ulm university universität
uulm

Fakultät für Mathematik und Wirtschaftswissenschaften

Institut für Numerische Mathematik

Masterarbeit

Efficient Implementation of the hp-Boundary Element Method for the Navier-Lamé Equation

vorgelegt von

Andreas Bantle

am 29. August 2012

Gutachter

1. Prof. Dr. S. Funken
2. Prof. Dr. K. Urban

Acknowledgements.

First and foremost, I would like to express my gratitude to my advisor, Prof. S. A. Funken, who always offered me advice and support throughout my research. Without his expertise and excellent supervision this thesis would not have been possible. I would also like to extend my gratitude to Prof. K. Urban for serving on my thesis committee.
Special thanks go to Markus Bantle for giving me advice, whenever I needed help, answering my questions very patiently and editing my thesis. I would also like to thank Natalie Selinski for editing my thesis.
Last but not least, I would like to thank my family and Franziska Merkle for supporting me during my master studies at University of Ulm.

Ulm, August 2012.

Contents

Chapter 1
Introduction

With the technical development of computers, the influence of mathematical modeling and simulation on research is increasing significantly. Instead of developing expensive and time consuming prototypes, mathematical models are derived and simulated on computers. Most of the (physical) problems can be described by mathematical models using partial differential equations (PDEs) or integral equations (IEs). In particular, problems in the field of linear elasticity can be modeled by an elliptic PDE, namely the **Navier-Lamé equation**. A common example in linear elasticity that can be modeled with the Navier-Lamé equation is the Cook's membrane example. Cook's membrane, which is illustrated in Figure 1.1, is fixed on the left-hand side and exposed to a traction on the right-hand side.

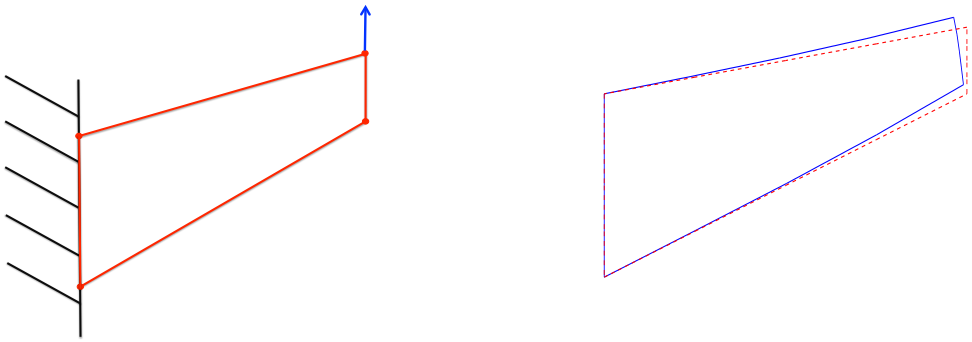

Fig. 1.1: Domain for the Cook's membrane example (left) with expected displacement (right).

In most cases the solution of the PDE cannot be calculated analytically and thus numerical methods are needed to approximate the solution. There are a variety of methods for approximating solutions to PDEs including the Finite Element Method (FEM), the Finite Difference Method (FDM) and the **Boundary Element Method (BEM)**. All methods have both advantages and disadvantages and thus a method is chosen based on the need of the application. One of the main uses of the BEM is to PDE models in linear elasticity, such as the Navier-Lamé equation. The BEM differs significantly from the other methods, since the main idea of the BEM is to transform the PDE into an equivalent boundary IE instead of discretizing the PDE itself. Thus, only the boundary of the domain has to be discretized. This can reduce the computational effort and consequentially the computational time, particularly with complicated domains. Moreover, problems on unbounded domains can be solved easily with the BEM, whereas several problems occur with the FEM or the FDM. However, for transforming the PDE in an equivalent integral equation

an analytical representation of a fundamental solution is required, but such a representation is not always provided.
The BEM is a Galerkin method, meaning the boundary IE is multiplied with test functions in a weak sense leading to the variational formulation. Discretizing this variational formulation yields a system of linear equations with a large number of degrees of freedom. The main computational effort arises from assembling this system of linear equations.
Particularly for practical problems, the accuracy and the reliability of the results are of fundamental importance for the calculations. In order to improve the accuracy of the numerical results there exist two main approaches. One is to refine the discretization of the boundary, which leads to an h-method. The other is to increase the polynomial degree of the numerical solution, which leads to a p-method. By combining both approaches cleverly we obtain an **hp-method**, where the advantages of both methods can be exploited. In general, the hp-method can achieve exponential convergence rates for the error, whereas the h- and the p-method only lead to algebraic convergence rates for the error. Figure 1.2 illustrates the estimated error for the Cook's membrane example.

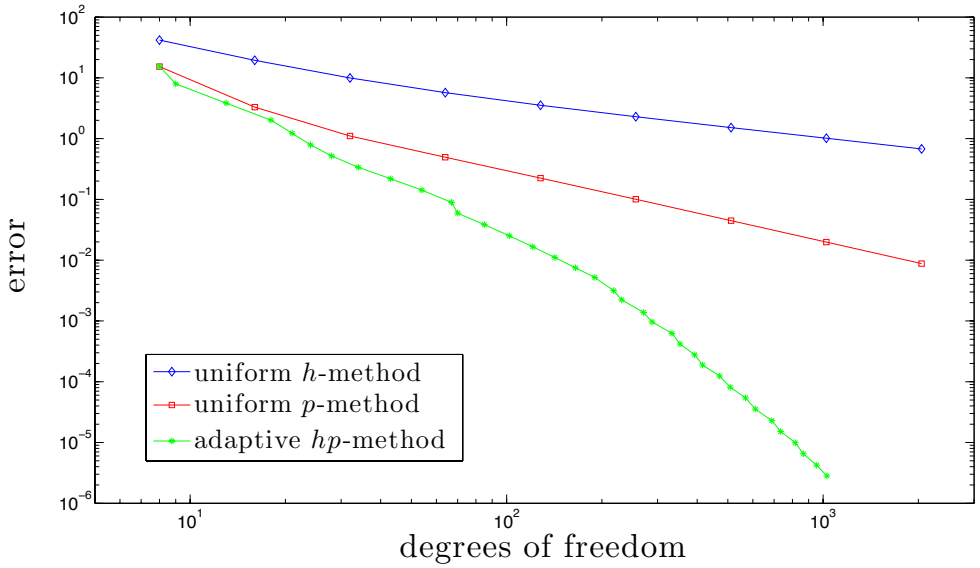

Fig. 1.2: Estimated error for the Cook's membrane example using the different approaches for reducing the error.

We see that with the exponential convergence rate of the hp-method accurate results can be achieved with a small number of degrees of freedom. On most computers, a similar accuracy cannot be achieved with the h- or the p-method due to the limited memory or at least long computational times are needed.

Goal of this Thesis

The goal of this thesis is to develop an efficient implementation of the hp-BEM for the Navier-Lamé equation, so that practical, two-dimensional problems for linear

elasticity can be solved. For this purpose, we want to reduce the error close to machine precision harnessing the exponential convergence rate of the *hp*-method. Therefore, we derive analytical formulas for the assembly of the Galerkin matrix that can be stably and efficiently calculated for high polynomial degrees. Moreover, we want to take advantage of progress in computer hardware in order to perform the calculations very efficiently. In particular, since most computers today are equipped with multi-core processors we want to parallelize the routines.
Additionally, this thesis focuses on integrating all routines into the epsBEM framework (see [10]), a software package for solving the Laplace equation with the *hp*-BEM. Thus, the structure of the routines for solving the Laplace and the Navier-Lamé equations must be similar, so that the user only has to adjust the main routines for solving the Navier-Lamé equation. For this purpose, the core routines have to be implemented in C, whereas the main routines have to be implemented in MATLAB.

Outline

This thesis is structured into of six main chapters. The first four chapters are devoted to the theory of the *hp*-BEM for the Navier-Lamé equation, and the implementation and the numerical results are presented in the last two chapters.
Chapter 2 gives an introduction to linear elasticity. We begin by giving the analytical background and introducing some basic notations. Subsequently, we derive the Navier-Lamé equation and its weak formulation by referencing to a model example.
An introduction to the theory of the *hp*-BEM for the Navier-Lamé equation is given in Chapter 3. In particular, we introduce the boundary integral operators and derive the system of linear equations for different types of boundary conditions, namely mixed, Dirichlet and Neumann boundary conditions.
In Chapter 4 we explore the associated Legendre functions and related functions. This chapter serves as a basis for developing an efficient implementation of the *hp*-BEM. The approach for implementing the *hp*-BEM, that is investigated in this thesis, is to take the Legendre polynomials and the Lobatto shape functions as ansatz-functions. Using this approach, the analytical computation of the entries in the Galerkin matrix can be reduced to evaluating integrals that are related to the associated Legendre functions. Consequently, we investigate the efficient calculation of these integrals, which is the basic requirement for the efficient implementation of the *hp*-BEM.
In Chapter 5 we describe the realization of the *hp*-BEM using analytical formulas. We begin by introducing the basis for the ansatz- and test-space, which is the main point for implementing an efficient *hp*-BEM, and by defining a parametrization of the boundary elements. The main part in this chapter is to derive analytical formulas for calculating the Galerkin matrices of the boundary integrals operators efficiently, using the special integrals related to the associated Legendre functions. This chapter closes by introducing some error estimators for the Dirichlet, the Neumann and the mixed problems that can be used for creating adaptive algorithms.
The implementation of the *hp*-version of the BEM is described in Chapter 6. Since all routines for the implementation of the *hp*-BEM have to be integrated into the epsBEM framework, we first give an overview on this software package. Furthermore, we describe all routines that are implemented within the scope of this thesis. In particular, we go into detail on the implementation of the functions introduced

in Chapter 4 and on the routines to calculate the integral operators with the formulas derived in Chapter 5. Additionally, we discuss the implementation of gliding conditions, i.e. a special type of boundary conditions, so that we can solve a wider range of practical problems in linear elasticity.
In the last chapter, we present the numerical results. We consider several standard examples and verify the convergence rates and the efficiency and reliability of the error estimator. This chapter closes with a discussion on practical examples of linear elasticity.

Chapter 2
The Navier-Lamé Equation

2.1 Theoretical Background

In this section, we give the theory of Sobolev spaces and introduce some important notations, that we need throughout this thesis, where we are guided by the work of [16]. Note that we only give a summary of the theory without proving the results. Throughout this section let $\Omega \subset \mathbb{R}^n$ $(n \geq 1)$ be a domain. Let $\mathcal{C}_c^\infty(\Omega) := \{u \in \mathcal{C}^\infty(\Omega) : u \text{ has compact support}\}$ and the Lebesgue space L^2 be defined as usual with the L^2 scalar product

$$(\, f, g \,)_{L^2(\Omega)} := \int_\Omega f(x)\, g(x)\, dx$$

and the induced norm $\| \cdot \|_{L^2(\Omega)}$.

Definition 2.1.1 *We call $u \in L^2(\Omega)$ weakly differentiable if there exist functions $g_1, ..., g_n \in L^2(\Omega)$ so that*

$$-\int_\Omega u(x)\, \frac{\partial}{\partial x_j} \phi(x)\, dx = \int_\Omega g_j(x)\, \phi(x)\, dx$$

$\forall j = 1, .., n$ and $\forall \phi \in \mathcal{C}_c^\infty(\Omega)$ and we call $(g_1, ..., g_n)^T$ the weak derivative of u. If u is differentiable, the usual derivative and the weak derivative are the same a.e.. Therefore, we write $\nabla u = (g_1, ..., g_n)^T$ for the weak derivative.

Definition 2.1.2 (Sobolev spaces on domains) *We define the Sobolev space $H^0(\Omega)$ by the Lebesgue Space $L^2(\Omega)$,*

$$H^0(\Omega) := L^2(\Omega)$$

equipped with the usual L^2 norm. The Sobolev space $H^1(\Omega)$ is given by

$$H^1(\Omega) := \left\{ u \in L^2(\Omega) \, : \, u \text{ is weakly differentiable} \, , \, \nabla u \in L^2(\Omega) \right\}.$$

A norm on H^1 is defined by

$$\|u\|^2_{H^1(\Omega)} := \|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{L^2(\Omega)}.$$

Moreover, we define the Sobolev space with positive integer order k by

$$H^k(\Omega) := \left\{ u \in L^2(\Omega) \, : \, u \text{ is weakly differentiable} \, , \, \nabla u \in H^{k-1}(\Omega) \right\}.$$

A norm on $H^k(\Omega)$ can be defined by

$$\|u\|^2_{H^k(\Omega)} := \|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{H^{k-1}(\Omega)}.$$

A semi-norm on $H^k(\Omega)$ is given by

$$|u|_{H^k(\Omega)} := \|D^k u\|_{L^2(\Omega)},$$

where D^k denotes the k-th weak derivative.

Lemma 2.1.3 *The Sobolev spaces $H^k(\Omega)$, $k \in \mathbb{N}_0$, are Hilbert spaces with continuous embedding $H^k(\Omega) \subset L^2(\Gamma)$.*

With the Riesz representation theorem we can define the dual space of the Sobolev spaces as extensions of L^2.

Definition 2.1.4 *The dual space of the Sobolev space $H^k(\Omega)$ is defined by the Sobolev space $\widetilde{H}^{-k}(\Omega)$ of negative integer order k equipped with the usual operator norm*

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

where the duality brackets $\langle \cdot, \cdot \rangle := \langle \cdot, \cdot \rangle_{H^{-k}(\Omega) \times H^k(\Omega)}$ are the extended L^2 scalar product.

We also need to define the Sobolev spaces with non-integer order on the boundary $\Gamma := \partial\Omega$. Therefore, we introduce the Sobolev-Slobodeckij semi-norm for $s \in (0,1)$ by

$$|u|_{s,\Gamma} := \left(\int_\Gamma \int_\Gamma \frac{|u(x) - u(y)|^2}{|x-y|^{n-1+2s}} \, ds_x \, ds_y \right)^{\frac{1}{2}}.$$

Definition 2.1.5 (Sobolev spaces on the boundary)

(i) Let $s \in (0,1)$. Then, the Sobolev space $H^s(\Gamma)$ is defined by

$$H^s(\Gamma) := \left\{ u \in L^2(\Gamma) \, : \, \|u\|_{H^s(\Gamma)} < \infty \right\}$$

with the norm

$$\|u\|^2_{H^s(\Gamma)} = \|u\|^2_{L^2(\Gamma)} + |u|^2_{s,\Gamma}.$$

(ii) For $\Gamma_0 \subset \Gamma$ and $s \in (0,1)$ we define

$$H^s(\Gamma_0) := \{ u = \widetilde{u}|_{\Gamma_0} \, : \, \widetilde{u} \in H^s(\Gamma) \}$$
$$\widetilde{H}^s(\Gamma_0) := \{ u = \widetilde{u}|_{\Gamma_0} \, : \, \widetilde{u} \in H^s(\Gamma), \ \operatorname{supp} \widetilde{u} \subset \overline{\Gamma_0} \}.$$

Lemma 2.1.6

(i) $H^{\frac{1}{2}}(\Gamma)$ *is a Hilbert space with continuous embedding* $H^{\frac{1}{2}}(\Gamma) \subset L^2(\Gamma)$.

(ii) $H^{\frac{1}{2}}(\Gamma)$ *is the trace space of* $H^1(\Omega)$, *i.e.* $H^{\frac{1}{2}}(\Gamma) = \{u = \widetilde{u}|_\Gamma \,:\, \widetilde{u} \in H^1(\Omega)\}$.

Again, the Riesz representation theorem provides a definition of the dual spaces of the Sobolev spaces with non-integer order.

Definition 2.1.7 *For* $s \in (0,1)$ *we denote the dual space of* $H^s(\Gamma)$ *by* $H^{-s}(\Gamma)$ *equipped with the usual operator norm on* $H^{-s}(\Gamma)$ *that is given by*

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

where the duality brackets $\langle \cdot, \cdot \rangle := \langle \cdot, \cdot \rangle_{H^{-s}(\Gamma)\times H^s(\Gamma)}$ *are the extended* L^2 *scalar product.*

Finally, we introduce the piecewise Sobolev spaces for a given partition $\mathcal{T}_h$ of the boundary Γ, i.e. $\forall\, \Gamma_\ell, \Gamma_k \in \mathcal{T}_h :\ |\Gamma_\ell \cap \Gamma_k| = \delta_{\ell,k}\,|\Gamma_\ell|$ and

$$\Gamma = \bigcup_{\Gamma_\ell \in \mathcal{T}_h} \overline{\Gamma_\ell}.$$

Definition 2.1.8 *For* $s \geq 0$ *the piecewise Sobolev space is defined by*

$$H^s_{pw}(\Gamma) := \left\{\psi \in H^{\min\{1,s\}}(\Gamma) \,:\, \psi\big|_{\Gamma_\ell} \in H^s(\Gamma_\ell),\ \forall\, \Gamma_\ell \in \mathcal{T}_h\right\}$$

equipped with the norm

$$\|\psi\|_{H^s_{pw}(\Gamma)} := \left(\sum_{\Gamma_\ell \in \mathcal{T}_h} \|\psi\|^2_{H^s(\Gamma_\ell)}\right)^{\frac{1}{2}}.$$

To simplify the notation we write $\|\cdot\|_s$ instead of $\|\cdot\|_{H^s(\Gamma)}$, $|\cdot|_s$ instead of $|\cdot|_{H^s(\Gamma)}$ and $\|\cdot\|_k$ instead of $\|\cdot\|_{H^k(\Omega)}$. Moreover, we restrict to write $\langle \cdot, \cdot \rangle$ for the different duality products that we defined above.
Since all functions that occur within this thesis are two dimensional functions $u : \mathbb{R}^2 \to \mathbb{R}^2$, we write $u \in V$ meaning $u_i \in V$ $(i = 1, 2)$ for all above defined function spaces. The corresponding norms are given by

$$\|u\|^2_V = \|u_1\|^2_V + \|u_2\|^2_V.$$

2.2 Derivation of the Navier-Lamé Equation and the Weak Formulation

In this section we give a short introduction the the Navier-Lamé equation. The most important application of the Navier-Lamé equation is to linear elasticity. Therefore,

we consider the mixed traction and displacement problem of homogeneous, isotropic and linearly elastic bodies. For a given body that is exposed to both volume forces and surface forces we want to calculate the displacement and the strain of the displaced body. Additionally, the body is fixed at some parts of the surface, in other words we prescribe the displacement to be zero, and at other parts of the surface a traction is given. An example is illustrated Figure 2.1.

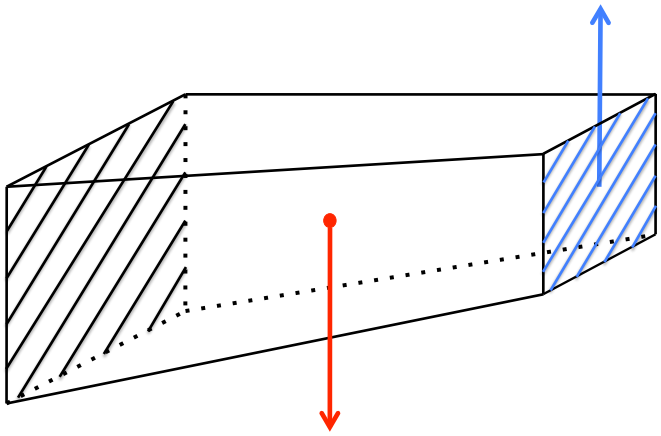

Fig. 2.1: Model of a bar that is fixed on left black shaded side. The bar is exposed to volume forces such as gravity (red arrow) and the right blue shaded side of the bar is exposed to a traction (blue arrow).

In many applications it is a good approximation to map the three-dimensional model of the body to a two-dimensional model. Especially, in the case of plain stain or plain stress reducing the dimension of the body is reasonable. In this work, we consider the plain strain state and thus, we only deal with two-dimensional models. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with Lipschitz-boundary $\Gamma := \partial\Omega$ describing the homogeneous, isotropic and linearly elastic body. Considering mixed traction and displacement problems we divide the boundary Γ into the Dirichlet boundary Γ_D and the Neumann boundary Γ_N. On the Dirichlet boundary, which is a closed subset of Γ with $|\Gamma_D| > 0$, the exact displacement $u_D : \Gamma_D \to \mathbb{R}^2$, $u_D \in \mathcal{C}(\Gamma_D)$, of the body is given, whereas on the Neumann boundary $\Gamma_N := \Gamma \backslash \Gamma_D$ we prescribe the traction by a function $g : \Gamma_N \to \mathbb{R}^2$, $g \in L^\infty(\Gamma_N)$. The volume forces that are applied to the body can be described by a function $f : \Omega \to \mathbb{R}^2$, $f \in \mathcal{C}(\overline{\Omega})$.
The sought displacement field of the body is given by $u : \Omega \to \mathbb{R}^2$, where we assume $u \in \mathcal{C}^2(\Omega) \cap \mathcal{C}(\overline{\Omega})$. Besides the visible displacement of the body, external forces also result in inner strain and stress. Therefore, we define the strain tensor σ and the stress tensor ε of the displaced body. Assuming a homogeneous, isotropic and linearly elastic material and small deformations, the strain and the stress tensors are symmetric 2×2 matrices, i.e. $\sigma, \varepsilon \in \mathbb{R}^{2\times 2}_{symm}$. Moreover, the strain tensor can be approximated by the symmetric part of the gradient of the displacement field u,

which reads

$$\varepsilon(u) \approx \frac{1}{2}\left(\nabla u + (\nabla u)^T\right).$$

In linear elasticity, strain and stress are linearly related, i.e. Hook's law holds:

$$\sigma(u) = 2\,\mu\,\varepsilon(u) + \lambda\,\mathrm{div}(u)\,\mathrm{I},$$

where $\mathrm{I} \in \mathbb{R}^{2\times 2}$ denotes the identity matrix and $\lambda, \mu \in \mathbb{R}$ are the Lamé constants. The Lamé constants are dependent on the parameters of material E, which is called the modulus of elasticity, and ν, which is called the Poisson's ratio. For homogeneous, isotropic and linearly elastic materials there holds $E > 0$ and $\nu \in \left(0, \frac{1}{2}\right)$. In the plain strain state, to which we refer in this work, the Lamé constants and the parameters of material are related as follows:

$$\lambda = \frac{E\,\nu}{(1+\nu)(1-2\,\nu)} \quad \text{and} \quad \mu = \frac{E}{2\,(1+\nu)}.$$

This relationship and the ranges of ν and E imply that $\mu > 0$ and $\lambda > -\frac{2}{3}\,\mu$.

In the state of equilibrium all forces add up to zero. Thus, we obtain

$$\begin{aligned} f + \mathrm{div}\,\sigma &= 0 \\ \sigma \cdot n &= g, \end{aligned}$$

where n denotes the outer unit normal vector. Putting all results together we can formulate the mixed traction and displacement problem:

For given volume forces $f \in \mathcal{C}(\overline{\Omega})$, traction $g \in L^\infty(\Gamma_N)$ and displacement $u_D \in \mathcal{C}(\Gamma_D)$, find $u \in \mathcal{C}^2(\Omega) \cap \mathcal{C}(\overline{\Omega})$ such that

$$\begin{cases} -\,\mathrm{div}\,\sigma(u) = f & \text{in } \Omega \\ \qquad\quad u = u_D & \text{on } \Gamma_D \\ \quad \sigma(u)\cdot n = g & \text{on } \Gamma_N, \end{cases} \tag{2.1}$$

with $\sigma(u) = 2\,\mu\,\varepsilon(u) + \lambda\,\mathrm{div}(u)\,\mathrm{I}$.

Since (2.1) does not always have a solution $u \in \mathcal{C}^2(\Omega) \cap \mathcal{C}(\overline{\Omega})$ we consider the weak formulation of the problem, where the solution u is supposed to be in $H^1(\Omega)$. To this end, we define two trace operators in order to be able to formulate the boundary conditions for the weak formulation.

Definition 2.2.1

(i) The trace operator γ_0 is defined by

$$\gamma_0 u(x) := \lim_{\Omega \ni \widetilde{x} \to x \in \Gamma} u(\widetilde{x}) \qquad \text{a.e. on } \Gamma \tag{2.2}$$

(ii) The co-normal derivative γ_1 is given by

$$\gamma_1 u(x) := \lim_{\Omega \ni \widetilde{x} \to x \in \Gamma} \sigma\left(u(\widetilde{x})\right) n \qquad \text{a.e. on } \Gamma \tag{2.3}$$

where n denotes the outer normal vector of x.

A detailed analysis of the trace operators can be found in [16], e.g. it is shown that these operators are well defined and can be extended to $H^1(\Omega)$:

$$\begin{aligned} \gamma_0 :&\ H^1(\Omega) \to H^{\frac{1}{2}}(\Gamma), \\ \gamma_1 :&\ H^1(\Omega) \to H^{-\frac{1}{2}}(\Gamma). \end{aligned}$$

Now we can state the weak formulation of (2.1):

For given volume forces $f \in \widetilde{H}^{-1}(\Omega)$, traction $g \in H^{-\frac{1}{2}}(\Gamma_N)$ and displacement $u_D \in H^{\frac{1}{2}}(\Gamma_D)$, find $u \in H^1(\Omega)$ such that

$$\begin{cases} -\operatorname{div} \sigma(u) = f & \text{in } \Omega \\ \gamma_0 u = u_D & \text{on } \Gamma_D \\ \gamma_1 u = g & \text{on } \Gamma_N. \end{cases} \tag{2.4}$$

Note that (2.4) holds in a weak sense.

Chapter 3
The hp-BEM for the Navier-Lamé Equation

In this chapter we give an introduction to the hp-BEM for the Navier-Lamé equation. As the basic theory is well known and a detailed analysis is given in [16], [14] and [15] we just give a summary of the main results.
Besides the mixed traction and displacement problem, that we investigate in the first section, we also consider two special cases in this chapter, namely the Dirichlet problem and the Neumann problem.

3.1 Mixed Traction and Displacement Problem

Throughout this section, we consider the weak formulation of the mixed traction and displacement problem (2.4). We assume $\Omega \subset \mathbb{R}^2$ to be a bounded domain with Lipschitz boundary $\Gamma := \partial\Omega$. The boundary is divided into the Neumann and the Dirichlet boundaries, i.e. $\Gamma = \Gamma_D \cup \Gamma_N$ with $|\Gamma_D \cap \Gamma_N| = 0$ and $|\Gamma_D| > 0$. Furthermore, we ignore volume forces, i.e. $f \equiv 0$. First, we state the representation formula, which is also called the Somigliana identity.

Theorem 3.1.1 *Let $u \in H^1(\Omega)$ with $-\mathrm{div}\sigma(u) = 0$ (in a weak sense). Then, there holds for all $x \in \Omega$*

$$u(x) = \int_\Gamma U(x,y)\,\gamma_1 u(y)\,ds_y - \int_\Gamma [\gamma_{1,y} U(x,y)]\,\gamma_0 u(y)\,ds_y, \tag{3.1}$$

where the Kelvin matrix $U(x,y)$ is the fundamental solution of (2.4) and the trace operator $\gamma_{1,y}$ with respect to y is applied to the columns of U, i.e. $\gamma_{1,y}[U^1(x,y), U^2(x,y)] := [\gamma_{1,y}U^1(x,y), \gamma_{1,y}U^2(x,y)]$. The Kelvin-matrix has the following representation regarding the Lamé parameters $\lambda, \mu \in \mathbb{R}$

$$U(x,y) = -\frac{\lambda+3\mu}{4\pi\mu\,(\lambda+2\mu)}\left(\log|x-y|\,\mathrm{I} - \frac{\lambda+\mu}{\lambda+3\mu}\,\frac{(x-y)(x-y)^T}{|x-y|^2}\right), \tag{3.2}$$

where $I \in \mathbb{R}^{2\times 2}$ denotes the identity matrix.

Additionally, it can be shown that the traction $T(x,y) := \gamma_{1,y}U(x,y)$ of the Kelvin matrix has the following representation formula

$$\begin{aligned} T(x,y) =& \frac{\mu}{2\pi(\lambda+2\mu)}\frac{(x-y)^T n(y)}{|x-y|^2}\,\mathrm{I} + \frac{\mu}{2\pi(\lambda+2\mu)}\frac{(x-y)^T t(y)}{|x-y|^2}\,\mathrm{I}\times\mathrm{I} \\ &+ \frac{\lambda+\mu}{\pi(\lambda+2\mu)}\frac{(x-y)^T n(y)}{|x-y|^4}(x-y)(x-y)^T, \end{aligned} \tag{3.3}$$

where $n(y)$ denotes the outer unit normal vector and $t(y)$ denotes the unit tangential vector with respect to $y \in \Gamma$. Moreover, the cross product for matrices is defined as the cross product of the rows and the columns which yields

$$\mathrm{I} \times \mathrm{I} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

The representation formula (3.1) motivates the definition of the following boundary integral operators.

Definition 3.1.2 *Let $x \in \mathbb{R}^2 \backslash \Gamma$, $\phi \in H^{-\frac{1}{2}}(\Gamma)$ and $\psi \in H^{\frac{1}{2}}(\Gamma)$. Then, the single layer potential $\widetilde{V}$ is given by*

$$\widetilde{V}\phi(x) := \int_\Gamma U(x,y)\,\phi(y)\,ds_y \tag{3.4}$$

and the double layer potential $\widetilde{K}$ reads

$$\widetilde{K}\psi(x) := \int_\Gamma [\gamma_{1,y} U(x,y)]\psi(y)\,ds_y. \tag{3.5}$$

One can show that both $\widetilde{V} : H^{-\frac{1}{2}}(\Gamma) \to H^1(\Omega)$ and $\widetilde{K} : H^{\frac{1}{2}}(\Gamma) \to H^1(\Omega)$ are linear and bounded operators. Plugging in the single and the double layer potential, the representation formula now reads

$$u(x) = \widetilde{V}\gamma_1 u(x) - \widetilde{K}\gamma_0 u(x), \quad \forall x \in \Omega. \tag{3.6}$$

Thus, using (3.6) we can calculate the solution u for all $x \in \Omega$ if we know the Cauchy-data $(\gamma_0 u, \gamma_1 u)$ on the boundary Γ. Since the Neumann data $\gamma_1 u$ is only given on the Neumann boundary and the Dirichlet data $\gamma_0 u$ is only given on the Dirichlet boundary, we have to compute the missing Cauchy-data.
In the following we derive a system of boundary integral equations that we use to compute the missing data. We first establish some analytical results.

Lemma 3.1.3 *Let $\widetilde{V} : H^{-\frac{1}{2}}(\Gamma) \to H^1(\Omega)$ be the single layer potential and $\widetilde{K} : H^{\frac{1}{2}}(\Gamma) \to H^1(\Omega)$ be the double layer potential. Moreover, let $x \in \Gamma$, α be the inner angle at x and $\mathrm{B}(x,\varepsilon)$ be the ε ball around x. Then, there holds*

(i) $\gamma_0\widetilde{V} : H^{-\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma)$ defines a linear and bounded operator. For $\phi \in L^\infty(\Gamma)$ there holds

$$\gamma_0\widetilde{V}\phi(x) = \int_\Gamma U(x,y)\,\phi(y)\,ds_y. \tag{3.7}$$

(ii) $\gamma_1\widetilde{V} : H^{-\frac{1}{2}}(\Gamma) \to H^{-\frac{1}{2}}(\Gamma)$ defines a linear and bounded operator. For $\phi \in H^{-\frac{1}{2}}(\Gamma)$ there holds

$$\gamma_1\widetilde{V}\phi(x) = \lim_{\varepsilon \to 0} \int_{\Gamma \backslash \mathrm{B}(x,\varepsilon)} [\gamma_{1,x} U(x,y)]\,\phi(y)\,ds_y + \frac{\alpha}{2\pi}\phi(x), \tag{3.8}$$

(iii) $\gamma_0\widetilde{K} : H^{\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma)$ *defines a linear and bounded operator. For* $\psi \in H^{\frac{1}{2}}(\Gamma)$ *we have to following formula*

$$\gamma_0\widetilde{K}\psi(x) = \left(-1 + \frac{\alpha}{2\pi}\right)\psi(x) + \lim_{\varepsilon\to 0}\int_{\Gamma\setminus \mathrm{B}(x,\varepsilon)} [\gamma_{1,y}U(x,y)]\,\psi(y)\,ds_y. \tag{3.9}$$

(iv) $\gamma_1\widetilde{K} : H^{\frac{1}{2}}(\Gamma) \to H^{-\frac{1}{2}}(\Gamma)$ *defines a linear and bounded operator.*

Note that the integrals in *(ii)* and *(iii)* exist and that $\alpha = \pi$ holds if $x \in \Gamma$ has a normal vector. Lemma 3.1.3 motivates the following definition.

Definition 3.1.4 *Let* $x \in \Gamma$, $\phi \in H^{-\frac{1}{2}}(\Gamma)$ *and* $\psi \in H^{\frac{1}{2}}(\Gamma)$. *Then, we define the following integral operators:*

(i) The single layer operator $V : H^{-\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma)$ *is given by*

$$V\phi(x) := \int_\Gamma U(x,y)\,\phi(y)\,ds_y. \tag{3.10}$$

(ii) The double layer operator $K : H^{\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma)$ *is given by*

$$K\psi(x) := \lim_{\varepsilon\to 0}\int_{\Gamma\setminus \mathrm{B}(x,\varepsilon)} [\gamma_{1,y}U(x,y)]\,\psi(y)\,ds_y. \tag{3.11}$$

(iii) The adjoint operator $K' : H^{-\frac{1}{2}}(\Gamma) \to H^{-\frac{1}{2}}(\Gamma)$ *is given by*

$$K'\phi(x) := \lim_{\varepsilon\to 0}\int_{\Gamma\setminus \mathrm{B}(x,\varepsilon)} [\gamma_{1,x}U(x,y)]\,\phi(y)\,ds_y. \tag{3.12}$$

(iv) The hypersingular operator $W : H^{\frac{1}{2}}(\Gamma) \to H^{-\frac{1}{2}}(\Gamma)$ *is given by*

$$W\psi(x) := -\gamma_1\,\widetilde{K}\psi(x). \tag{3.13}$$

The properties of the integral operators can be summarized by the following lemma.

Lemma 3.1.5

(i) The operators V, K, K' *and* W *are linear and bounded.*

(ii) The single layer operator V *is symmetric with respect to the duality product,*

$$\langle\phi_1, V\phi_2\rangle = \langle\phi_2, V\phi_1\rangle \quad \forall\phi_1,\phi_2 \in H^{-\frac{1}{2}}(\Gamma).$$

(iii) With appropriate scaling of the domain Ω the single layer operator is $H^{-\frac{1}{2}}$-elliptic, in other words $\exists\, \beta > 0$ and $\exists\, c > 0$:

$$\langle\, \phi, V_\beta\, \phi \,\rangle \geq\ c\, \|\phi\|^2_{-\frac{1}{2}} \qquad \forall \phi \in H^{-\frac{1}{2}}(\Gamma),$$

where V_β is given by

$$V_\beta \phi(x) = \int_\Gamma U_\beta(x,y)\, \phi(y)\, ds_y,$$
$$U_\beta(x,y) = -\frac{\lambda + 3\mu}{4\pi\mu\,(\lambda + 2\mu)} \left(\log|\beta(x-y)|\, \mathrm{I} - \frac{\lambda+\mu}{\lambda+3\mu}\, \frac{(x-y)(x-y)^T}{|x-y|^2} \right).$$

(iv) The hypersingular operator is $H^{\frac{1}{2}}$-semi-elliptic, i.e. $\exists\, c > 0$:

$$\langle\, W\psi, \psi \,\rangle \geq\ c\, |\psi|^2_{\frac{1}{2}} \qquad \forall \psi \in H^{\frac{1}{2}}(\Gamma).$$

(v) K' is the adjoint of K with respect to the duality product, i.e

$$\langle\, K'\phi, \psi \,\rangle = \langle\, \phi, K\psi \,\rangle \quad \forall\, \phi \in H^{-\frac{1}{2}}(\Gamma),\, \psi \in H^{\frac{1}{2}}(\Gamma). \tag{3.14}$$

(vi) Let $\frac{d}{ds}$ denote the arc length derivative. Then, there holds $\forall\, \psi_1, \psi_2 \in H^{\frac{1}{2}}(\Gamma)$

$$\langle\, W\psi_1, \psi_2 \,\rangle = \langle\, V^* \frac{d}{ds}\psi_1, \frac{d}{ds}\psi_2 \,\rangle \tag{3.15}$$

with

$$V^*\phi(x) = \int_\Gamma U^*(x,y)\, \phi(y)\, ds_y$$

and

$$U^*(x,y) = -\frac{\mu\,(\lambda+\mu)}{\pi\,(\lambda+2\mu)} \left(\log|x-y|\, \mathrm{I} - \frac{(x-y)(x-y)^T}{|x-y|^2} \right).$$

Note that we do not amplify the choice of the scaling factor β in (*iii*), as the single layer operator is elliptic in the examples considered in the subsequent chapters. We refer to [16] for a detailed description of how to construct an appropriate β.

In order to get a method for computing the missing Cauchy data we apply the trace operators γ_0 and γ_1 to the representation formula (3.6) which yields for $x \in \Gamma$

$$\gamma_0 u(x) = \int_\Gamma U(x,y)\, \gamma_1 u(y)\, ds_y - \left(-1 + \frac{\alpha}{2\pi}\right) \gamma_0 u(x) - \lim_{\varepsilon \to 0} \int_{\Gamma \setminus \mathrm{B}(x,\varepsilon)} [\gamma_{1,y} U(x,y)]\, \gamma_0 u(y)\, ds_y$$

and

$$\gamma_1 u(x) = \lim_{\varepsilon \to 0} \int_{\Gamma \setminus \mathrm{B}(x,\varepsilon)} [\gamma_{1,x} U(x,y)]\, \gamma_1 u(y)\, ds_y + \frac{\alpha}{2\pi} \gamma_1 u(x) - \gamma_{1,x} \widetilde{K} \gamma_0 u(x).$$

Assuming that x has a normal vector, i.e. $\alpha = \pi$, and plugging in the boundary integral operators we write shorter

$$\begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\,\mathrm{I} - K & V \\ W & \frac{1}{2}\,\mathrm{I} + K' \end{pmatrix} \begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix}. \tag{3.16}$$

This system is called the Caldéron-system and can be used to compute the missing Cauchy-data. We choose an arbitrary extension $\overline{u}_D \in H^{\frac{1}{2}}(\Gamma)$ and $\overline{g} \in H^{-\frac{1}{2}}(\Gamma)$ of the given data $u_D \in H^{\frac{1}{2}}(\Gamma_D)$ and $g \in H^{-\frac{1}{2}}(\Gamma_N)$. Then, the missing Cauchy-data is given by

$$\begin{pmatrix} u_N \\ g_D \end{pmatrix} := \begin{pmatrix} \gamma_0 u \\ \gamma_1 u \end{pmatrix} - \begin{pmatrix} \overline{u}_D \\ \overline{g} \end{pmatrix}.$$

It can be proven that $u_N \in \widetilde{H}^{\frac{1}{2}}(\Gamma_N)$ and $g_D \in \widetilde{H}^{-\frac{1}{2}}(\Gamma_D)$. Thus, we can reformulate (3.16) to

$$A \begin{pmatrix} u_N \\ g_D \end{pmatrix} = \left(\frac{1}{2} - A \right) \begin{pmatrix} \overline{u}_D \\ \overline{g} \end{pmatrix} \quad \text{on } \Gamma_N \times \Gamma_D, \tag{3.17}$$

where

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

denotes the Caldéron-projector. To proceed to the equivalent variational formulation we define $\mathcal{V} := \widetilde{H}^{\frac{1}{2}}(\Gamma_N) \times \widetilde{H}^{-\frac{1}{2}}(\Gamma_D)$ and its dual space $\mathcal{V}^* := H^{\frac{1}{2}}(\Gamma_D) \times H^{-\frac{1}{2}}(\Gamma_N)$ using the duality product

$$\langle (\phi_D, \psi_N), (\phi_N, \psi_D) \rangle_{\mathcal{V}^* \times \mathcal{V}} := \langle \psi_D, \phi_D \rangle_{\Gamma_D} + \langle \psi_N, \phi_N \rangle_{\Gamma_N}.$$

Here, $\langle \cdot, \cdot \rangle_{\Gamma_N}$ and $\langle \cdot, \cdot \rangle_{\Gamma_D}$ denote the extended L^2 scalar product on Γ_N and Γ_D, respectively. One can show that the Coldéron-projector defines a linear and bounded operator $\mathrm{A} : \mathcal{V} \to \mathcal{V}^*$. Moreover, we define the continuous bilinear form a on $\mathcal{V} \times \mathcal{V}$ by

$$a\,(\,(u_N, g_D), (\phi_N, \psi_D)\,) := \langle A(u_N, g_D), (\phi_N, \psi_D) \rangle_{\mathcal{V}^* \times \mathcal{V}}. \tag{3.18}$$

The variational formulation that is equivalent to (3.17) reads:

Find $(u_N, g_D) \in \mathcal{V}$ such that $\forall\, (\phi_N, \psi_D) \in \mathcal{V}$ there holds

$$a\,(\,(u_N, g_D), (\phi_N, \psi_D)\,) = \langle \left(\frac{1}{2} - A \right) (\overline{u}_D, \overline{g}), (\phi_N, \psi_D) \rangle_{\mathcal{V}^* \times \mathcal{V}}. \tag{3.19}$$

In order to be able to make a statement concerning the existence and the uniqueness of a solution we cite the following lemma from [16].

Lemma 3.1.6 (Lax-Milgram) *Let X be a Hilbert space, X^* be the dual space of X and let $A : X \to X^*$ be bounded and X-elliptic. Then, there holds*

$$\forall\, f \in X^*\ \exists!\, u \in X :\ Au = f.$$

Theorem 3.1.7 (Existence of a unique solution) *The integral formulation of the mixed traction and displacement problem* (3.17) *and the equivalent variational formulation* (3.19)*, respectively, have an unique solution.*

Proof. Due to the previous results and the Lax-Milgram lemma it remains to show that the operator A is $\mathcal{V}$-elliptic. Using the $H^{-\frac{1}{2}}$-ellipticity of V, the $H^{\frac{1}{2}}$-semi-ellipticity of W and the continuity of both operators one can show that

$$\|(u_N, g_D)\|_A^2 := \langle V g_D, g_D\rangle_{\Gamma_D} + \langle W u_N, u_N\rangle_{\Gamma_N}$$

defines a norm on $\mathcal{V}$ that is equivalent to the usual product norm $\|\cdot\|_{\mathcal{V}}$. Moreover, let $(u_N, g_D) \in \mathcal{V}$, then there holds

$$\begin{aligned}\langle A\,(u_N, g_D), (u_N, g_D)\rangle_{\mathcal{V}^*\times\mathcal{V}} &= \langle -K u_N + V g_D, g_D\rangle_{\Gamma_D} + \langle W u_N + K' g_D, u_N\rangle_{\Gamma_N}\\ &= \langle V g_D, g_D\rangle_{\Gamma_D} + \langle W u_N, u_N\rangle_{\Gamma_N}\\ &= \|(u_N, g_D)\|_{\mathrm{A}}^2 \geq c\,\|(u_N, g_D)\|_{\mathcal{V}}^2,\end{aligned}$$

which proofs the $\mathcal{V}$-ellipticity of A. □

In order to solve the variational formulation (3.19) numerically, we prescribe that $\Omega \subset \mathbb{R}^2$ is a polygonal Lipschitz domain with the boundary $\Gamma := \partial\Omega$. Note that if Ω is not a polygonal domain we have to approximate it by a polygonal domain which leads to an approximation error. However, this case is not considered in this work. Let $\mathcal{T}_h$ be a triangulation of the boundary Γ with

$$\mathcal{T}_h := \left\{ \Gamma_\ell = \mathrm{conv}\{A_\ell, B_\ell\},\ \ell = 1, ..., N_{el} :\ \Gamma = \bigcup_{\ell=1}^{N_{el}} \Gamma_\ell,\ |\Gamma_\ell \cap \Gamma_m| = \delta_{\ell m}|\Gamma_\ell| \right\},$$

where $h := \max\{|\Gamma_\ell| \,:\, 1 \leq \ell \leq N_{el}\}$. Note that we call the elements of $\mathcal{T}_h$ boundary elements. In order to be able to solve the mixed problem with different boundary conditions, we define the triangulations $\mathcal{T}_{h,D}$ of the Dirichlet and $\mathcal{T}_{h,N}$ of the Neumann boundaries, where we assume $\mathcal{T}_h = \mathcal{T}_{h,D} \cup \mathcal{T}_{h,N}$. Note that this definition implies that for all $\Gamma_\ell \in \mathcal{T}_h$ we get either $\Gamma_\ell \in \mathcal{T}_{h,D}$ or $\Gamma_\ell \in \mathcal{T}_{h,N}$. Furthermore, we denote by $N_{el,D}$ and $N_{el,N}$ the number of boundary elements on the Dirichlet and the Neumann boundaries, respectively.
In order to get a discrete problem, we approximate the infinite-dimensional space $\mathcal{V} = \widetilde{H}^{\frac{1}{2}}(\Gamma_N) \times \widetilde{H}^{-\frac{1}{2}}(\Gamma_D)$ by an appropriate finite-dimensional space $\mathcal{V}_h := S^1(\mathcal{T}_{h,N}, p_1) \times S^0(\mathcal{T}_{h,D}, p_0)$ where $p_0 \in \mathbb{N}_0^{N_{el,D}}$, $p_1 \in \mathbb{N}^{N_{el,N}}$. $S^1(\mathcal{T}_{h,N}, p_1)$ contains piecewise polynomial, globally continuous functions, i.e.

$$S^1(\mathcal{T}_{h,N}, p_1) := \left\{ \psi_h^{p_1} \in C(\Gamma_N) \,:\, \psi_h^{p_1}\big|_{\Gamma_\ell} \in \mathbb{P}_{p_{1,\ell}}, \ell = 1, ..., N_{el,N} \right\}.$$

and $S^0(\mathcal{T}_{h,D}, p_0)$ contains piecewise polynomial, globally discontinuous functions, i.e.

$$S^0(\mathcal{T}_{h,D}, p_0) := \left\{ \phi_h^{p_0} \in L^2(\Gamma_D) \,:\, \phi_h^{p_0}\big|_{\Gamma_\ell} \in \mathbb{P}_{p_{0,\ell}}, \ell = 1, ..., N_{el,D} \right\}.$$

Note that we obtain a conforming method for solving the discrete problem, since there holds $\mathcal{V}_h \subset \mathcal{V}$. The discrete problem reads

Find $(u_{N,h}, g_{D,h}) \in \mathcal{V}_h$ such that $\forall\, (\phi_{N,h}, \psi_{D,h}) \in \mathcal{V}_h$ there holds

$$a\,(\,(u_{N,h}, g_{D,h}), (\phi_{N,h}, \psi_{D,h})\,) = \langle \left(\frac{1}{2} - A\right)(\overline{u}_D, \overline{g}), (\phi_{N,h}, \psi_{D,h})\rangle_{\mathcal{V}^* \times \mathcal{V}}. \tag{3.20}$$

To make a statement concerning the solvability of the discrete problem, we cite the following lemma from [16].

