degree function, 13
p_i, p_T	polynomial degree on T_i or T, 13
$\underline{p}$, $\overline{p}$	minimum and maximum polynomial degree, 13
γ_T, γ_i	affine mapping from $[-1,1]$ to T and T_i, respectively, 71
$P_k^{(i)}$, $P_k^{(T)}$	Legendre polynomial on T_i and T, respectively, 72
$N_k^{(i)}$, $N_k^{(T)}$	Lobatto shape function on T_i and T, respectively, 72
ρ	mesh parameter for geometric meshes, 16
$S_0(\mathcal{T},\mathbf{h},\mathbf{p}_0)$,	space of piecewise polynomials w.r.t. the mesh $\mathcal{T}$, 13,72
$S_1(\mathcal{T},\mathbf{h},\mathbf{p}_1)$	space of piecewise polynomial globally continuous functions w.r.t. the mesh $\mathcal{T}$, 13,72

$\mathcal{N}_0, \mathcal{N}_1$	dimension of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$ and $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$, respectively, 13
$\mathcal{N}$	dimension of discrete space if clear which space is meant, 13
ϕ_h^p	discrete Solution of Symm's integral equation in $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p}_0)$, 13
ψ_h^p	discrete Solution of the hypersingular integral equation in $S_1(\mathcal{T}, \mathbf{h}, \mathbf{p}_1)$, 14

Legendre functions and modified Legendre functions

J	Joukowski transform, maps a circle to an ellipse, 22
P_k, N_k	Legendre Polynomial and Lobatto shape function, 23,24
P_k^m, Q_k^m	associated Legendre function of the first and the second kind, 25
$\widetilde{Q}_k^m$, R_k^m	modified associated Legendre functions, $m \in \mathbb{Z}$, 27
$I_{k,\ell}^m$, $O_{k,\ell}^m$, $U_{k,\ell}^m$	basic double integrals, 53
$T(a, b)$	line segment $\mathrm{conv}(a - b, a + b)$ in the complex plane ($a, b \in \mathbb{C}$), 52
$\widehat{T}$	$\widehat{T} = [-1, 1] = T(0, 1)$, 52

Error Estimators and Adaptive Algorithms

$S_{h,p}$	$S_{h,p} = S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, 99
$\widehat{S}_{h,p}$	fine space, $S_{h,p} \subset \widehat{S}_{h,p}$, 99
$\widehat{\phi}_h^p$	discrete solution of Symm's integral equation in $\widehat{S}_{h,p}$, 99
ω	weight function on $(-1, 1)$,102
ω^T	$\omega \circ \gamma_T^{(-1)}$, weight function on $T \in \mathcal{T}$, 102
$(\cdot, \cdot)_{L^2(\Gamma),\omega}$	weighted L^2 scalar product, 102
$\|\cdot\|_{L^2(\Gamma),\omega}$	weighted L^2 norm, 103
$\Pi_{h,p}$	element wise L^2 projection from $H^{-1/2}(\Gamma)$ to $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, 96
$\Pi_{h,p}^\omega$	element wise weighted L^2 projection from $H^{-1/2}(\Gamma)$ to $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, 103
η_0, $\widetilde{\eta}_0$	$h - h/2$ error estimators from [FLP08] with energy norm, 96
μ_0, $\widetilde{\mu}_0$	$h - h/2$ error estimators from [FLP08] with L^2 norm, 96
η, $\widetilde{\eta}$	generalized $h - h/2$ error estimators with energy norm, 99
μ, $\widetilde{\mu}$	generalized $h - h/2$ error estimators with L^2 norm, 99
η^ω, $\widetilde{\eta}^\omega$	generalized $h - h/2$ error estimators with energy norm and weighted projection, 103
μ^ω, $\widetilde{\mu}^\omega$	generalized $h - h/2$ error estimators with weighted L^2 norm and weighted L^2 projection, 103
θ_h, θ_p	adaptivity parameters for Algorithm 4.1, 119
θ, τ	adaptivity parameters for Algorithm 4.2, 122