Lemma 3.1.8 (Céa) *Let X be a Hilbert space, $a : X \times X \to \mathbb{R}$ a continuous and X-elliptic bilinear form and $f \in X'$ a continuous linear form. Moreover, let $X_h \subset X$ be a finite-dimensional test- and ansatz-space. Then, the discrete variational formulation*

$$a\,(u_h, v_h) = f(v_h) \quad \forall\, v_h \in X_h$$

has a unique solution $u_h \in X_h$ and u_h is the best approximation in the sense that

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

In particular, the Galerkin orthogonality holds, i.e.

$$\langle u - u_h, v_h\rangle_X = 0 \quad \forall\, v_h \in X_h.$$

The fact that the Caldéron-projector A is $\mathcal{V}$-elliptic and continuous implies the continuity and the ellipticity of the bilinear form a and thus, we can apply the Céa lemma to (3.20). Hence, (3.20) is well defined and the solution $(u_{N,h}, g_{D,h})$ converges quasi-optimal to the exact solution.
To solve (3.20) we choose a basis $(\Theta_j)_{j=1,\dots,\dim \mathcal{V}_h}$ of $\mathcal{V}_h$ and we write $(u_{N,h}, g_{D,h}) = \sum_{k=1}^{\dim \mathcal{V}_h} \alpha_k \Theta_k$, $\alpha_k \in \mathbb{R}$. Testing (3.20) with the basis functions Θ_j and plugging in the representation of $(u_{N,h}, g_{D,h})$ yields

$$\sum_{k=1}^{\dim \mathcal{V}_h} \alpha_k\, a\,(\Theta_k, \Theta_j) = \langle \left(\frac{1}{2} - A\right)(\overline{u}_D, \overline{g}), \Theta_j\rangle_{\mathcal{V}^* \times \mathcal{V}} \quad \forall\, j = 1, \dots, \dim \mathcal{V}_h. \tag{3.21}$$

In order to write (3.21) shorter, we define the Galerkin matrix $\mathbf{A} := (a_{j,k})_{j,k=1,\dots,\dim \mathcal{V}_h}$ by

$$a_{j,k} := a\,(\,\Theta_k, \Theta_j\,),$$

the mass matrix $\mathbf{M} = (m_{j,k})_{j,k=1,\dots,\dim \mathcal{V}_h}$ by

$$m_{j,k} := \langle\, \Theta_j, \Theta_k\,\rangle_{\mathcal{V}^* \times \mathcal{V}},$$

the coefficient vector $\mathbf{x} := (\alpha_k)_{k=1,\dots,\dim \mathcal{V}_h}$ and the coefficient vector $\mathbf{b} := (b_j)_{j=1,\dots,\dim \mathcal{V}_h}$ of $(\overline{u}_D, \overline{g})$. Finally, the discrete problem is given by the system of linear equations

$$\mathbf{A}\,\mathbf{x} = \left(\frac{1}{2}\,\mathbf{M} - \mathbf{A}\right)\mathbf{b}.$$

Describing the Galerkin matrix more closely we denote by $(\Phi_j)_{j=1,\dots,\dim S^0(\mathcal{T}_h,p_0)}$ the basis of $S^0(\mathcal{T}_h,p_0)$ and by $(\Psi_j)_{j=1,\dots,\dim S^1(\mathcal{T}_h,p_1)}$ the basis of $S^1(\mathcal{T}_h,p_1)$. Then, the Galerkin matrix of the single layer operator is defined by $\mathbf{V} := (V_{i,j})_{i,j=1,\dots,\dim S^0(\mathcal{T}_h,p)}$ with $V_{i,j} := \langle V\Phi_j, \Phi_i\rangle$, the Galerkin matrix of the double layer operator is defined by $\mathbf{K} := (K_{i,j})_{i=1,\dots,\dim S^0(\mathcal{T}_h,p)}^{j=1,\dots,\dim S^1(\mathcal{T}_h,p_1)}$ with $K_{i,j} := \langle K\Psi_j, \Phi_i\rangle$ and the Galerkin matrix of the hypersingular operator is given $\mathbf{W} := (W_{i,j})_{i,j=1,\dots,\dim S^1(\mathcal{T}_h,p)}$ with $W_{i,j} := \langle W\Psi_j, \Psi_i\rangle$. Then, there holds

$$\mathbf{A} = \begin{pmatrix} -\mathbf{K} & \mathbf{V} \\ \mathbf{W} & \mathbf{K}^T \end{pmatrix}.$$

Note that we used the fact that K' the adjoint of K which yields that the Galerkin matrix of K' is the transposed Galerkin matrix of the double layer operator.

In the end of this section we give an important result concerning the a priori error analysis for the mixed traction and displacement problem.

Theorem 3.1.9 *Let $(\psi,\phi) \in \mathcal{V}$ be the solution of* (3.19)*. Moreover, let $\mathcal{T}_h$ be a triangulation of Γ with $h := \max\{|\Gamma_\ell| : \Gamma_\ell \in \mathcal{T}_h\}$ and $(\psi_h^{p_1}, \phi_h^{p_0}) \in \mathcal{V}_h = S^1(\mathcal{T}_h,p_1)\times S^0(\mathcal{T}_h,p_0)$ denote the unique solution of the discrete problem* (3.20) *with polynomial degrees $p_0 \in \mathbb{N}_0$ and $p_1 \in \mathbb{N}$ on the mesh $\mathcal{T}_h$. Prescribing the regularity of the exact solution $(\psi,\phi) \in H^t_{pw}(\Gamma_N) \times H^s_{pw}(\Gamma_D)$ for $s,t \geq 0$, there holds*

$$\begin{aligned} &\left\|(\psi,\phi) - (\psi_h^{p_1},\phi_h^{p_0})\right\|_{H^{\frac{1}{2}}(\Gamma)\times H^{-\frac{1}{2}}(\Gamma)} \\ &\quad\lesssim \left(h^{\min\{t-\frac{1}{2},p_1+\frac{1}{2}\}}\left\|\psi\right\|_{H^t_{pw}(\Gamma_N)} + h^{\min\{s+\frac{1}{2},p_0+\frac{3}{2}\}}\left\|\phi\right\|_{H^s_{pw}(\Gamma_D)}\right). \end{aligned} \tag{3.22}$$

Proof. See [15], page 174. □

(3.22) shows that if the exact solution is smooth, the convergence rate of the error is only bounded by the polynomial degree of the discrete solution and if the exact solution is not smooth, e.g. in the neighborhood of a singularity, small polynomial degrees suffice to obtain the maximal convergence rate. This motivates the implementation of an *hp*-method. The idea of the *hp*-method is that we increase the polynomial degree on the boundary elements, on which the exact solution is smooth, and refine the boundary mesh near a singularity of the exact solution in order to obtain the maximal convergence rate. Thereby, we implement both geometric and adaptive *hp*-methods. Implementing a geometric *hp*-method, we know a priori where the singularity of the exact solution is and thus, we prescribe a geometric *hp*-refinement, i.e. we refine the boundary near the singularity and choose $p = 0$ on the smallest element and increase the polynomial degree on the bigger elements. Implementing an adaptive *hp*-method we control the *hp*-refinement by an appropriate error estimator. The numerical results in Chapter 7 show an exponential convergence rate for both the geometric and the adaptive *hp*-methods. Note that the exponential convergence is not yet proven theoretically for adaptive *hp*-methods, whereas in [17] there is a proof for the geometric *hp*-method.
In addition to the *hp*-method, we also implement adaptive *h*- and uniform *h*- and *p*-methods. Note that in [17] it is proven that the convergence rate of the uniform *p*-method is twice as big compared to the convergence rate of the uniform *h*-method.

3.2 The Dirichlet Problem and Symm's Integral Equation

In this section we investigate the Dirichlet problem, which reads

$$\begin{cases} -\operatorname{div}\sigma(u) = 0 & \text{in } \Omega \\ \qquad \gamma_0 u = u_D & \text{on } \Gamma, \end{cases} \tag{3.23}$$

where $u_D \in H^{\frac{1}{2}}(\Gamma)$ denotes the given Dirichlet data. In order to solve the Dirichlet problem we compute the missing Neumann data $\phi \in H^{-\frac{1}{2}}(\Gamma)$ using the first equation of the Caldéron-sytem, which is called Symm's integral equation and reads

$$V\phi = \left(K + \frac{1}{2}\,\mathrm{I}\right) u_D. \tag{3.24}$$

Moreover, we consider Symm's integral equation in a more general way, i.e. we solve

$$V\phi = f \tag{3.25}$$

for an arbitrary function $f \in H^{\frac{1}{2}}(\Gamma)$. Defining the bilinear-form a on $H^{-\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$ by

$$a(\phi, w) := \langle V\phi, w\rangle$$

leads to the equivalent variational formulation:

$$a(\phi, w) = \langle f, w\rangle \quad \forall\, w \in H^{-\frac{1}{2}}(\Gamma). \tag{3.26}$$

Since the single layer operator V is $H^{-\frac{1}{2}}$-elliptic, the lemma of Lax-Milgram implies the existence of a unique solution.
With the triangulation $\mathcal{T}_h$ of Γ and the polynomial degree $p := p_0 \in \mathbb{N}_0^{N_{el}}$ we obtain the discrete problem:

Find $\phi_h^p \in S^0(\mathcal{T}_h, p)$, such that $\forall\, w_h^p \in S^0(\mathcal{T}_h, p)$ there holds

$$a(\phi_h^p, w_h^p) = \langle f, w_h^p\rangle. \tag{3.27}$$

Note that we can apply the Céa lemma since the single layer operator is $H^{-\frac{1}{2}}$-elliptic and thus, (3.27) has a unique solution. Choosing a basis $(\Phi_j)_{j=1,\dots,\dim S^0(\mathcal{T}_h,p)}$ we can rewrite (3.27) to a system of linear equations. Therefore, we define the coefficient vector $\mathbf{x}$ of ϕ_h^p with respect to basis Φ_i and the right-hand side $\mathbf{b} := (b_i)_{i=1,\dots,\dim S^0(\mathcal{T}_h,p)}$ with $b_i = \langle f, \Phi_i\rangle$. Then, (3.27) reads

$$\mathbf{V}\mathbf{x} = \mathbf{b},$$

where $\mathbf{V}$ denotes the Galerkin matrix of the single layer operator. In the special case of the Dirichlet problem (3.24), we define the mass matrix $\mathbf{M} := (M_{i,j})_{i=1,\dots,\dim S^0(\mathcal{T}_h,p)}^{j=1,\dots,\dim S^1(\mathcal{T}_h,p_1)}$ with $M_{i,j} := \langle \Psi_j, \Phi_i\rangle$ and assume $\mathbf{u_D}$ to be the coefficient

vector of $u_{D,h}$ with respect to the basis Ψ_j. Using the Galerkin matrix of the double layer operator we obtain the following system:

$$\mathbf{Vx} = \left(\mathbf{K} + \frac{1}{2}\mathbf{M}\right) \mathbf{u_D}.$$

Concerning the a priori error analysis we state a result that is proven in [15], page 161:

Theorem 3.2.1 (Dirichlet problem) *Let $\mathcal{T}_h$ be a triangulation of Γ with $h := \max\{|\Gamma_\ell| : \Gamma_\ell \in \mathcal{T}_h\}$. Moreover, let $s \geq 0$ and $\phi \in H^s(\Gamma)$ be the exact solution of the Dirichlet problem (3.24) and $\phi_h^p \in S^0(\mathcal{T}_h, p)$ the solution of the discrete problem with polynomial degree $p \in \mathbb{N}_0$ on the discrete mesh $\mathcal{T}_h$. Then, there holds*

$$\|\phi - \phi_h^p\|_{H^{-\frac{1}{2}}(\Gamma)} \lesssim h^{\frac{1}{2}+\min\{s,p+1\}} \|\phi\|_{H^s(\Gamma)}. \tag{3.28}$$

3.3 The Neumann Problem and the Hypersingular Integral Equation

In this section we solve the Neumann problem

$$\begin{cases} -\mathrm{div}\sigma(u) = 0 & \text{in } \Omega \\ \gamma_1 u = g & \text{on } \Gamma. \end{cases} \tag{3.29}$$

We compute the missing Dirichlet data $u \in H^{\frac{1}{2}}(\Gamma)$ with the second equation of the Caldéron system, namely the hypersingular integral equation, which reads

$$Wu = \left(\frac{1}{2}\mathrm{I} - K'\right) g. \tag{3.30}$$

Here, $g \in H^{-\frac{1}{2}}(\Gamma)$ denotes the Neumann data. To solve (3.30) we consider the variational formulation:

$$\text{Find } u \in H^{\frac{1}{2}}(\Gamma): \qquad \langle Wu, v\rangle = \langle\left(\frac{1}{2}\mathrm{I} - K'\right) g, v\rangle \quad \forall\, v \in H^{\frac{1}{2}}(\Gamma). \tag{3.31}$$

Since the hypersingular operator W is not $H^{\frac{1}{2}}$-elliptic, we cannot apply Lemma 3.1.6 (Lax-Milgram) and thus, (3.31) does not have a unique solution. In order to obtain a problem that has a unique solution we introduce some constraints in the following.
We first consider the homogeneous integral equation , i.e. $g \equiv 0$ in (3.30). Let

$$\mathcal{R} := \mathrm{span}\{v_1,\, v_2,\, v_3\} \tag{3.32}$$

with $v_1 = (1,0)^T$, $v_2 = (0,1)^T$ and $v_3 = (x_2, -x_1)^T$. Plugging in v_k $(k = 1, ..., 3)$ into the second equation of the Caldéron-system (3.16) yields $Wv_k = 0$, where

we use the fact that $\gamma_1 v_k = 0$ and $\sigma(v_k) = 0$. Thus, $\mathcal{R}$ is the solution space of the homogeneous integral equation. Note that in linear elasticity, v_1 and v_2 correspond to the translations of the body in x_1- and x_2-direction, respectively, and v_3 corresponds to the rotation of the body and consequentially, $\mathcal{R}$ is the set of all rigid body motions in two dimensions. Thus, the solution of the non-homogeneous integral equation is unique except for the rigid body motions. In order to obtain a problem with a unique solution, we define the space $H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma) := \left\{ v \in H^{\frac{1}{2}}(\Gamma) \,:\, \langle v, w\rangle = 0,\ \forall w \in \mathcal{R} \right\}$ of all $H^{\frac{1}{2}}(\Gamma)$-functions that are orthogonal to the rigid body motions and consider the variational formulation

$$\text{Find } u \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma): \qquad \langle Wu, v\rangle = \langle \left(\frac{1}{2} - K'\right) g, v\rangle \qquad \forall v \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma). \tag{3.33}$$

Lemma 3.3.1 *(i) The hypersingular operator W is $H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$-elliptic, i.e.*

$$\exists\, c > 0 \,:\quad \langle Wv, v\rangle \,\geq c\, \|v\|^2_{\frac{1}{2}} \qquad \forall v \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma).$$

(ii) Let $g \in H^{-\frac{1}{2}}(\Gamma)$ be the given Neumann data. Then, for all $v \in \mathcal{R}$ there holds

$$\langle \gamma_0 v,\, g\rangle = 0, \tag{3.34}$$

which is a constraint to the Neumann data for the solvability of the problem.

Proof.

ad(i) See [16], page 158.

ad(ii) The second Betti formula which is proven in [16], page 18, reads

$$\begin{aligned} &-\int_\Omega \mathrm{div}\sigma(u)^T v\, dx + \int_\Gamma \gamma_0 v^T\, \gamma_1 u\, ds_x \\ &\quad = -\int_\Omega \mathrm{div}\sigma(v)^T u\, dx + \int_\Gamma \gamma_0 u^T\, \gamma_1 v\, ds_x. \end{aligned} \tag{3.35}$$

Let u be the solution of (3.29) with $-\mathrm{div}\sigma(u) = 0$ and $\gamma_1 u = g$ (in a weak sense) and $v \in \mathcal{R}$. Plugging in u and v into the second Betti formula yields

$$\langle \gamma_0 v,\, g\rangle = 0.$$

□

Due to (i) in Lemma 3.3.1 we are in the framework of the Lax-Milgram lemma and thus, (3.33) has a unique solution $u \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$. However, instead of discretizing $H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$ which is numerically difficult due to the restrictions $\langle w,\, v\rangle = 0, \forall w \in H^{\frac{1}{2}}(\Gamma)$

and $\forall v \in \mathcal{R}$, we use the restrictions to build a stabilization of the hypersingular operator. We define for $u, v \in H^{\frac{1}{2}}(\Gamma)$

$$\langle\!\langle u, v \rangle\!\rangle_{W+S} := \langle Wu,\, v\rangle + \sum_{k=1}^{3} \langle u,\, v_k\rangle\, \langle v_k,\, v\rangle,$$

where v_k denote the translations in x_1- and x_2-direction and the rotation, respectively, and solve the modified variational problem

$$\text{Find } u \in H^{\frac{1}{2}}(\Gamma): \qquad \langle\!\langle u, v \rangle\!\rangle_{W+S} = \langle \left(\frac{1}{2} - K'\right) g, v\rangle \qquad \forall v \in H^{\frac{1}{2}}(\Gamma). \tag{3.36}$$

In the following lemma we proof the equivalence of (3.33) and (3.36).

Lemma 3.3.2 *The variational formulations* (3.33) *and* (3.36) *are equivalent.*

Proof. First, we proof that if u is the solution of (3.36) we get $u \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$. Since $\{v_k, k = 1, ..., 3\}$ is a (non-orthogonal) basis of $\mathcal{R}$ we choose $\{w_1, w_2, w_3\} \subset \mathcal{R}$ to be an orthonormal basis of $\mathcal{R}$. There holds

$$v_k = \sum_{j=1}^{3} a_{j,k} w_j \quad k = 1, ..., 3, \tag{3.37}$$

where the matrix $A = (a_{j,k})_{j,k=1,..,3}$ is regular. Moreover, there holds

$$\begin{aligned}\langle \left(\frac{1}{2}\,\mathrm{I} - K'\right) g, w_j\rangle &= \underbrace{\langle g, w_j\rangle}_{=0} - \langle \left(\frac{1}{2}\,\mathrm{I} + K'\right) g, w_j\rangle \\ &= -\langle g, \left(\frac{1}{2}\,\mathrm{I} + K\right) w_j\rangle = 0,\end{aligned}$$

where we used (ii) in Lemma 3.3.1 to obtain the second identity, and the first equation of the Caldéron System and $\gamma_1 w_j = 0$ to obtain the last identity. Applying the second equation of the Caldéron System we get

$$\langle Wu, w_j\rangle = 0.$$

Testing (3.36) with w_j we obtain

$$\langle\!\langle u, w_j \rangle\!\rangle_{W+S} = \langle Wu,\, w_j\rangle + \sum_{k=1}^{3} \langle u,\, v_k\rangle\, \langle v_k,\, w_j\rangle = \langle \left(\frac{1}{2} - K'\right) g, w_j\rangle$$

which reduces to

$$\begin{aligned} &\sum_{k=1}^{3} \langle u,\, v_k\rangle\, \langle v_k,\, w_j\rangle = 0, \quad j = 1, .., 3 \\ \Leftrightarrow \quad & A \begin{pmatrix} \langle u, v_1\rangle \\ \langle u, v_2\rangle \\ \langle u, v_3\rangle \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \end{aligned}$$

Since A is regular we have $\langle u, v_k\rangle = 0$, $k = 1, ..., 3$, which implies $u \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$. Additionally, if $u \in H^{\frac{1}{2}}_{\mathcal{R}}(\Gamma)$ solves (3.33), (3.36) reduces to

$$\langle Wu,\, v\rangle = \langle \left(\frac{1}{2} - K'\right)\, g, v\rangle.$$

Thus, both problems are equivalent. □

The discrete problem of (3.36) reads with $p = p_1$

$$\text{Find } u_h^p \in S^1(\mathcal{T}_h, p): \qquad \langle\!\langle u_h^p, v_h^p\rangle\!\rangle_{W+S} = \langle \left(\frac{1}{2} - K'\right)\, g_h, v_h^p\rangle \qquad \forall v_h^p \in S^1(\mathcal{T}_h, p),$$

where g_h denotes the projection of the Neumann data to $S^0(\mathcal{T}_h, p_0)$. In order to solve the discrete problem we define the stabilization matrix $\mathbf{S} := (S_{i,j})_{i,j=1,..,p+1}$ with

$$S_{i,j} := \sum_{k=1}^{3} \langle \Psi_j, v_k\rangle\, \langle v_k, \Psi_i\rangle.$$

Moreover, let $\mathbf{M}$ be the mass matrix and $\mathbf{K}$ be the Galerkin matrix of the double layer operator. Defining the coefficient vectors $\mathbf{g}$ of g_h with respect to the basis of $S^0(\mathcal{T}_h, p_0)$ and $\mathbf{u}$ of u_h^p with respect to the basis of $S^1(\mathcal{T}_h, p)$ the system of linear equations reads

$$(\mathbf{W} + \mathbf{S})\mathbf{u} = \left(\frac{1}{2}\mathbf{M} - \mathbf{K}^T\right)\mathbf{g}.$$

Besides the Neumann problem we also consider the hypersingular equation in a more general way, i.e. we solve

$$Wu = h \tag{3.38}$$

for an arbitrary function $h \in H^{-\frac{1}{2}}(\Gamma)$. For solving (3.38) we proceed as above, i.e. the corresponding system of linear equations with stabilization reads

$$(\mathbf{W} + \mathbf{S})\mathbf{u} = \mathbf{b},$$

where $\mathbf{b} := (b_i)_{i=1,...,\dim S^1(\mathcal{T}_h,p)}$ with $b_i := \langle h, \Psi_i\rangle$.

Concerning the a priori error analysis we state a result that is proven in [15], page 171:

Theorem 3.3.3 (Neumann problem) *Let $\mathcal{T}_h$ be a triangulation of Γ with $h := \max\{|\Gamma_\ell| : \Gamma_\ell \in \mathcal{T}_h\}$. Moreover, let $s \geq \frac{1}{2}$ and $\psi \in H^s_{pw}(\Gamma)$ be the exact solution of the Neumann problem* (3.30) *and $\psi_h^p \in S^1(\mathcal{T}_h, p)$ the solution of the discrete problem with polynomial degree $p \in \mathbb{N}$ on the discrete mesh $\mathcal{T}_h$. Then, there holds*

$$\|\psi - \psi_h\|_{H^{\frac{1}{2}}(\Gamma)} \lesssim h^{\min\{s,p+1\}-\frac{1}{2}}\, \|\psi\|_{H^s_{pw}(\Gamma)}. \tag{3.39}$$

Chapter 4
Associated Legendre Functions and Related Functions

In this chapter, we establish a basis for the assembly of the Galerkin matrices. As it is shown in Chapter 5, we can reduce the calculation of the Galerkin entries to the calculation of double integrals of a specific type. Therefore, we investigate the calculation of these integrals, here.
In the first section, we give the theory we need for the calculation of the double integrals. We introduce the Legendre polynomials, the Lobatto shape functions and state some important properties. Note that we use the Legendre polynomials and the Lobatto shape functions for the definition of the basis functions of the ansatz- and test-spaces $S^0(\mathcal{T}_h, p_0)$ and $S^1(\mathcal{T}_h, p_1)$ in order to derive the *hp*-BEM for the Navier-Lamé equation for an arbitrary polynomial degree p. Furthermore, we introduce the associated Legendre functions and related functions that we need to discuss the calculation of the double integrals in the second section.

4.1 Associated Legendre Functions and Lobatto Shape Functions

First, we define the Legendre polynomials by a three-term recurrence relation.

Definition 4.1.1 *The Legendre polynomials are defined for $z \in \mathbb{C}$ by*

$$P_{k+1}(z) = \frac{2k+1}{k+1}\, z\, P_k(z) - \frac{k}{k+1}\, P_{k-1}(z), \quad k \geq 1. \tag{4.1}$$

with the initial values $P_0(z) = 1,\ P_1(z) = z$.

Note that this definition shows one of the most important advantages of the Legendre polynomials. Due to the three-term recurrence relation it is possible to evaluate the Legendre polynomials $P_k(z)$ with complexity $\mathcal{O}(k)$. Moreover, the computation is well-conditioned for $|z| < 1$, especially for $z \in [-1, 1]$ (see [11]). Some elementary properties are stated in the following lemma.

Lemma 4.1.2

(i) The Legendre polynomials are orthogonal with respect to the L^2 scalar product,

$$\int_{-1}^{1} P_k(t)\, P_m(t)\, dt = \frac{2}{2k+1}\, \delta_{km}\,, \ \ k, m \in \mathbb{N}_0. \tag{4.2}$$

(ii) For all $k \in \mathbb{N}_0$ there holds $P_k(1) = 1$ and $P_k(-1) = (-1)^k$.

(iii) The antiderivative of the Legendre polynomial P_k is given by

$$\int_{-1}^{t} P_k(\xi)\, d\xi = \frac{1}{2k+1}(P_{k+1}(t) - P_{k-1}(t))\,,\ k \geq 1. \tag{4.3}$$

(iv) There holds

$$\int_{-1}^{1}\int_{-1}^{1} P_k(s)\, P_m(t)\, ds\, dt = 4\delta_{k0}\,\delta_{m0}\,,\ k, m \in \mathbb{N}_0 \tag{4.4}$$

and

$$\int_{-1}^{1}\int_{-1}^{1} \frac{P_k(s)\, P_m(t)}{s-t}\, ds\, dt = \begin{cases} \dfrac{4}{(k+m+1)(m-k)} & ,m-k \text{ odd} \\ 0 & \text{else} \end{cases} \tag{4.5}$$

Proof. A proof of (i), (iii) and (iv) can be found in [10], and (ii) is proven in [13] page 79. □

Property (iii) in Lemma 4.1.2 motivates the definition of the Lobatto shape functions as antiderivatives of the Legendre polynomials.

Definition 4.1.3 *Let $k \geq 3$ and $t \in [-1, 1]$. Then, the Lobatto shape functions are defined by*

$$N_1(t) = \frac{1-t}{2}, \quad N_2(t) = \frac{1+t}{2}, \quad N_k(t) = \int_{-1}^{t} P_{k-2}(\xi)\, d\xi.$$

Note that in the literature, e.g. in [17], the Lobatto shape functions N_k are scaled with the factor $\frac{1}{\|P_{k-2}\|}$. However, for our application to integral equations we omit this factor.
The following properties can be derived immediately from the properties of the Legendre polynomials.

Lemma 4.1.4

(i) The Lobatto shape functions are related to the Legendre polynomials by

$$N_k(t) = \frac{1}{2k-3}(P_{k-1}(t) - P_{k-3}(t))\,,\ k \geq 3$$

(ii) The Lobatto shape functions fulfill the three-term recurrence relation

$$N_{k+1}(t) = \frac{2k-3}{k}\, t\, N_k(t) - \frac{k-3}{k}\, N_{k-1}(t)\,,\ k \geq 4$$

with the initial values $N_3(t) = \frac{1}{2}(t^2-1)$, $N_4(t) = \frac{1}{2}\, t\,(t^2-1)$

(iii) *There holds* $N_k(\pm 1) = 0$, $k \geq 3$.

(iv) *For* $m \geq 3$, *there holds*

$$\int_{-1}^{1}\int_{-1}^{1} \frac{P_k(s)\, N_m(t)}{s-t}\, ds\, dt = \begin{cases} -\frac{8}{(m-k-1)(m+k)(m-k-3)(m+k-2)} & , \text{ for } m-k \text{ even} \\ 0 & , \text{ otherwise,} \end{cases}$$

for $m = 2$ *we obtain*

$$\int_{-1}^{1}\int_{-1}^{1} \frac{P_k(s)\, N_2(t)}{s-t}\, ds\, dt = \begin{cases} -\frac{2}{k(k+1)} & , \text{ for } k \text{ odd} \\ -\frac{2}{(k-1)(k+2)} & , \text{ otherwise} \end{cases}$$

and for $m = 1$ *we get*

$$\int_{-1}^{1}\int_{-1}^{1} \frac{P_k(s)\, N_1(t)}{s-t}\, ds\, dt = \begin{cases} -\frac{2}{k(k+1)} & , \text{ for } k \text{ odd} \\ \frac{2}{(k-1)(k+2)} & , \text{ otherwise.} \end{cases}$$

Proof. The definition of the Lobatto shape functions and Lemma 4.1.2 (*iii*) imply (*i*). For the proof of the three-term recurrence relation (*ii*) see [11]. (*iii*) results from Lemma 4.1.2 (*ii*) and from (*i*). The representation of the Lobatto function of (*i*) and Lemma 4.1.2 (*iv*) imply (*iv*) . □

For the introduction of the Legendre functions we are guided by the works of [12] and [13] and summarize the results that are relevant for the application to the *hp*-BEM.
Let us consider the associated Legendre differential equation which is given by

$$\frac{d}{dz}\left\{(1-z^2)\,\frac{du}{dz}\right\} + \left[k(k+1) - \frac{m^2}{z^2-1}\right] u = 0, \tag{4.6}$$

where we assume m, $k \in \mathbb{N}_0$ and $z \in \mathbb{C}\backslash[-1,1]$. We start investigating (4.6) for $m = 0$. In this case, (4.6) is called the Legendre differential equation and all solutions of (4.6) are called Legendre functions. The Legendre polynomials that we introduced above solve the Legendre equation and hence, they are also called the Legendre functions of first kind. It is shown in [12] (page 51) that every solution of (4.6) has the following representation

$$u(z) = A\, P_k(z) + B\, \left(\frac{1}{2} P_k(z)\, \log\frac{z+1}{z-1} - W_{k-1}(z)\right), \quad A, B \in \mathbb{C} \tag{4.7}$$

with polynomials $W_{k-1}(z)$ that are defined by

$$W_{-1}(z) = 0, \quad W_0(z) = 1, \quad W_k(z) = \frac{2k+1}{k+1}\, z\, W_{k-1}(z) - \frac{k}{k+1}\, W_{k-2}(z). \tag{4.8}$$

This motivates the definition of the Legendre functions of second kind Q_k by

$$Q_k(z) := \frac{1}{2} P_k(z)\, \log\frac{z+1}{z-1} - W_{k-1}(z). \tag{4.9}$$

An important relationship to the Legendre polynomials is given by Neumann's formula which is stated in the following theorem.

Theorem 4.1.5 *Let $k \in \mathbb{N}_0$ and $z \in \mathbb{C}\backslash[-1,1]$. Then, there holds*

$$Q_k(z) = \frac{1}{2} \int_{-1}^{1} \frac{P_k(t)}{z-t}\, dt. \tag{4.10}$$

Proof. See [12], page 63/67. □

Considering the general case $m \in \mathbb{N}_0$ we define the associated Legendre functions as solutions of the associated Legendre equation. The associated Legendre functions of second kind $Q_k^m(z)$ are given by

$$Q_k^m(z) := (1-z^2)^{\frac{m}{2}} \frac{d^m}{dz^m} Q_k(z). \tag{4.11}$$

and solve (4.6). With this definition we can generalize Neumann's formula as follows.

Theorem 4.1.6 *Let $k, m \in \mathbb{N}_0$ and $z \in \mathbb{C}\backslash[-1,1]$. Then, there holds*

$$Q_k^m(z) := (-1)^m \frac{1}{2} m!\, (1-z^2)^{\frac{m}{2}} \int_{-1}^{1} \frac{P_k(t)}{(z-t)^{m+1}}\, dt. \tag{4.12}$$

Proof. See [12], page 116. □

Since we only need the integral in (4.12) for the application to the *hp*-BEM, we give the following definition.

Definition 4.1.7 *Let $k, m \in \mathbb{N}_0$ and $z \in \mathbb{C}\backslash[-1,1]$. We define*

$$\widetilde{Q}_k^m(z) := \int_{-1}^{1} \frac{P_k(t)}{(z-t)^{m+1}}\, dt \tag{4.13}$$

$$\widetilde{Q}_k^{-1}(z) := \int_{-1}^{1} P_k(t)\, \log(z-t)\, dt \tag{4.14}$$

$$\widetilde{Q}_k^{-2}(z) := \int_{-1}^{1} P_k(t)\, (z-t) \log(z-t)\, dt. \tag{4.15}$$

Referring to [10] and [12] we summarize the properties of the function $\widetilde{Q}_k^m(z)$ without a proof.

Lemma 4.1.8 *Let $z \in \mathbb{C}\backslash[-1,1]$.*

(i) For $m \geq -1$ and $k \geq \max\{1, 1-m\}$, there holds the three-term recurrence relation

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

(ii) For $k \in \mathbb{N}$ there holds the following symmetry property

$$\begin{aligned}
&\operatorname{Re}\widetilde{Q}_k^{-1}(z) = (-1)^k \operatorname{Re}\widetilde{Q}_k^{-1}(-z)\\
&\operatorname{Im}\widetilde{Q}_k^{-1}(z) = (-1)^k \operatorname{Im}\widetilde{Q}_k^{-1}(-z)\\
&\operatorname{Re}\widetilde{Q}_0^{-1}(z) = (-1)^k \operatorname{Re}\widetilde{Q}_0^{-1}(-z)\\
&\operatorname{Im}\widetilde{Q}_0^{-1}(z) = \operatorname{Im}\widetilde{Q}_0^{-1}(-z) \begin{cases} +2\pi\, i & , \text{ for } 0 < \arg(z) < \pi \\ -2\pi\, i & , \text{ otherwise.} \end{cases}
\end{aligned} \tag{4.16}$$

(iii) The functions $\widetilde{Q}_k^{-1}(z)$ and $\widetilde{Q}_k^{-2}(z)$ can be continuously extended to $z = 1$, i.e. the limits $\lim_{z\to 1}\widetilde{Q}_k^{-1}(z)$ and $\lim_{z\to 1}\widetilde{Q}_k^{-2}(z)$ exist.

(iv) The function $\widetilde{Q}_0^{-1}(z)$ is not one-valued for all $z \in (-\infty, -1]$, i.e the imaginary part of $\widetilde{Q}_0^{-1}(z)$ has a jump.

(v) $\widetilde{Q}_k^0$ can be represented by

$$\widetilde{Q}_k^0(z) = P_k(z)\,\log\frac{z+1}{z-1} - 2\,W_{k-1}(z), \tag{4.17}$$

where the polynomials W_k are defined by (4.8).

4.2 Special Integrals Related to the Associated Legendre Functions

In this section we first define some special integrals we need for the calculation of the Galerkin matrices. Referring to [19], we introduce the following notations.

Definition 4.2.1 *Let $a, b \in \mathbb{C}$, with $a + b \notin [-1, 1]$ and $a - b \notin [-1, 1]$. Then, we define for $j, k, p \in \mathbb{N}_0$*

$$I_{j,k}^{-1}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\,P_k(t)\,\log(a + b\,s - t)\,dt\,ds\ , \tag{4.18}$$

$$I_{j,k}^{p}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\,P_k(t)\,\frac{1}{(a + b\,s - t)^{p+1}}\,dt\,ds \tag{4.19}$$

and for $j, p \in \mathbb{N}_0$ and $k \in \mathbb{N}$

$$O_{j,k}^{-1}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\,N_k(t)\,\log(a + b\,s - t)\,dt\,ds\ , \tag{4.20}$$

$$O_{j,k}^{p}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\,N_k(t)\,\frac{1}{(a + b\,s - t)^{p+1}}\,dt\,ds. \tag{4.21}$$

Moreover, we extend this definition and introduce two more integrals we need for the calculation of the Galerkin matrices.

Definition 4.2.2 *Let $a, b \in \mathbb{C}$, with $a + b \notin [-1, 1]$ and $a - b \notin [-1, 1]$. Then, we define for $j, k, p \in \mathbb{N}_0$*

$$\widetilde{I}^{-1}_{j,k}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(a + b\, s)\, \log(a + b\, s - t)\, dt\, ds\ , \tag{4.22}$$

$$\widetilde{I}^{p}_{j,k}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(a + b\, s)\, \frac{1}{(a + b\, s - t)^{p+1}}\, dt\, ds, \tag{4.23}$$

and for $j, p \in \mathbb{N}_0$ and $k \in \mathbb{N}$

$$\widetilde{O}^{-1}_{j,k}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, N_k(t)\, \mathrm{Im}(a + b\, s)\, \log(a + b\, s - t)\, dt\, ds\ , \tag{4.24}$$

$$\widetilde{O}^{p}_{j,k}(a,b) := \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, N_k(t)\, \mathrm{Im}(a + b\, s)\, \frac{1}{(a + b\, s - t)^{p+1}}\, dt\, ds. \tag{4.25}$$

Since we only need $I^{-1}_{j,k}$, $\widetilde{I}^{0}_{j,k}$, $O^{0}_{j,k}$ and $\widetilde{O}^{1}_{j,k}$ for the application to the BEM we restrict to investigate these integrals.
The calculation via recurrence relations and the stable and efficient implementation of $I^{p}_{j,k}$ and $O^{p}_{j,k}$ ($p \in \mathbb{N}_0$) is described in [19] for $a \pm b \notin [-1, 1]$. However, for calculating the Galerkin matrices we get the special case $a \pm b = \pm 1$ considering neighboring elements (see Section 5.1). Therefore, we continuously extend the integrals to $a \pm b = \pm 1$ and obtain the following lemma.

Lemma 4.2.3 *Let $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}} \in \mathbb{C}\backslash[-1, 1]$ with $\lim_{n\to\infty} a_n = a \in \mathbb{C}$, $\lim_{n\to\infty} b_n = b \in \mathbb{C}$ and $a \pm b = \pm 1$. Then, there holds for $j \in \mathbb{N}_0$, $k \in \mathbb{N}$*

(i) $\lim_{n\to\infty} I^{-1}_{j,k}(a_n, b_n) = I^{-1}_{j,k}(a, b)$

(ii) $\lim_{n\to\infty} I^{0}_{j,k}(a_n, b_n) = I^{0}_{j,k}(a, b)$

(iii) $\lim_{n\to\infty} \widetilde{I}^{0}_{j,k}(a_n, b_n) = \widetilde{I}^{0}_{j,k}(a, b)$

(iv) $\lim_{n\to\infty} O^{0}_{j,k}(a_n, b_n) = O^{0}_{j,k}(a, b)$

In particular, all limits in $(i) - (iv)$ exist.

Proof. Since all integrals are calculated via recurrence relations we only investigate the initial values.

ad(i) See [10].

ad(ii) According to [19] the initial values of $I^{0}_{j,k}(a_n, b_n)$ are calculated with $\widetilde{Q}^{-1}_{k}(a_n + b_n)$, $\widetilde{Q}^{-1}_{k}(a_n - b_n)$ and $\widetilde{Q}^{-1}_{k}\left(\frac{1-a_n}{b_n}\right)$, $\widetilde{Q}^{-1}_{k}\left(\frac{-1-a_n}{b_n}\right)$, respectively. Note that $a+b = \pm 1 \Leftrightarrow \frac{\pm 1-a}{b} = 1$ and $a - b = \pm 1 \Leftrightarrow \frac{\pm 1-a}{b} = -1$. Therefore, we have to investigate the limit $\lim_{z\to\pm 1} \widetilde{Q}^{-1}_{k}(z)$ in order to proof (ii). Due to Lemma

4.1.8 (*iii*) the limit exists for $z \to 1$. For $z \to -1$, we use the symmetry of $\widetilde{Q}_k^{-1}(z)$ (Lemma 4.1.8 (*ii*)) which yields

$$\lim_{z\to -1} \widetilde{Q}_k^{-1}(z) = (-1)^{k+1} \lim_{z\to 1} \widetilde{Q}_k^{-1}(z) + \kappa\, 2\pi\, i,$$

where $\kappa = \pm 1$ depends on the argument of z.

ad(iii) With the linearity of the integral we get

$$\widetilde{I}^0_{j,k}(a_n, b_n) = \mathrm{Im}(a_n)\, I^0_{j,k}(a_n, b_n) + \mathrm{Im}(b_n) \int_{-1}^{1}\int_{-1}^{1} \frac{s\, P_k(t)\, P_j(s)}{a_n + b_n\, s - t}\, ds\, dt$$

Using the three-term recurrence relation of the Legendre polynomials (4.1) yields for $j \geq 1$

$$s\, P_j(s) = \frac{j+1}{2j+1}\, P_{j+1}(s) + \frac{j}{2j+1}\, P_{j-1}(s)$$

and $s\, P_0(s) = P_1(s)$, respectively. Thus, there holds for $j \geq 1$

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} \frac{s\, P_k(t)\, P_j(s)}{a_n + b_n\, s - t}\, ds\, dt \\
&= \left[\frac{j+1}{2j+1} \int_{-1}^{1}\int_{-1}^{1} \frac{P_k(t)\, P_{j+1}(s)}{a_n + b_n\, s - t}\, ds\, dt + \frac{j}{2j+1} \int_{-1}^{1}\int_{-1}^{1} \frac{P_k(t)\, P_{j-1}(s)}{a_n + b_n\, s - t}\, ds\, dt \right]
\end{aligned}$$

and

$$\int_{-1}^{1}\int_{-1}^{1} \frac{s\, P_k(t)\, P_0(s)}{a_n + b_n\, s - t}\, ds\, dt = \int_{-1}^{1}\int_{-1}^{1} \frac{P_k(t)\, P_1(s)}{a_n + b_n\, s - t}\, ds\, dt.$$

Hence,

$$\begin{aligned}
\widetilde{I}^0_{j,k}(a_n, b_n) &= \mathrm{Im}(a_n)\, I^0_{j,k}(a_n, b_n) \\
&\quad + \mathrm{Im}(b_n) \left[\frac{j+1}{2j+1}\, I^0_{j+1,k}(a_n, b_n) + \frac{j}{2j+1}\, I^0_{j-1,k}(a_n, b_n) \right] \qquad (4.26)
\end{aligned}$$

and

$$\widetilde{I}^0_{0,k}(a_n, b_n) = \mathrm{Im}(a_n)\, I^0_{0,k}(a_n, b_n) + \mathrm{Im}(b)\, I^0_{1,k}(a_n, b_n). \qquad (4.27)$$

With (*ii*) we obtain (*iii*).

ad(iv) With the definition of the Lobatto shape functions N_1 and N_2 and Lemma 4.1.4 (*i*) we get for $k \geq 3$

$$\begin{aligned}
O^0_{j,1}(a_n, b_n) &= \frac{1}{2} \left(I^0_{j,0}(a_n, b_n) - I^0_{j,1}(a_n, b_n) \right) \\
O^0_{j,2}(a_n, b_n) &= \frac{1}{2} \left(I^0_{j,0}(a_n, b_n) + I^0_{j,1}(a_n, b_n) \right) \\
O^0_{j,k}(a_n, b_n) &= \frac{1}{2k-3} \left(I^0_{j,k-1}(a_n, b_n) - I^0_{j,k-3}(a_n, b_n) \right).
\end{aligned}$$

Applying (*ii*) completes the proof.

□

With this result we see that we can use the results of [19], (4.26) and (4.27) for the calculation of $I_{j,k}^{-1}$, $\widetilde{I}_{j,k}^{0}$ and $O_{j,k}^{0}$. Therefore, we do not go into detail about the calculation of these integrals.
It still remains to investigate the integral $\widetilde{O}_{j,k}^{1}$ for the calculation of the Galerkin matrices. According to [19], the initial values are calculated by $\widetilde{Q}_k^0(z)$, which cannot be extended to $z = 1$. Thus, we cannot prove the existence of an extension to $a \pm b = \pm 1$ in the same way as in Lemma 4.2.3.
In the following, we first discuss the calculation of $\widetilde{O}_{j,k}^{1}$ for $a \pm b \notin [-1, 1]$, where we derive formulas that are different from the formulas in [19], and then we show that $\widetilde{O}_{j,k}^{1}$ can be extended to $a \pm b = \pm 1$.
By analogy with (4.26) and (4.27) we get

$$\widetilde{O}_{j,k}^{1}(a,b) = \operatorname{Im}(a)\, O_{j,k}^{1}(a,b) + \operatorname{Im}(b) \left[\frac{j+1}{2j+1} O_{j+1,k}^{1}(a,b) + \frac{j}{2j+1} O_{j,k}^{1}(a,b) \right] \tag{4.28}$$

and

$$\widetilde{O}_{0,k}^{1}(a,b) = \operatorname{Im}(a)\, O_{0,k}^{1}(a,b) + \operatorname{Im}(b)\, O_{1,k}^{1}(a,b). \tag{4.29}$$

Hence, we discuss the calculation of $O_{j,k}^{1}(a,b)$ which is done in the following way:

$$\left(\begin{array}{c|c|c|ccccc} \vdots & \vdots & \vdots & \cdots & \cdots & O_{0,k}^{1} & \cdots & \cdots \\ \hline \vdots & \vdots & \vdots & \cdots & \cdots & O_{1,k}^{1} & \cdots & \cdots \\ \hline O_{j,1}^{1} & O_{j,2}^{1} & O_{j,3}^{1} & & & & & \\ \vdots & \vdots & \vdots & & & O_{j,k}^{1} & & \\ \vdots & \vdots & \vdots & & & & & \end{array}\right). \tag{4.30}$$

This structure for the computation results from the fact that the Lobatto shape functions N_1 and N_2 do not fulfill a recurrence relation, and thus we have to compute the first an the second column, separately. Moreover, we calculate the $O_{j,k}^{1}(a,b)$, $j \geq 2$, $k \geq 4$, via the four-term recurrence relation

$$\begin{array}{ccc} & * & \\ * & * & \\ & * & \end{array}$$

which is given in [19] (Theorem 19) by

$$O_{j,k}^{1}(a,b) = -\frac{a}{b}\frac{2j-1}{j+k} O_{j-1,k}^{1}(a,b) - \frac{j-k-1}{j+k} O_{j-2,k}^{1}(a,b) + \frac{1}{b}\frac{2j-1}{j+k} O_{j-1,k-1}^{1}(a,b).$$

The initial values for this recurrence relation are $O_{0,k}^{1}$, $O_{1,k}^{1}$ and $O_{j,3}^{1}$. Thus, these values have to be calculated first.
We start investigating the first and second column and obtain the following lemma.

Lemma 4.2.4 *Let $a, b \in \mathbb{C}$, with $a \pm b \notin [-1,1]$ and $j \in \mathbb{N}$. Then, there holds*

$$O^1_{0,1}(a,b) = \frac{1}{2}\left\{\widetilde{Q}_0^{-1}\left(\eta_1 \frac{-1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1 \frac{1-a}{b}\right)\right\} + \frac{1}{b}\widetilde{Q}_0^0\left(\frac{-1-a}{b}\right)$$

$$O^1_{j,1}(a,b) = \frac{1}{2(2j+1)}\left\{\widetilde{Q}_{j+1}^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_{j-1}^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_{j+1}^0\left(\frac{1-a}{b}\right) + \widetilde{Q}_{j-1}^0\left(\frac{1-a}{b}\right)\right\} + \frac{1}{b}\widetilde{Q}_j^0\left(\frac{-1-a}{b}\right)$$

and

$$O^1_{0,2}(a,b) = -\frac{1}{2}\left\{\widetilde{Q}_0^{-1}\left(\eta_1 \frac{-1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1 \frac{1-a}{b}\right)\right\} - \frac{1}{b}\widetilde{Q}_0^0\left(\frac{1-a}{b}\right)$$

$$O^1_{j,2}(a,b) = \frac{1}{2(2j+1)}\left\{\widetilde{Q}_{j+1}^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_{j-1}^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_{j+1}^0\left(\frac{-1-a}{b}\right) + \widetilde{Q}_{j-1}^0\left(\frac{-1-a}{b}\right)\right\} - \frac{1}{b}\widetilde{Q}_j^0\left(\frac{1-a}{b}\right)$$

with

$$\eta_1 = \begin{cases} 1 & , \textit{for } \operatorname{conv}\left\{\frac{1-a}{b}, \frac{-1-a}{b}\right\} \cap \{x \in \mathbb{R} \,:\, x \le -1\} = \emptyset \\ -1 & , \textit{otherwise.} \end{cases} \tag{4.31}$$

Proof. We start proving the representation of $O^1_{0,1}$ and $O^1_{0,2}$. For $a, b \in \mathbb{C}$ with $a \pm b \notin [-1,1]$, there holds

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} \frac{t}{(a+b\,s-t)^2}\,ds\,dt \\
&\quad= \int_{-1}^{1}\left(\log(-a-b\,s+t) + \frac{a+b\,s}{a+b\,s-t}\right)\Big|_{t=-1}^{1} ds \\
&\quad= \int_{-1}^{1} \log\left(b\left(\frac{1-a}{b}-s\right)\right) - \log\left(b\left(\frac{-1-a}{b}-s\right)\right) ds \\
&\qquad+ \int_{-1}^{1} \frac{a+b\,s}{a+b\,s-1} - \frac{a+b\,s}{a+b\,s+1}\,ds.
\end{aligned}$$

For the first integral we obtain with

$$\log(ab) = log(a) + \log(b) + \begin{cases} -2\pi\,i, & \text{for } \arg(a)+\arg(b) \ge \pi \\ +2\pi\,i, & \text{for } \arg(a)+\arg(b) < \pi \end{cases}$$

depending on the arguments of the complex numbers $\frac{1-a}{b}$, $\frac{1-a}{b}$ and b

$$\begin{aligned}
&\int_{-1}^{1} \log\left(b\left(\frac{1-a}{b}-s\right)\right) - \log\left(b\left(\frac{-1-a}{b}-s\right)\right) ds \\
&= \int_{-1}^{1} \log\left(\frac{1-a}{b}-s\right) - \log\left(\frac{-1-a}{b}-s\right) ds + \int_{-1}^{1} \kappa\, 2\pi i\, ds \\
&= \widetilde{Q}_0^{-1}\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\frac{-1-a}{b}\right) + \kappa\, 4\pi i,
\end{aligned}$$

where $\kappa \in \{-1, 0, 1\}$. Note that instead of adding $\kappa\, 4\pi i$ explicitly we use the symmetry property of $\widetilde{Q}_0^{-1}$ in Lemma 4.1.8 (ii) which yields

$$\begin{aligned}&\int_{-1}^{1} \log\left(b\left(\frac{1-a}{b} - s\right)\right) - \log\left(b\left(\frac{-1-a}{b} - s\right)\right) ds \\ &= \widetilde{Q}_0^{-1}\left(\eta_1 \frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1 \frac{-1-a}{b}\right),\end{aligned}$$

with

$$\eta_1 = \begin{cases} 1 & \text{, for conv}\left\{\frac{1-a}{b}, \frac{-1-a}{b}\right\} \cap \{x \in \mathbb{R} : x \le -1\} = \emptyset \\ -1 & \text{, otherwise.}\end{cases}$$

For the second integral we obtain

$$\begin{aligned}&\int_{-1}^{1} \frac{a + b\,s}{a + b\,s - 1} - \frac{a + b\,s}{a + b\,s + 1}\, ds \\ &= -\frac{1}{b}\int_{-1}^{1} \frac{a + b\,s}{\frac{1-a}{b} - s} - \frac{a + b\,s}{\frac{-1-a}{b} - s}\, ds \\ &= -\frac{1}{b}\int_{-1}^{1} a\left(\frac{1}{\frac{1-a}{b} - s} - \frac{1}{\frac{-1-a}{b} - s}\right) + b\left(\frac{s}{\frac{1-a}{b} - s} - \frac{s}{\frac{-1-a}{b} - s}\right) ds. \qquad (4.32)\end{aligned}$$

With $\frac{s}{z-s} = \frac{z}{z-s} - 1$, $z \in \mathbb{C}$, we simplify (4.32) and get

$$\begin{aligned}&\int_{-1}^{1} \frac{a + b\,s}{a + b\,s - 1} - \frac{a + b\,s}{a + b\,s + 1}\, ds \\ &= -\frac{1}{b}\int_{-1}^{1} \frac{1}{\frac{1-a}{b} - s} - \frac{1}{\frac{-1-a}{b} - s}\, ds \\ &= \frac{1}{b}\left(\widetilde{Q}_0^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_0^0\left(\frac{1-a}{b}\right)\right).\end{aligned}$$

Thus, we have

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} \frac{t}{(a + b\,s - t)^2}\, ds\, dt \\ &= \widetilde{Q}_0^{-1}\left(\eta_1 \frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1 \frac{-1-a}{b}\right) + \frac{1}{b}\left(\widetilde{Q}_0^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_0^0\left(\frac{1-a}{b}\right)\right).\end{aligned}$$

With

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} \frac{1}{(a + b\,s - t)^2}\, dt\, ds \\ &= \int_{-1}^{1} \frac{1}{a + b\,s - 1} - \frac{1}{a + b\,s + 1}\, ds \\ &= -\frac{1}{b}\int_{-1}^{1} \frac{1}{\frac{1-a}{b} - s} - \frac{1}{\frac{-1-a}{b} - s}\, ds \\ &= -\frac{1}{b}\left(\widetilde{Q}_0^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^0\left(\frac{-1-a}{b}\right)\right)\end{aligned}$$

and the definition of $N_1(t) = \frac{1}{2}(1-t)$ and $N_2(t) = \frac{1}{2}(1+t)$ we get the formula for $O^1_{0,1}(a,b)$ and $O^1_{0,2}(a,b)$.
For $j \in \mathbb{N}$ we use the following result which is proven in [19] (Theorem 16):

$$I^1_{j,1}(a,b) = b\,\frac{j}{2j+1}\,I^1_{j+1,0}(a,b) + b\,\frac{j+1}{2j+1}\,I^1_{j-1,0}(a,b) + a\,I^1_{j,0}(a,b). \tag{4.33}$$

With the definition of $N_1(t)$ and (4.33) we obtain

$$\begin{aligned} O^1_{j,1}(a,b) &= \frac{1}{2}\left(I^1_{j,0}(a,b) - I^1_{j,1}(a,b)\right) \\ &= \frac{1}{2}\left((1-a)\,I^1_{j,0}(a,b) - b\,\frac{j}{2j+1}\,I^1_{j+1,0}(a,b) - b\,\frac{j+1}{2j+1}\,I^1_{j-1,0}(a,b)\right). \end{aligned}$$

With the initial values of $I^1_{j,k}$ ([19], Lemma 20 combined with Lemma 17):

$$I^1_{j,0}(a,b) = \frac{1}{b}\left(\widetilde{Q}^0_0\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_0\left(\frac{1-a}{b}\right)\right) \tag{4.34}$$

we obtain

$$\begin{aligned} O^1_{j,1}(a,b) = \frac{1}{2}\Bigg\{ &\frac{1-a}{b}\left[\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_j\left(\frac{1-a}{b}\right)\right] \\ &- \frac{j}{2j+1}\left[\widetilde{Q}^0_{j+1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j+1}\left(\frac{1-a}{b}\right)\right] \\ &- \frac{j+1}{2j+1}\left[\widetilde{Q}^0_{j+1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j+1}\left(\frac{1-a}{b}\right)\right]\Bigg\}. \end{aligned}$$

Applying the three-term recurrence relation of $\widetilde{Q}^0_k$ (Lemma 4.1.8 (i)), which reads

$$z\widetilde{Q}^0_j(z) = \frac{j+1}{2j+1}\widetilde{Q}^0_{j+1}(z) + \frac{j}{2j+1}\widetilde{Q}^0_{j-1}(z),$$

to $\widetilde{Q}^0_j\left(\frac{1-a}{b}\right)$ and $\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right)$, we obtain

$$\begin{aligned} O^1_{j,1}(a,b) = \frac{1}{2(2j+1)}\Bigg\{ &\widetilde{Q}^0_{j+1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j-1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j+1}\left(\frac{1-a}{b}\right) \\ &+ \widetilde{Q}^0_{j-1}\left(\frac{1-a}{b}\right)\Bigg\} + \frac{1}{b}\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right). \end{aligned}$$

Note that we added and subtracted $\frac{1}{b}\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right)$. The proof of the formula for $O^1_{j,2}(a,b)$ can be done similarly and is omitted here. □

The formula for the calculation of the third column is given in the next lemma.

Lemma 4.2.5 *Let $a, b \in \mathbb{C}$, with $a \pm b \notin [-1,1]$, $j \geq 2$ and η_1 as defined in* (4.31). *Then, there holds*

$$\begin{aligned} O^1_{0,3}(a,b) = &\,a\left[\widetilde{Q}^{-1}_0\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}^{-1}_0\left(\eta_1\frac{-1-a}{b}\right)\right] + b\left[\widetilde{Q}^{-1}_1\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}^{-1}_1\left(\eta_1\frac{-1-a}{b}\right)\right] \\ &- \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{1}{3}b\right)\left[\widetilde{Q}^0_0\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_0\left(\frac{-1-a}{b}\right)\right] \\ &- a\left[\widetilde{Q}^0_1\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_1\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}b\left[\widetilde{Q}^0_2\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_2\left(\frac{-1-a}{b}\right)\right] + 2, \end{aligned}$$

$$
\begin{aligned}
O^1_{1,3}(a,b) = a&\left[\widetilde{Q}^{-1}_1\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}^{-1}_1\left(\eta_1\frac{-1-a}{b}\right)\right] + \frac{2}{3}b\left[\widetilde{Q}^{-1}_2\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}^{-1}_2\left(\eta_1\frac{-1-a}{b}\right)\right]\\
&+ \frac{1}{3}b\left[\widetilde{Q}^{-1}_0\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}^{-1}_0\left(\eta_1\frac{-1-a}{b}\right)\right]\\
&- \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{3}{5}b\right)\left[\widetilde{Q}^0_1\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_1\left(\frac{-1-a}{b}\right)\right]\\
&- \frac{2}{3}a\left[\widetilde{Q}^0_2\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_2\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}a\left[\widetilde{Q}^0_0\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_0\left(\frac{-1-a}{b}\right)\right]\\
&- \frac{1}{5}b\left[\widetilde{Q}^0_3\left(\frac{1-a}{b}\right) - \widetilde{Q}^0_3\left(\frac{-1-a}{b}\right)\right] + 2,
\end{aligned}
$$

and

$$
\begin{aligned}
O^1_{j,3}(a,b) = a&\,\frac{j}{(j+2)(2j+1)}\left\{\widetilde{Q}^0_{j+1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j+1}\left(\frac{1-a}{b}\right)\right\}\\
&+ \left[\frac{a^2-1}{b}\,\frac{1}{j+2} + b\,\frac{j-1}{(j+2)(2j-1)}\right]\left\{\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_j\left(\frac{1-a}{b}\right)\right\}\\
&+ a\,\frac{3j+2}{(j+2)(2j+1)}\left\{\widetilde{Q}^0_{j-1}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j-1}\left(\frac{1-a}{b}\right)\right\}\\
&+ b\,\frac{j}{(j+2)(2j-1)}\left\{\widetilde{Q}^0_{j-2}\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_{j-2}\left(\frac{1-a}{b}\right)\right\}.
\end{aligned}
$$

Proof. The proof can be found in Appendix A. □

The formulas for the calculation of the first and the second row of (4.30) is given in the following lemma.

Lemma 4.2.6 *Let $a, b \in \mathbb{C}$, with $a \pm b \notin [-1,1]$ and $k \geq 3$. Then there holds*

$$
O^1_{0,k}(a,b) = \frac{1}{b\,(2k-3)}\left[\widetilde{Q}^0_{k-1}(a-b) - \widetilde{Q}^0_{k-1}(a+b) - \widetilde{Q}^0_{k-3}(a-b) + \widetilde{Q}^0_{k-3}(a+b)\right]
$$

and

$$
\begin{aligned}
O^1_{1,k}(a,b) = \frac{1}{b^2\,(2k-3)}\Big\{&\widetilde{Q}^{-1}_{k-1}(\eta_2(a+b)) - \widetilde{Q}^{-1}_{k-1}(\eta_2(a-b))\\
&- \widetilde{Q}^{-1}_{k-3}(\eta_2(a+b)) + \widetilde{Q}^{-1}_{k-3}(\eta_2(a-b))\\
&- \frac{1}{b}\left[\widetilde{Q}^0_{k-1}(a+b) + \widetilde{Q}^0_{k-1}(a-b) - \widetilde{Q}^0_{k-3}(a+b) - \widetilde{Q}^0_{k-3}(a-b)\right]\Big\}.
\end{aligned}
$$

with

$$
\eta_2 = \begin{cases} 1 & , \textit{for } \operatorname{conv}\{a+b,\, a-b\} \cap \{x \in \mathbb{R} :\ x \leq -1\} = \emptyset \\ -1 & , \textit{otherwise.} \end{cases} \tag{4.35}
$$

Proof. We start with proving the representation of $O^1_{0,k}$. There holds

$$\begin{aligned}
O^1_{0,k} &= \int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)}{(a+b\,s-t)^2}\,ds\,dt \\
&= \frac{1}{b^2}\int_{-1}^{1} N_k(t) \int_{-1}^{1} \frac{1}{\left(-\frac{a}{b}+\frac{1}{b}t-s\right)^2}\,ds\,dt \\
&= \frac{1}{b^2}\int_{-1}^{1} N_k(t) \left(\frac{1}{-1-\frac{a}{b}+\frac{1}{b}t} - \frac{1}{1-\frac{a}{b}+\frac{1}{b}t}\right) dt \\
&= -\frac{1}{b}\int_{-1}^{1} N_k(t) \left(\frac{1}{a+b-t} - \frac{1}{a-b-t}\right) dt.
\end{aligned}$$

Applying the representation of the Lobatto shape functions in Lemma 4.1.4 (i) and using the definition of $\widetilde{Q}^0_k$ we get the formula for $O^1_{0,k}$. For the second identity we obtain analogously

$$\begin{aligned}
O^1_{1,k} &= \int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\,s}{(a+b\,s-t)^2}\,ds\,dt \\
&= \frac{1}{b^2}\int_{-1}^{1} N_k(t) \int_{-1}^{1} \frac{s}{\left(-\frac{a}{b}+\frac{1}{b}t-s\right)^2}\,ds\,dt \\
&= \frac{1}{b^2}\int_{-1}^{1} N_k(t) \left(\log\left(s+\frac{a}{b}-\frac{1}{b}t\right) + \left(\frac{1}{b}t-\frac{a}{b}\right)\frac{1}{-\frac{a}{b}+\frac{1}{b}t-s}\right)\Bigg|_{s=-1}^{1} dt \\
&= \frac{1}{b^2}\int_{-1}^{1} N_k(t) \left\{\log\left(1+\frac{a}{b}-\frac{1}{b}t\right) - \log\left(-1+\frac{a}{b}-\frac{1}{b}t\right)\right. \\
&\qquad\qquad \left. + \left(\frac{1}{b}t-\frac{a}{b}\right)\left(\frac{1}{-\frac{a}{b}+\frac{1}{b}t-1} - \frac{1}{-\frac{a}{b}+\frac{1}{b}t+1}\right)\right\} dt. \qquad (4.36)
\end{aligned}$$

As it is done in the proof of Lemma 4.2.4 we simplify the logarithm terms in (4.36) by

$$\begin{aligned}
&\int_{-1}^{1} N_k(t) \left\{\log\left(1+\frac{a}{b}-\frac{1}{b}t\right) - \log\left(-1+\frac{a}{b}-\frac{1}{b}t\right)\right\} dt \\
&\quad= \int_{-1}^{1} N_k(t) \left\{\log\left(\frac{1}{b}\,(a+b-t)\right) - \log\left(\frac{1}{b}\,(a-b-t)\right)\right\} dt \\
&\quad= \int_{-1}^{1} N_k(t) \left\{\log\left(a+b-t\right) - \log\left(a-b-t\right)\right\} dt + \kappa\,2\pi i \int_{-1}^{1} N_k(t)\,dt,
\end{aligned}$$

where $\kappa \in \{-1,0,1\}$. Since $\int_{-1}^{1} N_k(t)\,dt = \frac{1}{2k-3}\left(\int_{-1}^{1} P_{k-1}(t)\,dt - \int_{-1}^{1} P_{k-3}(t)\,dt\right) = -\frac{2}{3}\,\delta_{k,3}$ we obtain

$$\begin{aligned}
&\int_{-1}^{1} N_k(t) \left\{\log\left(1+\frac{a}{b}-\frac{1}{b}t\right) - \log\left(-1+\frac{a}{b}-\frac{1}{b}t\right)\right\} dt \\
&\quad= \frac{1}{2k-3}\left\{\widetilde{Q}^{-1}_{k-1}\left(\eta_2\left(a+b\right)\right) - \widetilde{Q}^{-1}_{k-1}\left(\eta_2\left(a-b\right)\right) - \widetilde{Q}^{-1}_{k-3}\left(\eta_2\left(a+b\right)\right) + \widetilde{Q}^{-1}_{k-3}\left(\eta_2\left(a-b\right)\right)\right\},
\end{aligned}$$

where we define η_2 by

$$\eta_2 = \begin{cases} 1 & \text{, for } \operatorname{conv}\{a+b,\, a-b\} \cap \{x \in \mathbb{R} \,:\, x \leq -1\} = \emptyset \\ -1 & \text{, otherwise.} \end{cases}$$

Again, we added $\kappa \frac{4}{3} \pi i \, \delta_{k,3}$ implicitly by point reflection of $a+b$ and $a-b$ using the symmetry property of $\widetilde{Q}_0^{-1}$. For the remaining terms in (4.36) we get

$$\begin{aligned} &\int_{-1}^{1} N_k(t) \left\{ \left(\frac{1}{b} t - \frac{a}{b} \right) \left(\frac{1}{-\frac{a}{b} + \frac{1}{b} t - 1} - \frac{1}{-\frac{a}{b} + \frac{1}{b} t + 1} \right) \right\} dt \\ &= \int_{-1}^{1} N_k(t) \left(\frac{a-t}{a+b-t} - \frac{a-t}{a-b-t} \right) dt \\ &= -b \int_{-1}^{1} N_k(t) \left(\frac{1}{a+b-t} - \frac{1}{a-b-t} \right) dt \\ &= \frac{b}{2k-3} \left(\widetilde{Q}_{k-1}^0(a+b) - \widetilde{Q}_{k-1}^0(a-b) - \widetilde{Q}_{k-3}^0(a+b) + \widetilde{Q}_{k-3}^0(a-b) \right). \end{aligned}$$

Note that we used $\frac{t}{z-t} = \frac{z}{z-t} - 1$, $z \in \mathbb{C}$, for the second identity. Putting the results together completes the proof. □

With the last three lemmas, (4.28) and (4.29) we can calculate $\widetilde{O}_{j,k}^1(a,b)$ provided that $a \pm b \neq \pm 1$. However, it still remains to investigate the existence of an extension of $\widetilde{O}_{j,k}^1$ for $a \pm b = \pm 1$ which is done in the following theorem.

Theorem 4.2.7 *Let* $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}} \in \mathbb{C}\backslash[-1,1]$ *with* $\lim_{n\to\infty} a_n = a \in \mathbb{C}$, $\lim_{n\to\infty} b_n = b \in \mathbb{C}$ *and* $a \pm b = \pm 1$. *Then, there holds for* $j \in \mathbb{N}_0$, $k \in \mathbb{N}$

$$\lim_{n\to\infty} \widetilde{O}_{j,k}^1(a_n, b_n) = \widetilde{O}_{j,k}^1(a,b).$$

Proof. To proof the existence of an extension we use the representation (4.17) of $\widetilde{Q}_k^0$ which reads

$$\widetilde{Q}_k^0(z) = P_k(z) \, \log \frac{z+1}{z-1} - 2\, W_{k-1}(z).$$

In particular, we divide the representation of $\widetilde{Q}_k^0(z)$ into a singular term with coefficient $P_k(z)$ and a polynomial $2\, W_{k-1}(z)$. Thus, we only consider the singular terms in the formulas for the calculation of $\widetilde{O}_{j,k}^1$ and show that they add up to zero. Since the way of proceeding is the same in all cases and since there are many cases due to the different formulas for the calculation of $\widetilde{O}_{j,k}^1$ we restrict to investigate $\widetilde{O}_{0,1}^1$. In this case we consider

$$\begin{aligned} a+b=1 \quad &\Leftrightarrow \quad \frac{1-a}{b} = 1 \\ a+b=-1 \quad &\Leftrightarrow \quad \frac{-1-a}{b} = 1 \\ a-b=1 \quad &\Leftrightarrow \quad \frac{1-a}{b} = -1 \\ a-b=-1 \quad &\Leftrightarrow \quad \frac{-1-a}{b} = -1. \end{aligned}$$

Let $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}}$ be as provided in the theorem. Then, we get

$$\begin{aligned}\widetilde{O}^1_{0,1}(a_n,b_n) = \frac{\mathrm{Im}(a_n)}{2}\left\{\widetilde{Q}_0^{-1}\left(\eta_1\,\frac{-1-a_n}{b_n}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\,\frac{1-a_n}{b_n}\right)\right\} + \frac{\mathrm{Im}(a_n)}{b_n}\widetilde{Q}_0^0\left(\frac{-1-a_n}{b_n}\right)\\ + \frac{\mathrm{Im}(b_n)}{6}\left\{\widetilde{Q}_2^0\left(\frac{-1-a_n}{b_n}\right) - \widetilde{Q}_0^0\left(\frac{-1-a_n}{b_n}\right)\right.\\ \left. -\widetilde{Q}_2^0\left(\frac{1-a_n}{b_n}\right) + \widetilde{Q}_0^0\left(\frac{1-a_n}{b_n}\right)\right\} + \frac{\mathrm{Im}(b_n)}{b_n}\widetilde{Q}_1^0\left(\frac{-1-a_n}{b_n}\right).\end{aligned}$$

In the case $a+b=1$ we define $z_n := \frac{1-a_n}{b_n} \to 1$ $(n\to\infty)$ to simplify the notation. For the singular terms there holds

$$\begin{aligned}&\lim_{n\to\infty}\left[\frac{\mathrm{Im}(b_n)}{6}\left(-\widetilde{Q}_2^0(z_n) + \widetilde{Q}_0^0(z_n)\right)\right]\\ &= \frac{\mathrm{Im}(b)}{6}\lim_{n\to\infty}\left[\underbrace{(-P_2(z_n)+P_0(z_n))}_{\to 0\ (n\to\infty)}\log\frac{z_n+1}{z_n-1}\right] + \frac{\mathrm{Im}(b)}{3}(W_1(1)-W_{-1}(1))\\ &= \frac{\mathrm{Im}(b)}{3}(W_1(1)-W_{-1}(1)).\end{aligned}$$

Note that the Legendre polynomials vanish due to Lemma 4.1.2 (ii). In the case $a+b=-1$ we define $z_n := \frac{-1-a_n}{b_n} \to 1$ $(n\to\infty)$. For the singular terms there holds

$$\begin{aligned}&\lim_{n\to\infty}\left[\frac{\mathrm{Im}(a_n)}{b_n}\widetilde{Q}_0^0(z_n) + \frac{\mathrm{Im}(b_n)}{b_n}\widetilde{Q}_1^0(z_n) + \frac{\mathrm{Im}(b_n)}{6}\left(\widetilde{Q}_2^0(z_n) - \widetilde{Q}_0^0(z_n)\right)\right]\\ &= \lim_{n\to\infty}\left[\left\{\underbrace{\frac{\mathrm{Im}(a_n)}{b_n}P_0(z_n) + \frac{\mathrm{Im}(b_n)}{b_n}P_1(z_n)}_{\to\frac{\mathrm{Im}(a+b)}{b}=0\ (n\to\infty)} + \frac{\mathrm{Im}(b_n)}{6}\underbrace{(P_2(z_n)-P_0(z_n))}_{\to 0\ (n\to\infty)}\right\}\log\frac{z_n+1}{z_n-1}\right]\\ &\quad - \frac{\mathrm{Im}(a)}{b}W_{-1}(1) - \frac{\mathrm{Im}(b)}{b}W_0(1) - \frac{\mathrm{Im}(b)}{3}(W_1(1)-W_{-1}(1))\\ &= -\frac{\mathrm{Im}(a)}{b}W_{-1}(1) - \frac{\mathrm{Im}(b)}{b}W_0(1) - \frac{\mathrm{Im}(b)}{3}(W_1(1)-W_{-1}(1)).\end{aligned}$$

In the case $a-b=1$ we define $z_n := \frac{1-a_n}{b_n} \to -1$ $(n\to\infty)$ and obtain

$$\begin{aligned}&\lim_{n\to\infty}\left[\frac{\mathrm{Im}(b_n)}{6}\left(-\widetilde{Q}_2^0(z_n) + \widetilde{Q}_0^0(z_n)\right)\right]\\ &= \frac{\mathrm{Im}(b)}{6}\lim_{n\to\infty}\left[\underbrace{(-P_2(z_n)+P_0(z_n))}_{\to 0\ (n\to\infty)}\log\frac{z_n+1}{z_n-1}\right] + \frac{\mathrm{Im}(b)}{3}(W_1(-1)-W_{-1}(-1))\\ &= \frac{\mathrm{Im}(b)}{3}(W_1(-1)-W_{-1}(-1)).\end{aligned}$$

For $a-b=-1$ we define $z_n := \frac{-1-a_n}{b_n} \to -1$ $(n\to\infty)$ and get

$$\begin{aligned}
&\lim_{n\to\infty}\left[\frac{\mathrm{Im}(a_n)}{b_n}\widetilde{Q}_0^0(z_n)+\frac{\mathrm{Im}(b_n)}{b_n}\widetilde{Q}_1^0(z_n)+\frac{\mathrm{Im}(b_n)}{6}\left(\widetilde{Q}_2^0(z_n)-\widetilde{Q}_0^0(z_n)\right)\right]\\
&=\lim_{n\to\infty}\left[\left\{\underbrace{\frac{\mathrm{Im}(a_n)}{b_n}P_0(z_n)+\frac{\mathrm{Im}(b_n)}{b_n}P_1(z_n)}_{\to\frac{\mathrm{Im}(a-b)}{b}=0\ (n\to\infty)}+\frac{\mathrm{Im}(b_n)}{6}\underbrace{(P_2(z_n)-P_0(z_n))}_{\to 0\ (n\to\infty)}\right\}\log\frac{z_n+1}{z_n-1}\right]\\
&\quad-\frac{\mathrm{Im}(a)}{b}W_{-1}(-1)-\frac{\mathrm{Im}(b)}{b}W_0(-1)-\frac{\mathrm{Im}(b)}{3}(W_1(-1)-W_{-1}(-1))\\
&=-\frac{\mathrm{Im}(a)}{b}W_{-1}(-1)-\frac{\mathrm{Im}(b)}{b}W_0(-1)-\frac{\mathrm{Im}(b)}{3}(W_1(-1)-W_{-1}(-1)).
\end{aligned}$$

Thus, we obtain finite values for $\widetilde{O}_{0,1}^1$ $(a_n\pm b_n\to\pm 1)$. □

Chapter 5
Realization of the hp-BEM for the Navier-Lamé Equation

In this chapter, we describe the realization the hp-BEM for the Navier-Lamé equation. We begin by introducing a parametrization and some notations that we need for the calculations in this chapter. The main point for implementing the hp-BEM for high polynomial degrees is the choice of the basis functions of the discrete spaces $S^0(\mathcal{T}_h, p_0)$ and $S^1(\mathcal{T}_h, p_1)$, which is described in the second section. The calculation of the Galerkin matrices is given in the third section. We close this chapter by introducing some a posteriori error estimators that can be used to create adaptive algorithms.

5.1 Geometrical Basics

In this section we introduce a parametrization of the boundary elements and state some geometrical results that we need for the calculation of the Galerkin matrices. Let $\Gamma_\ell \in \mathcal{T}_h$ be a boundary element. Defining the center $m_\ell := \frac{A_\ell + B_\ell}{2}$ of the boundary element Γ_ℓ we introduce the following parametrization

$$\gamma_\ell : \begin{cases} [-1,1] & \to & \Gamma_\ell \\ t & \mapsto & \dfrac{B_\ell - A_\ell}{2}\, t + m_\ell. \end{cases} \tag{5.1}$$

Note that we obtain $\gamma_\ell'(t) = \frac{B_\ell - A_\ell}{2}$ and $|\gamma_\ell'(t)| = \frac{|\Gamma_\ell|}{2}$, and

$$\gamma_\ell^{-1}(x) = \frac{2}{\|B_\ell - A_\ell\|^2}(B_\ell - A_\ell)^T(x - m_\ell),$$

which implies

$$\nabla\gamma_\ell^{-1}(x) = 2\frac{B_\ell - A_\ell}{\|B_\ell - A_\ell\|^2}.$$

As we see in the subsequent sections all calculations for the Galerkin matrices are performed for a fixed combination of boundary elements. Therefore, we consider the boundary elements Γ_ℓ, $\Gamma_m \in \mathcal{T}_h$ with $x \in \Gamma_\ell$ and $y \in \Gamma_m$. The configuration is given in Figure 5.1.
Using the parametrization and the vectors $u := \frac{B_m - A_m}{2}$ and $v := \frac{B_\ell - A_\ell}{2}$ we get $x = m_\ell + s\,v$ and $y = m_m + t\,u$ with $s, t \in [-1, 1]$.
In the following we take a closer look at the term $|x - y|^2$. With $w := m_m - m_\ell$

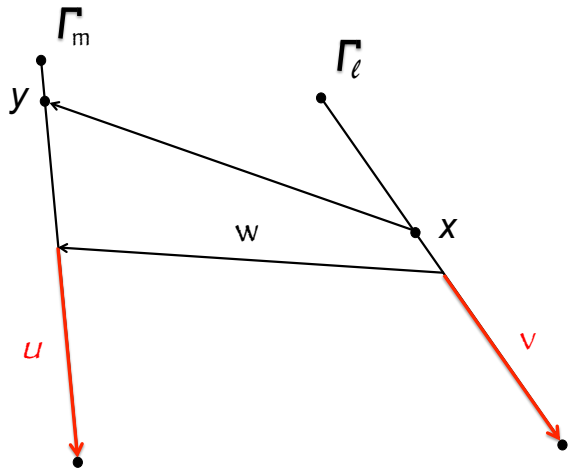

Fig. 5.1: Configuration of the boundary elements Γ_ℓ and Γ_m.

there holds $x - y = m_\ell - m_m + s\,v - t\,u = s\,v - w - t\,u$ and hence

$$\begin{aligned} |x-y|^2 &= |s\,v - w - t\,u|^2 \\ &= (s\,v - w - t\,u)^T (s\,v - w - t\,u) \\ &= u^T u\, t^2 - 2\,(s\,v - w)^T u\, t + (s\,v - w)^T (s\,v - w) \\ &= u^T u (z(s) - t)(\overline{z}(s) - t\,), \end{aligned} \tag{5.2}$$

where $z(s)$ is one of the zeros of the quadratic equation with

$$z(s) = \frac{(s\,v - w)^T u + \sqrt{[(s\,v - w)^T u]^2 - u^T u\,(s\,v - w)^T (s\,v - w)}}{u^T u}.$$

Applying the Lagrangian identity we get

$$[(s\,v - w)^T u]^2 - u^T u\,(s\,v - w)^T (s\,v - w) = -[u \times (s\,v - w)]^2$$

which yields

$$z(s) = \frac{(s\,v - w)^T u + i\,u \times (s\,v - w)}{u^T u}.$$

To simplify the notation we define $a := \frac{-u^T w - i\,u \times w}{u^T u}$ and $b := \frac{u^T v + i\,u \times v}{u^T u}$ and get

$$z(s) = a + b\,s. \tag{5.3}$$

Note that in the special case of identical elements $\Gamma_m = \Gamma_\ell$ we get $x - y = (s - t)\,u$ and $z(s) = s$. Moreover, one can show that in the case of neighboring elements, i.e. $A_\ell = B_m$ or $A_m = B_\ell$, we get $a \pm b = \pm 1$.

5.2 Basis Functions of the Discrete Function Spaces

In order to calculate the Galerkin matrices it remains to choose an appropriate basis of the discrete spaces $S^0(\mathcal{T}_h, p_0)$ and $S^1(\mathcal{T}_h, p_1)$. Thereby, we do not choose the monomials, as it is done in the standard approach, but we choose the Legendre polynomials and the Lobatto shape functions, as they fulfill recurrence relations and can be evaluated efficiently up to a high polynomial degree. Additionally, this choice of the basis functions leads to recurrence relations the Galerkin entries can be computed with very efficiently.
In the following we describe the basis functions more precisely. Let $\mathcal{T}_h$ be a triangulation of the polygonal boundary Γ with N_c corner points and $N_{el} := |\mathcal{T}_h|$. Additionally, let $p_0 \in \mathbb{N}_0^{N_{el}}$ and $p_1 \in \mathbb{N}^{N_{el}}$.
As the functions of $S^0(\mathcal{T}_h, p_0)$ are piecewise polynomial, globally discontinuous functions, we choose the transformed Legendre polynomials as basis functions, i.e. the basis is given by

$$\begin{pmatrix} \widetilde{P}_j^{(\ell)}(x) \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 \\ \widetilde{P}_j^{(\ell)}(x) \end{pmatrix},$$

where the transformed Legendre polynomials are defined by $\widetilde{P}_j^{(\ell)}(x) := (P_j \circ \gamma_\ell^{-1})(x)$ for $\ell = 1, ..., N_{el}$ and $j = 0, ..., p_{0,\ell}$. Note that

$$\operatorname{supp}\left\{\begin{pmatrix} \widetilde{P}_j^{(\ell)}(x) \\ 0 \end{pmatrix}\right\} = \operatorname{supp}\left\{\begin{pmatrix} 0 \\ \widetilde{P}_j^{(\ell)}(x) \end{pmatrix}\right\} = \Gamma_\ell$$

and thus, we have a local basis, i.e every basis function only lives on one boundary element. For the implementation we sort the basis functions as follows:

1. The basis functions are sorted by their non-zero entries, i.e all basis functions whose first entry is non-zero are listed first.
2. The basis functions are sorted by the boundary element they live on.
3. The basis functions are sorted by their polynomial degree.

All functions in $S^1(\mathcal{T}_h, p_1)$ are globally continuous and hence, we cannot choose the Legendre polynomials as basis functions. However, the Lobatto shape functions $N_j(t)$ vanish at the endpoints ± 1 for all $j \geq 3$ and thus, we choose the transformed Lobatto shape functions as basis functions for $j \geq 3$

$$\begin{pmatrix} \widetilde{N}_j^{(\ell)}(x) \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 \\ \widetilde{N}_j^{(\ell)}(x) \end{pmatrix},$$

with $\widetilde{N}_j^{(\ell)}(x) := (N_j \circ \gamma_\ell^{-1})(x)$. Again, there holds

$$\operatorname{supp}\left\{\begin{pmatrix} \widetilde{N}_j^{(\ell)}(x) \\ 0 \end{pmatrix}\right\} = \operatorname{supp}\left\{\begin{pmatrix} 0 \\ \widetilde{N}_j^{(\ell)}(x) \end{pmatrix}\right\} = \Gamma_\ell.$$

However, the first two Lobatto shape functions N_1 and N_2 do not vanish at the endpoints. Hence, to ensure the global continuity we add hat functions to the basis,

that live on two neighboring elements. Thus, we have both linear basis functions that live on two elements and basis functions of higher polynomial degree that live on one element. For the implementation we introduce the following oder of the basis functions:

1. The basis functions are sorted by their non-zero entries, i.e all basis functions whose first entry is non-zero are listed first.
2. All hat functions are listed before all other basis functions with higher polynomial degree.
3. The basis functions are sorted by the boundary element they live on.
4. The basis functions are sorted by their polynomial degree.

Figure 5.2 shows an example of the basis functions for three elements and $p_0 = p_1 = (1, 2, 3)$. In particular, the figure shows the enumeration of the basis functions of $S^1(\mathcal{T}_h, p_1)$ within one component.

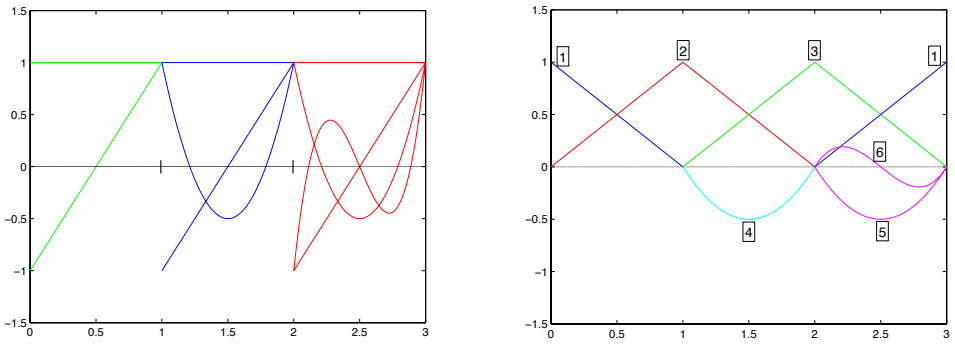

Fig. 5.2: Basis function for $S^0(\mathcal{T}_h, p_0)$ (left) and $S^1(\mathcal{T}_h, p_1)$ (right) for three elements and $p_0 = p_1 = (1, 2, 3)$. Source: [3].

Finally, we define the number of degrees of freedom with respect to the discrete space $S^0(\mathcal{T}_h, p_0)$ by

$$\mathcal{N}^0 := \dim S^0(\mathcal{T}_h, p_0) = 2 \sum_{k=1}^{N_{el}} (p_{0,k} + 1)$$

and the degrees of freedom with respect to $S^1(\mathcal{T}_h, p_1)$ by

$$\mathcal{N}^1 := \dim S^1(\mathcal{T}_h, p_1) = 2 \left(\sum_{k=1}^{N_{el}} (p_{1,k} - 1) + N_c \right).$$

Considering different types of boundary conditions we denote the degrees of freedom on the Dirichlet boundary by $\mathcal{N}_D^i := \dim S^i(\mathcal{T}_{h,D}, p_i)$, on the Neumann boundary by $\mathcal{N}_N^i := \dim S^i(\mathcal{T}_{h,N}, p_i)$ and on the gliding boundary (subsequently defined) by $\mathcal{N}_G^i := \dim S^i(\mathcal{T}_{h,G}, p_i)$ $(i = 0, 1)$.

5.3 Computation of the Galerkin Matrices

In the section we describe the calculation of the Galerkin matrices. In particular, we derive analytical formulas that depend on the double integrals that we introduced in Section 4.2. Hence, we can assemble the Galerkin matrices efficiently and stably for high polynomial degrees. In the first section we investigate the Galerkin matrix of the single layer operator and in the second section we describe the calculation of the Galerkin matrix of the double layer operator. Finally, we derive formulas for the entries of the Galerkin matrix of the hypersingular operator.

5.3.1 The Single Layer Operator

With the definition of the basis functions for $S^0(\mathcal{T}_h, p)$ the Galerkin matrix for a fixed combination of boundary elements Γ_ℓ and Γ_m with polynomial degrees p_ℓ and p_m reads:

$$\left(\begin{array}{ccc|ccc} \langle V\begin{pmatrix}\widetilde{P}_0^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_0^{(\ell)}\\0\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}\widetilde{P}_{p_m}^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_0^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}\widetilde{P}_0^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_0^{(\ell)}\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}\widetilde{P}_{p_m}^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_0^{(\ell)}\end{pmatrix}\rangle \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \langle V\begin{pmatrix}\widetilde{P}_0^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_{p_\ell}^{(\ell)}\\0\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}\widetilde{P}_{p_m}^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_{p_\ell}^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}\widetilde{P}_0^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_{p_\ell}^{(\ell)}\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}\widetilde{P}_{p_m}^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_{p_\ell}^{(\ell)}\end{pmatrix}\rangle \\ \hline \langle V\begin{pmatrix}0\\\widetilde{P}_0^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_0^{(\ell)}\\0\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}0\\\widetilde{P}_{p_m}^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_0^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}0\\\widetilde{P}_0^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_0^{(\ell)}\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}0\\\widetilde{P}_{p_m}^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_0^{(\ell)}\end{pmatrix}\rangle \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \langle V\begin{pmatrix}0\\\widetilde{P}_0^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_{p_\ell}^{(\ell)}\\0\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}0\\\widetilde{P}_{p_m}^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_{p_\ell}^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}0\\\widetilde{P}_0^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_{p_\ell}^{(\ell)}\end{pmatrix}\rangle & \cdots & \langle V\begin{pmatrix}0\\\widetilde{P}_{p_m}^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_{p_\ell}^{(\ell)}\end{pmatrix}\rangle \end{array}\right).$$

Using the given block structure of the Galerkin matrix, we compute the 2×2 matrix of the (j,k)-th entry ($j = 0, ..., p_\ell$ and $k = 0, ..., p_m$) of each block, since there holds

$$\begin{pmatrix} \langle V\begin{pmatrix}\widetilde{P}_k^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_j^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}\widetilde{P}_k^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_j^{(\ell)}\end{pmatrix}\rangle \\ \langle V\begin{pmatrix}0\\\widetilde{P}_k^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_j^{(\ell)}\\0\end{pmatrix}\rangle & \langle V\begin{pmatrix}0\\\widetilde{P}_k^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_j^{(\ell)}\end{pmatrix}\rangle \end{pmatrix} = \underbrace{\int_{\Gamma_\ell}\widetilde{P}_j^{(\ell)}(x)\int_{\Gamma_m}U(x,y)\,\widetilde{P}_k^{(m)}(y)\,ds_y\,ds_x}_{=:\,V_{j,k}^{(\ell,m)}}.$$

For the calculations, we distinguish the case of identical elements $\Gamma_\ell = \Gamma_m$ and non-identical elements $|\Gamma_\ell \cap \Gamma_m| = 0$ and investigate both cases, separately.
Starting the calculation of $V_{j,k}^{(\ell,m)}$ for identical elements, we need the following result that is proven in [10].