Data Compression

$\mathcal{I}$	index set for indices of the basis function of $S_0(\mathcal{T}, \mathbf{h}, \mathbf{p})$, 138
σ, τ	clusters, $\sigma, \tau \subset \mathcal{I}$, 138
B_σ	axis parallel bounding box for the cluster σ, 138
$\mathbb{T}$, $\mathbb{T}_e$	cluster tree for indices of basis functions and elements, 138
C_{leaf}	leaf size of a cluster tree, 138
C_{sp}	sparsity constant for a partitioning, 140
$\mathbb{P}$	block partitioning, $\mathbb{P} \subset \mathcal{I} \times \mathcal{I}$, 139
η	admissibility constant, 139
N_{st}	storage requirement, 140
N_{mv}	complexity of the matrix vector multiplication, 140
$I^{\nu}_{x,\sigma}$	νth order tensor product Chebyshev interpolation operator on B_σ, 142
κ	kernel function $\kappa(x, y) = \log \lvert x - y \rvert$, 142

Curriculum Vitae

Personal Data

NAME	Markus Bantle
DATE OF BIRTH	August 22, 1984
PLACE OF BIRTH	Laupheim, Germany
NATIONALITY	German
E-MAIL	mabantle@gmail.com

Education

SINCE MAY 2010	Doctoral studies, Institute for Numerical Mathematics, University of Ulm, Germany
MAY 2010	Diplom in Mathematics (Grade: 1.1)
OCT 2004 - MAY 2010	Studies in Mathematics & Computer Science, University of Ulm, Germany
JUL 2009	Master of Science in Mathematics (GPA 3.9)
SEP 2008 - JUL 2009	Master Studies in Mathematics, San Diego State University, San Diego, CA, USA.
2003	Abitur (Grade: 1.2)

Work Experience

SINCE OCT 2009	Wissenschaftlicher Mitarbeiter, Institute for Numerical Analysis, University of Ulm, Germany
SEP 2008 - JUL 2009	Graduate Teaching Assistant, San Diego State University, San Diego, CA, USA
JAN 2008 - JUL 2008	Forschungsstudent, Graduiertenkolleg, University of Ulm, Germany
2005-2007	Student Teaching Assistant, University of Ulm, Germany

Publications

M. Bantle, S. Funken.
Data Compression Techniques for the p-Version of the Boundary Element Method for the Laplace Equation in Two Dimensions.
IABEM 2013 Symposium of the International Association for Boundary Element Methods, Santiago, 09.01.2013-11.01.2013, IABEM 2013 Proceedings, Pontificia Universidad Católica de Chile, (2013), ISBN: 978-956-351-579-4, pp. 13-18.

A. Bantle, M. Bantle, and S. Funken.
epsBEM, Efficient P-Stable Matlab Implementation of 2D BEM for Laplace and Lamé Problems.
Technical Report, University of Ulm, Germany, 2013.

S.S.P. Shen, M. Bantle, A. S. Donahue, R. Wong, and C. K. Lee.
EOF-MSE adaptive method to assess an acid deposition monitoring network over Alberta, Canada.
Advances in Adaptive Data Analysis, 5, No. 1, 2012.

M. Bantle (supervisor S. Funken).
Efficient Implementation of Collocation- and Boundary Element Methods for Integral Equations.
Diplomarbeit, University of Ulm, Germany, 2009.

M. Bantle (supervisor S.S.P. Shen).
Error Analysis, Assessment and Design of the Alberta Acid Deposition Network.
Master Thesis, San Diego State University, San Diego, Ca, USA.

Selected Talks

A. Bantle, M. Bantle, and S. Funken.
epsBEM - efficient p-stable Boundary Element Methods, A MATLAB *Software Package.*
WONAPDE 2013 Fourth Chilean Workshop on Numerical Analysis of Partial Differential Equations, Concepcion, Chile, Jan. 14 - 18, 2013.

M. Bantle, S. Funken.
Data Compression Techniques for the p-Version of the Boundary Element Method for the Laplace Equation in two Dimensions.
IABEM 2013 Symposium of the International Association for Boundary Element Methods, Santiago, Chile, Jan. 09 - 11, 2013.

Erklärung

Ich, Markus Bantle, versichere hiermit, dass ich die vorliegende Arbeit selbständig angefertigt habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe. Ich erkläre außerdem, dass die von mir vorgelegte Dissertation bisher nicht im In- oder Ausland in dieser oder ähnlicher Form in einem anderen Promotionsverfahren vorgelegt wurde. Ich versichere ferner die Richtigkeit der im Lebenslauf gemachten Angaben.

Ulm, den 22.4.2013

(Unterschrift)