Lemma 5.3.1 *Let $\Gamma_\ell \in \mathcal{T}_h$ be a boundary element with $h_\ell := |\Gamma_\ell|$ and $k, j \in \mathbb{N}_0$. Then, there holds*

$$\begin{aligned}&\int_{\Gamma_\ell}\widetilde{P}_j^{(\ell)}(x)\int_{\Gamma_\ell}\log|x-y|\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x \\ &= \begin{cases} \frac{h_\ell^2}{2}\left(\log\left(h_\ell^2\right)-3\right) & ,\ \textit{for } j=k=0 \\ \frac{2h_\ell^2}{(j+k+2)(j+k)((j-k)^2-1)} & ,\ \textit{for } j+k>0 \textit{ and } j-k \textit{ even} \\ 0 & ,\ \textit{otherwise.}\end{cases}\end{aligned}$$

With Lemma 5.3.1 we obtain the following explicit formula in the case of identical elements.

Theorem 5.3.2 *Let $\Gamma_\ell \in \mathcal{T}_h$ be a boundary element and u be the directional vector of Γ_ℓ as defined in Section 5.1. Moreover, let $j,k \in \mathbb{N}_0$. Then, there holds in the case of identical elements*

$$V_{j,k}^{(\ell,\ell)} = -\frac{\lambda+3\mu}{4\pi\mu(\lambda+2\mu)}\, u^T u \,\left(\theta_{j,k}\,\mathrm{I} - \frac{\lambda+\mu}{\lambda+3\mu}\,4\delta_{k0}\,\delta_{j0}\,\frac{uu^T}{u^T u}\right),$$

where

$$\theta_{k,j} = \begin{cases} \left(2\log\left(4\,v^T v\right) - 6\right) & ,\ \textit{for } j=k=0 \\ \frac{8}{(j+k+2)(j+k)((j-k)^2-1)} & ,\ \textit{for } j+k>0 \textit{ and } j-k \textit{ even} \\ 0 & ,\ \textit{otherwise.} \end{cases}$$

Proof. For reasons of clarity we split the Kelvin matrix (3.2) into the two parts $U^{(1)}$ and $U^{(2)}$ with

$$U(x,y) = -\frac{\lambda+3\mu}{4\pi\mu\,(\lambda+2\mu)}\Bigg(\underbrace{\log|x-y|\,\mathrm{I}}_{=:\,U^{(1)}(x,y)} - \frac{\lambda+\mu}{\lambda+3\mu}\underbrace{\frac{(x-y)(x-y)^T}{|x-y|^2}}_{=:\,U^{(2)}(x,y)}\Bigg) \tag{5.4}$$

and investigate both parts, separately. We get

$$\begin{aligned} &\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} U(x,y)\,\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x \\ &= -\frac{\lambda+3\mu}{4\pi\mu\,(\lambda+2\mu)}\Bigg\{\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} U^{(1)}(x,y)\,\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x \\ &\qquad\qquad - \frac{\lambda+\mu}{\lambda+3\mu}\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} U^{(2)}(x,y)\,\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x\Bigg\} \end{aligned}$$

Using Lemma 5.3.1 there holds for the first integral

$$\begin{aligned} &\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} U^{(1)}(x,y)\,\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x \\ &= \int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} \log|x-y|\,\widetilde{P}_k^{(\ell)}(y)\,ds_y\,ds_x\,\mathrm{I} = \theta_{k,j}\,\mathrm{I} \end{aligned}$$

with

$$\theta_{k,j} := \begin{cases} 2\log\left(4\,u^T u\right) - 6 & ,\ \text{for } j=k=0 \\ \frac{8}{(j+k+2)(j+k)((j-k)^2-1)} & ,\ \text{for } j+k>0 \text{ and } j-k \text{ even} \\ 0 & ,\ \text{otherwise.} \end{cases}$$

Note that we used the fact that $h_\ell = 2\,|u|$. For calculating the second term we plug in the parametrization and use $(x-y) = (s-t)\,u$, which we already stated in Section 5.1. Thus, we obtain

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} U^{(2)}(x,y)\, \widetilde{P}_k^{(\ell)}(y)\, ds_y\, ds_x \\
&= \int_{\Gamma_\ell}\int_{\Gamma_\ell} \frac{(x-y)(x-y)^T}{|x-y|^2}\, \widetilde{P}_k^{(\ell)}(y) \widetilde{P}_j^{(\ell)}(x)\, ds_y\, ds_x \\
&= \frac{|\Gamma_\ell|^2}{4} \int_{-1}^{1}\int_{-1}^{1} P_k(t) P_j(s) ds\, dt\, \frac{uu^T}{u^T u} \\
&= 4\delta_{k0}\,\delta_{j0}\; uu^T.
\end{aligned}$$

Note that we used (iv) in Lemma 4.1.2 and $u^T u = |u|^2 = \frac{|\Gamma_\ell|^2}{4}$ for the last identity. Putting the results together completes the proof. □

In the case of non-identical elements we reduce the calculation of $V_{j,k}^{(\ell,m)}$ to the calculation of the integrals $I_{j,k}^{-1}$ and $\widetilde{I}_{j,k}^{0}$ that we introduced in Chapter 4. For reasons of clarity we split the Kelvin matrix into two parts and investigate both parts separately, as it is already done in the proof of Theorem 5.3.2 (See (5.4)). We start investigating the first part of the Kelvin matrix and obtain the following lemma.

Lemma 5.3.3 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section 5.1. Then, there holds for $j,k \in \mathbb{N}_0$*

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(1)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x \\
&= |u|\,|v| \left\{ 2\,\log(u^T u)\,\delta_{k0}\delta_{j0} + \mathrm{Re}\left(I_{j,k}^{-1}(a,b)\right)\right\} \mathrm{I}.
\end{aligned}$$

Proof. We start plugging in the parametrization and obtain with $|u| = \frac{|\Gamma_\ell|}{2}$, $|v| = \frac{|\Gamma_m|}{2}$ and $z := z(s) = a + b\,s$

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(1)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x \\
&= \int_{\Gamma_\ell}\int_{\Gamma_m} \log|x-y|\, \widetilde{P}_k^{(m)}(y)\, \widetilde{P}_j^{(\ell)}(x) ds_y\, ds_x\; \mathrm{I} \\
&= \frac{|u|\,|v|}{2} \int_{-1}^{1}\int_{-1}^{1} \log|s\,v - w - t\,u|^2 P_k(t)\, P_j(s)\, dt\, ds\; \mathrm{I} \\
&= \frac{|u|\,|v|}{2} \int_{-1}^{1}\int_{-1}^{1} \log(u^T u(z-t)(\overline{z}-t))\; P_k(t)\, P_j(s)\, dt\, ds\; \mathrm{I} \\
&= \frac{|u|\,|v|}{2} \left(\log(u^T u) \int_{-1}^{1}\int_{-1}^{1} P_k(t)\, P_j(s)\; dt\, ds + \int_{-1}^{1}\int_{-1}^{1} \log(z-t)\; P_k(t)\, P_j(s)\; dt\, ds \right. \\
&\qquad \left. + \int_{-1}^{1}\int_{-1}^{1} \log(\overline{z}-t)\; P_k(t)\, P_j(s)\; dt\, ds \right) \mathrm{I}. \qquad (5.5)
\end{aligned}$$

Since there holds $\log \overline{z} = \overline{\log z},\ \forall z \in \mathbb{C}$, we obtain

$$\int_{-1}^{1}\int_{-1}^{1} \log(\overline{z}-t)\ P_k(t)P_j(s)\ dt\,ds = \overline{\int_{-1}^{1}\int_{-1}^{1} \log(z-t)\ P_k(t)P_j(s)\ dt\,ds}$$

and hence,

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} \log(z-t)\ P_k(t)P_j(s)\ dt\,ds + \int_{-1}^{1}\int_{-1}^{1} \log(\overline{z}-t)\ P_k(t)P_j(s)\ dt\,ds\\ &= 2\,\mathrm{Re}\left(\int_{-1}^{1}\int_{-1}^{1} \log(z-t)\ P_k(t)P_j(s)\ dt\,ds\right).\end{aligned}$$

With the definition of $I_{j,k}^{-1}$ and (iv) in Lemma 4.1.2 we can write (5.5) shorter

$$\begin{aligned}&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(1)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x\\ &= |u|\,|v|\left\{2\,\log(u^T u)\,\delta_{k0}\delta_{j0} + \mathrm{Re}\left(I_{j,k}^{-1}(a,b)\right)\right\} \mathrm{I}.\end{aligned}$$

□

Before we investigate the second part of the Kelvin matrix we state the following partial fraction decompositions that we need for the calculations.

Lemma 5.3.4 *Let $t \in [-1,1]$ and $z \in \mathbb{C}\backslash[-1,1]$. Then, there holds*

$$(i)\quad \frac{t^2}{(z-t)(\overline{z}-t)} = 1 - \frac{\mathrm{Re}(z)^2 - \mathrm{Im}(z)^2}{2\,i\,\mathrm{Im}(z)}\left(\frac{1}{z-t} - \frac{1}{\overline{z}-t}\right) - \mathrm{Re}(z)\left(\frac{1}{z-t} + \frac{1}{\overline{z}-t}\right)$$

$$(ii)\quad \frac{t}{(z-t)(\overline{z}-t)} = -\frac{1}{2}\left(\frac{1}{z-t} + \frac{1}{\overline{z}-t}\right) - \frac{\mathrm{Re}(z)}{2i\,\mathrm{Im}(z)}\left(\frac{1}{z-t} - \frac{1}{\overline{z}-t}\right)$$

$$(iii)\quad \frac{1}{(z-t)(\overline{z}-t)} = -\frac{1}{2i\,\mathrm{Im}(z)}\left(\frac{1}{z-t} - \frac{1}{\overline{z}-t}\right).$$

Proof. The proof can be done by multiplying the equations with $(z-t)(\overline{z}-t)$ and comparing the coefficients. □

For the second part of the Kelvin matrix we obtain the following result.

Lemma 5.3.5 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section 5.1. Then, there holds for $j,k \in \mathbb{N}_0$*

$$\begin{aligned}&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(2)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x\\ &= |u|\,|v|\Bigg\{4\delta_{k0}\,\delta_{j0}\,\frac{uu^T}{u^T u} + \mathrm{Im}\left(\widetilde{I}_{j,k}^{0}(a,b)\right)\frac{uu^T - u_\perp u_\perp{}^T}{u^T u}\\ &\qquad - \mathrm{Re}\left(\widetilde{I}_{j,k}^{0}(a,b)\right)\frac{u_\perp u^T + uu_\perp{}^T}{u^T u}\Bigg\}.\end{aligned} \tag{5.6}$$

Proof. First, we investigate $U^{(2)}(x,y)$. With the parametrization and $z := z(s)$ we obtain

$$\begin{aligned}
U^{(2)}(x,y) &= \frac{(x-y)(x-y)^T}{|x-y|^2} \\
&= \frac{(sv-(w+t\,u))(sv-(w+t\,u))^T}{u^Tu(z-t)(\overline{z}-t)} \\
&= \frac{t^2}{u^Tu(z-t)(\overline{z}-t)}\,uu^T - \frac{t}{u^Tu(z-t)(\overline{z}-t)}\left((s\,v-w)u^T + u(s\,v-w)^T\right) \\
&\quad + \frac{1}{u^Tu(z-t)(\overline{z}-t)}\,(s\,v-w)(s\,v-w)^T.
\end{aligned}$$

In general, every vector $b \in \mathbb{R}^2$ can be represented by a linear combination of u and $u_\perp := (u_2, -u_1)^T$, i.e.

$$b = \frac{u^Tb}{u^Tu}\,u - \frac{u \times b}{u^Tu}\,u_\perp.$$

Applying this identity on $s\,v-w$ yields

$$\begin{aligned}
s\,v-w &= \left(\frac{u^T(s\,v-w)}{u^Tu}\,u - \frac{u\times(s\,v-w)}{u^Tu}\,u_\perp\right) \\
&= \mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp,
\end{aligned}$$

where we used the representation of $z(s)$ in (5.3). Hence, we get

$$\begin{aligned}
U^{(2)}(x,y) &= \frac{t^2}{u^Tu(z-t)(\overline{z}-t)}\,uu^T \\
&\quad - \frac{t}{u^Tu(z-t)(\overline{z}-t)}\left(u\,(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)^T + (\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)\,u^T\right) \\
&\quad + \frac{1}{u^Tu(z-t)(\overline{z}-t)}\,(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)\,(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)^T,
\end{aligned}$$

which we can further simplify by applying Lemma 5.3.4 as follows

$$\begin{aligned}
U^{(2)}(x,y) &= \frac{1}{u^Tu}\Bigg[\left\{1 - \frac{\mathrm{Re}(z)^2-\mathrm{Im}(z)^2}{2\,i\,\mathrm{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right) - \mathrm{Re}(z)\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\right\}uu^T \\
&\quad + \left\{\frac{1}{2}\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right) + \frac{\mathrm{Re}(z)}{2i\,\mathrm{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\right\}\Big(u\,(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)^T \\
&\qquad + (\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)\,u^T\Big) \\
&\quad - \frac{1}{2i\,\mathrm{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)(\mathrm{Re}(z)\,u - \mathrm{Im}(z)\,u_\perp)^T\Bigg] \\
&= \frac{uu^T}{u^Tu} + \frac{\mathrm{Im}(z)}{2i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T - u_\perp u_\perp{}^T}{u^Tu} \\
&\quad - \frac{\mathrm{Im}(z)}{2}\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\frac{u_\perp u^T + uu_\perp{}^T}{u^Tu}.
\end{aligned} \tag{5.7}$$

With this representation of $U^{(2)}$ and the linearity of the integral we obtain

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(2)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x \\
&= |u|\,|v| \Bigg\{ \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, dt\, ds\ \frac{uu^T}{u^T u} \\
&\quad + \frac{1}{2i} \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(z) \left(\frac{1}{z-t} - \frac{1}{\overline{z}-t} \right) dt\, ds\ \frac{uu^T - u_\perp u_\perp{}^T}{u^T u} \\
&\quad - \frac{1}{2} \int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(z) \left(\frac{1}{z-t} + \frac{1}{\overline{z}-t} \right) dt\, ds\ \frac{u_\perp u^T + uu_\perp{}^T}{u^T u} \Bigg\}.
\end{aligned}$$

In general, for all $z \in \mathbb{C}$ there holds $\frac{1}{\overline{z}} = \overline{\frac{1}{z}}$ which implies that

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(z) \left(\frac{1}{z-t} - \frac{1}{\overline{z}-t} \right) dt\, ds \\
&= \int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds - \overline{\int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds} \\
&= 2i\, \mathrm{Im} \left(\int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds \right)
\end{aligned} \tag{5.8}$$

and

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} P_j(s)\, P_k(t)\, \mathrm{Im}(z) \left(\frac{1}{z-t} + \frac{1}{\overline{z}-t} \right) dt\, ds \\
&= \int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds + \overline{\int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds} \\
&= 2\,\mathrm{Re} \left(\int_{-1}^{1}\int_{-1}^{1} \frac{P_j(s)\, P_k(t)\, \mathrm{Im}(z)}{z-t}\, dt\, ds \right).
\end{aligned} \tag{5.9}$$

With the definition of $\widetilde{I}^0_{j,k}(a,b)$ and Lemma 4.1.2 (iv) we finally have

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U^{(2)}(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x \\
&= |u|\,|v| \Bigg\{ 4\delta_{k0}\,\delta_{j0}\, \frac{uu^T}{u^T u} + \mathrm{Im} \left(\widetilde{I}^0_{j,k}(a,b) \right) \frac{uu^T - u_\perp u_\perp{}^T}{u^T u} \\
&\qquad - \mathrm{Re} \left(\widetilde{I}^0_{j,k}(a,b) \right) \frac{u_\perp u^T + uu_\perp{}^T}{u^T u} \Bigg\}.
\end{aligned}$$

□

Putting the results of the previous lemmas together we get a formula for the computation of the single layer operator.

Theorem 5.3.6 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be a boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$ and u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section*

5.1. Moreover, let $j,k \in \mathbb{N}_0$. Then, there holds

$$\begin{aligned} V_{j,k}^{(\ell,m)} = &-\frac{\lambda+3\mu}{4\pi\mu(\lambda+2\mu)}|u|\,|v|\Big\{ \Big[2\,\log(u^Tu)\,\delta_{k0}\delta_{j0} + \mathrm{Re}\left(I_{j,k}^{-1}(a,b)\right)\Big]\,\mathrm{I} \\ &- \frac{\lambda+\mu}{\lambda+3\mu}\Big[4\delta_{k0}\,\delta_{j0}\,\frac{uu^T}{u^Tu} + \mathrm{Im}\left(\widetilde{I}_{j,k}^{0}(a,b)\right)\frac{uu^T - u_\perp u_\perp{}^T}{u^Tu} \\ &- \mathrm{Re}\left(\widetilde{I}_{j,k}^{0}(a,b)\right)\frac{u_\perp u^T + uu_\perp{}^T}{u^Tu}\Big]\Big\}. \end{aligned} \tag{5.10}$$

5.3.2 The Double Layer Operator

The Galerkin matrices of the single and the double layer operator have the same 2×2 block structure, as the basis functions of $S^0(\mathcal{T}_h,p_0)$ and $S^1(\mathcal{T}_h,p_1)$ are sorted similarly by their components. Therefore, we compute the entries of the Galerkin matrix of the double layer operator similarly to the entries of the matrix of the single layer operator. Hence, we calculate for a fixed combination of boundary elements Γ_ℓ and Γ_m with polynomial degrees $p_{1,\ell}$ and $p_{1,m}$ ($j=1,...,p_{1,\ell}$ and $k=1,...,p_{1,m}$):

$$\begin{pmatrix} \langle K\begin{pmatrix}\widetilde{N}_k^{(m)}\\0\end{pmatrix},\begin{pmatrix}\widetilde{P}_j^{(\ell)}\\0\end{pmatrix}\rangle & \langle K\begin{pmatrix}\widetilde{N}_k^{(m)}\\0\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_j^{(\ell)}\end{pmatrix}\rangle \\ \langle K\begin{pmatrix}0\\\widetilde{N}_k^{(m)}\end{pmatrix},\begin{pmatrix}\widetilde{P}_j^{(\ell)}\\0\end{pmatrix}\rangle & \langle K\begin{pmatrix}0\\\widetilde{N}_k^{(m)}\end{pmatrix},\begin{pmatrix}0\\\widetilde{P}_j^{(\ell)}\end{pmatrix}\rangle \end{pmatrix} = \underbrace{\int_{\Gamma_\ell}\widetilde{P}_j^{(\ell)}(x)\int_{\Gamma_m} T(x,y)\,\widetilde{N}_k^{(m)}(y)\,ds_y\,ds_x}_{=:\,K_{j,k}^{(\ell,m)}}.$$

Again, we distinguish the cases of identical elements and non-identical elements. In the case of identical elements we derive the following explicit formula.

Theorem 5.3.7 *Let $\Gamma_\ell \in \mathcal{T}_h$ be a boundary element and u be the directional vector of Γ_ℓ as defined in Section 5.1. Moreover, let $j\in\mathbb{N}_0$ and $k\in\mathbb{N}$. Then, there holds in the case of identical elements*

$$K_{j,k}^{(\ell,\ell)} = \frac{\mu}{2\pi(\lambda+2\mu)}\,|u|\;\eta_{j,k}\;\mathrm{I}\times\mathrm{I}$$

where for $k\geq 3$

$$\eta_{j,k} := \begin{cases} -\frac{8}{(k-j-1)(k+j)(k-j-3)(k+j-2)} & ,\ \textit{for } j-k \textit{ even} \\ 0 & ,\ \textit{otherwise,} \end{cases}$$

and

$$\eta_{j,2} := \begin{cases} -\frac{2}{j(j+1)} & ,\ \textit{for } j \textit{ odd} \\ -\frac{2}{(j-1)(j+2)} & ,\ \textit{otherwise} \end{cases}, \quad \eta_{j,1} := \begin{cases} -\frac{2}{j(j+1)} & ,\ \textit{for } j \textit{ odd} \\ \frac{2}{(j-1)(j+2)} & ,\ \textit{otherwise.} \end{cases}$$

Proof. In the case of identical elements there holds $(x-y)^Tn(y) = (s-t)\,u^Tn(y) = 0$. Plugging in the parametrization we obtain for the co-normal derivative $T(x,y)$ of the Kelvin matrix that is given in (3.3) with $t := t(y) = \frac{u}{|u|}$

$$\begin{aligned} T(x,y) &= \frac{\mu}{2\pi(\lambda+2\mu)}\frac{(x-y)^Tt}{|x-y|^2} \\ &= \frac{\mu}{2\pi(\lambda+2\mu)}\,\frac{1}{|u|}\,\frac{1}{(s-t)}\,\mathrm{I}\times\mathrm{I}\,. \end{aligned}$$

and thus

$$\begin{aligned} K_{j,k}^{(\ell,\ell)} &= \int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_\ell} T(x,y)\, \widetilde{N}_k^{(\ell)}(y)\, ds_y\, ds_x \\ &= \frac{\mu}{2\pi(\lambda+2\mu)}\, |u| \int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\, P_j(s)}{s-t}\, ds\, dt \;\; \mathrm{I}\times\mathrm{I}\,. \end{aligned}$$

Simplifying the notation we define $\eta_{j,k} := \int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\,P_j(s)}{s-t}\, ds\, dt$. Thus, we obtain

$$K_{j,k}^{(\ell,\ell)} = \frac{\mu}{2\pi(\lambda+2\mu)}\, |u|\, \eta_{j,k} \;\; \mathrm{I}\times\mathrm{I}\,.$$

With (iv) in Lemma 4.1.4 we get for $k \geq 3$

$$\eta_{j,k} = \begin{cases} -\frac{8}{(k-j-1)(k+j)(k-j-3)(k+j-2)} & , \text{ for } j-k \text{ even} \\ 0 & , \text{ otherwise,} \end{cases}$$

and

$$\eta_{j,2} = \begin{cases} -\frac{2}{j(j+1)} & , \text{ for } j \text{ odd} \\ -\frac{2}{(j-1)(j+2)} & , \text{ otherwise} \end{cases}, \quad \eta_{j,1} = \begin{cases} -\frac{2}{j(j+1)} & , \text{ for } j \text{ odd} \\ \frac{2}{(j-1)(j+2)} & , \text{ otherwise,} \end{cases}$$

which completes the proof. □

In the case of non-identical elements, we derive a formula for the computation of the Galerkin entries that depends on the integrals $O_{j,k}^0(a,b)$ and $\widetilde{O}_{j,k}^1(a,b)$. To simplify the calculations, we split $T(x,y)$ into three parts:

$$\begin{aligned} T(x,y) =& \frac{\mu}{2\pi(\lambda+2\mu)} \underbrace{\frac{(x-y)^T n(y)}{|x-y|^2}\,\mathrm{I}}_{=:\, T^{(1)}(x,y)} + \frac{\mu}{2\pi(\lambda+2\mu)} \underbrace{\frac{(x-y)^T t(y)}{|x-y|^2}\,\mathrm{I}\times\mathrm{I}}_{=:\, T^{(2)}(x,y)} \\ &+ \frac{\lambda+\mu}{\pi(\lambda+2\mu)} \underbrace{\frac{(x-y)^T n(y)}{|x-y|^4}(x-y)(x-y)^T}_{=:\, T^{(3)}(x,y)}. \end{aligned} \tag{5.11}$$

In the following we successively investigate all parts stating the next three lemmas. In the of this section, we summarize the results in Theorem 5.3.11.

Lemma 5.3.8 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section 5.1. Then, there holds for $j \in \mathbb{N}_0$ and $k \in \mathbb{N}$*

$$\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} T^{(1)}(x,y)\, \widetilde{N}_k^{(m)}(y)\, ds_y\, ds_x = |v|\, \mathrm{Im}\left(O_{j,k}^0(a,b)\right)\, \mathrm{I} \tag{5.12}$$

Proof. First, we investigate $T^{(1)}(x,y)$. Plugging in the parametrization we obtain with $n := n(y)$, $z := z(s)$ and (5.2)

$$\begin{aligned} T^{(1)}(x,y) &= \frac{(x-y)^T n}{|x-y|^2}\ \mathrm{I} \\ &= \frac{(s\,v - w - t\,u)^T n}{u^T u\,(z-t)(\overline{z}-t)}\ \mathrm{I} \\ &= \frac{(s\,v-w)^T n}{u^T u}\,\frac{1}{(z-t)(\overline{z}-t)}\ \mathrm{I} \\ &= -\frac{\mathrm{Im}(z)}{|u|}\,\frac{1}{(z-t)(\overline{z}-t)}\ \mathrm{I}, \end{aligned}$$

where we used

$$\frac{(s\,v-w)^T n}{u^T u} = \frac{(s\,v-w)\times u}{u^T u\,|u|} = -\frac{\mathrm{Im}(z)}{|u|}.$$

With the partial fraction decomposition in Lemma 5.3.4 (*iii*) we get

$$T^{(1)}(x,y) = \frac{1}{2i\,|u|}\left(\frac{1}{z-t} - \frac{1}{\overline{z}-t}\right). \tag{5.13}$$

Thus, we obtain for the integral

$$\begin{aligned} &\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} T^{(1)}(x,y)\,\widetilde{N}_k^{(m)}(y)\,ds_y\,ds_x \\ &= |u|\,|v|\,\frac{1}{2i\,|u|}\int_{-1}^{1}\int_{-1}^{1} N_k(t)\,P_j(s)\left(\frac{1}{z-t} - \frac{1}{\overline{z}-t}\right)ds\,dt\ \mathrm{I} \\ &= |v|\,\mathrm{Im}\left(\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)}{z-t}\,ds\,dt\right)\mathrm{I}. \end{aligned}$$

Note that we obtain the last identity with (5.8) by substituting $P_k(t)$ with $N_k(t)$. Plugging in the definition of $O_{j,k}^0(a,b)$ yields (5.12). □

Lemma 5.3.9 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section 5.1. Then, there holds for $j \in \mathbb{N}_0$ and $k \in \mathbb{N}$*

$$\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} T^{(2)}(x,y)\,\widetilde{N}_k^{(m)}(y)\,ds_y\,ds_x = |v|\,\mathrm{Re}\left(O_{j,k}^0(a,b)\right)\ \mathrm{I}\times\mathrm{I}. \tag{5.14}$$

Proof. As in the proof of the previous lemma we start with investigating $T^{(2)}(x,y)$. With the parametrization, $t(y) = \frac{u}{|u|}$, $z := z(s)$ and (5.2) there holds

$$\begin{aligned} T^{(2)}(x,y) &= \frac{(x-y)^T t(y)}{|x-y|^2}\,\mathrm{I}\times\mathrm{I} \\ &= \frac{(s\,v-w-t\,u)^T u}{|u|\,u^T u\,(z-t)(\overline{z}-t)}\,\mathrm{I}\times\mathrm{I} \\ &= \frac{1}{|u|}\left\{\frac{(s\,v-w)^T u}{u^T u}\,\frac{1}{(z-t)(\overline{z}-t)} - \frac{t}{(z-t)(\overline{z}-t)}\right\}\mathrm{I}\times\mathrm{I} \end{aligned}$$

Applying (i) and (ii) in Lemma 5.3.4 we get

$$\begin{aligned} T^{(2)}(x,y) &= \frac{1}{|u|}\Bigg\{ \left[\frac{\operatorname{Re}(z)}{2i\operatorname{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)+\frac{1}{2}\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\right] \\ &\qquad -\frac{\operatorname{Re}(z)}{2i\operatorname{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\Bigg\}\,\mathrm{I}\times\mathrm{I} \\ &= \frac{1}{2\,|u|}\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\,\mathrm{I}\times\mathrm{I}. \end{aligned} \tag{5.15}$$

Hence, by using (5.9) we obtain

$$\begin{aligned} &\int_{\Gamma_\ell}\widetilde{P}_j^{(\ell)}(x)\int_{\Gamma_m} T^{(2)}(x,y)^T\,\widetilde{N}_k^{(m)}(y)\,ds_y\,ds_x \\ &= |u|\,|v|\,\frac{1}{2\,|u|}\int_{-1}^{1}\int_{-1}^{1} N_k(t)\,P_j(s)\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)ds\,dt\;\mathrm{I}\times\mathrm{I} \\ &= |v|\operatorname{Re}\left(\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)}{z-t}\,ds\,dt\right)\;\mathrm{I}\times\mathrm{I}. \end{aligned}$$

Plugging in the definition of $O_{j,k}^0(a,b)$ completes the proof. □

Lemma 5.3.10 *Let $\Gamma_\ell,\Gamma_m\in\mathcal{T}_h$ be boundary elements with $|\Gamma_\ell\cap\Gamma_m|=0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s)=a+b\,s$ as defined in Section 5.1. Then, there holds for $j\in\mathbb{N}_0$ and $k\in\mathbb{N}$*

$$\begin{aligned} &\int_{\Gamma_\ell}\widetilde{P}_j^{(\ell)}(x)\int_{\Gamma_m} T^{(3)}(x,y)^T\,\widetilde{N}_k^{(m)}(y)\,ds_y\,ds_x \\ &= \frac{|v|}{2}\Bigg\{\operatorname{Im}\left(O_{j,k}^0(a,b)\right)\,\mathrm{I}-\operatorname{Re}\left(\widetilde{O}_{j,k}^1(a,b)\right)\,\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu} \\ &\qquad -\operatorname{Im}\left(\widetilde{O}_{j,k}^1(a,b)\right)\,\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\Bigg\} \end{aligned} \tag{5.16}$$

Proof. We start simplifying $T^{(3)}(x,y)$. Since there holds

$$T^{(3)}(x,y)=T^{(1)}(x,y)\cdot U^{(2)}(x,y)$$

we use the representations (5.7) and (5.13) and obtain

$$\begin{aligned} T^{(3)}(x,y) &= \frac{1}{2i\,|u|}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\left\{\frac{uu^T}{u^Tu}+\frac{\operatorname{Im}(z)}{2i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu}\right. \\ &\qquad \left.-\frac{\operatorname{Im}(z)}{2}\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\right\} \\ &= \frac{1}{|u|}\Bigg\{\frac{1}{2i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T}{u^Tu}-\frac{\operatorname{Im}(z)}{4}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)^2\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu} \\ &\qquad -\frac{\operatorname{Im}(z)}{4i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\Bigg\}. \end{aligned}$$

With

$$\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)^2=\left(\frac{1}{(z-t)^2}+\frac{1}{(\overline{z}-t)^2}\right)+\frac{1}{i\operatorname{Im}(z)}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)$$

and

$$\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\left(\frac{1}{z-t}+\frac{1}{\overline{z}-t}\right)=\left(\frac{1}{(z-t)^2}-\frac{1}{(\overline{z}-t)^2}\right)$$

we get

$$\begin{aligned}T^{(3)}(x,y)=\frac{1}{|u|}\Bigg\{&\frac{1}{2i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T}{u^Tu}-\frac{\operatorname{Im}(z)}{4}\left(\frac{1}{(z-t)^2}+\frac{1}{(\overline{z}-t)^2}\right)\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu}\\&-\frac{1}{4i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu}\\&-\frac{\operatorname{Im}(z)}{4i}\left(\frac{1}{(z-t)^2}-\frac{1}{(\overline{z}-t)^2}\right)\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\Bigg\}\\=\frac{1}{|u|}\Bigg\{&\frac{1}{4i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\frac{uu^T+u_\perp u_\perp{}^T}{u^Tu}\\&-\frac{\operatorname{Im}(z)}{4}\left(\frac{1}{(z-t)^2}+\frac{1}{(\overline{z}-t)^2}\right)\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu}\\&-\frac{\operatorname{Im}(z)}{4i}\left(\frac{1}{(z-t)^2}-\frac{1}{(\overline{z}-t)^2}\right)\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\Bigg\}.\end{aligned}$$

Simple calculus shows that $\frac{uu^T+u_\perp u_\perp{}^T}{u^Tu}=\mathrm{I}$ and thus

$$\begin{aligned}T^{(3)}(x,y)=\frac{1}{|u|}\Bigg\{&\frac{1}{4i}\left(\frac{1}{z-t}-\frac{1}{\overline{z}-t}\right)\mathrm{I}-\frac{\operatorname{Im}(z)}{4}\left(\frac{1}{(z-t)^2}+\frac{1}{(\overline{z}-t)^2}\right)\frac{uu^T-u_\perp u_\perp{}^T}{u^Tu}\\&-\frac{\operatorname{Im}(z)}{4i}\left(\frac{1}{(z-t)^2}-\frac{1}{(\overline{z}-t)^2}\right)\frac{(u_\perp u^T+uu_\perp{}^T)}{u^Tu}\Bigg\}.\end{aligned}$$

In general, there holds $\frac{1}{(\overline{z}-t)^2}=\overline{\frac{1}{(z-t)^2}},\ t\in\mathbb{R},\ z\in\mathbb{C}$, which implies

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}N_k(t)\,P_j(s)\operatorname{Im}(z)\left(\frac{1}{(\overline{z}-t)^2}-\frac{1}{(\overline{z}-t)^2}\right)ds\,dt\\&=\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt-\overline{\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt}\\&=2i\operatorname{Im}\left(\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt\right)\end{aligned}\tag{5.17}$$

and

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}N_k(t)\,P_j(s)\operatorname{Im}(z)\left(\frac{1}{(\overline{z}-t)^2}+\frac{1}{(\overline{z}-t)^2}\right)ds\,dt\\&=\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt+\overline{\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt}\\&=2\operatorname{Re}\left(\int_{-1}^{1}\int_{-1}^{1}\frac{N_k(t)\,P_j(s)\operatorname{Im}(z)}{(z-t)^2}\,ds\,dt\right).\end{aligned}\tag{5.18}$$

Thus, we obtain for the integration with (5.8), (5.9), (5.17) and (5.18)

$$\begin{aligned}
&\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} T^{(3)}(x,y)^T\, \widetilde{N}_k^{(m)}(y)\, ds_y\, ds_x \\
&= \frac{|v|}{2}\Bigg\{ \mathrm{Im}\left(\int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\, P_j(s)}{z-t}\, ds\, dt \right) \mathrm{I} \\
&\quad - \mathrm{Re}\left(\int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\, P_j(s)\, \mathrm{Im}(z)}{(z-t)^2}\, ds\, dt \right) \frac{uu^T - u_\perp u_\perp{}^T}{u^T u} \\
&\quad - \mathrm{Im}\left(\int_{-1}^{1}\int_{-1}^{1} \frac{N_k(t)\, P_j(s)\, \mathrm{Im}(z)}{(z-t)^2}\, ds\, dt \right) \frac{(u_\perp u^T + u u_\perp{}^T)}{u^T u} \Bigg\}.
\end{aligned}$$

Applying the definition of $O_{j,k}^0(a,b)$ and $\widetilde{O}_{j,k}^1(a,b)$ completes the proof. □

Summarizing the results we have the following formula for the computation of the Galerkin entries for the double layer operator.

Theorem 5.3.11 *Let $\Gamma_\ell, \Gamma_m \in \mathcal{T}_h$ be boundary elements with $|\Gamma_\ell \cap \Gamma_m| = 0$. Moreover, let u and v be the directional vectors of Γ_ℓ and Γ_m and $z(s) = a + b\,s$ as defined in Section 5.1. Then, there holds for $j \in \mathbb{N}_0$ and $k \in \mathbb{N}$*

$$\begin{aligned}
K_{j,k}^{(\ell,m)} = \frac{|v|}{2\pi}\mathrm{Im}\left(O_{j,k}^0(a,b)\right)\mathrm{I} + \frac{\mu\,|v|}{2\pi(\lambda+2\mu)}\mathrm{Re}\left(O_{j,k}^0(a,b)\right)\mathrm{I}\times\mathrm{I} \\
- \frac{(\lambda+\mu)\,|v|}{2\pi(\lambda+2\mu)}\left\{ \mathrm{Re}\left(\widetilde{O}_{j,k}^1(a,b)\right)\frac{uu^T - u_\perp u_\perp{}^T}{u^T u} \right. \\
\left. + \mathrm{Im}\left(\widetilde{O}_{j,k}^1(a,b)\right)\frac{(u_\perp u^T + u u_\perp{}^T)}{u^T u} \right\}.
\end{aligned} \tag{5.19}$$

5.3.3 The Hypersingular Operator

Instead of deriving an analytical formula for the computation of the Galerkin matrix of the hypersingular operator, as it is already done in the previous sections, we proceed as in [5] and use the formulas for the single layer operator and the property (v) in Lemma 3.1.5, i.e.

$$\langle\, W\psi_1, \psi_2 \,\rangle = \langle\, V^* \frac{d}{ds}\psi_1, \frac{d}{ds}\psi_2 \,\rangle \quad \forall\, \psi_1, \psi_2 \in H^{\frac{1}{2}}(\Gamma). \tag{5.20}$$

We calculate for a fixed combination of boundary elements Γ_ℓ and Γ_m and for fixed polynomial degrees j $(1 \le j \le p_\ell)$ and k $(1 \le k \le p_m)$ the 2×2-matrix

$$\begin{pmatrix}
\langle V^* \frac{d}{ds}\begin{pmatrix}\widetilde{N}_k^{(m)}\\ 0\end{pmatrix}, \frac{d}{ds}\begin{pmatrix}\widetilde{N}_j^{(\ell)}\\ 0\end{pmatrix}\rangle & \langle V^* \frac{d}{ds}\begin{pmatrix}\widetilde{N}_k^{(m)}\\ 0\end{pmatrix}, \frac{d}{ds}\begin{pmatrix}0\\ \widetilde{N}_j^{(\ell)}\end{pmatrix}\rangle \\
\langle V^* \frac{d}{ds}\begin{pmatrix}0\\ \widetilde{N}_k^{(m)}\end{pmatrix}, \frac{d}{ds}\begin{pmatrix}\widetilde{N}_j^{(\ell)}\\ 0\end{pmatrix}\rangle & \langle V^* \frac{d}{ds}\begin{pmatrix}0\\ \widetilde{N}_k^{(m)}\end{pmatrix}, \frac{d}{ds}\begin{pmatrix}0\\ \widetilde{N}_j^{(\ell)}\end{pmatrix}\rangle
\end{pmatrix}.$$

Thereby, the arc length derivative of $\widetilde{N}_j^{(\ell)}$ is given by

$$\begin{aligned}
\frac{d}{ds}\widetilde{N}_j^{(\ell)}(x) &= \frac{d}{ds}\left(N_j \circ \gamma_\ell^{-1}\right)(x) \\
&= \left(N_j' \circ \gamma_\ell^{-1}\right)(x) \cdot t^T \nabla \gamma_\ell^{-1},
\end{aligned}$$

where t denotes the unit tangential vector of Γ_ℓ with $t = \frac{u}{|u|}$ and u as defined in Section 5.1. Then, we get with $B_\ell - A_\ell = 2\,u$

$$t^T \nabla \gamma_\ell^{-1} = \frac{1}{|u|} u^T \frac{2\,u}{u^T u} = \frac{2}{|u|}.$$

Moreover, using the definition of the Lobatto shape functions we obtain for $j \geq 3$ $N_j'(t) = P_{j-2}(t)$, $N_2'(t) = \frac{1}{2}\,P_0(t)$ and $N_1'(t) = -\frac{1}{2}\,P_0(t)$ which yields

$$\begin{aligned} \frac{d}{ds}\,\widetilde{N}_j^{(\ell)}(x) &= \frac{2}{|u|}\,\widetilde{P}_{j-2}^{(\ell)}(x), \\ \frac{d}{ds}\,\widetilde{N}_2^{(\ell)}(x) &= \frac{1}{|u|}\,\widetilde{P}_0^{(\ell)}(x) \end{aligned}$$

and

$$\frac{d}{ds}\,\widetilde{N}_1^{(\ell)}(x) = -\frac{1}{|u|}\,\widetilde{P}_0^{(\ell)}(x).$$

Thus, we calculate for $j, k \geq 3$

$$\frac{4}{|u|\,|v|} \begin{pmatrix} \langle V^* \begin{pmatrix} \widetilde{P}_{k-2}^{(m)} \\ 0 \end{pmatrix}, \begin{pmatrix} \widetilde{P}_{j-2}^{(\ell)} \\ 0 \end{pmatrix} \rangle & \langle V^* \begin{pmatrix} \widetilde{P}_{k-2}^{(m)} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \widetilde{P}_{j-2}^{(\ell)} \end{pmatrix} \rangle \\ \langle V^* \begin{pmatrix} 0 \\ \widetilde{P}_{k-2}^{(m)} \end{pmatrix}, \begin{pmatrix} \widetilde{P}_{j-2}^{(\ell)} \\ 0 \end{pmatrix} \rangle & \langle V^* \begin{pmatrix} 0 \\ \widetilde{P}_{k-2}^{(m)} \end{pmatrix}, \begin{pmatrix} 0 \\ \widetilde{P}_{j-2}^{(\ell)} \end{pmatrix} \rangle \end{pmatrix}. \tag{5.21}$$

Note that the other cases $j \leq 3$ or $k \leq 3$ arise analogously.
As we already stated in the introduction the operators V and V^* are only distinguished from the factors of the kernel function. Hence, we use the result from Lemma 5.3.3 and Lemma 5.3.5 and obtain a formula for the calculation of the entries of the Galerkin matrix of the hypersingular operator.

5.4 A Posteriori Error Estimators

In this section we introduce error estimators for the Dirichlet, the Neumann and the mixed problem. Since all estimators are provided by the epsBEM framework we do not go into detail on the calculation and the implementation.

5.4.1 The Dirichlet Problem and Symm's Integral Equation

There are three different types of error estimator that are considered within this work for solving Symm's integral equation:

(i) Error estimators that are created by space enrichment with respect to the mesh-size (h - $h/2$ estimators).

(ii) Error estimators that are created by space enrichment with respect to the polynomial degree (p - p^* estimators).

(iii) Residual based error estimators.

In order to define the error estimators we introduce the following notations. Let $\phi \in H^{-\frac{1}{2}}(\Gamma)$ be the exact solution of Symm's integral equation (3.26), $\phi_h^p \in S^0(\mathcal{T}_h, p)$ be the numerical solution on a given triangulation $\mathcal{T}_h$ and $\phi_{h/2}^p \in S^0(\mathcal{T}_{h/2}, p)$ be the numerical solution on the uniformly refined triangulation $\mathcal{T}_{h/2}$. Moreover, we denote by Π_h^p the L^2-projection on $S^0(\mathcal{T}_h, p)$ and by $\|\!| \cdot |\!\|$ the energy norm on Γ which is induced by the single layer operator, i.e. $|\!|\!|\phi|\!|\!|^2 := \langle V\phi, \phi\rangle$.

Definition 5.4.1 (h-$h/2$ error estimators) *The h-$h/2$ error estimators for Symm's integral equation are given by*

$$(i) \quad \eta_h := |\!|\!| \phi_{h/2}^p - \phi_h^p |\!|\!| \tag{5.22}$$

$$(ii) \quad \widetilde{\eta}_h := |\!|\!| \left(1 - \Pi_h^p\right) \phi_{h/2}^p |\!|\!| \tag{5.23}$$

$$(iii) \quad \mu_h := \left\| \frac{h^{1/2}}{p+1} \left(\phi_{h/2}^p - \phi_h^p\right) \right\|_{L^2(\Gamma)} \tag{5.24}$$

$$(iv) \quad \widetilde{\mu}_h := \left\| \frac{h^{1/2}}{p+1} \left(1 - \Pi_h^p\right) \phi_{h/2}^p \right\|_{L^2(\Gamma)}. \tag{5.25}$$

Definition 5.4.2 (p-p^* error estimators) *For $p^* \geq p$, the p-p^* error estimators for Symm's integral equation are given by*

$$(i) \quad \eta_p := |\!|\!| \phi_h^{p^*} - \phi_h^p |\!|\!| \tag{5.26}$$

$$(ii) \quad \widetilde{\eta}_p := |\!|\!| \left(1 - \Pi_h^p\right) \phi_h^{p^*} |\!|\!| \tag{5.27}$$

$$(iii) \quad \mu_p := \left\| \frac{h^{1/2}}{p^*+1} \left(\phi_h^{p^*} - \phi_h^p\right) \right\|_{L^2(\Gamma)} \tag{5.28}$$

$$(iv) \quad \widetilde{\mu}_p := \left\| \frac{h^{1/2}}{p^*+1} \left(1 - \Pi_h^p\right) \phi_h^{p^*} \right\|_{L^2(\Gamma)}. \tag{5.29}$$

Definition 5.4.3 (Residual based error estimator) *The residual based error estimator for Symm's integral equation is given by*

$$\nu := \left\| \frac{h^{1/2}}{p+1} \frac{d}{ds} \left(V\phi_h^p - f\right) \right\|_{L^2(\Gamma)}, \tag{5.30}$$

where $\frac{d}{ds}$ denotes the arc length derivative on Γ.

Note that the η estimators cannot be used to create adaptive algorithms, since the energy norm cannot be localized meaning we cannot estimate the error on one boundary element. However, the μ estimators can be calculated locally and are consequentially appropriate for controlling the refinement in adaptive algorithms.
In [8] the efficiency and reliability of the h-$h/2$ estimators for $p = 0$ is proven, whereas there is no proof for the other error estimators. A detailed description of the calculations of all above defined estimators can be found in [10].

5.4.2 The Neumann Problem and the Hypersingular Integral Equation

For the hypersingular integral equation we introduce two different types of error estimators, namely the h-$h/2$ estimators and p-p^* estimators for $p^* \geq p$.
In order to defining the error estimators we introduce the following notation. Let $u \in H^{\frac{1}{2}}(\Gamma)$ be the exact solution of the hypersingular integral equation (3.31), $u_h^p \in S^1(\mathcal{T}_h, p)$ be the numerical solution on a given triangulation $\mathcal{T}_h$ and $u_{h/2}^p \in S^1(\mathcal{T}_{h/2}, p)$ be the numerical solution on the uniformly refined triangulation $\mathcal{T}_{h/2}$. Moreover, we denote by Π_h^p the L^2-projection on $S^1(\mathcal{T}_h, p)$ and by $\|\!|\cdot|\!\|$ the energy norm on Γ that is induced by the hypersingular operator, i.e. $|\!|\!|u|\!|\!|^2 := \langle Wu, u\rangle$ (If Γ is connected we define $|\!|\!|u|\!|\!|^2 := \langle\!\langle u, u\rangle\!\rangle_{W+S}$).

Definition 5.4.4 (h-$h/2$ error estimators) *The h-$h/2$ error estimators for the hypersingular integral equation are given by*

$$(i) \quad \eta_h := |\!|\!| u_{h/2}^p - u_h^p |\!|\!| \tag{5.31}$$

$$(ii) \quad \widetilde{\eta}_h := |\!|\!| \left(1 - \Pi_h^p\right) u_{h/2}^p |\!|\!| \tag{5.32}$$

$$(iii) \quad \mu_h := \left\| \frac{h^{1/2}}{p+1} \frac{d}{ds}(u_{h/2}^p - u_h^p) \right\|_{L^2(\Gamma)} \tag{5.33}$$

$$(iv) \quad \widetilde{\mu}_h := \left\| \frac{h^{1/2}}{p+1} \frac{d}{ds} \left(\left(1 - \Pi_h^p\right) u_{h/2}^p \right) \right\|_{L^2(\Gamma)}. \tag{5.34}$$

Definition 5.4.5 (p-p^* error estimators) *For $p^* \geq p$ the p-p^* error estimators for the hypersingular integral equation are given by*

$$(i) \quad \eta_p := |\!|\!| u_h^{p^*} - u_h^p |\!|\!| \tag{5.35}$$

$$(ii) \quad \widetilde{\eta}_p := |\!|\!| \left(1 - \Pi_h^p\right) u_h^{p^*} |\!|\!| \tag{5.36}$$

$$(iii) \quad \mu_p := \left\| \frac{h^{1/2}}{p^*+1} \frac{d}{ds}(u_h^{p^*} - u_h^p) \right\|_{L^2(\Gamma)} \tag{5.37}$$

$$(iv) \quad \widetilde{\mu}_p := \left\| \frac{h^{1/2}}{p^*+1} \frac{d}{ds} \left(\left(1 - \Pi_h^p\right) u_h^{p^*} \right) \right\|_{L^2(\Gamma)}. \tag{5.38}$$

Again, only the μ estimators are suitable for implementing adaptive algorithms.
The efficiency and the reliability of the h-$h/2$ estimators for $p = 0$ is proven in [7], whereas there is no proof for the other error estimator.

5.4.3 The Mixed Problem

For the mixed problem we introduce the h-$h/2$ and p-p^* estimators ($p^* \geq p$).
In order to define the error estimators we introduce the following notation. Let $\eta_{D,h}$, $\widetilde{\eta}_{D,h}$, $\eta_{D,p}$, $\widetilde{\eta}_{D,p}$ and $\mu_{D,h}$, $\widetilde{\mu}_{D,h}$, $\mu_{D,p}$ and $\widetilde{\mu}_{D,p}$ be the error estimators for Symm's

integral equation restricted to the Dirichlet boundary. Moreover, let $\eta_{N,h}$, $\widetilde{\eta}_{N,h}$, $\eta_{N,p}$, $\widetilde{\eta}_{N,p}$ and $\mu_{N,h}$, $\widetilde{\mu}_{N,h}$, $\mu_{N,p}$ and $\widetilde{\mu}_{N,p}$ be the error estimators for the hypersingular integral equation restricted to the Neumann boundary.

Definition 5.4.6 (h-$h/2$ error estimators) *The h-$h/2$ error estimators for the mixed problem are given by*

$$(i) \quad \eta_h^2 := \eta_{D,h}^2 + \eta_{N,h}^2 \tag{5.39}$$

$$(ii) \quad \widetilde{\eta}_h^2 := \widetilde{\eta}_{D,h}^2 + \widetilde{\eta}_{N,h}^2 \tag{5.40}$$

$$(iii) \quad \mu_h^2 := \mu_{D,h}^2 + \mu_{N,h}^2 \tag{5.41}$$

$$(iv) \quad \widetilde{\mu}_h^2 := \widetilde{\mu}_{D,h}^2 + \widetilde{\mu}_{N,h}^2. \tag{5.42}$$

Definition 5.4.7 (p-p^* error estimators) *For $p^* \geq p$ the p-p^* error estimators for the hypersingular integral equation are given by*

$$(i) \quad \eta_p^2 := \eta_{D,p}^2 + \eta_{N,p}^2 \tag{5.43}$$

$$(ii) \quad \widetilde{\eta}_p^2 := \widetilde{\eta}_{D,p}^2 + \widetilde{\eta}_{N,p}^2 \tag{5.44}$$

$$(iii) \quad \mu_p^2 := \mu_{D,p}^2 + \mu_{N,p}^2 \tag{5.45}$$

$$(iv) \quad \widetilde{\mu}_p^2 := \widetilde{\mu}_{D,p}^2 + \widetilde{\mu}_{N,p}^2. \tag{5.46}$$

This definition implies that the results concerning the efficiency and the reliability of the estimators can be transferred from the previous two subsections.

Chapter 6

Implementation

In this chapter we discuss the implementation. Since it is one of the main goals of this thesis to integrate all functions for solving the Navier-Lamé equation into the epsBEM framework, we give a short overview on the software package in the first section.
In the second section we go into detail on the implementation of the double integrals that we introduced in Section 4.2 and finally, we describe the implementation of the *hp*-BEM for the Navier-Lamé equation.

6.1 Overview on the epsBEM Package

epsBEM (**e**fficient and **p**-**s**table **B**oundary **E**lement **M**ethods) is a software package to solve the Laplace and the Navier-Lamé equations with the *hp*-BEM. Figure 6.1 shows an overview on all routines of the software package.

The focus of this software package is on the efficient and stable implementation such that the results are accurate close to machine precision. The main routines are supposed to be implemented in MATLAB, such that an easy handling for the user is provided.
Two C-libraries to calculate the associated Legendre functions and integrals thereof (bottom row in Figure 6.1) are the core of the software package. Using openMP and multi-precision libraries these C-libraries provide an efficient and stable calculation of the basic integrals.
Based on the C-libraries, the routines for the calculation of the Galerkin matrices are implemented in both C and MATLAB. On the one hand the code is supposed to be understandable and short (MATLAB-routines), on the other hand the routines are supposed to be efficient and stable (C-routines). Additionally, there are routines for assembling and solving the linear system of equations, routines for displaying the solution and routines for refining the mesh using different mesh refining strategies.
On the top layer, the error estimators and routines for calculating the exact error are implemented.
Within the scope of this thesis, we add the C-functions **O1**, **U1** and **Ut1** to the double integral library, implement the MATLAB- and C-routines for the the calculation of the Galerkin matrices and adjust some plot routines for displaying the displacement field. Moreover, we investigate the incorporation of gliding conditions (subsequently defined) for the Navier-Lamé problem.

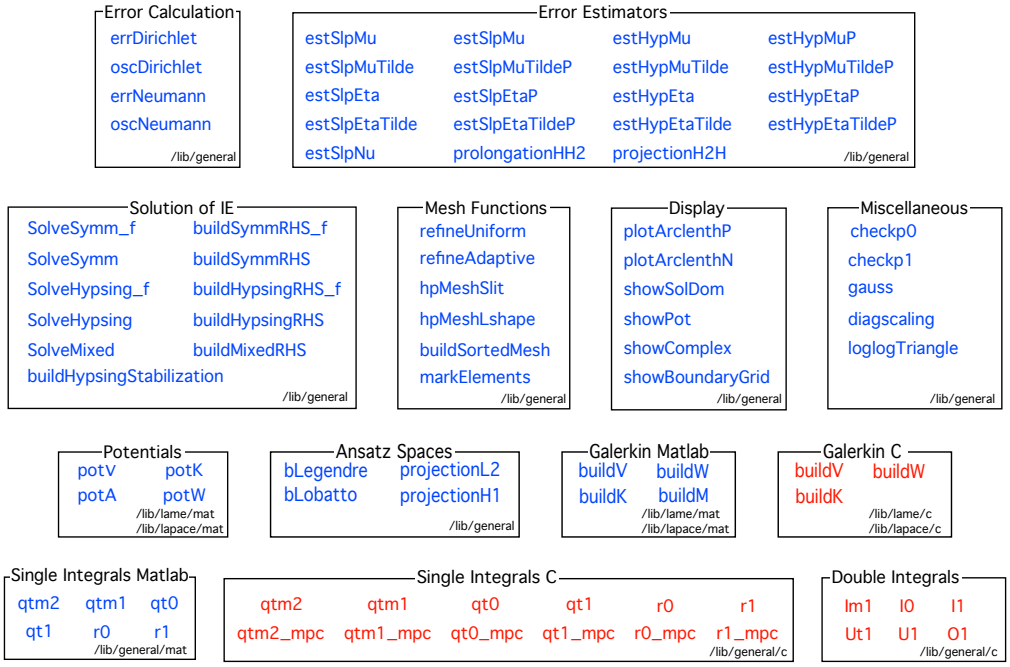

Fig. 6.1: Overview on the functions of the epsBEM package, where C-routines are depicted red and MATLAB-routines are depicted blue. Source:[10].

6.2 Integrals of the Associated Legendre Functions

The functions for the calculation of the double integrals $I^m_{j,k}$ and $O^m_{j,k}$ are implemented according to formulas that are derived in [19] and - in the case of $O^1_{j,k}$ - with the formulas that are proven in Section 4.2, respectively. Note that there are no routines for the calculation of the integrals $\widetilde{I}^0_{j,k}$ and $\widetilde{O}^1_{j,k}$ since these integrals are computed directly in the routines for the calculation of the Galerkin matrices using (4.26), (4.27), (4.28) and (4.29).
The routines for the calculation of $I^{-1}_{j,k}$, $I^0_{j,k}$, $O^0_{j,k}$ and $O^1_{j,k}$ are implemented in the C-library **liblegendre.c** and connected to MATLAB via mex-files. Thereby, the double integrals are calculated via recurrence relations in order to get an efficient implementation for high polynomial degrees p with complexity $\mathcal{O}(p^2)$ $(j = 0, .., p,\ k = 1, .., p+1)$. Since the recurrence relations are not stable we use the multi-precision libraries **mpfr** (The **M**ultiple **P**recision **F**loating-Point **R**eliable Library, see [9]) and **mpc** (The **M**ultiple **P**recision **C**omplex Library, see [6]) in order to ensure results that are accurate close to machine precision, i.e. the tolerance 10^{-14} is used. Using the libraries **mpfr** and **mpc** the number of digits that are used for the calculations is computed heuristically with the function **getdigits** that is provided by the epsBEM framework.
In the following, we exemplarily describe the implementation of $O^1_{j,k}$, as this function is one of the functions that is implemented within the scope of this thesis.

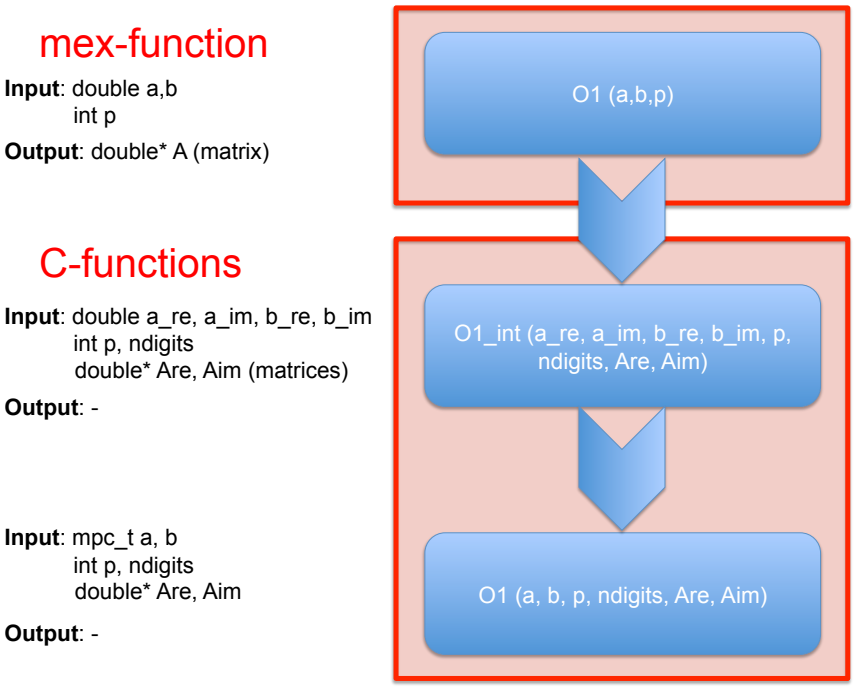

Fig. 6.2: Overview on the functions for the calculation of $O^1_{j,k}(a,b)$.

An overview on the routines for the calculation of $O^1_{j,k}$ is given in Figure 6.2. We see that the calculations are performed in two different levels (red boxes). Thereby, the first level is the interface between MATLAB and C and the second level converts the input parameters that are given in double precision to **mpc** variables and performs the computations. In the following we describe the implementation of the two levels more closely.
The first level is implemented by the mex-function **O1**. The input parameters are the complex numbers **a**, **b** (in double precision) and the polynomial degree **p**, and the output is the matrix $\mathbf{A} \in \mathbb{C}^{(p+1)\times(p+1)}$ which is given by

$$A = \begin{pmatrix} O^1_{0,1}(a,b) & \cdots & O^1_{0,p+1}(a,b) \\ \vdots & \ddots & \vdots \\ O^1_{p,1}(a,b) & \cdots & O^1_{p,p+1}(a,b) \end{pmatrix}.$$

The mex-function computes the number of digits that are needed for the calculation, allocates the memory for the output matrix **A** and calls the C-function **O1_int**.

Listing 6.1: Excerpt of liblegendre.c: O1_int.

```
1 void O1_int(double a_re, double a_im, double b_re, double
2     b_im, int p, int ndigits, double* Are, double* Aim)
3 {
4     mpc_t a, b;
```

```
5       mpc_init2  (a,  ndigits);
6       mpc_init2  (b,  ndigits);
7       /* set a and b */
8       mpc_set_d_d(a, a_re, a_im, MPC_RNDNN);
9       mpc_set_d_d(b, b_re, b_im, MPC_RNDNN);
10      /* call O1 */
11      O1(a,b,p,ndigits, Are, Aim);
12      /* clear a and b */
13      mpc_clear(a);
14      mpc_clear(b);
15  }
```

The second layer is implemented by the functions **O1_int** and **O1**.
Listing 6.1 shows the code of **O1_int**. In the lines 4-6 the **mpc** variables **a** and **b** are first defined and initialized, then the function **O1**, which performs the calculations, is called (line 11). Finally, **a** and **b** are deleted (lines 13-14).
The computation of the matrix **A** is performed in the C-function **O1** according to (4.30) with the formulas that are derived in Lemmas 4.2.4, 4.2.5 and 4.2.6. Therefore, we do not go into detail on the complete implementation, here. However, we describe the computation of η_2 and $\widetilde{Q}_k^{-1}(\eta_2\,(a \pm b))$ and take a closer look at the implementation for $a \pm b = \pm 1$ in the following.

Listing 6.2: Excerpt of liblegendre.c: O1 (computation of η_2).

```
1   eta2=1;
2   if ( (a_im*a_im - b_im*b_im)<0 &&
3        (a_re*b_im - a_im*b_re)/b_im<= -1+1e-14 )
4   {
5       eta2=-1;
6   } else {
7       int cond1 ,cond2;
8       cond1 = fabs(a_im+b_im) < 1e-14;
9       cond2 = fabs(a_im-b_im) < 1e-14;
10      if ( (cond1 && cond2  && (a_re-b_re) <= -1+1e-14) ||
11           (cond1 && (a_re+b_re) <= -1+1e-14) ||
12           (cond2 && (a_re-b_re) <= -1+1e-14) )
13          {
14              eta2=-1;
15          }
16  }
```

Listing 6.2 shows the calculation of η_2. The four situations in which $\mathrm{conv}\{a+b, a-b\} \cap \{x \in \mathbb{R} : x < -1\} \neq \emptyset$ and thus $\eta_2 = -1$ are illustrated in Figure 6.3. In line 2-3 in Listing 6.2 we check the first situation. Thereby, we check if $\mathrm{conv}\{a+b, a-b\}$

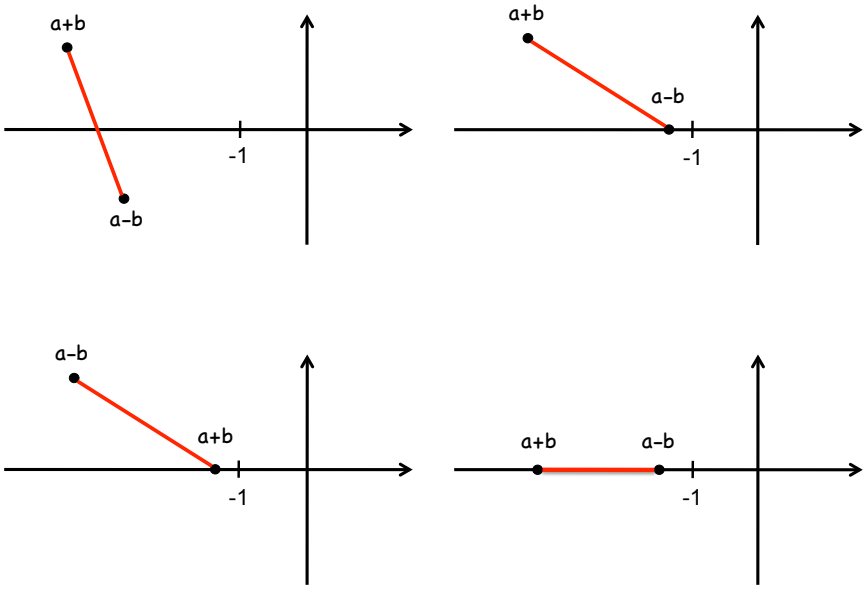

Fig. 6.3: Configurations with $\eta_2 = -1$.

intersects the real axis ($\mathrm{Im}(a+b) \cdot \mathrm{Im}(a-b) < 0$, line 2) and if the intersection is smaller than -1 (line 3). The remaining three situations are treated in line 10-12. We check if $a+b \in \mathbb{R}$ with $a+b \leq -1$ (line 10), if $a-b \in \mathbb{R}$ with $a-b \leq -1$ (line 11) or if $a \pm b \in \mathbb{R}$ with $a \pm b \leq -1$ (line 12). As already mentioned above, the tolerance 10^{-14} is used for the case differentiation.

Listing 6.3: Excerpt of liblegendre.c: O1 (calculation of $\widetilde{Q}_k^{-1}(\eta_2(a \pm b))$).

```
1  mpc_add(z1, a, b, MPC_RNDNN);   /* z1 = a+b;*/
2  mpc_sub(z2, a, b, MPC_RNDNN);   /* z2 = a-b;*/
3  if (eta2==-1){
4      mpc_neg  (z1, z1, MPC_RNDNN);
5      mpc_neg  (z2, z2, MPC_RNDNN);
6  }
7  qtm1_ex (z1, p+1, ndigits, c5);
8  qtm1_ex (z2, p+1, ndigits, c6);
9  if (eta2==1)
10     for (i=0; i<=p+1; i++)
11         mpc_sub(c5[i], c5[i], c6[i], MPC_RNDNN);
12
13 else{ /* multiply with (-1)^k */
14     for (i=0; i<=p+1; i+=2) /* even indices: c5-c6 */
15         mpc_sub(c5[i], c5[i], c6[i], MPC_RNDNN);
```

```
16      for (i=1; i<=p+1; i+=2) /* odd indices: c6-c5 */
17          mpc_sub(c5[i], c6[i], c5[i], MPC_RNDNN);
18  }
```

Listing 6.3 shows the calculation of $\widetilde{Q}_k^{-1}(\eta_2(a+b)) - \widetilde{Q}_k^{-1}(\eta_2(a-b))$. In line 1-6 we compute $z_1 = a+b$ and $z_2 = a-b$ and, if $\eta_2 = -1$, $z_1 = -a-b$ and $z_2 = -a+b$, respectively. Then, we call the function **qtm1_ex** that computes $\widetilde{Q}_k^{-1}(z)$, $k = 0, ..., p+1$, and save the result in the **mpc** arrays **c5** and **c6**. In the case that $\eta_2 = 1$ we compute $\widetilde{Q}_k^{-1}(\eta_2(a+b)) - \widetilde{Q}_k^{-1}(\eta_2(a-b))$ (line 9-11), whereas we calculate $(-1)^k \left[\widetilde{Q}_k^{-1}(\eta_2(a+b)) - \widetilde{Q}_k^{-1}(\eta_2(a-b))\right]$ in the case that $\eta_2 = -1$ (line 13-18). We change the order in the subtraction for all odd indices k (line 16-17). Hence, we add implicitly $4\pi\, i$ in the case of $k = 0$. Note that the implementation of η_1 and $\widetilde{Q}_k^{-1}\left(\eta_1 \frac{\pm 1-a}{b}\right)$ is done analogously.
Finally, we describe the implementation in the special case that $a \pm b = \pm 1$, where we only refer to the case $a + b = 1$. As we already stated in the proof of Theorem 4.2.7, the coefficients of the singular terms in the formulas for the computation of $\widetilde{O}_{j,k}^1(a,b)$ vanish. Therefore, we set $\widetilde{Q}_k^0(1)$ to its finite part, i.e.

$$\widetilde{Q}_k^0(1) := -2W_{k-1}(1),$$

where we again use the tolerance of 10^{-14}. We want to stress that this method only works if either only $\widetilde{Q}_k^0(a+b)$ or $\widetilde{Q}_k^0\left(\frac{1-a}{b}\right)$ occur in the formula for the calculation of $\widetilde{O}_{j,k}^1(a,b)$. This effect is due to the fact that the sequences $(a_n + b_n)_{n\in\mathbb{N}}$ and $(\frac{1-a_n}{b_n})_{n\in\mathbb{N}}$ converge to 1 from different directions. Thus, combining $\widetilde{Q}_k^0(a+b)$ and $\widetilde{Q}_k^0\left(\frac{1-a}{b}\right)$ results in a different value for $a + b = 1$, which yields a discontinuous behavior of the double integral $\widetilde{O}_{j,k}^1(a,b)$ in the neighborhood of 1. Note that the formulas that we derived for the computation of $\widetilde{O}_{j,k}^1$ fulfill this constraint, whereas the formulas that are stated in [19] do not fulfill this constraint and cannot be used for the implementation.

6.3 Implementation of the *hp*-BEM

In this section we describe the implementation of the *hp*-BEM for the Dirichlet, the Neumann and the mixed problems. Since the main routines and the routines for solving the three different problems are provided by the epsBEM framework and were only adjusted for the Navier-Lamé equation, we only describe the procedure in solving the problems but not the implementation. However, in the subsequent sections we go into detail on the implementation of the routines that are different from the routines for the Laplace equation and implemented within this work.
As we already derived in Chapter 3 for solving the mixed, the Dirichlet and the Neumann problems we solve

$$\mathbf{Ax} = \left(\frac{1}{2}\mathbf{M} - \mathbf{A}\right)\begin{pmatrix}\mathbf{g}\\ \mathbf{u_D}\end{pmatrix}, \tag{6.1}$$

$$\mathbf{Vx} = \left(\mathbf{K} + \frac{1}{2}\mathbf{M}\right) \mathbf{u_D} \tag{6.2}$$

and

$$(\mathbf{W} + \mathbf{S})\mathbf{u} = \left(\frac{1}{2}\mathbf{M} - \mathbf{K}^T\right) \mathbf{g}. \tag{6.3}$$

We proceed as follows:

(i) We project the Dirichlet data that is given by the function handle u_D and the Neumann data that is given by the function handle g to the discrete spaces $S^1(\mathcal{T}_h, p_1)$ and $S^0(\mathcal{T}_h, p_0)$, respectively. For this, we use the MATLAB functions **projectionH1.m** and **projectionL2.m** and as a result we obtain the coefficient vectors $\mathbf{u_D}$ with respect to basis of $S^1(\mathcal{T}_h, p_1)$ and $\mathbf{g}$ with respect to basis of $S^0(\mathcal{T}_h, p_0)$.

(ii) Depending on the problem we assemble the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$ and $\mathbf{W}$ by calling the functions **buildV.m**, **buildK.m** and **buildW.m**, the stabilization matrix by calling **buildHypsingStabilization.m** and the mass matrix by calling **buildM.m**. In the case of the mixed problem we restrict the matrices $\mathbf{V}$, $\mathbf{K}$ and $\mathbf{W}$ to the degrees of freedom of the Neumann and the Dirichlet boundaries, since (6.1) holds on $\Gamma_N \times \Gamma_D$. Thus, we solve a $(\mathcal{N}_N^1 + \mathcal{N}_D^0) \times (\mathcal{N}_N^1 + \mathcal{N}_D^0)$ system.

(iii) We solve the linear system of equations (6.1) with the backslash operator of MATLAB and obtain the coefficient vector of the numerical solution with respect to the bases of $S^0(\mathcal{T}_h, p_0)$ and $S^1(\mathcal{T}_h, p_1)$.

In the subsequent sections we amplify the implementation of the Galerkin matrices and investigate another type of boundary conditions, namely gliding conditions.

6.3.1 Implementation of the Galerkin Matrices

For the calculation of the Galerkin matrices there are both C-functions and MATLAB functions. We describe the implementation of the C- and the MATLAB functions by taking the example of the single layer operator. Note that the complete code for calculating the Galerkin matrices can be found in Appendix B.
An overview on the MATLAB functions for the calculation of the Galerkin entries is given in Figure 6.4. All functions in the red boxes are implemented in the MATLAB file **buildV.m**.

Figure 6.4 shows that the calculation is implemented on three different levels (red boxes). In the lowest level we perform the calculation of the 2×2 block matrix that we introduced in Section 5.3.1 for one fixed combination of boundary elements. As we already stated in Section 5.3 we distinguish the case of identical elements (**galV_id**) and non-identical elements (**galV_ex**).
In the following we describe the implementation of **galV_ex**. The input parameters of **galV_ex** are

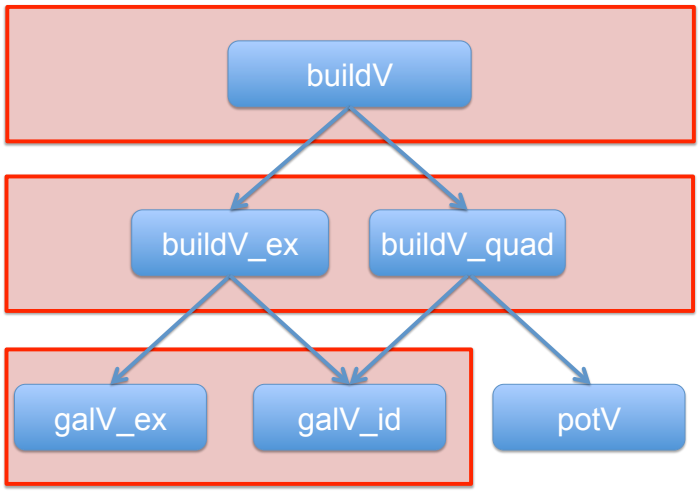

Fig. 6.4: Overview on the functions for the calculation of the Galerkin matrix for the single layer operator.

- **vert_x** $\in \mathbb{R}^{2\times 2}$: Matrix that contains the edges of Γ_ℓ, i.e. A_ℓ corresponds to the first row and B_ℓ corresponds to the second row.
- **vert_y** $\in \mathbb{R}^{2\times 2}$: Matrix that contains the edges of Γ_m, i.e. A_m corresponds to the first row and B_m corresponds to the second row.
- **p** $\in \mathbb{N}_0$: Polynomial degree.
- **options**: Struct that contains the Lamé coefficients λ and μ.

The output **val** is the 2×2 block matrix that is introduced in Section 5.3.1. Listing 6.4 shows an excerpt of the code.

Listing 6.4: Excerpt of buildV.m: galV_ex.

```
%*** compute u,v,w,a,b
...
%*** compute matrices
uut    = u'*u / utu;
usust = [u(2);-u(1)]*[u(2),-u(1)] / utu;
uust   = u'*[u(2),-u(1)] / utu;
usut   = [u(2);-u(1)]*u / utu;
%*** compute integrals Im1
im1 = real(Im1(a,b,p));
tmpm1 = [im1,zeros(p+1);zeros(p+1),im1];
%*** compute It0 => combination of rows!
i0   = [zeros(1,p+2);I0(a,b,p+1)];
```

```
13 c1 = repmat((1:(p+1))./[1,3:2:(2*p+1)],p+1,1)';
14 c2 = repmat((0:p)./[1,3:2:(2*p+1)],p+1,1)';
15 tmp0 = a2*i0(2:end-1,1:p+1)+...
16        b2*(c1.*i0(3:end,1:p+1)+c2.*i0(1:end-2,1:p+1));
17 %*** compute matrices  multiplied with intgrals
18 U1 = [(uut(1,1)-usust(1,1))*imag(tmp0),...
19       (uut(1,2)-usust(1,2))*imag(tmp0);...
20       (uut(2,1)-usust(2,1))*imag(tmp0),...
21       (uut(2,2)-usust(2,2))*imag(tmp0)];
22 U2 = [(uust(1,1)+usut(1,1))*real(tmp0),...
23       (uust(1,2)+usut(1,2))*real(tmp0);...
24       (uust(2,1)+usut(2,1))*real(tmp0),...
25       (uust(2,2)+usut(2,2))*real(tmp0)];
26 %*** compute single layer potential
27 val=-fac*(tmpm1-fac2*(U1-U2));
28 %*** add deltak0*deltaj0
29 tmp=fac*2*(log(utu)*eye(2)-2*fac2*uut);
30 val([1,p+2],[1,p+2]) = val([1,p+2],[1,p+2]) - tmp;
```

After having computed the vectors u, v and w according to Figure 5.1 and $a, b \in \mathbb{C}$ according to (5.3) we compute the matrices

$$\frac{uu^T}{u^Tu}, \quad \frac{u_\perp u_\perp{}^T}{u^Tu} \quad \text{and} \quad \frac{uu_\perp{}^T + u_\perp u^T}{u^Tu}$$

in line 4-7. In line 9-10 we compute the real part of the integral $\left(I^{-1}_{j,k}(a,b)\right)_{j,k=0,\dots,p}$ by calling the C-function **Im1** and assemble the 2×2 block matrix

$$\mathbf{tmpm1} = \left(\begin{array}{c|c} \mathrm{Re}\left(I^{-1}_{j,k}(a,b)\right)_{j,k=0,..,p} & 0 \\ \hline 0 & \mathrm{Re}\left(I^{-1}_{j,k}(a,b)\right)_{j,k=0,\dots,p} \end{array}\right).$$

Afterwards, we compute the matrix $\left(\widetilde{I}^{0}_{j,k}(a,b)\right)_{j,k=0,\dots,p}$ in lines 12-16. Therefore, we call the C-function **I0** to calculate the matrix $\left(I^{0}_{j,k}(a,b)\right)_{j,k=0,\dots,p}$ and combine the rows according to (4.26) and (4.27). Note that defining $I^0_{-1,k}(a,b) := 0$ we can treat (4.27) similar to (4.26) for $j = 0$ and thus the first row of **i0** is initialized with zeros (line 12). Since MATLAB can perform matrix operations efficiently, we combine all rows at once without a loop. We build the coefficient matrices $\mathbf{c1} := (c1_{j,k})_{j,k=0,\dots,p}$ and $\mathbf{c2} := (c2_{j,k})_{j,k=0,\dots,p}$ with

$$c1_{j,k} = \frac{j}{2j+1} \quad \text{and} \quad c2_{j,k} = \frac{j+1}{2j+1}$$

and combine the whole sub matrices with the elementwise multiplication (line 15-16). In line 18-24 we compute the 2×2 block matrices

$$\mathbf{U1} = \left(\begin{array}{c|c} \mathrm{Im}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu^T - u_\perp u_\perp^T}{u^T u}\right)_{1,1} & \mathrm{Im}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu^T - u_\perp u_\perp^T}{u^T u}\right)_{1,2} \\ \hline \mathrm{Im}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu^T - u_\perp u_\perp^T}{u^T u}\right)_{2,1} & \mathrm{Im}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu^T - u_\perp u_\perp^T}{u^T u}\right)_{2,2} \end{array} \right)$$

and

$$\mathbf{U2} = \left(\begin{array}{c|c} \mathrm{Re}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu_\perp{}^T + u_\perp u^T}{u^T u}\right)_{1,1} & \mathrm{Re}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu_\perp{}^T + u_\perp u^T}{u^T u}\right)_{1,2} \\ \hline \mathrm{Re}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu_\perp{}^T + u_\perp u^T}{u^T u}\right)_{2,1} & \mathrm{Re}\left(\widetilde{I}^0_{j,k}(a,b)\right)_{j,k=0,..,p} \left(\frac{uu_\perp{}^T + u_\perp u^T}{u^T u}\right)_{2,2} \end{array} \right).$$

Finally, we compute Galerkin matrix according to (5.10), where we add the term with $\delta_{k,0}\delta_{j,0}$ in lines 28-29.
Note that in **galV_ex** we distinguish the case that $|\Gamma_\ell| < |\Gamma_m|$ in which we proceed as described above and the case that $|\Gamma_\ell| \geq |\Gamma_m|$ in which we change the order of the integration, as the formulas for the calculation of the number of digits for **Im1** and **I0** prescribe that the element of the outer integral is smaller. One can show that by changing the order of integration we get

$$\begin{array}{ccc} v & \leftrightsquigarrow & u \\ w & \rightsquigarrow & -w \\ a & \rightsquigarrow & -\frac{a}{b} \\ b & \rightsquigarrow & \frac{1}{b}. \end{array}$$

Besides these changes, the calculation that is described above is exactly the same in the other case.
In the second level the routines **buildV_ex** and **buildV_quad** are implemented. The input parameters of both functions are given by:

- **coordinates** $\in \mathbb{R}^{N_c \times 2}$: Matrix that contains the vertices of the boundary Γ; each row contains the x_1- and x_2-coordinate of one edge. (N_c denotes the number of edges of the triangulation.)
- **elements** $\in \mathbb{R}^{N_{el} \times 2}$: Matrix that describes the boundary elements; each row contains the indices of the edges of one boundary element.
- **p** $\in \mathbb{N}_0^{N_{el}}$: Vector that contains the polynomial degree of each boundary element.
- **options**: Struct that contains the Lamé parameters λ and μ.

Both routines return the Galerkin matrix **V** of the single layer operator. However, the routines **build_ex** and **buildV_quad** differ in the calculation of the entries of

the Galerkin matrix. **buildV_ex** calculates the entries analytically by calling the functions **galV_ex** and **galV_id**, whereas the entries for non-identical elements are computed semi-analytically in **buildV_quad**, i.e. the inner integral in

$$\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x$$

is computed analytically with the function **potV** that is described in [2] and the outer integral is computed vice quadrature. Thus, we obtain

$$\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \underbrace{\int_{\Gamma_m} U(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y}_{=\mathbf{potV}(x)}\, ds_x \approx \frac{|\Gamma_\ell|}{2} \sum_{\nu=1}^{N_g} \omega_\nu^g\, P_j(x_\nu^g)\, \mathbf{potV}(x_\nu^g), \tag{6.4}$$

where N_g denotes the number of Gauss nodes and ω_ν^g and x_ν^g denote the Gauss weights and points. Defining the matrix $\mathbf{P} = (\mathbf{p}_{\nu,j})_{\nu=1,\dots,N_g}^{j=0,\dots,p}$ with $\mathbf{p}_{\nu,j} = P_j(x_\nu^g)$ and the matrix $\mathbf{W} \in \mathbb{R}^{N_g \times (p+1)}$ that contains the Gauss weights we can compute (6.4) for all $j = 0, ..., p$ and $\nu = 1, .., N_g$ efficiently by

$$\int_{\Gamma_\ell} \widetilde{P}_j^{(\ell)}(x) \int_{\Gamma_m} U(x,y)\, \widetilde{P}_k^{(m)}(y)\, ds_y\, ds_x = \frac{|\Gamma_\ell|}{2} (\mathbf{W}. * \mathbf{P})^T * \mathbf{potV}.$$

Here, $.*$ denotes the elementwise product and $*$ denotes the matrix-matrix multiplication.
The top level in Figure 6.4 is used for exception handling and calling the sub-routines for the analytical and semi-analytical computations.

The C-routines for the analytical computation of the Galerkin matrices are implemented in the C-library **libGalerkinLame.c**. The structure of the routines is similar the structure of the MATLAB functions. Since the recurrence relations for the calculation of the double integral $I_{j,k}^{-1}(a,b)$ and $I_{j,k}^{0}(a,b)$ are very sensitive concerning the input parameters a and b, the multi-precision libraries **mpfr** and **mpc** are used. In order to reduce the computational times we only compute a and b with the multi-precision libraries, the other calculations are performed in double precision.
Moreover, the routines for the calculation of the Galerkin matrices are parallelized with openMP so that every thread calculates the sub-matrix for a fixed combination of boundary elements. For a detailed description of the parallelization we refer the [3].
An analysis of the computational times for the assembly of the Galerkin matrices with both the C- and the MATLAB routines is given in Figure 6.5. The calculations were performed on a computer with the following setup:

Processor: AMD Phenom II X6 1090T
Operating System: Ubuntu 10.4
RAM: 16GB DDR3.

Figure 6.5 shows the computational time for several different methods, i.e. a geometric *hp*-method with 660 degrees of freedom (green), a uniform *h*-method with 512 elements and $p = 0$ (blue), a uniform *hp*-method with 32 elements and $p = 32$

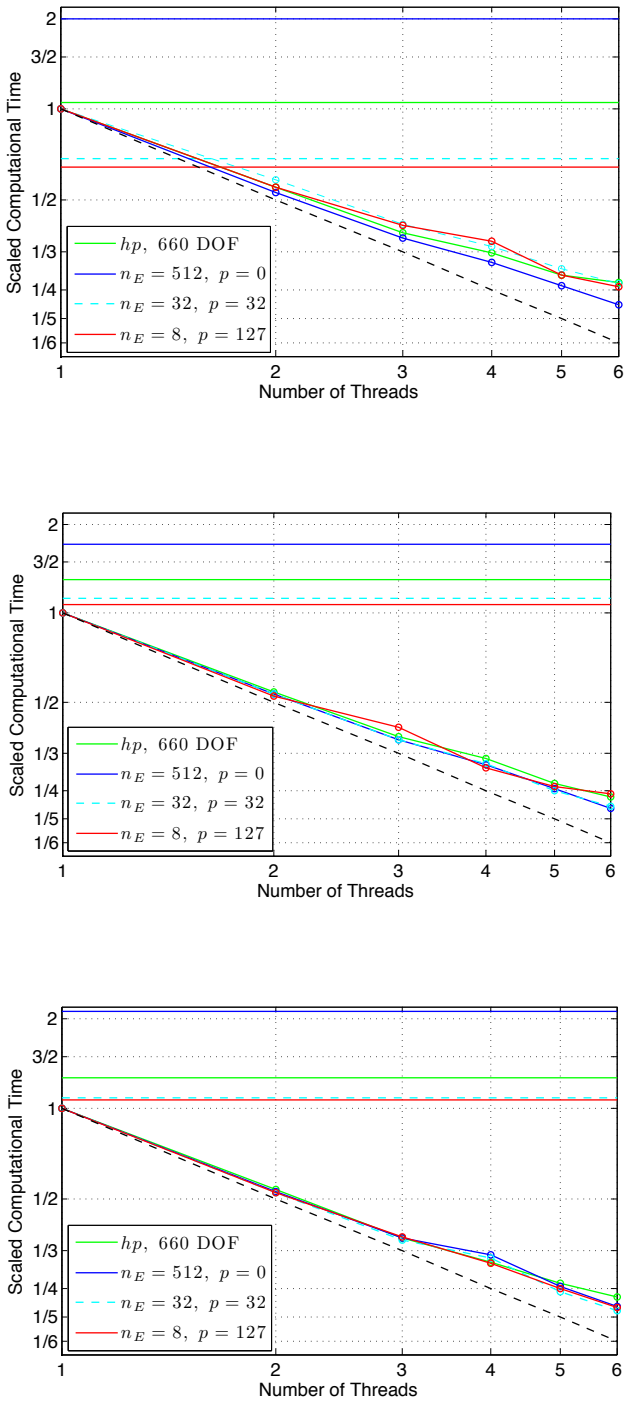

Fig. 6.5: Computational times for the assembly of the Galerkin matrices $\mathbf{V}$ (top), $\mathbf{K}$ (middle) and $\mathbf{W}$ (bottom) with the C-functions depending on the number of threads. The black dashed line is the optimal scaling and the horizontal lines denote the computational time for the assembly with MATLAB routines.

(dashed light blue) and a uniform p-method with 8 elements and $p = 127$ (red). The times that are illustrated in the plots are the average times of three runs each and scaled with the computational time of the C-routines with one thread.

The results that we obtain for the Navier-Lamé equation are similar to the results that are described in [10] for the Laplace equation. All three plots show that we do not achieve the optimal scaling (black dashed line) for the computations with the C-routines, the maximal scaling factor that is achieved using 6 threads is in the range of $3.9 - 4.4$. This is what we expected, since every thread calculates a fixed number of blocks of the Galerkin matrices, i.e. the blocks are not distributed dynamically to the threads. Thus, if the number of blocks is not divisible by the number of threads, some threads are finished while other threads calculate an extra block. Additionally, the threads block each other while writing the results into the Galerkin matrix, since the entries for the hat function are added up and thus exclusive writing is necessary.
Moreover, we see that the scaling factor for $\mathbf{V}$ is worse than the scaling factors for $\mathbf{K}$ and $\mathbf{W}$. This is due to the fact that the computational effort for computing a matrix block is higher for the double layer and the hypersingular operator than for the single layer operator and hence the parallelization has a bigger impact.
Comparing the MATLAB routines to the C-routines with one thread we see that the MATLAB routines are slower than the C-routines except for the calculations of $\mathbf{V}$ with high polynomial degrees. This is due to the fact that C can perform simple calculations very efficiently, whereas MATLAB can perform matrix operations very efficiently. Thus, the MATLAB routines for the calculation of $\mathbf{V}$ are faster than the C-routines for high polynomials degrees, since we only used matrix operations in **galV_ex**. For h-methods the matrix blocks are smaller and the matrix operations have less impact and consequentially the MATLAB functions are slower. Furthermore, since the computational effort for computing $\mathbf{K}$ and $\mathbf{W}$ is bigger than for $\mathbf{V}$ the efficient matrix operations of MATLAB have a smaller impact and thus, the MATLAB routines for the double layer and the hypersingular operator are slower than the C-routines even with high polynomial degrees.

6.3.2 Mixed Problem with Gliding Conditions

In this section, we discuss the implementation of another type of boundary conditions that occurs in many applications in the field of linear elasticity, namely the gliding conditions. Besides the Dirichlet boundary Γ_D where the displacement is given and the Neumann boundary Γ_N where the traction is given, we denote the gliding boundary by Γ_G and prescribe the gliding conditions, i.e. the displacement of the body is fixed in one specified direction $n \in \mathbb{R}^2$ and the traction is fixed in another specified direction $m \in \mathbb{R}^2$. Thus, we get for $h \in H^{\frac{1}{2}}(\Gamma_G) \times H^{-\frac{1}{2}}(\Gamma_G)$ the following condition

$$\begin{pmatrix} n^T \gamma_0 \, u \\ m^T \gamma_1 u \end{pmatrix} = h \quad \text{on } \Gamma_G. \tag{6.5}$$

We do not go into detail on the theory of the mixed problems with gliding conditions, i.e. the existence and the uniqueness of a solution, but discuss the numerical realization.

A standard example in linear elasticity that can be solved using gliding conditions is the membrane with a hole that is illustrated in Figure 6.6.

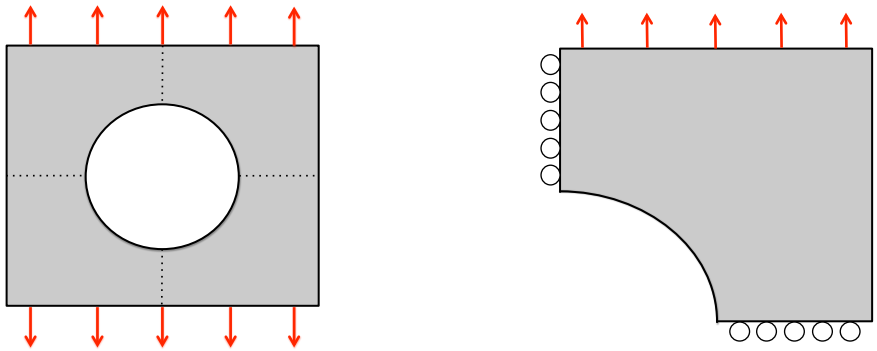

Fig. 6.6: Membrane with a hole where a surface force is applied on the upper and lower boundary (left), part of the membrane with gliding conditions displayed by the circles (right).

As we can see in Figure 6.6 (left) there holds $\Gamma = \Gamma_N$ with $\gamma_1 u = g(x) := (0, -1)^T$ on the lower boundary, $\gamma_1 u = g(x) := (0, 1)^T$ on the upper boundary and $\gamma_1 u = g(x) := (0, 0)^T$ on the left and right boundary. Instead of solving this Neumann problem, we split the membrane in four parts and only calculate the solution on one of the parts exploiting the symmetry of the solution (see Figure 6.6, right). Thereby, we get gliding conditions on the cut edges, i.e. there holds

$$\begin{pmatrix} \gamma_0 u_1 \\ \gamma_1 u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

on the left boundary and

$$\begin{pmatrix} \gamma_0 u_2 \\ \gamma_1 u_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

on the lower boundary.
For incorporating the gliding conditions, we cannot proceed as we did for the mixed problem in Section 3.1, to be more precisely, we cannot use the symmetric formulation of the Caldéron system (3.17). Instead we rearrange the Caldéron system (3.16) as follows

$$\begin{pmatrix} V & -\frac{1}{2}\mathrm{I} - K \\ -\frac{1}{2}\mathrm{I} + K' & W \end{pmatrix} \begin{pmatrix} \gamma_1 u \\ \gamma_0 u \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

For the implementation we prescribe a sorted mesh, i.e.

$$\mathbf{elements} = \begin{pmatrix} \mathbf{dirichlet} \\ \mathbf{neumann} \\ \mathbf{gliding} \end{pmatrix}$$

and we denote the coefficient vector of $u_h^{p_1}$ by

$$\mathbf{u} = \left(\mathbf{u_1^{(D)}}, \mathbf{u_1^{(N)}}, \mathbf{u_1^{(G)}}, \mathbf{u_2^{(D)}}, \mathbf{u_2^{(N)}}, \mathbf{u_2^{(G)}}\right)^T$$

and the coefficient vector of $\phi_h^{p_0}$ by

$$\mathbf{x} = \left(\mathbf{x_1^{(D)}}, \mathbf{x_1^{(N)}}, \mathbf{x_1^{(G)}}, \mathbf{x_2^{(D)}}, \mathbf{x_2^{(N)}}, \mathbf{x_2^{(G)}}\right)^T.$$

Using the previously defined Galerkin matrices we obtain the discrete system

$$\begin{pmatrix} \mathbf{V} & -\frac{1}{2}\mathbf{M} - \mathbf{K} \\ -\frac{1}{2}\mathbf{M} + \mathbf{K}^T & \mathbf{W} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

Then, the boundary conditions are incorporated by extending the linear system of equations to

$$\begin{pmatrix} \mathbf{V} & -\frac{1}{2}\mathbf{M} - \mathbf{K} & \mathbf{M_1}^T \\ -\frac{1}{2}\mathbf{M} + \mathbf{K}^T & \mathbf{W} & \mathbf{M_2}^T \\ \mathbf{M_1} & \mathbf{M_2} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{\Lambda} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{f} \end{pmatrix}, \tag{6.6}$$

where $\mathbf{\Lambda}$ denotes the Lagrange parameter and the vector

$$\mathbf{f} = (\mathbf{u_{D,1}}\ ,\ \mathbf{u_{D,2}}\ ,\ \mathbf{g_1}\ ,\ \mathbf{g_2}\ ,\ \mathbf{h_1}\ ,\ \mathbf{h_2})^T$$

denotes the coefficient vector of the given Dirichlet, Neumann and gliding data. The matrices $\mathbf{M_1}$ and $\mathbf{M_2}$ are given by

$$\mathbf{M_1} = \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & \mathrm{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1\,\mathrm{I} & 0 & 0 & m_2\,\mathrm{I} \end{array}\right)$$

and

$$\mathbf{M_2} = \left(\begin{array}{cccccc} \mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & n_1\,\mathrm{I} & 0 & 0 & n_2\,\mathrm{I} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right),$$

where the upper part of both matrices corresponds the Dirichlet data, the middle part to the Neumann data and the lower part to the gliding data. Thus, the given boundary conditions are fulfilled due to last equation of the extended system

$$\mathbf{M_1 x} + \mathbf{M_2 u} = \mathbf{f}.$$

Note that the system (6.6) is not appropriate for solving the mixed problem without gliding conditions as there are also disadvantages compared to the system of linear

equations for the mixed problem (6.1). First, system (6.6) is not symmetric and thus, efficient algorithms, such as the cg-method, cannot be used for solving the system. Moreover, using (6.6) we have to solve a bigger system of linear equations. Besides the extension of the Caldéron system by the matrices $\mathbf{M_1}$ and $\mathbf{M_2}$ we do not restrict the Galerkin matrices $\mathbf{V}$, $\mathbf{K}$ and $\mathbf{K}$ to the degrees of freedom of the Dirichlet and the Neumann boundary as it is done in (6.1). Therefore, the extended system is of the size

$$(\mathcal{N}^0 + \mathcal{N}^1 + \mathcal{N}_D^1 + \mathcal{N}_N^0) \times (\mathcal{N}^0 + \mathcal{N}^1 + \mathcal{N}_D^1 + \mathcal{N}_N^0).$$

The assembly and the solving of the linear system of equations (6.6) is implemented in the MATLAB function **solveMixedGliding.m**. The input parameters are:

- **coordinates** $\in \mathbb{R}^{N_c \times 2}$: See input parameters **buildV_ex**.
- **dirichlet** $\in \mathbb{R}^{N_{el,D} \times 2}$, **neumann** $\in \mathbb{R}^{N_{el,N} \times 2}$, **gliding** $\in \mathbb{R}^{N_{el,G} \times 2}$: Matrix that describes the boundary elements on the Dirichlet, the Neumann and the gliding boundaries; each row contains the indices of the edges of one boundary element. ($N_{el,D}$, $N_{el,N}$ and $N_{el,G}$ denote the number of elements on Γ_D, Γ_N and Γ_G.)
- **u**, **g**: Function handles of the Dirichlet and the Neumann data.
- $\mathbf{m}, \mathbf{n} \in \mathbb{R}^{N_{el,G} \times 2}$: The directions in which the displacement and the traction are fixed for every gliding boundary.

In the following we describe the assembly of the matrices $\mathbf{M_1}$ and $\mathbf{M_2}$ more closely. Although the matrices $\mathbf{M_1}$ and $\mathbf{M_2}$ are sparse, we abdicate the sparse format and initialize both matrices as full matrices, since the Galerkin matrices $\mathbf{V}, \mathbf{K}$ and $\mathbf{W}$ and thus the complete matrix in the system of linear equations are in general densely populated.
In order to assemble the lower part of the matrices $\mathbf{M_1}$ and $\mathbf{M_2}$, we first extend the matrices $\mathbf{m}$ and $\mathbf{n}$ to the size $\mathcal{N}_G^0 \times 2$ and $\mathcal{N}_G^1 \times 2$ and save the result in the variables **ctmp0** and **ctmp1**. The corresponding excerpt of the code of **solveMixedGliding** is presented in Listing 6.5. Note that the complete code can be found in Appendix C.

Listing 6.5: Excerpt of solveMixedGliding.m.

```
1 freenodesG=unique(gliding);
2 ctmp0 = []; ctmp1 = [];
3 for k=1:length(freenodesG)
4   [idx,~]=find(gliding == freenodesG(k));
5   ctmp1 = [ctmp1; n(idx(1),:)];
6 end
7 for k=1:size(gliding,1)
8   % dof on k-th gliding boundary with respect to S1
9   idx1 = [cp1(nED+nEN+k):cp1(nED+nEN+k+1)-1]+1;
```

```
10    % dof on k-th gliding boundary with respect to S0
11    idx0 = cp0(nED+nEN+k-1)+1:cp0(nED+nEN+k);
12      % extend m and n
13    ctmp0 = [ctmp0;repmat(m(k,:),length(idx0),1)];
14    ctmp1 = [ctmp1;repmat(n(k,:),length(idx1),1)];
15  end
```

Extending m is implemented with a loop over all elements on Γ_G. For extending n we need both a loop over all nodes and a loop over all elements on Γ_G, since in $S^1(\mathcal{T}_h, p_1)$ there are both nodal basis functions (hat functions) and basis functions of higher polynomial degree that live on one element. After having extracted the indices of all nodes on the gliding boundary (line 1) we loop over all nodes on Γ_G and append the row of n that corresponds the the k-th node to **ctmp1** (line 2-6). The loop over all elements of the gliding boundary is implemented in lines 7-15. First, we extract the indices of the degrees of freedom on the k-th boundary element with respect to the basis of $S^0(\mathcal{T}_h, p_0)$ (**idx0**) and the basis of $S^1(\mathcal{T}_h, p_1)$ (**idx1**) in lines 8-11. Using the MATLAB-function **repmat** we extend the rows of m and n that correspond to the k-th boundary element and append it to **ctmp0** and **ctmp1**, respectively.
Finally, the entries for the gliding data in $\mathbf{M_1}$ are created by the MATLAB expression **diag(ctmp0(:,1))** and **diag(ctmp0(:,2))**, the entries in $\mathbf{M_2}$ by **ctmp1(n(:,1))** and **diag(ctmp1(:,2))** and inscribed into the matrices.

Chapter 7
Numerical Results

In this chapter we present the numerical results. One focus is on the verification of the convergences rates and the efficiency and reliability of the error estimators for several refinement strategies. As we already stated in the introduction the results have to be accurate close to machine precision. However, in the implementation we calculate the squared error estimators and the squared energy error and extract the square root afterwards, where we loose half the precision. Hence, we want to reduce the error and the error estimators by 10^{-7}-10^{-8}.
In the first section we illustrate the results for Symm's integral equation and the Dirichlet problem, and in the second section the results for the hypersingular integral equation and the Neumann problem are presented. This chapter closes by showing the results for the mixed problem with and without gliding conditions.

7.1 The Dirichlet Problem

Slit Example

As a first example, we calculate

$$V\phi = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

on the slit, i.e. $\Gamma := [-1, 1]$. For all computations on the slit we choose $\lambda = 600$ and $\mu = 300$. Since there is no analytical solution for this problem, we compute the energy norm of the exact solution $|\!|\!|\phi|\!|\!|$ by extrapolation in order to compute the error in the energy norm by

$$|\!|\!|\phi - \phi_h^p|\!|\!|^2 \overset{Gal.-Orth.}{=} |\!|\!|\phi|\!|\!|^2 - |\!|\!|\phi_h^p|\!|\!|^2.$$

In the following we verify the convergence rate that is given in (3.28). For a constant right-hand side f it can be shown that $\phi \in H^{-\varepsilon}(\Gamma)$, $\varepsilon > 0$, and thus we expect a convergence rate of $\mathcal{O}\big(\,[\mathcal{N}^0]^{-\frac{1}{2}}\,\big)$ for a uniform h-method and $\mathcal{O}(\,[\mathcal{N}^0]^{-1}\,)$ for a uniform p-method.
In Figure 7.1 the numerical results for uniform h- and p-refinements are illustrated. The plots show the error and the error estimators plotted against the degrees of freedom with a double logarithmic scaling. Note that for all $p-p^*$ estimators $p^* = 2p+1$ is used.

We see that all lines run parallel with the slope of $-\frac{1}{2}$ for the uniform h-method and -1 for the uniform p-method, i.e. the convergence rates coincide with the expected convergence rates. Note that it can be shown numerically that increasing the polynomial degree for the uniform h-method or pre-refining the mesh for the uniform

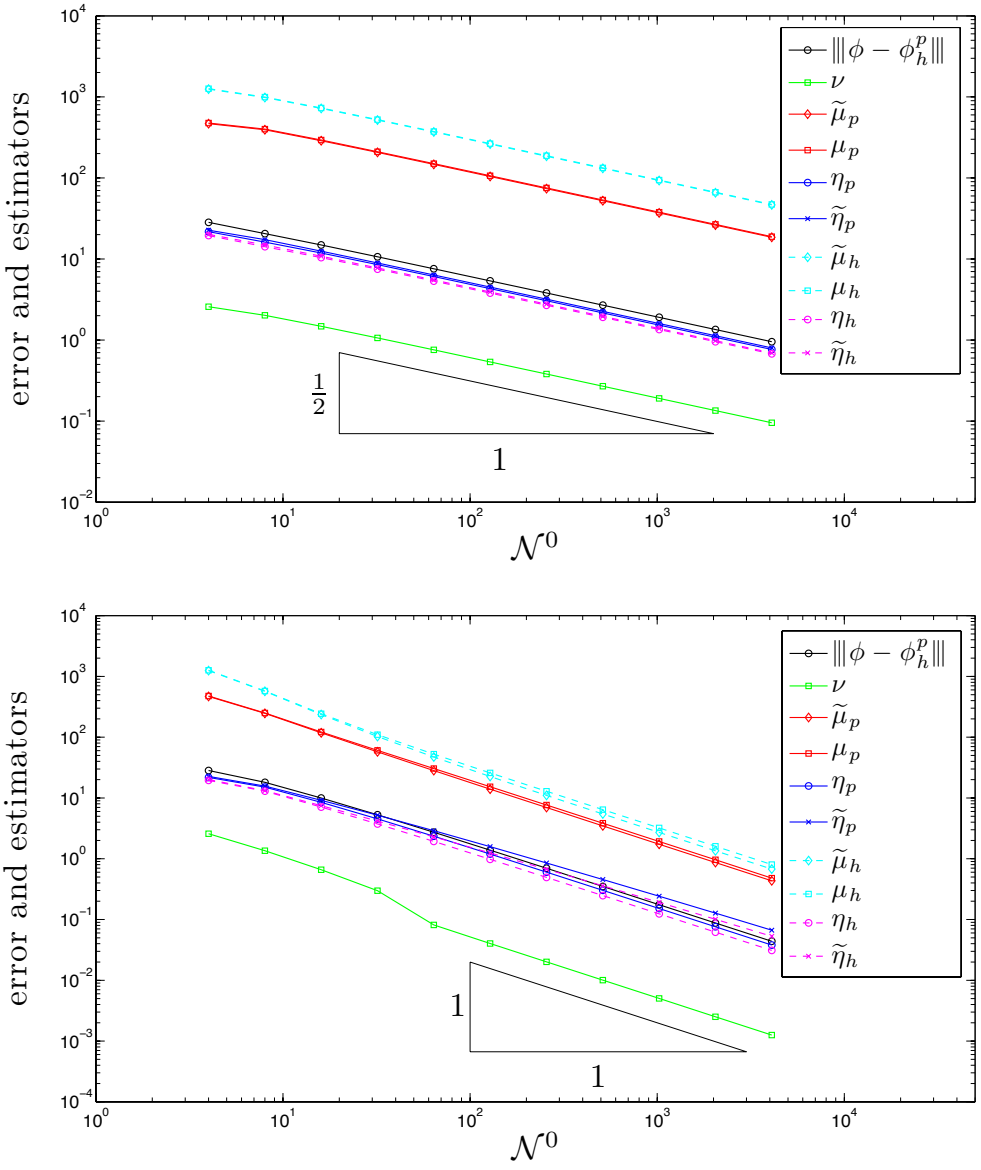

Fig. 7.1: Error and error estimators for a uniform h-refinement with $p = 0$ (top) and a uniform p-refinement with two elements (bottom) for $V\phi = (1, 1)^T$ on the slit.

p-method does not yield a better convergence rate, but we obtain a better constant in (3.28).
Figure 7.2 shows the error and the error estimators plotted against the degrees of freedom using an adaptive h-refinement with $p = 0$ in the first plot and $p = 4$ in the second plot. Again, the plots are scaled double logarithmically and $p^* = 2p + 1$ is used for the $p - p^*$ estimators. Moreover, we mark the elements in the adaptive algorithms with the Dörfler criterion ($\theta = 0.55$) using the estimator $\widetilde{\mu}_h$.

We see that we get a convergence rate of $\mathcal{O}\big([\mathcal{N}^0]^{-\frac{3}{2}}\big)$ for $p = 0$ and $\mathcal{O}\big([\mathcal{N}^0]^{-\frac{11}{2}}\big)$ for $p = 4$, which corresponds to the optimal convergence rate of $\mathcal{O}\big([\mathcal{N}^0]^{-\left(p+\frac{3}{2}\right)}\big)$

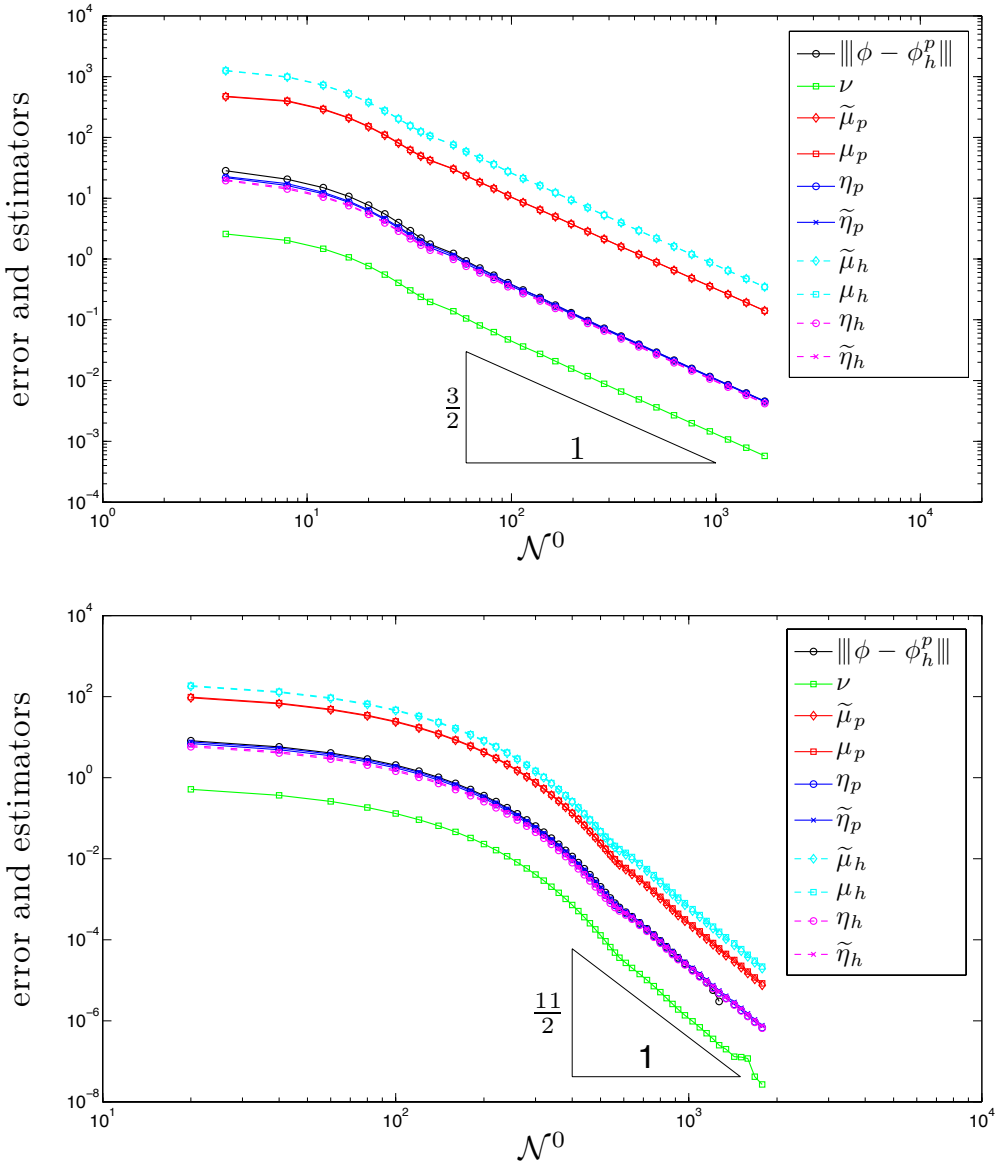

Fig. 7.2: Error and error estimators for an adaptive h-refinement with $p = 0$ (top) and $p = 4$ (bottom) for $V\phi = (1, 1)^T$ on the slit.

that can be reached according to (3.28). Thus, using an adaptive h-refinement the regularity of the solution does not spoil the convergence rate. Note that for $p = 4$ the slope of the estimator ν varies from $-\frac{11}{2}$ in the last three steps. However, we already reduced the estimator by 7 digits and cannot assume that the results are accurate anymore.

In Figure 7.3 we see the error and the error estimators plotted against the square root of the degrees of freedom for an adaptive hp-refinement in the first plot and a geometric hp-refinement in the second plot. Here, only the y-axis is scaled logarithmically. We use the error estimator $\widetilde{\mu}_h$ and the Dörfler criterion for both marking the elements for mesh refinement ($\theta_h = 0.55$) and for marking the elements for increas-

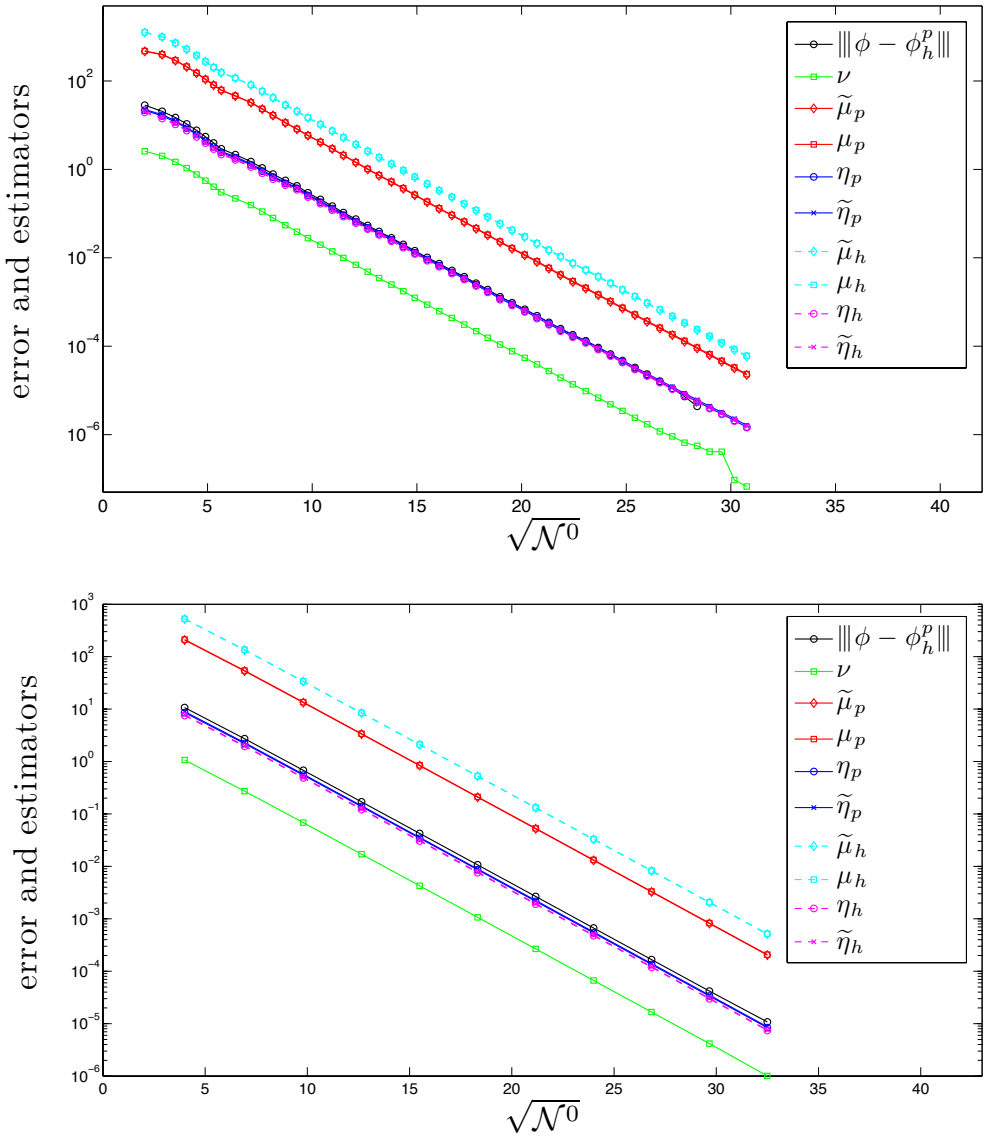

Fig. 7.3: Error and error estimators for an adaptive *hp*-refinement (top) and a geometric *hp*-refinement (bottom) for $V\phi = (1,1)^T$ on the slit.

ing the polynomial degree ($\theta_p = 0.8$). According to [17] we expect an exponential convergence rate, to be more precisely, there holds

$$|||\phi - \phi_h^p||| \lesssim e^{-\beta\sqrt{\mathcal{N}}}, \quad \beta > 0. \tag{7.1}$$

Thus, we expect a straight line with slope $-\beta \log(e)$ in the plots.
Figure 7.3 shows that we obtain an exponential convergence rate for both the adaptive and the geometric *hp*-refinement. However, we see that we obtain a better constant for the adaptive algorithm than for the geometric algorithm, i.e. there hods $\beta_{adap.} > \beta_{geom.}$. This can be explained by the fact that we use a posteriori knowledge for the adaptive algorithm, whereas we prescribe the refinement a priori in the geometric algorithm. Additionally, we see that the slope of the residual based estimator in the first plots varies in the last steps. Again, we already reduced the

estimator to machine precision and cannot assume the results to be accurate anymore.
All figures for the slit example show that the error estimators run parallel to the energy error, which proofs the efficiency and the reliability of these estimators numerically for the slit example.

Rotated L-shape Example

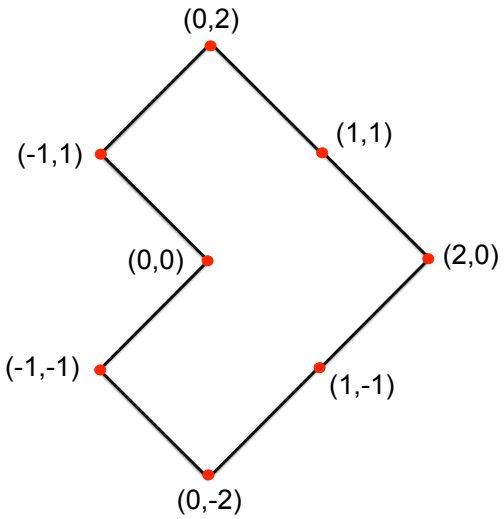

Fig. 7.4: Rotated L-shape domain.

The second example that is treated in this section is a common benchmark example on the rotated L-shape that is also investigated in [1], [4] and [18]. The rotated L-shape is illustrated in Figure 7.4 and the exact solution $u := (u_r, u_\varphi)^T$ is given in polar coordinates by

$$u_r(r,\varphi) = \frac{r^\alpha}{2\mu}\left\{-(\alpha+1)\,\cos\left[(\alpha+1)\,\varphi\right] + \left[C_2-(\alpha+1)\right]C_1\,\cos\left[(\alpha-1)\,\varphi\right]\right\}$$
$$u_\varphi(r,\varphi) = \frac{r^\alpha}{2\mu}\left\{(\alpha+1)\,\sin\left[(\alpha+1)\,\varphi\right] + \left[C_2+\alpha-1\right]C_1\,\sin\left[(\alpha-1)\,\varphi\right]\right\},$$

where the parameters are defined by $C_1 = -\frac{\cos[(\alpha+1)\omega]}{\cos[(\alpha-1)\omega]}$, $C_2 = \frac{2(\lambda+2\mu)}{\lambda+\mu}$ with $\omega = \frac{3}{4}\,\pi$. Moreover, $\alpha \approx 0.54448373$ is the positive solution of $\alpha\,\sin(2\omega) + \sin(2\omega\,\alpha) = 0$. In order to compare the numerical results with [1] we choose the parameters of material of bronze ($E = 100000$ and $\nu = 0.3$), which implies that $\lambda \approx 5.7692 \cdot 10^4$ and $\mu = 3.8462 \cdot 10^4$.

In Figure 7.5 the h-$h/2$ estimator μ_h is plotted against the degrees of freedom for a uniform h and a uniform p-refinement, an adaptive h-refinement with $p = 4$ and an adaptive hp-refinement. For both adaptive algorithms the Dörfler criterion with

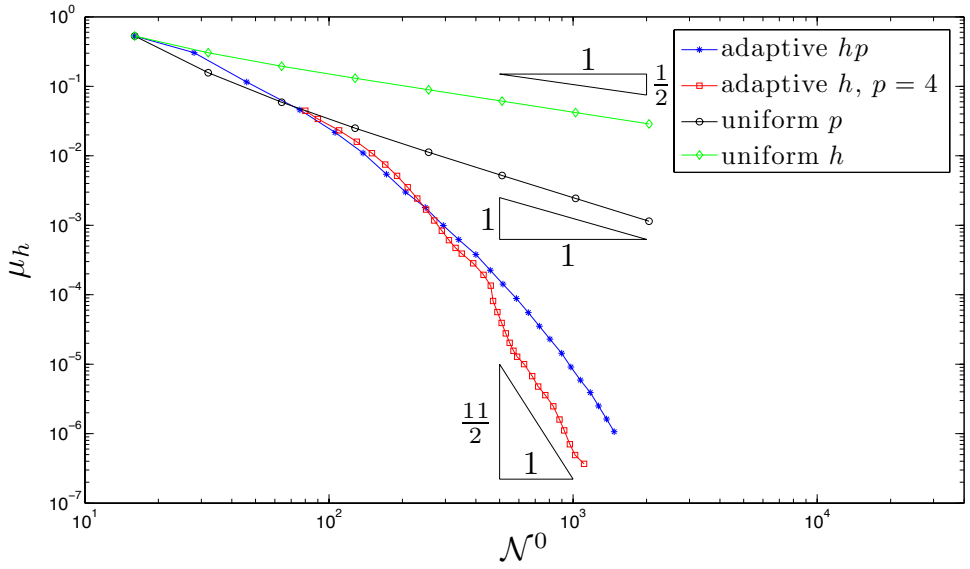

Fig. 7.5: μ_h for a uniform h- and p-refinement, an adaptive h-refinement with $p = 4$ and an adaptive hp-refinement.

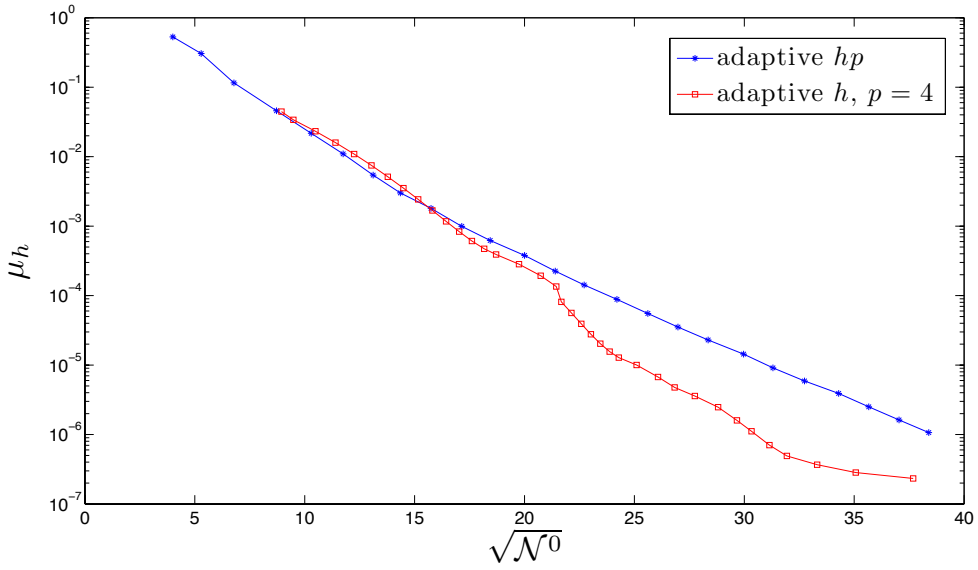

Fig. 7.6: L-shape: μ_h for an adaptive h-refinement with $p = 4$ and an adaptive hp-refinement with semi-logarithmical scaling.

$\theta = 0.55$ for the adaptive h-method, and $\theta_h = 0.55$ and $\theta_p = 0.8$ for the adaptive hp-method is used. Additionally, we mark the elements using the estimator μ_h. Figure 7.5 shows the following results:

- Even with the adaptive algorithms the error estimator is only reduced by 6 digits. This is due to the fact that for approximating the given Dirichlet data we need a very high polynomial degree p_1, i.e. $p_1 = 20\,(p_0 + 1)$ is chosen in

this example. Thus, we have to store big matrices and need a large number of digits for the computations with the multi-precision libraries. This results in very long computational times.

- The adaptive h-method converges faster than the adaptive hp-method. This can be explained by the fact that the parameters for marking the elements are better chosen for the adaptive h-method than for the adaptive hp-method. However, Figure 7.6 shows, that we do not obtain an exponential convergence rate for the adaptive h-method. The adaptive hp-method converges exponentially, although the convergence is slow.

- For all refinement strategies we obtain the same convergence rates as for the slit example which indicates numerically that $\phi \in H^{-\varepsilon}(\Gamma)$.

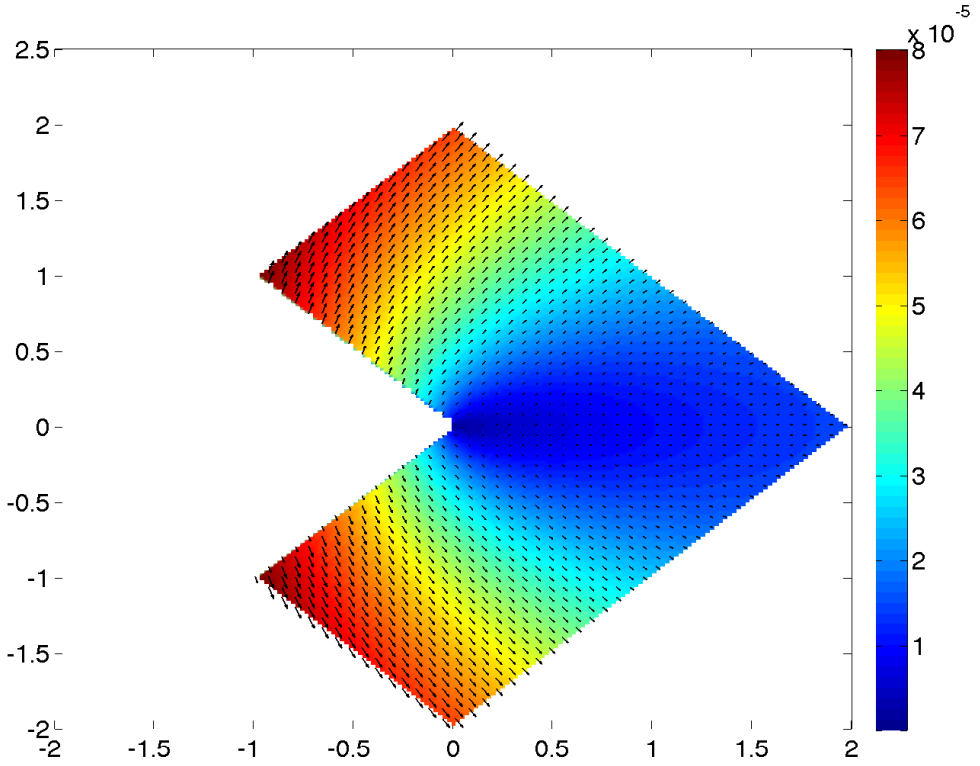

Fig. 7.7: L-shape: Absolute value of displacement with displacement field on the domain Ω.

Figure 7.7 shows the exact solution of the Dirichlet problem plotted in the domain Ω, where the vector field denotes the displacement on a grid and the colors denote the absolute value of the displacement in every point of the grid.

Figure 7.8 shows the boundary Γ (red dashed) and the deformed boundary $\widetilde{\Gamma}$, i.e. $\widetilde{\Gamma} \ni \widetilde{x} := x + u(x),\ \forall x \in \Gamma$. In order to visualize the displacement, it is scaled with the factor 3000 as it is also done in [18] and [1]. Both figures coincide with the

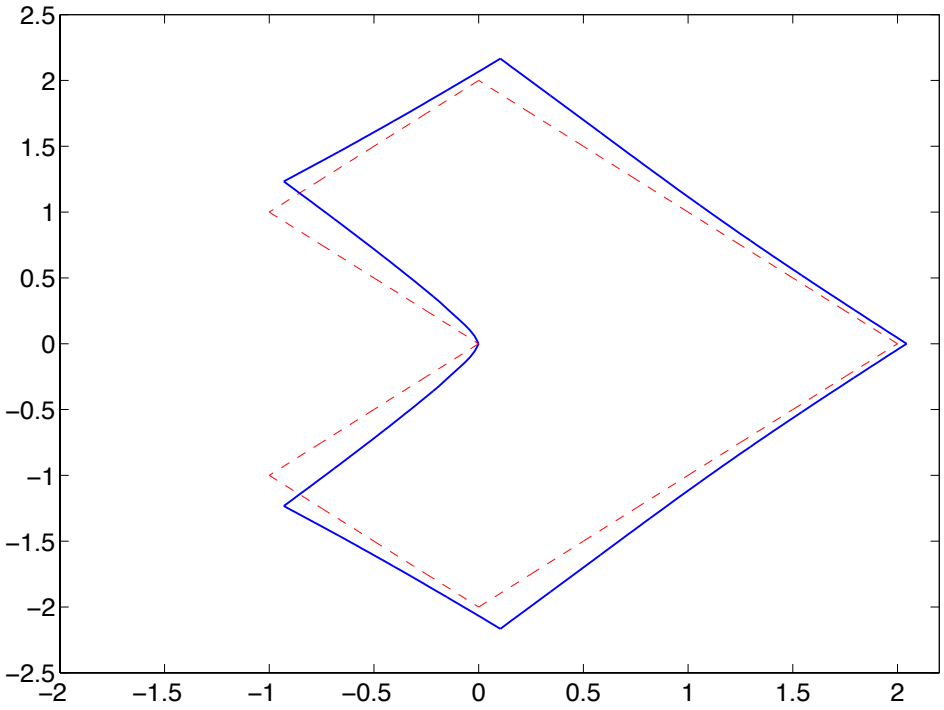

Fig. 7.8: L-shape: Undeformed boundary Γ (red dashed) and deformed boundary $\widetilde{\Gamma}$ (blue) scaled with factor 3000.

figures that are illustrated in [18] and [1], although no exact values are presented, there.

7.2 The Neumann Problem

Slit Example

As a first Neumann example we solve

$$Wu = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

on the slit. As we already did for solving Symm's integral equation, we prescribe the Lamé parameters $\lambda = 600$ and $\mu = 300$ and determine the energy of the exact solution $|||u|||$ by extrapolation since there is no analytical solution. Again, we verify the convergence rates that we stated in Section 3.3 (See (3.39)). For a constant right-hand side it is known that $u \in H^{1-\varepsilon}(\Gamma)$, $\varepsilon > 0$, and therefore (3.39) implies a convergence rate of $\mathcal{O}\big([\mathcal{N}^1]^{-\frac{1}{2}} \big)$ for a uniform h-refinement and a convergence rate of $\mathcal{O}([\mathcal{N}^1]^{-1})$ for the uniform p-refinement. Figure 7.9 shows the errors and the estimator plotted against the degrees of freedom in a double logarithmic scaling. We see that the convergence rate of the uniform h-method is $\mathcal{O}([\mathcal{N}^1]^{-\frac{1}{2}})$ and the

convergence rate of uniform p-method is $\mathcal{O}(\,[\mathcal{N}^1]^{-1}\,)$, which is what we expected.

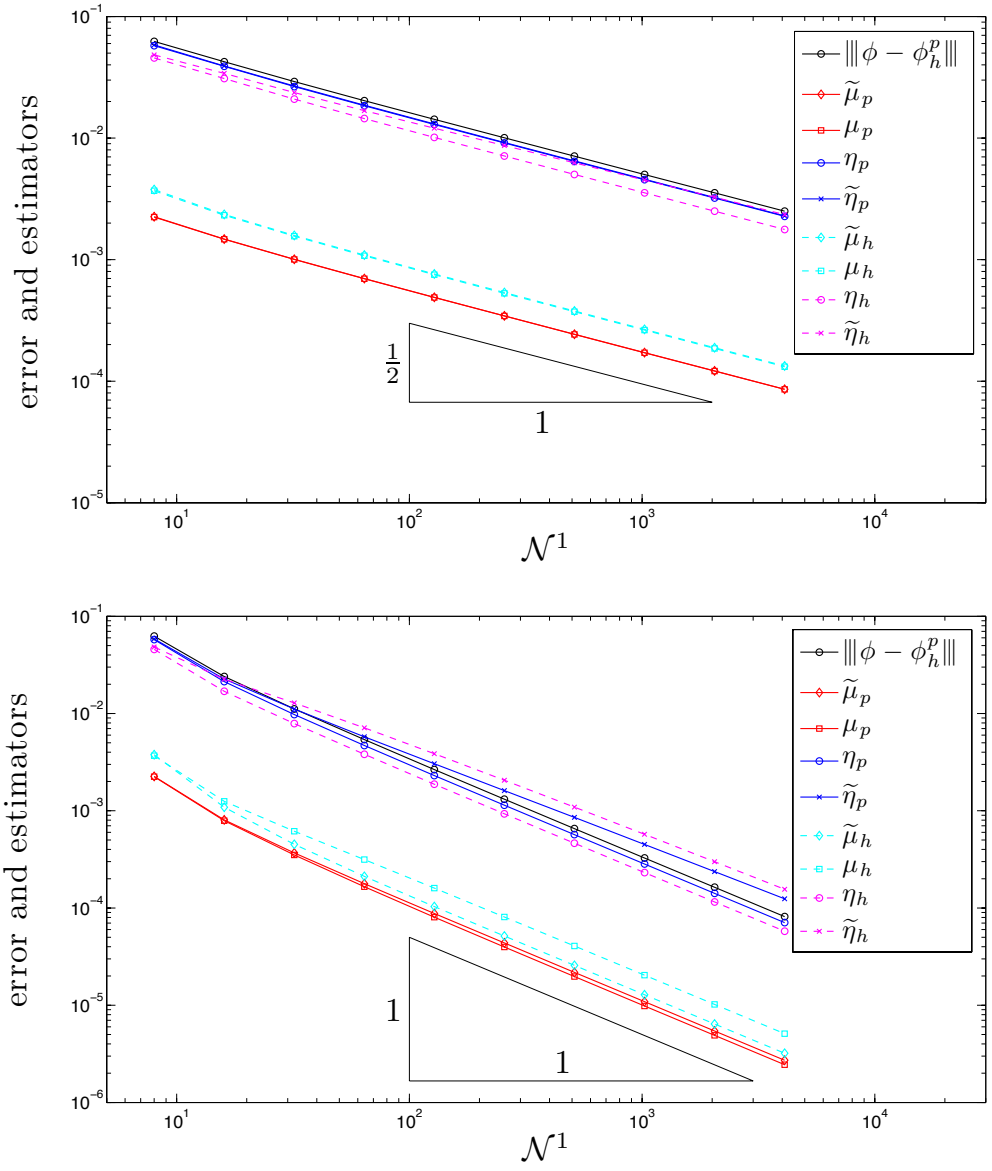

Fig. 7.9: Error and error estimators for a uniform h-refinement with $p = 1$ (top) and a uniform p-refinement with two elements (bottom) for $Wu = (1, 1)^T$ on the slit.

Figure 7.10 shows the error and the error estimators plotted against the degrees of freedom for an adaptive h-method, where we used the Dörfler criterion with $\theta = 0.55$ and μ_h for refining the mesh adaptively. Referring to (3.39), the optimal convergence rate that we can obtain by refining the mesh is $\mathcal{O}\big([\mathcal{N}^1]^{-(p+\frac{1}{2})}\big)$. Again, the numerical results coincide with the theory, since we get a numerical convergence rate of $\mathcal{O}\big([\mathcal{N}^1]^{-\frac{3}{2}}\big)$ for $p = 1$ and $\mathcal{O}\big([\mathcal{N}^1]^{-\frac{13}{2}}\big)$ for $p = 6$.

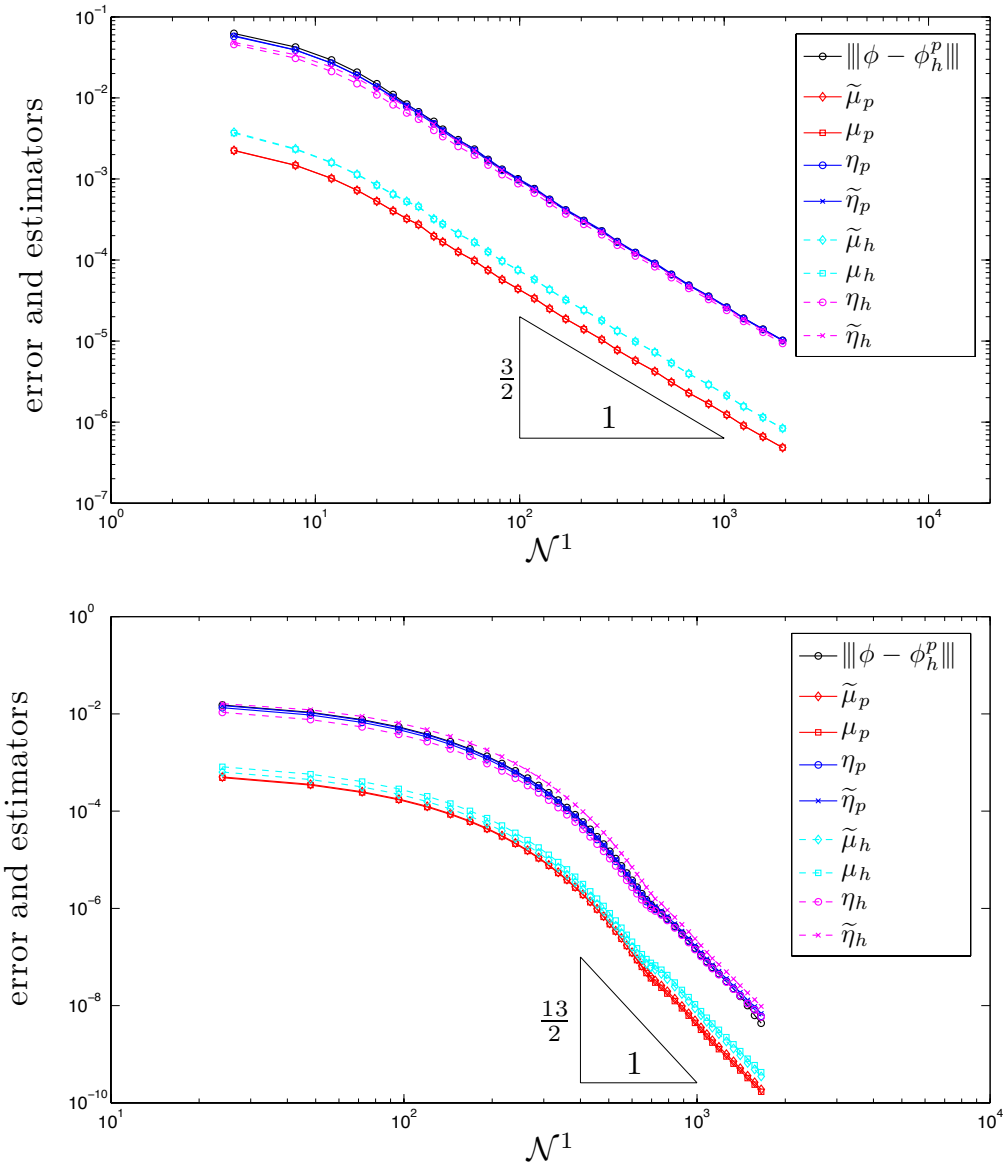

Fig. 7.10: Error and error estimators for an adaptive h-refinement with $p = 1$ (top) and $p = 6$ (bottom) for $Wu = (1, 1)^T$ on the slit.

For the geometric hp-refinement it is proven in [17] that there holds an error estimation that is similar to (7.1) for the hypersinguar equation, i.e. we can expect an exponential convergence rate. The adaptive and geometric hp-methods are illustrated in Figure 7.11 semi-logarithmically plotted against the square root of the

degrees of freedom. We see that both methods converge exponentially and the the constant in the estimation is better for the adaptive *hp*-refinement. Thus, the results are similar to the results for Symm's integral equation.

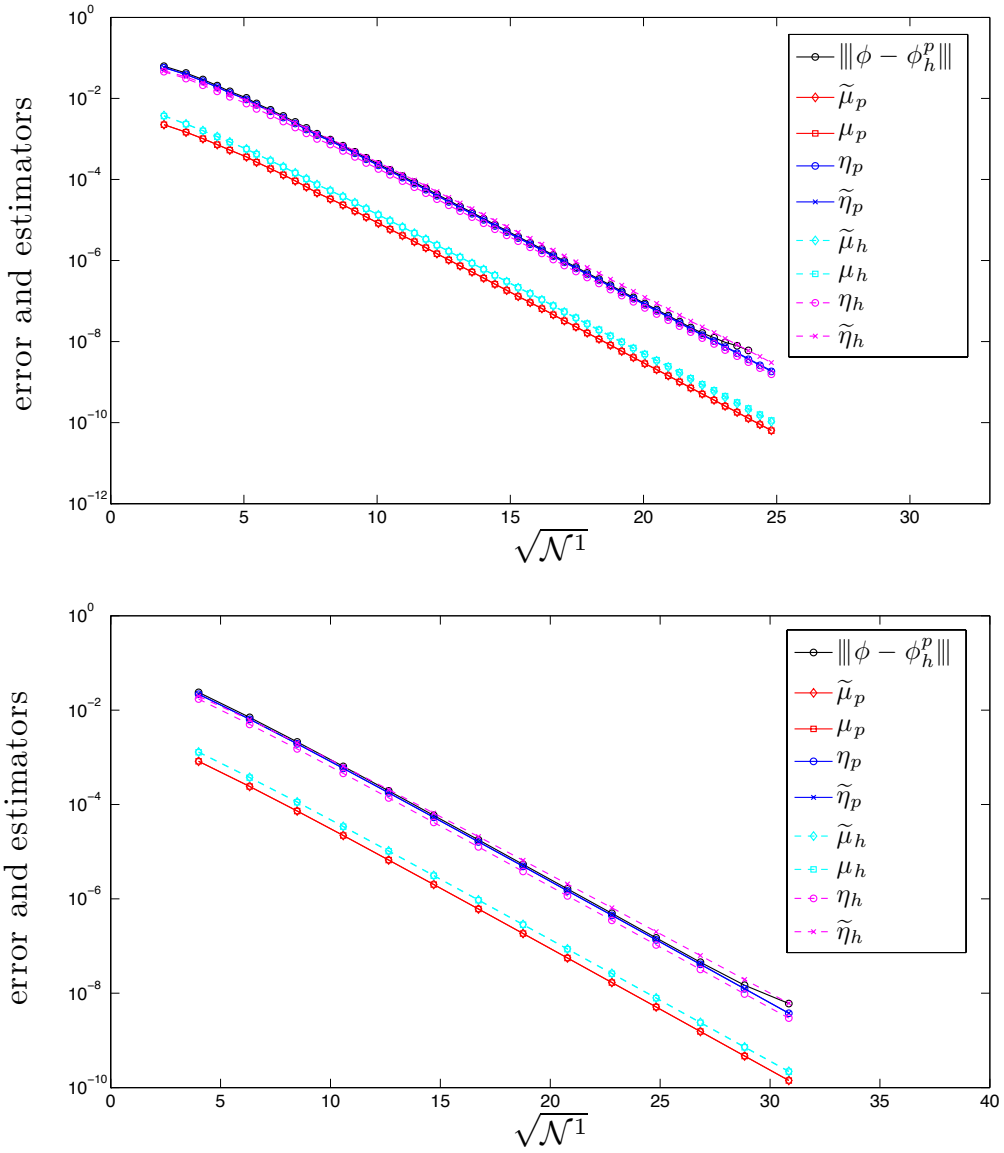

Fig. 7.11: Error and error estimators for an adaptive *hp*-refinement (top) and a geometric *hp*-refinement (bottom) for $Wu = (1,1)^T$ on the slit.

Square Membrane Example

As a second Neumann example we illustrate the square membrane that is given by $\Omega := [-3,3]^2$ and that is stretched on the upper and lower side, i.e. a traction $g = n$ is applied, where n denotes the outer normal vector. The Lamé coefficients are given by $\lambda = 600$ and $\mu = 300$. The results are illustrated in Figure 7.12 and 7.13. Note that we scaled the displacement with the factor 100 for the visualization in Figure 7.13.

Both Figures show that the membrane contracts in x_1-direction and expands in x_2-direction and the maximal displacement of $3.6 \cdot 10^{-3}$ is located at the edges. Moreover, we see that the displacement is symmetric to the center, i.e. this example can also be calculated by partitioning the domain in four parts using gliding condition.

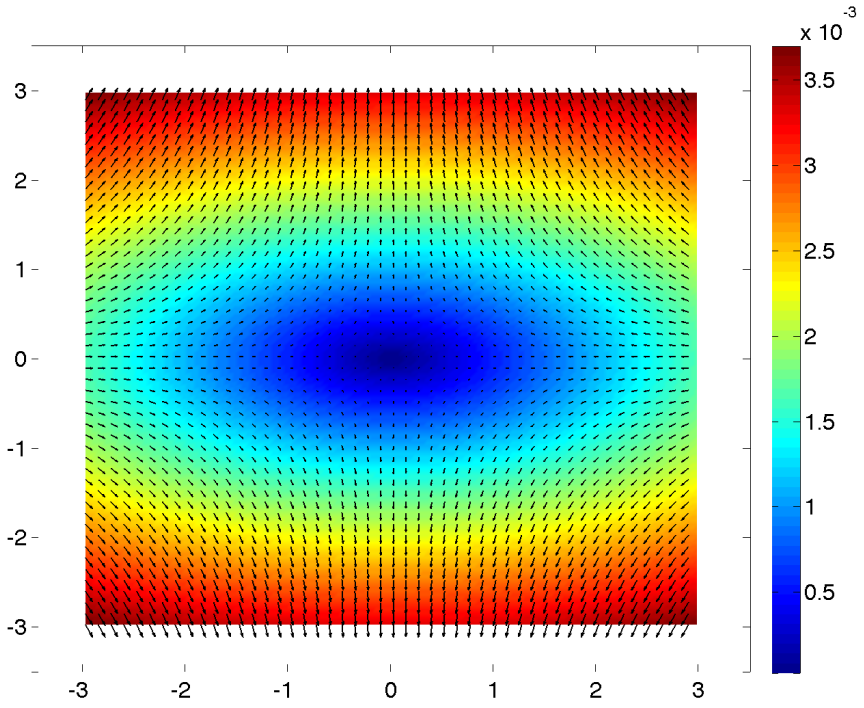

Fig. 7.12: Square Membrane: Absolute value of displacement with displacement field on the domain Ω.

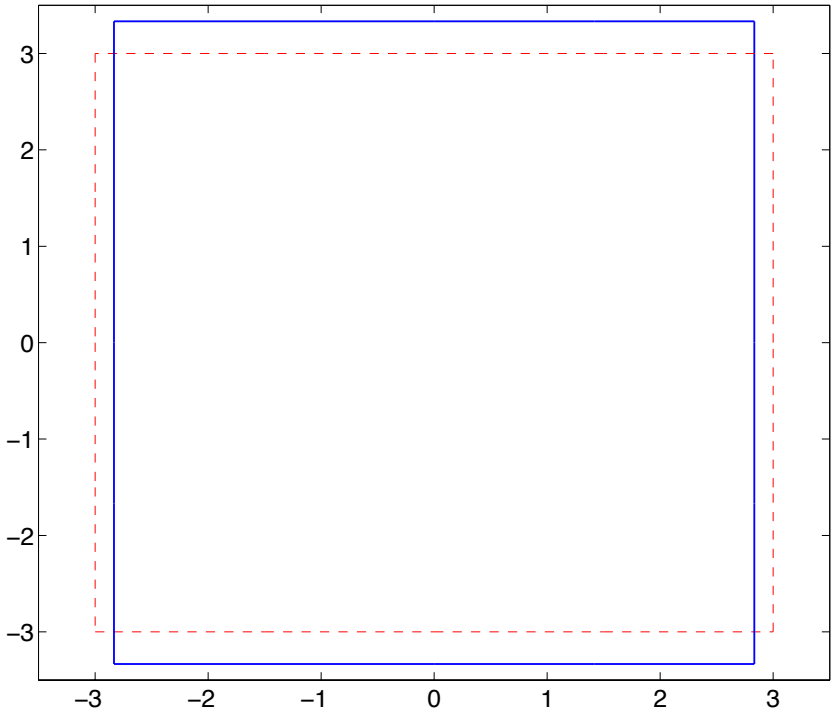

Fig. 7.13: Square Membrane: Undeformed boundary Γ (red dashed) and deformed boundary $\widetilde{\Gamma}$ (blue) scaled with factor 100.

7.3 The Mixed Traction and Displacement Problem

Cook's Membrane Example

As a first example we choose a common example of linear elasticity that is also treated in [1] and [18] and that corresponds to the two-dimensional model of the bar that we introduced in Section 2.2, namely the Cook's membrane example. The membrane that we investigate in the following is given in Figure 7.14.

The membrane is fixed on the left-hand side, i.e. we have a Dirichlet boundary with $u_D(x) = (0,0)^T$ and a shearing load is applied on the right-hand side, i.e. we have a Neumann boundary with $g(x) = (0,1)^T$. Moreover, the top and the bottom boundary are also Neumann boundaries with $g(x) = (0,0)^T$ and volume forces vanish. Thus, we have a mixed problem. According to [1] we choose the parameter of material of Plexiglass, i.e. $E = 2900$ and $\nu = 0.4$, which implies that $\lambda = 4.1429 \cdot 10^3$ and $\mu = 1.0357 \cdot 10^3$.
The error estimator μ_h is illustrated in Figure 7.15 for different refinement strategies.

We see that we obtain a exponential convergence rate for the adaptive *hp*-method and algebraic convergence rates with the slope of $-\frac{1}{2}$ for the uniform *h*-method,

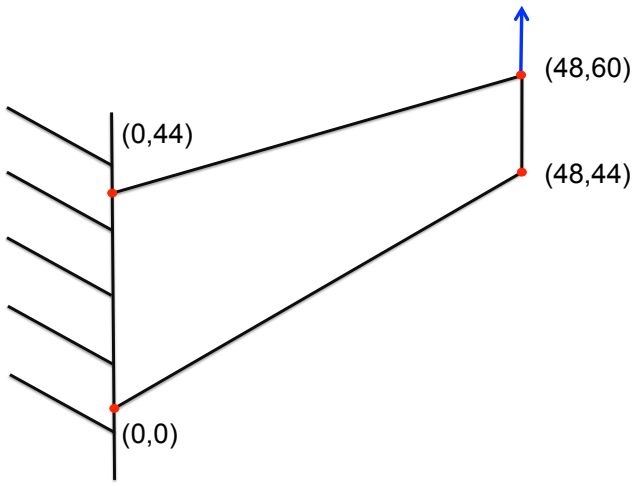

Fig. 7.14: Cook's membrane with $\Omega = \text{conv}\{(0,0),\ (48,44),\ (48,60),\ (0,44)\}$ and a surface force applied on the right-hand side.

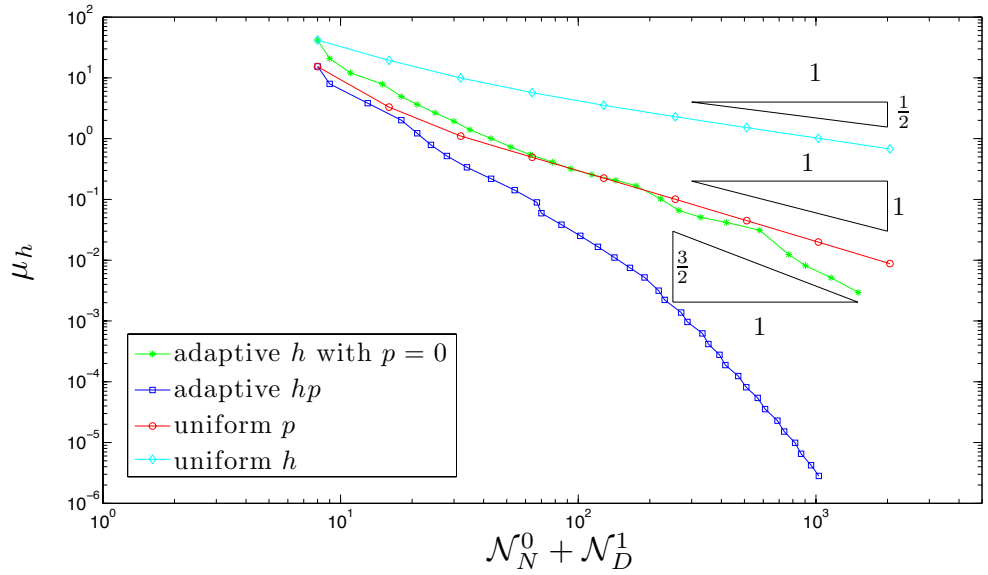

Fig. 7.15: Cook's membrane: μ_h for a uniform h- and p-refinement, an adaptive h-refinement with $p = 0$ and an adaptive hp-refinement.

-1 for the uniform p-method and $-\frac{3}{2}$ for the adaptive h-method with polynomial degree $p = 0$. Using the adaptive hp-algorithm we reduce the estimator by 7 digits and thus we are accurate close to machine precision.
Figure 7.16 and 7.17 show the displaced boundary and the displacement field, respectively. Note that we used a scaling factor of 50 for illustrating the displaced boundary. We see that the membrane bends up, whereas the left-hand side is fixed. The displacement proceeds continuously from the left to the right side, where the

maximal displacement of 0.14 is located on the upper right edge of the membrane. The second Figure shows that the upper side of the membrane is bulged and the lower side is stretched. Thus, we see the typical displacement of a bar that is loaded as described above. Moreover, in [1] and [18] similar result are illustrated, although no exact values for a comparison are given.

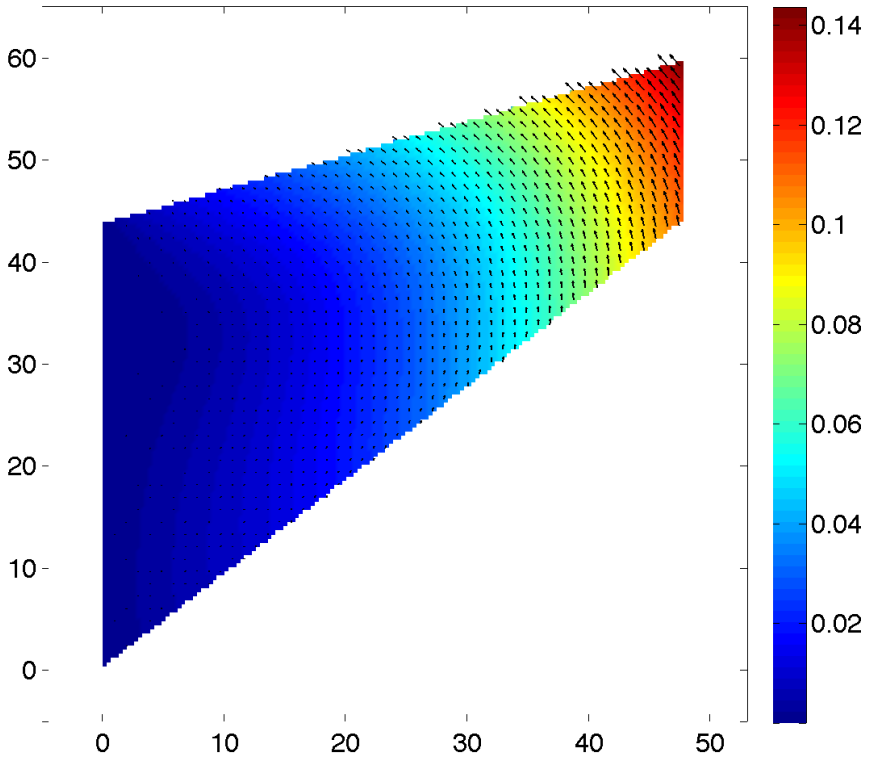

Fig. 7.16: Cook's Membrane: Absolute value of displacement with displacement field on the domain Ω.

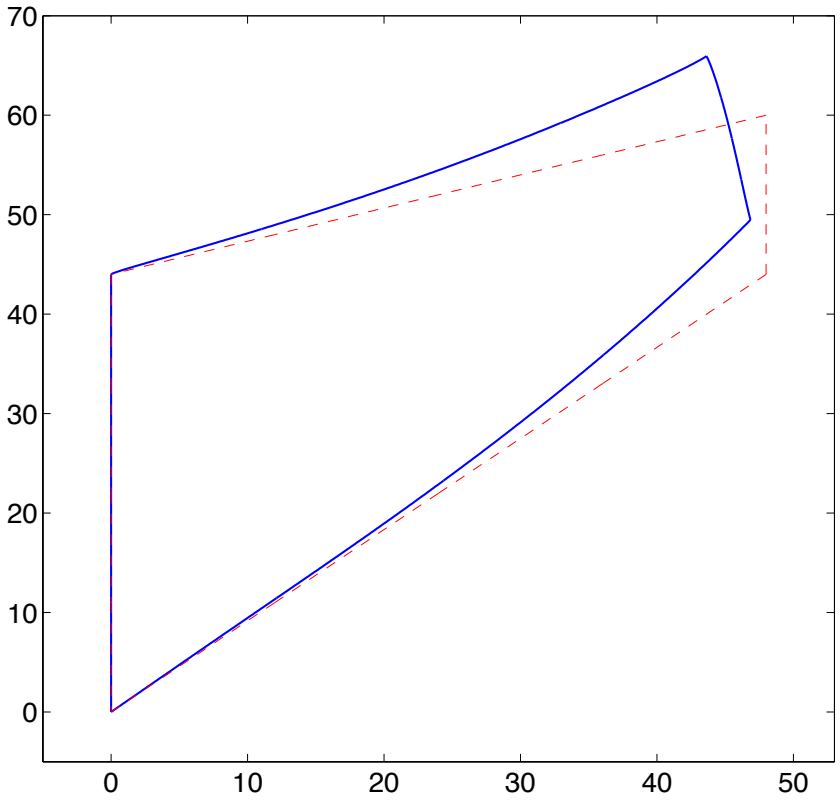

Fig. 7.17: Cook's Membrane: Undeformed boundary Γ (red dashed) and deformed boundary $\widetilde{\Gamma}$ (blue) scaled with factor 50.

Membrane with a Hole

As a last example we investigate the membrane with a hole that we introduce in Figure 6.6. We solve the problem according to the right figure in 6.6 using gliding conditions. Figure 7.18 shows the geometry with the initial mesh that is used for the calculations, where the gliding boundary is denoted by the circles.

Again, we choose the parameters of material of Plexiglass. The directions in which the displacement and the traction is fixed on the gliding boundary are given by $n = (1,0)^T$ and $m = (0,1)^T$ for the left boundary and by $n = (0,1)^T$ and $m = (1,0)^T$ for the lower boundary. On the upper boundary we prescribe a traction $g(x) := (0,1)^T$, whereas we set $g(x) = (0,0)^T$ on all other Neumann elements.

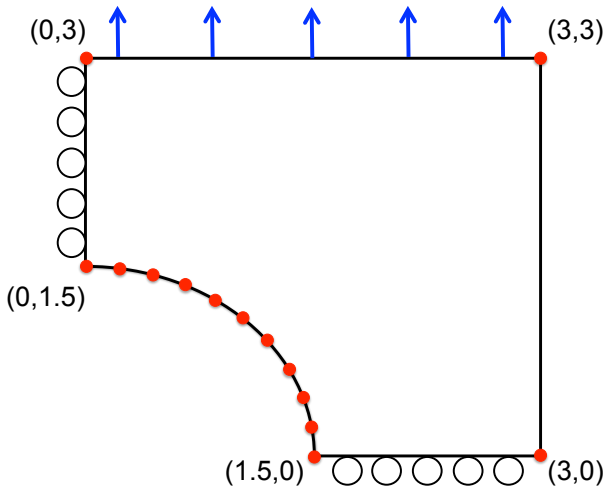

Fig. 7.18: Right upper part of the membrane with a hole with initial mesh (red points) and a traction on the upper side (blue arrows).

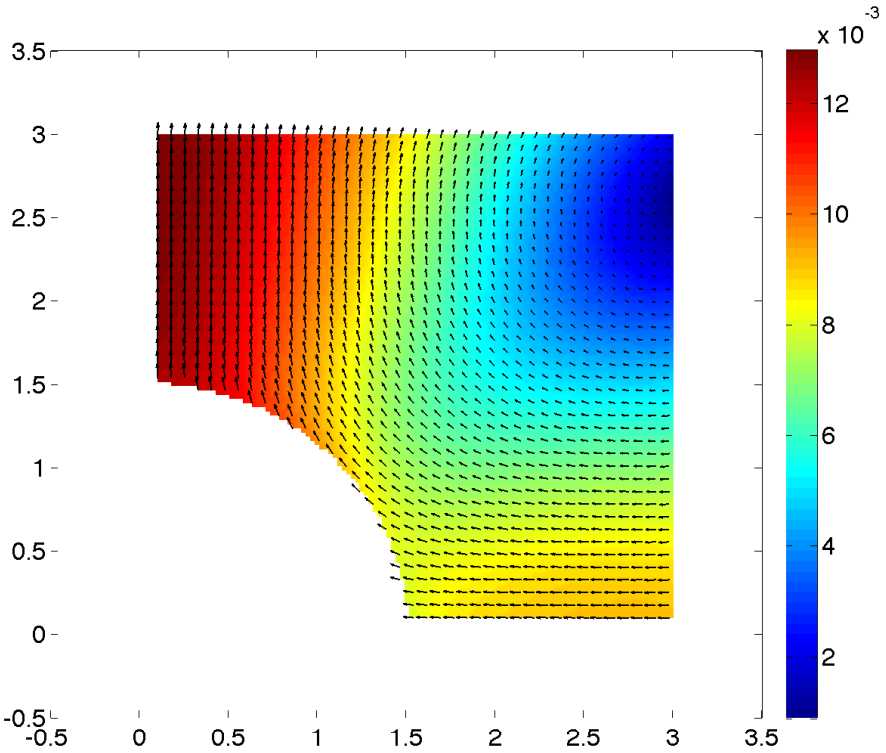

Fig. 7.19: Membrane with a hole: Absolute value of displacement with displacement field on the domain Ω.

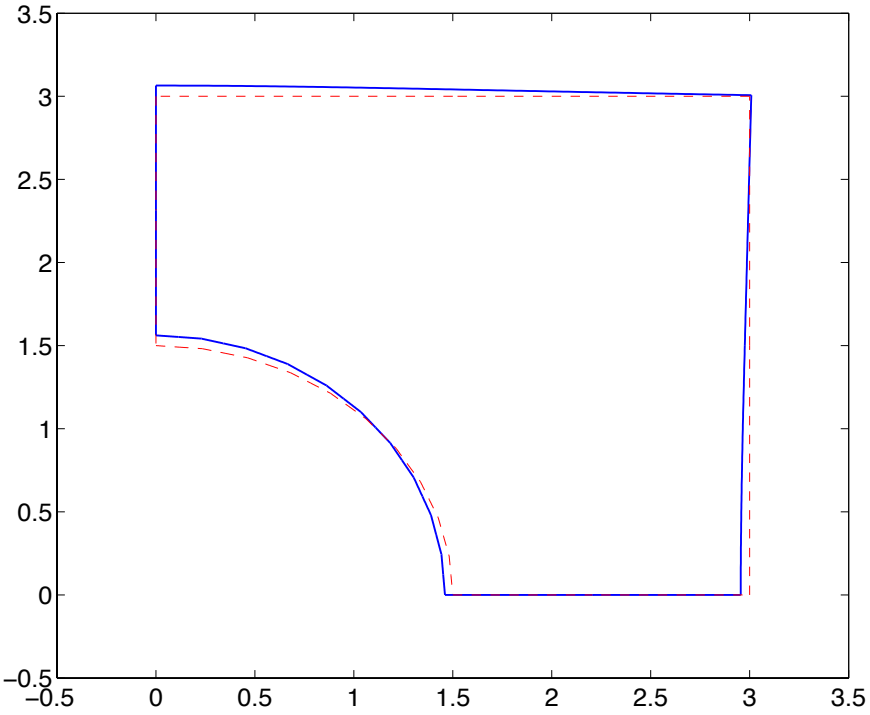

Fig. 7.20: Membrane with a hole: Undeformed boundary Γ (red dashed) and deformed boundary $\widetilde{\Gamma}$ (blue) scaled with factor 10 .

The displacement of the membrane is presented in the Figures 7.19 and 7.20. We see that the gliding boundaries only move orthogonal to the directions n, i.e. the displacement in the directions n vanishes. Moreover, we see that the maximal displacement is located on the left boundary, which corresponds to the vertical axis in the middle of the whole membrane. Since on this axis there is the least material we can expect the maximal displacement in x_2-direction. Thus, the results that we obtain coincide with the expected results, and - as far as it is comparable - the results coincide with the results that are stated in [1].

Chapter 8
Conclusion

In this thesis, we investigated how to derive an efficient and stable implementation of the *hp*-BEM for the Navier-Lamé equation. In particular, a wide range of practical problems in two-dimensional linear elasticity should be solved. The focus of the implementations was on two main aspects. First, the routines that were developed for solving the Navier-Lamé equation should be integrated into the epsBEM framework. Second, the routines should be fast and stable, so that the error can be reduced close to machine precision.
With the theory of the associated Legendre functions we derived recurrence relations for some special integrals that are related to the associated Legendre functions and that can be evaluated efficiently up to high polynomial degrees. Choosing the Legendre polynomials and the Lobatto shape functions as ansatz-functions, we used these recurrence relations to derive analytical formulas for assembling the Galerkin matrices of the boundary integral operators very efficiently.
All implementations are completely integrated into epsBEM framework, so that the user only has to adjust the main routines for solving the Laplace and Navier-Lamé equations. The core of the implementation are two C-libraries for evaluating the special integrals and assembling the Galerkin matrices. Using the multi-precision libraries **mpfr** and **mpc** provided a stable implementation of the core routines even for high polynomial degrees.
For all examples we could reduce the error at least up to a precision of 10^{-14} using the adaptive *hp*-refinement strategies. Additionally, we were able to reduce computational time by parallelizing the assembly of the Galerkin matrices with openMP. However, research can be performed on incorporating data compression techniques that further reduce computational time and that are currently developed for the Laplace equation (see [10]).
Besides the *hp*-BEM we also implemented uniform *h*- and *p*-, and adaptive *h*-methods with arbitrary polynomial degrees. The numerical results showed that we can calculate the solution up to machine precision exploiting the exponential convergence rate of the adaptive *hp*-refinements. Moreover, we could reproduce the theoretical results concerning the convergence rate of the uniform refinements and regain the optimal convergence rate using the adaptive *h*-refinement.
Practical examples in linear elasticity could be solved by incorporating gliding conditions. However, it still remains to develop error estimators for the problems with gliding conditions in order to create adaptive algorithms for these problems. Furthermore, within the scope of this thesis we ignored volume forces. It is another task to implement the Newton potential, so that volume forces such as gravity can be considered.

Appendix A
Calculation of the Integral $O^1_{j,k}$

Lemma A.0.1 *Let $a, b \in \mathbb{C}$, with $a \pm b \notin [-1,1]$, $j \geq 2$ and η_1 as defined in (4.31). Then, there holds*

$$\begin{aligned} O^1_{0,3}(a,b) = {} & a\left[\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] + b\left[\widetilde{Q}_1^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\ & - \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{1}{3}b\right)\left[\widetilde{Q}_0^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^0\left(\frac{-1-a}{b}\right)\right] \\ & - a\left[\widetilde{Q}_1^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^0\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}b\left[\widetilde{Q}_2^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^0\left(\frac{-1-a}{b}\right)\right] + 2, \end{aligned}$$

$$\begin{aligned} O^1_{1,3}(a,b) = {} & a\left[\widetilde{Q}_1^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] + \frac{2}{3}b\left[\widetilde{Q}_2^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_2^{-1}\left(\frac{-1-a}{b}\right)\right] \\ & + \frac{1}{3}b\left[\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\ & - \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{3}{5}b\right)\left[\widetilde{Q}_1^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^0\left(\frac{-1-a}{b}\right)\right] \\ & - \frac{2}{3}a\left[\widetilde{Q}_2^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^0\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}a\left[\widetilde{Q}_0^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^0\left(\frac{-1-a}{b}\right)\right] \\ & - \frac{1}{5}b\left[\widetilde{Q}_3^0\left(\frac{1-a}{b}\right) - \widetilde{Q}_3^0\left(\frac{-1-a}{b}\right)\right] + 2. \end{aligned}$$

and

$$\begin{aligned} O^1_{j,3}(a,b) = {} & a\,\frac{j}{(j+2)(2j+1)}\left\{\widetilde{Q}_{j+1}^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_{j+1}^0\left(\frac{1-a}{b}\right)\right\} \\ & + \left[\frac{a^2-1}{b}\,\frac{1}{j+2} + b\,\frac{j-1}{(j+2)(2j-1)}\right]\left\{\widetilde{Q}_j^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_j^0\left(\frac{1-a}{b}\right)\right\} \\ & + a\,\frac{3j+2}{(j+2)(2j+1)}\left\{\widetilde{Q}_{j-1}^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_{j-1}^0\left(\frac{1-a}{b}\right)\right\} \\ & + b\,\frac{j}{(j+2)(2j-1)}\left\{\widetilde{Q}_{j-2}^0\left(\frac{-1-a}{b}\right) - \widetilde{Q}_{j-2}^0\left(\frac{1-a}{b}\right)\right\}, \end{aligned}$$

where η_2 is defined as in Lemma 4.2.4.

Proof. We start proving the first identity. Using the definition of $N_3(t)$, we have

$$O^1_{0,3}(a,b) = \frac{1}{2}\left(\int_{-1}^1\int_{-1}^1 \frac{t^2}{(a+b\,s-t)^2}\,ds\,dt - \int_{-1}^1\int_{-1}^1 \frac{1}{(a+b\,s-t)^2}\,ds\,dt\right). \quad \text{(A.1)}$$

Therefore, we investigate both integrals separately. There holds

$$\begin{aligned}\int_{-1}^{1} \frac{t^2}{(a+b\,s-t)^2}\,dt &= t + 2\,(a+b\,s)\,\log(-a-b\,s+t)\Big|_{-1}^{1} \\ &\quad + \frac{(a+b\,s)^2}{a+b\,s-t}\Big|_{-1}^{1} \\ &= 2 + 2(a+b\,s)\,[\log(-a-b\,s+1) - \log(-a-b\,s-1)] \\ &\quad + (a+b\,s)^2\left[\frac{1}{a+b\,s-1} - \frac{1}{a+b\,+1}\right].\end{aligned}$$

Thus we obtain

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} \frac{t^2}{(a+b\,s-t)^2}\,ds\,dt \\ &= 4 + 2\int_{-1}^{1} (a+b\,s)\,[\log(-a-b\,s+1) - \log(-a-b\,s-1)] \\ &\qquad + (a+b\,s)^2\left[\frac{1}{a+b\,s-1} - \frac{1}{a+b\,+1}\right] ds.\end{aligned}$$

We proceed as in Lemma 4.2.4 and simplify the logarithm terms by point reflection. Thus, we get

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1} \frac{t^2}{(a+b\,s-t)^2}\,ds\,dt \\ &= 4 + 2\int_{-1}^{1} (a+b\,s)\,\left[\log\left(\eta_1\left(\frac{1-a}{b} - s\right)\right) - \log\left(\eta_1\left(\frac{-1-a}{b} - s\right)\right)\right] \\ &\qquad + \frac{(a+b\,s)^2}{b}\left[\frac{1}{\frac{1-a}{b} - s} - \frac{1}{\frac{-1-a}{b} - s}\right] ds \\ &= 2a\left[\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] + 2b\left[\widetilde{Q}_1^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\ &\quad - \left(\frac{a^2}{b} + \frac{1}{3}\,b\right)\left[\widetilde{Q}_0^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^{0}\left(\frac{-1-a}{b}\right)\right] - 2a\left[\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right] \\ &\quad - \frac{2}{3}\,b\left[\widetilde{Q}_2^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^{0}\left(\frac{-1-a}{b}\right)\right] + 4.\end{aligned}$$

For the second integral in A.1 we get

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{1} \frac{1}{(a+b\,s-t)^2}\,ds\,dt &= \int_{-1}^{1} \frac{1}{(a+b\,s-1)} - \frac{1}{(a+b\,s+1)}\,ds \\ &= \frac{1}{b}\left[\widetilde{Q}_0^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^{0}\left(\frac{-1-a}{b}\right)\right]\end{aligned}$$

Putting the result together, we get the first identity

$$\begin{aligned}O^1_{0,3}(a,b) &= a\left[\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] + b\left[\widetilde{Q}_1^{-1}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\frac{-1-a}{b}\right)\right] \\ &\quad - \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{1}{3}b\right)\left[\widetilde{Q}_0^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^{0}\left(\frac{-1-a}{b}\right)\right] \\ &\quad - a\left[\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}b\left[\widetilde{Q}_2^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^{0}\left(\frac{-1-a}{b}\right)\right] + 2.\end{aligned}$$

□

For the second identity, we get

$$O^1_{1,3}(a,b) = \frac{1}{2}\left(\int_{-1}^{1}\int_{-1}^{1} \frac{s\,t^2}{(a+b\,s-t)^2}\,ds\,dt - \int_{-1}^{1}\int_{-1}^{1} \frac{s}{(a+b\,s-t)^2}\,ds\,dt\right). \quad \text{(A.2)}$$

By integrating similarly as above, we obtain

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} \frac{s\,t^2}{(a+b\,s-t)^2}\,ds\,dt \\
&= 4 + 2\int_{-1}^{1} s\,(a+b\,s)\,\left[\log\left(\eta_1\left(\frac{1-a}{b}-s\right)\right) - \log\left(\eta_1\left(\frac{-1-a}{b}-s\right)\right)\right] \\
&\qquad + \frac{s\,(a+b\,s)^2}{b}\left[\frac{1}{\frac{1-a}{b}-s} - \frac{1}{\frac{-1-a}{b}-s}\right] ds \\
&= 2a\,\left[\widetilde{Q}_1^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\
&\quad + 2b\,\left[\frac{2}{3}\widetilde{Q}_2^{-1}\left(\eta_1\frac{1-a}{b}\right) - \frac{2}{3}\widetilde{Q}_2^{-1}\left(\eta_1\frac{-1-a}{b}\right) + \frac{1}{3}\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \frac{1}{3}\widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\
&\quad - \left(\frac{a^2}{b}+\frac{1}{3}\,b\right)\left[\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right] \\
&- 2a\,\left[\frac{2}{3}\widetilde{Q}_2^{-1}\left(\eta_1\frac{1-a}{b}\right) - \frac{2}{3}\widetilde{Q}_2^{-1}\left(\eta_1\frac{-1-a}{b}\right) + \frac{1}{3}\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \frac{1}{3}\widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\
&\quad - \frac{2}{3}\,b\left[\frac{3}{5}\widetilde{Q}_3^{0}\left(\frac{1-a}{b}\right) - \frac{3}{5}\widetilde{Q}_3^{0}\left(\frac{-1-a}{b}\right) + \frac{2}{5}\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \frac{2}{5}\widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right] + 4.
\end{aligned}$$

where we exploited that

$$\begin{aligned}
s\,P_0(s) &= P_1(s) \\
s\,P_1(s) &= \frac{2}{3}\,P_2(s) + \frac{1}{2}\,P_0(s) \\
s\,P_2(s) &= \frac{3}{5}\,P_3(s) + \frac{2}{5}\,P_1(s).
\end{aligned}$$

For the second integral in A.2 we get

$$\begin{aligned}
\int_{-1}^{1}\int_{-1}^{1} \frac{s}{(a+b\,s-t)^2}\,ds\,dt &= \int_{-1}^{1} \frac{s}{(a+b\,s-1)} - \frac{s}{(a+b\,s+1)}\,ds \\
&= \frac{1}{b}\left[\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right]
\end{aligned}$$

and finally we obtain

$$\begin{aligned}
O^1_{1,3}(a,b) = {} & a\left[\widetilde{Q}_1^{-1}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{-1}\left(\frac{-1-a}{b}\right)\right] + \frac{2}{3}b\left[\widetilde{Q}_2^{-1}\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^{-1}\left(\frac{-1-a}{b}\right)\right] \\
& + \frac{1}{3}b\left[\widetilde{Q}_0^{-1}\left(\eta_1\frac{1-a}{b}\right) - \widetilde{Q}_0^{-1}\left(\eta_1\frac{-1-a}{b}\right)\right] \\
& - \frac{1}{2}\left(\frac{a^2-1}{b} - \frac{3}{5}b\right)\left[\widetilde{Q}_1^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_1^{0}\left(\frac{-1-a}{b}\right)\right] \\
& - \frac{2}{3}a\left[\widetilde{Q}_2^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_2^{0}\left(\frac{-1-a}{b}\right)\right] - \frac{1}{3}a\left[\widetilde{Q}_0^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_0^{0}\left(\frac{-1-a}{b}\right)\right] \\
& - \frac{1}{5}b\left[\widetilde{Q}_3^{0}\left(\frac{1-a}{b}\right) - \widetilde{Q}_3^{0}\left(\frac{-1-a}{b}\right)\right] + 2.
\end{aligned}$$

For proving the last identity we use two results that are stated in [19], i.e.

$$I^1_{j,1}(a,b) = b\,\frac{j}{2j+1}\,I^1_{j+1,0}(a,b) + b\,\frac{j+1}{2j+1}\,I^1_{j-1,0}(a,b) + a\,I^1_{j,0}(a,b)$$

and

$$I^1_{j,2}(a,b) = b\,\frac{j-1}{2j+1}\,I^1_{j+1,1}(a,b) + b\,\frac{j+2}{2j+1}\,I^1_{j-1,1}(a,b) + a\,I^1_{j,1}(a,b).$$

Combining both result yields

$$\begin{aligned} I^1_{j,2}(a,b) = {} & ab\,\frac{3j}{(j+2)(2j+1)} I^1_{j+1,0}(a,b) + \left[a^2\,\frac{3}{j+2} + \frac{j-1}{j+2} + b^2\,\frac{3(j-1)}{(j+2)(2j-1)}\right] I^1_{j,0}(a,b) \\ & + ab\,\frac{3(3j+2)}{(j+2)(2j+1)} I^1_{j-1,0}(a,b) + b^2\,\frac{3j}{(j+2)(2j-1)} I^1_{j-2,0}(a,b). \end{aligned}$$

Thus, we obtain

$$\begin{aligned} O^1_{j,3}(a,b) &= \frac{1}{3}\left(I^1_{j,2}(a,b) - I^1_{j,0}(a,b)\right) \\ &= ab\,\frac{j}{(j+2)(2j+1)} I^1_{j+1,0}(a,b) + \left[a^2\,\frac{1}{j+2} - \frac{1}{j+2} + b^2\,\frac{j-1}{(j+2)(2j-1)}\right] I^1_{j,0}(a,b) \\ &\quad + ab\,\frac{3j+2}{(j+2)(2j+1)} I^1_{j-1,0}(a,b) + b^2\,\frac{j}{(j+2)(2j-1)} I^1_{j-2,0}(a,b). \end{aligned}$$

Plugging the initial values of $I^1_{j,0}(a,b)$, that are given by

$$I^1_{j,0}(a,b) = \frac{1}{b}\left[\widetilde{Q}^0_j\left(\frac{-1-a}{b}\right) - \widetilde{Q}^0_j\left(\frac{1-a}{b}\right)\right],$$

completes the proof.

Appendix B
Matlab Programs for the Assembly of the Galerkin Matrices

Listing B.1: buildV.m

```
1
2  function V = buildV(coordinates,elements,p,options)
3
4  if nargin <4
5    error('At least four input arguments required.')
6  end
7  p = checkp0(p,size(elements,1));
8  %*** Try using mpc-routines
9  try
10   [V,F] = buildV_ex(coordinates,elements,p,options);
11   if F==1
12     disp(' Minimal mesh size smaller than eps. Results...
13          are inaccurate.')
14   end
15   return
16 catch exception
17   disp(['Error in buildV_ex: ', exception.message])
18 end
19 %*** Try using quadrature-routines
20 try
21   V = buildV_quad(coordinates, elements, p, options);
22   return
23 catch exception
24   disp(['Error in buildV_quad: ', exception.message])
25 end
26 error('Neither buildV_ex nor buildV_quad were ...
27       successful in buildV.')
28
29
30 function V = galVid(u,p,options)
31 %*** compute general Parameters
32 u=u(:);
33 utu = u'*u;
34 uut = u*u';
35 lam = options.lambda;
36 mu = options.mu;
37 V = zeros(2*(p+1),2*(p+1));
38 %*** factors
39 fac =  (lam+3*mu)*utu/(4*pi*mu*(lam+2*mu));
40 fac2 = 4*(lam+mu)*uut/((lam+3*mu)*utu);
41 %*** add delta_k0*delta_j0
42 V([1,p+2],[1,p+2]) = -fac*((2*log(4*utu)-6)*eye(2)-fac2);
43 if p>=1
```

```
  %*** diagonal enries
  for i=2:p+1
    V([i,i+p+1],[i,i+p+1])=fac*(2/(i*(i-1))*eye(2));
  end
  %*** off-diagonal entries
  %*** first row
  for j = 1:2:(p-1)
    V([1,p+2],[j+2,j+p+3]) = ...
                  -fac*(8/(j*(j+1)*(j+2)*(j+3))*eye(2));
  end
  %*** remaining part of upper triangular part
  for i = 2:p
    for k = 1:floor((p-i+1)/2)
      V(i,(i+2*k)) = ...
               -2*fac/((i+k-1)*(i+k)*(2*k-1)*(2*k+1));
      V(i+p+1,(i+2*k)+p+1) = V(i,(i+2*k));
    end
  end
  %*** symmetric part
  for i = 1:2*(p+1)
    for j = i+2:2:2*(p+1)
      V(j,i) = V(i,j);
    end
  end
end


function [val,flag] = galV_ex(vert_x,vert_y,p,options)
%*** compute general parameters
v = (vert_x(2,:)-vert_x(1,:))/2;
u = (vert_y(2,:)-vert_y(1,:))/2;
w = (vert_y(1,:)+vert_y(2,:)-vert_x(1,:)-vert_x(2,:))/2;
utu = u*u';
vtv = v*v';

flag=0;
if (sqrt(utu) < eps) || (sqrt(vtv)<eps)
  flag=1;
  val = zeros(2*(p+1),2*(p+1));
  return
end

%*** compute factors
lam = options.lambda;
mu = options.mu;
fac = (lam+3*mu)*sqrt(utu)*sqrt(vtv)/(4*pi*mu*(lam+2*mu));
fac2 = (lam+mu)/(lam+3*mu);
if(vtv>utu) % switch u and v s.t. |b|<1

  %*** matrices
  vvt=v'*v/vtv;
  vsvst=[v(2);-v(1)]*[v(2),-v(1)]/vtv;
  vvst=v'*[v(2),-v(1)]/vtv;
  vsvt=[v(2);-v(1)]*v/vtv;

  %*** compute z=a+bs
  a1= (v(:,1).*w(:,1)+v(:,2).*w(:,2))/vtv;
  a2= (v(:,1).*w(:,2)-v(:,2).*w(:,1))/vtv;
  a = complex(a1, a2);
  b1= (v(:,1).*u(:,1)+v(:,2).*u(:,2))/vtv;
  b2= (v(:,1).*u(:,2)-v(:,2).*u(:,1))/vtv;
  b = complex( b1, b2);

  %*** compute integrals Im1 and I0xIm(z)
  im1 = real(Im1(a,b,p))';
  i0= [zeros(p+2,1),I0(a,b,p+1).'];
  tmpm1=[im1,zeros(p+1);zeros(p+1),im1];
  c1=repmat((1:(p+1))./[1,3:2:(2*p+1)],p+1,1);
  c2=repmat((0:p)./[1,3:2:(2*p+1)],p+1,1);
  tmp0=a2*i0(1:p+1,2:end-1)+...
       b2*(c1.*i0(1:p+1,3:end)+c2.*i0(1:p+1,1:end-2));
  %*** resize matrices and multiply with intgrals
  V1 = [(vvt(1,1)-vsvst(1,1))*imag(tmp0),...
        (vvt(1,2)-vsvst(1,2))*imag(tmp0);...
```

```
        (vvt(2,1)-vsvst(2,1))*imag(tmp0),...
        (vvt(2,2)-vsvst(2,2))*imag(tmp0)];
  V2 = [(vvst(1,1)+vsvt(1,1))*real(tmp0),...
        (vvst(1,2)+vsvt(1,2))*real(tmp0);...
        (vvst(2,1)+vsvt(2,1))*real(tmp0),...
        (vvst(2,2)+vsvt(2,2))*real(tmp0)];

  %*** compute single layer potential
  val=-fac*(tmpm1-fac2*(V1-V2));

  %*** add deltak0*deltaj0
  tmp=fac*2*(log(vtv)*eye(2)-2*fac2*(v'*v)/vtv);
  val([1,p+2],[1,p+2]) = val([1,p+2],[1,p+2]) - tmp;
else
  %*** compute z=a+bs
  a1=-(u(:,1).*w(:,1)+u(:,2).*w(:,2))/utu;
  a2=-(u(:,1).*w(:,2)-u(:,2).*w(:,1))/utu;
  a = complex(a1, a2);
  b1= (u(:,1).*v(:,1)+u(:,2).*v(:,2))/utu;
  b2= (u(:,1).*v(:,2)-u(:,2).*v(:,1))/utu;
  b = complex( b1, b2);
  %*** compute matrices
  uut=u'*u/utu;
  ususst=[u(2);-u(1)]*[u(2),-u(1)]/utu;
  uust=u'*[u(2),-u(1)]/utu;
  usut=[u(2);-u(1)]*u/utu;
  %*** compute integrals Im1 and I0xIm(z)=> combination of
      rows!
  im1 = real(Im1(a,b,p));
  i0= [zeros(1,p+2);I0(a,b,p+1)];
  tmpm1=[im1,zeros(p+1);zeros(p+1),im1];
  c1=repmat((1:(p+1))./[1,3:2:(2*p+1)],p+1,1)';
  c2=repmat((0:p)./[1,3:2:(2*p+1)],p+1,1)';
  tmp0=a2*i0(2:end-1,1:p+1)+...
       b2*(c1.*i0(3:end,1:p+1)+c2.*i0(1:end-2,1:p+1));

  %*** resize matrices and multiply with intgrals
  U1 = [(uut(1,1)-ususst(1,1))*imag(tmp0),...
        (uut(1,2)-ususst(1,2))*imag(tmp0);...
        (uut(2,1)-ususst(2,1))*imag(tmp0),...
        (uut(2,2)-ususst(2,2))*imag(tmp0)];
  U2 = [(uust(1,1)+usut(1,1))*real(tmp0),...
        (uust(1,2)+usut(1,2))*real(tmp0);...
        (uust(2,1)+usut(2,1))*real(tmp0),...
        (uust(2,2)+usut(2,2))*real(tmp0)];

  %*** compute single layer potential
  val=-fac*(tmpm1-fac2*(U1-U2));

  %*** add deltak0*deltaj0
  tmp=fac*2*(log(utu)*eye(2)-2*fac2*uut);
  val([1,p+2],[1,p+2]) = val([1,p+2],[1,p+2]) - tmp;
end

function [V,Flag] = buildV_ex(coordinates,elements,...
                              p,options)
sp = cumsum(([0;p+1]));
V = zeros(2*sum(p+1));
Flag = 0;
for k1=1:size(elements,1)
  vert = coordinates(elements(k1,:),:);
  u=(vert(2,:)-vert(1,:))/2;
  %*** index 1 in global matrix
  index1 = sp(k1):sp(k1+1)-1;
  ind1 = [index1, sp(end)+index1]+1;
  V(ind1,ind1) = galVid(u,p(k1),options);
  for k2=k1+1:size(elements,1)
    %*** maximum polynomial degree for indices
    m=max(p(k1),p(k2));
    idx2=[1:p(k2)+1,m+2:m+p(k2)+2];
    %*** index 2 in global matrix
    index2 = sp(k2):sp(k2+1)-1;
    ind2 = [index2,sp(end)+index2]+1;
    %*** compute submatrix
```

```
    [tmp,flag] = galV_ex(coordinates(elements(k1,:),:),...
                         coordinates(elements(k2,:),:),...
                         m,options);
    Flag = flag | Flag;
    %*** components (1,1), (1,2)
    V(index1+1 ,ind2) = tmp(1:p(k1)+1,idx2);
    %*** components (2,1), (2,2)
    V(index1+sp(end)+1,ind2) = tmp(m+2:m+p(k1)+2,idx2);
    V(ind2,ind1) = V(ind1,ind2)';
  end
end


function V = buildV_quad(coordinates,elements,p,options)
%*** resize p
if(numel(p)==1)
  p = p*ones(size(elements,1),1);
end
max_p = max(p);
%*** set number of Gauss points
ngauss = 32;
while ngauss <= 2*max_p + 64
  ngauss = 2*ngauss;
end

[wg,xg] = gauss(ngauss);
ngauss  = length(wg);
pleg = bLegendre(xg,max_p);

%*** Get edges of the elements
a = (coordinates(elements(:,2),:)...
     -coordinates(elements(:,1),:))/2;
b = (coordinates(elements(:,1),:)...
     +coordinates(elements(:,2),:))/2;
hh = sqrt(sum(a.^2,2)); %h/2

V = zeros(2*sum(p+1));
W = repmat(wg(:),1,max(p)+1);
cp = cumsum([0;p(:)+1]);
for k1 = 1:size(elements,1)
  %*** map Gauss Points to Element
  points = xg * a(k1,:) +  ones(size(xg))*b(k1,:);
  %*** index 1 in global matrix
  index1 = cp(k1):cp(k1+1)-1;
  ind1 = [index1, cp(end)+index1]+1;
  dummy = hh(k1)* (W(:,1:p(k1)+1).*pleg(:,1:p(k1)+1))';
  tmp = galVid(a(k1,:),p(k1),options);
  V(ind1,ind1) = tmp;
  for k2 = k1+1:size(elements,1)
    %*** Calculate the inner integral
    tmp = potV(coordinates(elements(k2,:),:),points,...
               p(k2),options);
    %*** index 2 in global matrix
    index2 = cp(k2):cp(k2+1)-1;
    ind2 = [index2,cp(end)+index2]+1;

    %***quadrature for outer integral --> Submatrix
    V(index1+1         ,ind2) = dummy * tmp(1:ngauss,:);
    V(index1+cp(end)+1,ind2) = dummy * tmp(ngauss+1:end,:);
    V(ind2,ind1) = V(ind1,ind2)';
  end
end
```

Listing B.2: buildK.m

```
function K = buildK(coordinates,elements,p0,p1,options)
if nargin <5
  error('Five input arguments required.')
end
nE = size(elements,1);
p0 = checkp0(p0,nE);
p1 = checkp1(p1,nE);
%*** Try using mpc-routines
try
  [K,Flag] = buildK_ex(coordinates,elements,p0,p1,options);
  if Flag==1
    disp(' Minimal mesh size smaller than eps. Results...
          are inaccurate.')
  end
  return
catch exception
  disp(['Error in buildK_ex: ', exception.message])
end
%*** Try using quadrature-routines
try
  [K,Flag] = buildK_quad(coordinates,elements,p0,p1,options);
  if Flag==1
    disp(' Minimal mesh size smaller than eps. Results...
          are inaccurate.')
  end
  return
catch exception
  disp(['Error in buildK_quad: ', exception.message])
end
error('Neither buildK_ex nor buildK_quad were ...
       successful in buildK.')

function K = galK_id(p0,p1,hh,options)
%*** Factor 2 allready simplified
fac = -options.mu/(pi*(options.lambda+2*options.mu))*hh;
K = zeros(2*(p0+1),2*(p1+1));
%*** Compute DLP for  (N_1,N_2) x P_k
for i0 = 0:2:p0
  %*** Case j even
  K(i0+p0+2,   1) = -fac/((i0-1)*(i0+2));
  K(i0+1   ,p1+2) = -K(i0+p0+2,1);
  K(i0+p0+2,   2) = -K(i0+p0+2,1);
  K(i0+1   ,p1+3) =  K(i0+p0+2,1);
end
for i0 = 1:2:p0
    %*** Case j odd
  K(i0+p0+2,   1) = fac/((i0)*(i0+1));
  K(i0+1   ,p1+2) = -K(i0+p0+2,1);
  K(i0+p0+2,   2) =  K(i0+p0+2,1);
  K(i0+1   ,p1+3) = -K(i0+p0+2,1);
end

%*** Compute DLP for P_k x (N_3,....)
for i1 = 2:2:p1
  for i0 = 1:2:p0
      %*** Case k-j even
      K(i0+p0+2, i1+1   ) = ...
              4*fac/((i0-i1)*(i0-i1+2)*(i0+i1-1)*(i0+i1+1));
      K(i0+1   , i1+p1+2) = -K(i0+p0+2,i1+1);
  end
end
for i1 = 3:2:p1
  for i0 = 0:2:p0
      %*** Case k-j odd
      K(i0+p0+2,i1+1   ) = ...
             4*fac/((i0-i1)*(i0-i1+2)*(i0+i1-1)*(i0+i1+1));
      K(i0+1   ,i1+p1+2) = -K(i0+p0+2,i1+1);
  end
end

function [val,flag] = galK_ex(vert_x,vert_y,p0,p1,options)
p=max(p0+1,p1);
%*** compute u,v,w
```

```
v = (vert_x(2,:)-vert_x(1,:))/2;
u = (vert_y(2,:)-vert_y(1,:))/2;
w = (vert_y(1,:)+vert_y(2,:))/2-(vert_x(1,:)+vert_x(2,:))/2;
utu = u*u';
vtv = v*v';
lam = options.lambda;
mu = options.mu;

val = zeros(2*(p0+1),2*(p1+1));
flag=0;
if (sqrt(utu) < eps) || (sqrt(vtv)<eps)
  flag=1;
  return
end

fac1 =  mu/(2*pi*(lam+2*mu));
fac2 = (lam+mu)/(2*pi*(lam+2*mu));
%*** assemble matrices
uut=u'*u./utu;
usust=[u(2);-u(1)]*[u(2),-u(1)]/utu;
%*** (v vs' + vs v')
uust=(u'*[u(2),-u(1)]+[u(2);-u(1)]*u)/utu;
I=[ones(p0+1,p1+1),zeros(p0+1,p1+1);...
   zeros(p0+1,p1+1),ones(p0+1,p1+1)];
IxI=[ zeros(p0+1,p1+1),ones(p0+1,p1+1);...
     -ones(p0+1,p1+1),zeros(p0+1,p1+1)];
UUT=[uut(1,1)*ones(p0+1,p1+1),uut(1,2)*ones(p0+1,p1+1);...
     uut(2,1)*ones(p0+1,p1+1),uut(2,2)*ones(p0+1,p1+1)];
USUST=[usust(1,1)*ones(p0+1,p1+1),usust(1,2)*ones(p0+1,p1+1)
    ;...
       usust(2,1)*ones(p0+1,p1+1),usust(2,2)*ones(p0+1,p1+1)];
UUST= [uust(1,1)*ones(p0+1,p1+1),uust(1,2)*ones(p0+1,p1+1);...
       uust(2,1)*ones(p0+1,p1+1),uust(2,2)*ones(p0+1,p1+1)];
%*** compute z=a+bt
a1=-(u(:,1).*w(:,1)+u(:,2).*w(:,2))/utu;
a2=-(u(:,1).*w(:,2)-u(:,2).*w(:,1))/utu;
a = complex(a1, a2);
b1= (u(:,1).*v(:,1)+u(:,2).*v(:,2))/utu;
b2= (u(:,1).*v(:,2)-u(:,2).*v(:,1))/utu;
b = complex( b1, b2);
%*** compute matrices with
%           /                              /
% tmp0(j,k)=|P_j N_k/(z-s) & tmp1(j,k)=|P_j N_k Im(z)/(z-s)^2
%           /                              /

%*** c0 = [1/2 , 1/2 , 1/3 , ... , 1/(2k-3)]
c0 = repmat(1./[2,2,3:2:(2*p-1)],p0+1,1);
%*** c1 = [1 , 2/3 , ... ,(j+1)/(2j+1) ]
c1=repmat((1:(p0+1))./[1,3:2:(2*p0+1)],p1+1,1)';
%*** c2 = [0 , 1/3 , ... , j/(2j+1) ]
c2=repmat((0:p0)./[1,3:2:(2*p0+1)],p1+1,1)';
if abs(b) > 1
  ctmp0 = -1/b*I0(-a/b,1/b,p).';
else
  ctmp0 = I0(a,b,p);
end
ctmp1=O1(a,b,p+1);
o1 = [zeros(1,p1+1);ctmp1(1:p0+2,1:p1+1)];
%*** combination of rows for Im(a+bs)
tmp1 = repmat( a2*o1(2:end-1,:)+b2*(c1.*o1(3:end,:)...
              +c2.*o1(1:end-2,:)),2,2);
i0 = ctmp0(1:p0+1,1:p1+1);
%*** combination of columns for N_k
tmp0=repmat([i0(:,1)-i0(:,2),i0(:,1)+i0(:,2), ...
             i0(:,3:p1+1)-i0(:,1:p1-1)].*c0(:,1:p1+1),2,2);
%*** computation of the double layer potential
val=fac1*IxI.*real(tmp0)+1/(2*pi)*I.*imag(tmp0)...
    -fac2*( (UUT-USUST).*real(tmp1) + UUST.*imag(tmp1));
val=val*sqrt(vtv);


function [K,Flag] = buildK_ex(coordinates,elements,...
                              p0,p1,options)

```

```
a = ( coordinates(elements(:,2),:)...
     -coordinates(elements(:,1),:))/2;
hh = sqrt(sum(a.^2,2));
sp0 = cumsum([0;p0(:)+1]);
sp1 = cumsum([size(coordinates,1);p1(:)-1]);
K = zeros(2*sp0(end),2*sp1(end));

Flag = 0;
for k1=1:size(elements,1)
  %*** index in global matrix
  index1 = sp0(k1):sp0(k1+1)-1;
  ind1 = [index1,index1+sp0(end)]+1;
  for k2=1:size(elements,1)
    index2 = [elements(k2,:)-1,sp1(k2):sp1(k2+1)-1];
    ind2 = [index2,index2+sp1(end)]+1;
    if k1~=k2
      %*** Case non-identical elements
      [tmp,flag] = galK_ex(coordinates(elements(k1,:),:), ...
                           coordinates(elements(k2,:),:), ...
                           p0(k1), p1(k2),options);
      Flag = Flag|flag;
      K(index1+1          ,ind2) = ...
             K(index1+1          ,ind2)+ tmp(1:p0(k1)+1   , :);
      K(index1+sp0(end)+1,ind2) = ...
             K(index1+sp0(end)+1,ind2)+ tmp(p0(k1)+2:end, :);
    else
      %*** Case identical elements
      K(ind1,ind2) = K(ind1,ind2) ...
                   + galK_id(p0(k1),p1(k1),hh(k1),options);

    end
  end
end



function [K,Flag] = buildK_quad(coordinates, elements, ...
                                p0, p1, options)
max_p = max(p0,p1);
ngauss = 32;
while ngauss <= 2*max_p + 64
  ngauss = 2*ngauss;
end

[wg,xg] = gauss(ngauss);
ngauss = length(xg);
pleg = bLegendre(xg,max(p0));
%Get left and right edges of the elements ...
a = ( coordinates(elements(:,2),:)...
     -coordinates(elements(:,1),:))/2;
b = ( coordinates(elements(:,1),:)...
     +coordinates(elements(:,2),:))/2;
hh = sqrt(sum(a.^2,2));
cp0 = cumsum([0;p0(:)+1]);
cp1 = cumsum([size(coordinates,1);p1(:)-1]);

K = zeros(2*cp0(end),2*cp1(end));
Flag = 0;
if(2*min(hh)<eps)
  Flag=1;
  return
end

W = repmat(wg(:),1,max(p0)+1);
for k1 = 1:size(elements,1);
  %*** map Gauss Points to Element
  points = xg * a(k1,:) +  ones(size(xg))*b(k1,:);
  index1 = cp0(k1):cp0(k1+1)-1;
  ind1 = [index1,index1+cp0(end)]+1;
  dummy = hh(k1)*(W(:,1:p0(k1)+1).*pleg(:,1:p0(k1)+1))';
  for k2 = 1:size(elements,1);
    index2 = [elements(k2,:)-1,cp1(k2):cp1(k2+1)-1];
    ind2 = [index2,index2+cp1(end)]+1;
    %*** calculate the inner integral
```

```
221       if k1~=k2
222         tmp = potK(coordinates(elements(k2,:),:),...
223                    points,p1(k2),options);
224         K(index1+1          ,ind2) = ...
225                 K(index1+1,ind2) + dummy*tmp(1:ngauss     , :);
226         K(index1+cp0(end)+1,ind2) = ...
227                 K(index1+cp0(end)+1,ind2) + dummy*tmp(ngauss+1:
                        end, :);
228       else
229       %*** case of identical elements
230         K(ind1,ind2) = K(ind1,ind2)...
231                        + galK_id(p0(k1),p1(k1),hh(k1),options);
232       end
233     end
234   end
```

Listing B.3: buildW.m

```
function [W,varargout] = buildW(coordinates,elements,...
                                p,varargin)
p = checkp1(p,size(elements,1));
%*** Try using mpc-routines
try
  if nargout ==1
    W = buildW_ex(coordinates,elements,p,varargin{1});
  else
    W = buildW_ex(coordinates,elements,p, varargin{1});
    varargout{1} = buildV(coordinates,elements,p-1, ...
                          varargin{1});
  end
return
catch exception
  disp(['Error in buildW_ex: ', exception.message])
end

%*** Try using quadrature-routines
try
  if nargout ==1
    W = buildW_quad(coordinates, elements, p, varargin{1});
  else
    W = buildW_quad(coordinates,elements,p, varargin{1});
    varargout{1} = buildV(coordinates,elements,p-1,...
                          varargin{1});
  end
  return
catch exception
  disp(['Error in buildW_quad: ', exception.message])
end
error('Neither buildW_ex nor buildW_quad were successful in
    buildW.')

function W = buildW_ex(coordinates,elements,p,varargin)
cp = cumsum([size(coordinates,1);p(:)-1]);
W = zeros(2*cp(end));
for j = 1:size(elements,1)
  vert1 = coordinates(elements(j,:),:);
  u=(vert1(2,:)-vert1(1,:))/2;
  index1 = [elements(j,:)-1,cp(j):cp(j+1)-1]+1;
  ind1 = [index1,index1+cp(end)];
  for k = 1:size(elements,1)
    index2 = [elements(k,:)-1,cp(k):cp(k+1)-1]+1;
    ind2 = [index2,index2+cp(end)];
    vert2  = coordinates(elements(k,:),:);
    if j~=k
      pstar=max(p(j)-1,p(k)-1);
      tmp = galW_ex(vert1, vert2,pstar,varargin{1});
      tmp =  tmp([1:p(j),pstar+2:pstar+p(j)+1],...
                 [1:p(k),pstar+2:pstar+p(k)+1]);
    else
      tmp = galWid(u,p(j)-1,varargin{1});
    end
    i = size(tmp)/2;
    tmp = tmp([1,1:i(1),i(1)+1,i(1)+1:end],[1,1:i(2),i(2)+1,i
        (2)+1:end]);
    tmp([1,i(1)+2],:) = -tmp([1,i(1)+2],:)/2;
    tmp([2,i(1)+3],:) = tmp([2,i(1)+3],:)/2;
    tmp(:,[1,i(2)+2]) = -tmp(:,[1,i(2)+2])/2;
    tmp(:,[2,i(2)+3]) = tmp(:,[2,i(2)+3])/2;
    W(ind1,ind2) = W(ind1,ind2) + tmp;
  end
end

function W = buildW_quad(coordinates, elements, p, varargin)
ngauss = 32;
while ngauss<2*p
  ngauss = 2*ngauss;
end
[weights,xg] = gauss(ngauss);
pleg = bLegendre(xg,max(p));
u = ( coordinates(elements(:,2),:)...
     -coordinates(elements(:,1),:))/2;
```

```
m = ( coordinates(elements(:,1),:)
     +coordinates(elements(:,2),:))/2;
cp = cumsum([size(coordinates,1);p(:)-1]);
W = zeros(2*cp(end));
wg = repmat(weights(:),1,max(p)+1);
for j = 1:size(elements,1)
  points = xg * u(j,:) +  ones(size(xg))*m(j,:);
  idx1 = [elements(j,:)-1,cp(j):cp(j+1)-1]+1;
  ind1 = [idx1,idx1+cp(end)];
  outer = (wg(:,1:p(j)).*pleg(:,1:p(j)))';
  for k = 1:size(elements,1)
    idx2 = [elements(k,:)-1,cp(k):cp(k+1)-1]+1;
    ind2 = [idx2,idx2+cp(end)];
    if j~=k
      pot = potVscaled(coordinates(elements(k,:),:),...
                       points,p(k)-1,varargin{1});
      tmp=[outer*pot(1:ngauss,:) ; outer*pot(ngauss+1:end,:)];
    else
      tmp = galWid(u(j,:),p(j)-1, varargin{1});
    end
    i = size(tmp)/2;
    tmp = tmp([1,1:i(1),i(1)+1,i(1)+1:end],...
              [1,1:i(2),i(2)+1,i(2)+1:end]);
    tmp([1,i(1)+2],:) = -tmp([1,i(1)+2],:)/2;
    tmp([2,i(1)+3],:) = tmp([2,i(1)+3],:)/2;
    tmp(:,[1,i(2)+2]) = -tmp(:,[1,i(2)+2])/2;
    tmp(:,[2,i(2)+3]) = tmp(:,[2,i(2)+3])/2;
    W(ind1,ind2) = W(ind1,ind2) + tmp;
  end
end

function W = galWid(u,p,options)
%*** compute general Parameters
u=u(:);
utu = u'*u;
uut = u*u';
lam = options.lambda;
mu = options.mu;
W = zeros(2*(p+1));

%*** factors
fac =  mu*(lam+mu)/(pi*(lam+2*mu));
fac2 = 4*uut/utu;% 4*(lam+mu)*uut/((lam+3*mu)*utu);

%*** add delta_k0*delta_j0
W([1,p+2],[1,p+2]) = -fac*((2*log(4*utu)-6)*eye(2)-fac2);
if p>=1
  %*** diagonal enries
  for i=2:p+1
    W([i,i+p+1],[i,i+p+1])=fac*(2/(i*(i-1))*eye(2));
  end
  %*** off-diagonal entries
  %*** first row
  for j = 1:2:(p-1)
    W([1,p+2],[j+2,j+p+3]) = -fac*(8/(j*(j+1)*(j+2)*(j+3))...
                                   *eye(2));
  end
  %*** remaining part of upper triangular part
  for i = 2:p
    for k = 1:floor((p-i+1)/2)
      W(i,(i+2*k)) = -2*fac/((i+k-1)*(i+k)*(2*k-1)*(2*k+1));
      W(i+p+1,(i+2*k)+p+1) = W(i,(i+2*k));
    end
  end
  %*** symmetric part
  for i = 1:2*(p+1)
    for j = i+2:2:2*(p+1)
      W(j,i) = W(i,j);
    end
  end
end

function val = galW_ex(vert_x,vert_y,p,options)
%*** compute general paramters
```

```
v = (vert_x(2,:)-vert_x(1,:))/2;
u = (vert_y(2,:)-vert_y(1,:))/2;
w = (vert_y(1,:)+vert_y(2,:))/2 - (vert_x(1,:)+vert_x(2,:))/2;
utu = u*u';
vtv = v*v';
lam = options.lambda;
mu = options.mu;
%*** compute factor
fac =  mu*(lam+mu)/(pi*(lam+2*mu));
if(vtv>utu) % switch u and v s.t. |b|<1
  %*** matrices
  vvt=v'*v/vtv;
  vsvst=[v(2);-v(1)]*[v(2),-v(1)]/vtv;
  vvst=v'*[v(2),-v(1)]/vtv;
  vsvt=[v(2);-v(1)]*v/vtv;
  %*** compute z=a+bt
  a1= (v(:,1).*w(:,1)+v(:,2).*w(:,2))/vtv;
  a2= (v(:,1).*w(:,2)-v(:,2).*w(:,1))/vtv;
  a = complex( a1, a2);
  b1= (v(:,1).*u(:,1)+v(:,2).*u(:,2))/vtv;
  b2= (v(:,1).*u(:,2)-v(:,2).*u(:,1))/vtv;
  b = complex( b1, b2);
  %*** compute integrals Im1 and I0xIm(z)
  im1 = real(Im1(a,b,p)).';
  i0= [zeros(p+2,1),I0(a,b,p+1).'];
  tmpm1=[im1,zeros(p+1);zeros(p+1),im1];
  c1=repmat((1:(p+1))./[1,3:2:(2*p+1)],p+1,1);
  c2=repmat((0:p)./[1,3:2:(2*p+1)],p+1,1);
  tmp0=repmat(a2*i0(1:p+1,2:end-1)+b2*(c1.*i0(1:p+1,3:end)...
               +c2.*i0(1:p+1,1:end-2)),2,2);
  %*** resize matrices
  VVT=[vvt(1,1)*ones(p+1),vvt(1,2)*ones(p+1);...
       vvt(2,1)*ones(p+1),vvt(2,2)*ones(p+1)];
  VSVST=[vsvst(1,1)*ones(p+1), vsvst(1,2)*ones(p+1);...
         vsvst(2,1)*ones(p+1), vsvst(2,2)*ones(p+1)];
  VVST= [(vvst(1,1)+vsvt(1,1))*ones(p+1), ...
         (vvst(1,2)+vsvt(1,2))*ones(p+1); ...
         (vvst(2,1)+vsvt(2,1))*ones(p+1), ...
         (vvst(2,2)+vsvt(2,2))*ones(p+1)];

  %*** compute single layer potential
  val=-fac*(tmpm1-(imag(tmp0).*(VVT-VSVST)-real(tmp0).*VVST));
  %*** add deltak0*deltaj0
  tmp=fac*2*(log(vtv)*eye(2)-2*vvt);
  val([1,p+2],[1,p+2]) = val([1,p+2],[1,p+2]) - tmp;
else
  %*** matrices
  uut=u'*u/utu;
  usust=[u(2);-u(1)]*[u(2),-u(1)]/utu;
  uust=u'*[u(2),-u(1)]/utu;
  usut=[u(2);-u(1)]*u/utu;

  %*** compute z=a+bt
  a1=-(u(:,1).*w(:,1)+u(:,2).*w(:,2))/utu;
  a2=-(u(:,1).*w(:,2)-u(:,2).*w(:,1))/utu;
  a = complex(a1, a2);
  b1= (u(:,1).*v(:,1)+u(:,2).*v(:,2))/utu;
  b2= (u(:,1).*v(:,2)-u(:,2).*v(:,1))/utu;
  b = complex( b1, b2);

  %*** compute integrals Im1 and I0xIm(z)
  im1 = real(Im1(a,b,p));
  i0= [zeros(1,p+2);I0(a,b,p+1)];
  tmpm1=[im1,zeros(p+1);zeros(p+1),im1];
  c1=repmat([1:(p+1)]./[1,3:2:(2*p+1)],p+1,1)';
  c2=repmat([0:p]./[1,3:2:(2*p+1)],p+1,1)';
  tmp0=repmat(a2*i0(2:end-1,1:p+1)+b2*(c1.*i0(3:end,1:p+1)...
               +c2.*i0(1:end-2,1:p+1)),2,2);

  %*** resize matrices
  UUT=[uut(1,1)*ones(p+1),uut(1,2)*ones(p+1);...
       uut(2,1)*ones(p+1),uut(2,2)*ones(p+1)];
  USUST=[usust(1,1)*ones(p+1), usust(1,2)*ones(p+1);...
         usust(2,1)*ones(p+1), usust(2,2)*ones(p+1)];
```

```
  UUST= [(uust(1,1)+usut(1,1))*ones(p+1), ...
         (uust(1,2)+usut(1,2))*ones(p+1); ...
         (uust(2,1)+usut(2,1))*ones(p+1), ...
         (uust(2,2)+usut(2,2))*ones(p+1)];

  %*** compute single layer potential
  val=-fac*(tmpm1-(imag(tmp0).*(UUT-USUST)-real(tmp0).*UUST));

  %*** add deltak0*deltaj0
  tmp=fac*2*(log(utu)*eye(2)-2*uut);
  val([1,p+2],[1,p+2]) = val([1,p+2],[1,p+2]) - tmp;
end

function V = potVscaled(vertices,x,p,options)
Npts=size(x,1);
V = zeros(2*Npts,2*(p+1));
%calculate vectors
m = (vertices(1,:)+vertices(2,:))/2;
u = (vertices(2,:)-vertices(1,:))/2;
utu = u*u';
v=u./sqrt(utu);
vs=[-v(2),v(1)];
%calculate z
w = [m(1)-x(:,1),m(2)-x(:,2)];
z_re =-(u(:,1).*w(:,1)+u(:,2).*w(:,2))/utu;
z_im =-(w(:,1).*u(:,2)-w(:,2).*u(:,1))/utu;
%calculate matrices
v0=eye(2);
v1=v'*v;
v2=v'*vs+vs'*v;
v3=v1-vs'*vs;
%factors
fac1 = options.mu*(options.lambda+options.mu)/...
       (pi*(options.lambda+2*options.mu));
%log-term
tmp0 = real(qtm1(complex(z_re,z_im),p,1).');
tmp0(:,1) = tmp0(:,1);
%second-term
tmp1 = qt0(complex(z_re,z_im),p,1).';
for k=1:size(x,1)
    tmp1(k,:) = tmp1(k,:)*z_im(k);
end
V(1:Npts     ,1  :p+1) = -fac1*(tmp0*v0(1,1)+...
                    real(tmp1)*v2(1,1)-imag(tmp1)*v3(1,1));
V(Npts+1:end,1  :p+1) = -fac1*(tmp0*v0(1,2)+...
                    real(tmp1)*v2(1,2)-imag(tmp1)*v3(1,2));
V(1:Npts     ,p+2:end) = -fac1*(tmp0*v0(2,1)+...
                    real(tmp1)*v2(2,1)-imag(tmp1)*v3(2,1));
V(Npts+1:end,p+2:end) = -fac1*(tmp0*v0(2,2)+...
                    real(tmp1)*v2(2,2)-imag(tmp1)*v3(2,2));

V(1:Npts     ,1  )=V(1:Npts     ,1  )-fac1*(log(utu)-2*v1(1,1));
V(Npts+1:end,1  )=V(Npts+1:end,1  )-fac1*(         -2*v1(1,2));
V(1:Npts     ,p+2)=V(1:Npts     ,p+2)-fac1*(         -2*v1(2,1));
V(Npts+1:end,p+2)=V(Npts+1:end,p+2)-fac1*(log(utu)-2*v1(2,2));
```

Appendix C
Matlab Programs for Solving Problems with Gliding Conditions

Listing C.1: solveMixedGliding.m

```
function [uh , phih, V, W] = ...
          solveMixedgliding(coordinates,dirichlet,neumann,...
                            gliding,m,n,u,g,p0,p1,varargin)

if nargin<10
  error('At least ten input arguments required.')
elseif nargin ==10
  options = [];
else
  options = varargin{1};
end

nED = size(dirichlet,1);
nEN = size(neumann,1);
nC = size(coordinates,1);
cp0 = cumsum(p0(:)+1);        % dof with respect to S0
cp1 = cumsum([nC;p1(:)-1]);   % dof with respect to S1
%  dof with respect to S1 on Gamma_D
dofD        = [unique(dirichlet)',cp1(1)+1:cp1(nED+1)];
%  dof with respect to S1 on Gamma_G
dofGD       = [unique(gliding)',cp1(nED+nEN+1)+1:cp1(end)];
%  dof with respect to S0 on Gamma_G
dofGN       = setdiff(1:cp0(end), 1:cp0(nED+nEN));
%  dof with respect to S0 on Gamma_N
dofN        = setdiff(1:cp0(end), [dofGN,1:cp0(nED)]);
%  freenodes on Gamma_G
freenodesG=setdiff(unique(gliding),unique(dirichlet));
%  intersection of dirichlet and gliding boundary
dofGD=setdiff(dofGD,dofD);
%  lenght of vectors
ndofD = length(dofD); ndofN = length(dofN);
ndofGD = length(dofGD); ndofGN = length(dofGN);

%*** Assembly of Galerkin matrices
elements=[dirichlet;neumann;gliding];
tic
W=buildW(coordinates,elements,p1,options);
V=buildV(coordinates,elements,p0,options);
disp(['Time for V&W   : ',num2str(toc)]);
M=buildM(coordinates,elements,2,p0,p1);
tic
K = buildK(coordinates,elements,p0,p1,options);
disp(['Time for K     : ',num2str(toc)]);
```

```
%*** project solution on discrete spaces
uDh    = projectionH1(coordinates,elements, p1, u, options);
tmpu   = uDh(dofD,:);
phiNh  = projectionL2(coordinates,elements,p0, g, options);
tmpphi = phiNh(dofN,:);

%*** compute matrices M1 and M2
M1=zeros(2*(ndofD+ndofN)+ndofGD+ndofGN,2*cp0(end));
M2=zeros(2*(ndofD+ndofN)+ndofGD+ndofGN,2*cp1(end));
%  incorporating Neumann-data in M1
M1(2*ndofD+1:ndofN+2*ndofD, dofN) = eye(ndofN);
M1(ndofN+2*ndofD+1: 2*(ndofN+ndofD), dofN+cp0(end)) = ...
       eye(ndofN);
%  incorporating Dirichlet-data in M2
M2(1:ndofD,dofD) = eye(ndofD);
M2(ndofD+1: 2*ndofD, dofD+cp1(end)) = eye(ndofD);

%  incorporate Gliding data
tmp0 = []; ctmp0 = []; tmp1 = []; ctmp1 = [];
for k=1:length(freenodesG)
  [idx,~]=find(gliding == freenodesG(k));
  tmp1(k)= n(idx(1),1) * uDh(freenodesG(k),1)+ ...
           n(idx(1),2) * uDh(freenodesG(k),2);
  ctmp1 = [ctmp1; n(idx(1),:)];
end
tmp1=tmp1(:);
for k=1:size(gliding,1)
  %  dof on k-th gliding boundary with respect to S1
  idx1 = [cp1(nED+nEN+k):cp1(nED+nEN+k+1)-1]+1;
  %  dof on k-th gliding boundary with respect to S0
  idx0 = cp0(nED+nEN+k-1)+1:cp0(nED+nEN+k);
  %  build RHS
  tmp0 = [tmp0;  m(k,1) * phiNh(idx0,1) + ...
                 m(k,2) * phiNh(idx0,2)];
  tmp1 = [tmp1;  n(k,1) * uDh(idx1,1)+ ...
                 n(k,2) * uDh(idx1,2)];
  ctmp0 = [ctmp0;repmat(m(k,:),length(idx0),1)];
  ctmp1 = [ctmp1;repmat(n(k,:),length(idx1),1)];
end

M1(end-ndofGN+1:end,dofGN          ) = diag(ctmp0(:,1));
M1(end-ndofGN+1:end,dofGN+cp0(end)) = diag(ctmp0(:,2));

M2(2*(ndofD+ndofN)+1:2*(ndofD+ndofN)+ndofGD,dofGD          )...
     = diag(ctmp1(:,1));
M2(2*(ndofD+ndofN)+1:2*(ndofD+ndofN)+ndofGD,dofGD+cp1(end))...
     = diag(ctmp1(:,2));

%*** Left-hand-side matrix
S = [V, -0.5*M-K, M1'; -0.5*M'+K', W, M2'; ...
     M1, M2 , zeros(size(M1,1))];
%*** build right hand side vector
b=[zeros(2*(cp0(end)+cp1(end)),1);tmpu(:);...
   tmpphi(:);tmp1;tmp0];
%*** Computation of approximation
tic
x=S\b(:);
disp(['Time for Solve: ',num2str(toc)]);

%***extract solution
phih = reshape(x(1:2*cp0(end)),cp0(end),2);
uh = reshape(x(2*cp0(end)+1:2*(cp0(end)+cp1(end))),cp1(end),2)
```

Appendix D
Content of the CD

The CD contains the epsBEM software package (version 2.0, August 2012). We first give an overview on the structure of the software package before we explain how to run the main routines.
The main directory consists of four subdirectories which we describe briefly. For a detailed explanation of the software package see [10].

- **examples**: This folder contains the main routines and subdirectories that contain the geometries and the function handles for the different problems of the Laplace and the Navier-Lamé equations. Furthermore, a graphical user interface is provided.

- **lib**: This directory is divided into three subdirectories. In the **lame** and the **laplace** folder the (compiled) C- and the MATLAB-functions for the assembly of the Galerkin matrices for the corresponding PDEs can be found. All other MATLAB-functions that are listed in the overview on the epsBEM package in Figure 6.1 can be found in the folder **general**.

- **src**: This folder contains the C- and the mex-files of the C-libraries for calculating the integrals and assembling the Galerkin matrices.

In following we give a short instruction how to run the main routines.

- First, the C-libraries have to be compiled with the make-file **make.m** that is located in the root directory. For compiling the parallelized routines for the assembly of the Galerkin matrices the option **p** has to be used. However, this options can only be used while working on a Linux or Unix system. Furthermore, the multi-precision libraries **mpfr** and **mpc** have to be installed.

- The main routines for the Dirichlet, the Neumann and mixed problem with and without gliding conditions can be found in the folder **examples**. In the beginning of each example file the user can chose the PDE, the problem, the geometry, the Lamé coefficients as well as the refinement parameters for the adaptive algorithms. The initial values are set in the way that the examples, which are treated in Chapter 7, are solved. The comment in the subsequent row denotes the alternative settings for all examples, that are implemented for the Navier-Lamé equation.

Bibliography

[1] J. Alberty, C. Carstensen, S.A. Funken, and R. Klose. *Matlab Implementation of the Finite Element Method in Elasticity*. Computing, 69(3), November 2002.

[2] A. Bantle. *Efficient Implementation of the Collocation Method for the Navier-Lame Equation*. June 2010.

[3] M. Bantle. *Efficient Implementation of Collocation and Boundary Element Methods for Integral Equations*. December 2010.

[4] C. Carstensen, G. Dolzmann, S.A. Funken, and D.S. Helm. *Locking-Free Adaptive Mixed Finite Elements Methods in Linear Elasticity*. April 2012.

[5] C. Castensen, S.A. Funken, and E.P. Stephan. *On the Adaptive Coupling of FEM and BEM in 2-d-Elasticity*. July 1997.

[6] Andreas Enge, Mickaël Gastineau, Philippe Théveny, and Paul Zimmermann. *mpc — A library for multiprecision complex arithmetic with exact rounding*, July 2012. http://mpc.multiprecision.org/.

[7] C. Erath, S.A. Funken, P. Goldenits, and D. Praetorius. *Simple Error Estimators for the Galerkin BEM for some Hypersingular Integral Equation in 2D*, 2009.

[8] S. Ferraz-Leite and D. Praetorius. *Simple A Posteriori Error Estimators for the h-Version of the Boundary Element Method*. Applicable Analysis: An International Journal, 61, 2002.

[9] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. *MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding*, June 2007.

[10] S.A. Funken and M. Bantle. *epsBEM: efficient and p-stable Boundary Element Methods, a* MATLAB *software package*. Work in progress.

[11] P.F. Heiter. *Stable Implementation of Three-Term Recurrence Relations*. June 2010.

[12] E.W. Hobson. *The Theory of Spherical and Ellipsoidal Harmonics*. Chelsea Publishing Company, 1965.

[13] T.M. Macrobert. *Spherical Harmonics: An Elementary Treatise on Harmonic Functions with Applications*. Pergamont Press Ltd., 1967.

[14] G. Of. *BETI-Gebietszerlegungsmethoden mit schnellen Randelementverfahren und Anwendungen*. April 2012.

[15] S. Sauter and C. Schwab. *Randelementmethoden: Analyse, Numerik und Implementierung schneller Algorithmen.* Teubner, 2004.

[16] O. Steinbach. *Numerische Näherungsverfahren für elliptische Randwertprobleme.* Vieweg+Teubner, 2010.

[17] E.P. Stephan. *The h-p Boundary Element Method for Solving 2- and 3-Dimensional Problems.* Computer methods in applied mechanics and engineering, 69(3), November 2002.

[18] J. Thiele. *A Posteriori Finite Element Analysis für die lineare Elastizitätstheorie.* March 2003.

[19] H. Weidle. *Rekursionen zur stabilen Berechnung von speziellen Integralen bei der Randelementmethode.* October 2010.

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben wird.

Ulm, den 29.08.2012 _____

(Unterschrift)