A Boundary Element Method for Eddy Currents

A Master's Thesis on the Magnetic Field Method for the Calculation of Eddy Currents on Surfaces with Complicated Geometries

Oded Stein, ETH Zürich

Supervised by:
Prof. Dr. Ralf Hiptmair, ETH Zürich

24. July, 2015

Abstract

This master's thesis explores a method to calculate the magnetic field generated by eddy currents. It was introduced in a previous work by R. Hiptmair. The method is valid for eddy currents generated by quasi-stationary exciting magnetic fields.

Inside the conductor a normal vectorial Maxwell problem is solved using vectorial boundary element methods. Outside the conductor, for domains with trivial first cohomology group, a scalar potential approach is used. For more complicated domains that do not have trivial first cohomology groups a more intricate special scalar potential with certain discontinuities is used. This allows to apply scalar boundary element methods on the outside of the conductor. For this case, new Calderon identities for the scalar problem are introduced. On the boundary of the conductor itself the equations are coupled to solve the full eddy current problem.

The method is tested for various geometries and compared to exact solutions where available.

If the domain outside the conductor does not have a trivial first cohomology group the method as constructed is dependent on the construction of cutting surfaces. This thesis manages to remove this dependence and reduce computations to the boundary of the cutting surfaces only. These boundaries correspond to homology cycles.

To remedy numerical instabilities caused by operations with discontinuous functions a suitable preprocessing is developed and introduced. The preprocessing manages to remedy some of these problems and simplify the system of linear equations that has to be solved.

Acknowledgements

I would like to thank Prof. Ralf Hiptmair of ETH Zurich for supervising me and helping me with all my problems during the writing process. Prof. Hiptmair wrote the paper which introduced the magnetic field based method and was the basis for this thesis.

I would also like to thank Dr. Lars Kielhorn and Elke Spindler of ETH Zurich who assisted me with BETL2 and all sorts of technical infrastructure without which all the calculations of this thesis would not have been possible. I also thank Nicolaus Heuer who helped me with the idea for the proof in Chapter 9.

I am grateful for the ETH Foundation's Excellence Scholarship and Opportunity Award which financed my master's degree studies and made this thesis possible.

Lastly, I would like to thank my family and friends who supported me throughout my studies at ETH Zurich and during the writing of this thesis (and who helped me with grammar and spelling).

Contents

1 Introduction

Eddy currents are currents induced in an electric conductor when it is moved through a magnetic field (ref. [12, p.298]). The eddy currents create magnetic fields which interact with the magnetic field present and often significantly change the dynamics of the system. As [12, p.298] states, "eddy currents are notoriously difficult to calculate." Nevertheless they are of importance as metals moving in magnetic fields are abundant in engineering applications.

This master's thesis explores a boundary element method to calculate eddy currents on the surfaces of conductors moving slowly through exciting magnetic fields. These magnetic fields are generated by closed electric currents (such as in a coil or a circular wire). This work deals with the motivation of the model, the derivation of the boundary element method, the implementation of the boundary element method and possible simplifications of it.

In published literature one can find multiple methods for calculating these eddy currents, such as a boundary element method based on the electric field (mentioned in [14] and [15]) or a mixed BEM-FEM approach as mentioned in [25].
In [14] Ralf Hiptmair derives multiple boundary element methods for the calculation of eddy currents and explores two of them in detail: one is based on formulating the problem with electric fields and the other is based on formulating the problem with magnetic fields. The magnetic field method from [14] relies on a scalar potential formulation of the magnetic field outside the conductor. This formulation allows the use of scalar boundary element methods outside the conductor but places some requirements on the geometry of the conductor. These geometrical constraints can lead to a complicated formulation. This thesis builds on [14] and explores the method based on magnetic fields formulated there. An attempt is made to simplify some of the complications stemming from the geometry of the problem. [14] has an error in formulating the Calderon identities for the case of objects with complicated geometries. This error is fixed in this thesis.

In Chapter 2 the eddy current model is introduced. This model is a simplification of the Maxwell equations under specific assumptions which are discussed there. The chapter also introduces standard notation for the electric and magnetic fields and their respective potentials.

Chapter 3 deals with the functional analysis framework in which the calculations of this thesis happen. It introduces the proper definition and setup of the function spaces and the traces for the boundary. Function spaces and trace spaces for the magnetic fields and the potentials are introduced and justified.

The basic theory for the boundary element method is introduced in Chapter 4. It consists of standard theory developed elsewhere that forms the basis for most boundary element methods. Also, a standard notation for all operators is introduced.

Chapter 5 then deals with the application of boundary element theory to construct boundary integral equations for the case where the conductor has a simple geometry. The method consists of coupling boundary integral equations on the inside and on the outside of the conductor. Simple geometry means that the complement of the conductor has trivial first cohomology group.

In Chapter 6 the boundary integral equations from Chapter 5 are discretized and a matrix equation is formulated to solve the problem computationally. An implementation is provided using the framework of BETL2 [21] for the case where the conductor is a sphere.

As the theory from Chapter 5 only holds for simple geometries, Chapter 7 introduces boundary integral equations that allow conductors with more general geometries. The corresponding boundary integral equations are derived; they differ substantially from the rather standard equations in Chapter 5.

Chapter 8 deals with the discretization and implementation of the equations derived in Chapter 5. First the new boundary integral equations are tested by themselves, then the coupled eddy current problem itself is solved.

The formulation in Chapter 7 and Chapter 8 includes cutting surfaces. These are special surfaces that are needed for the method, however the method is to some degree independent of the specific choice of cutting surfaces. In fact, the method can be reduced to just the boundary of these cutting surfaces. This is discussed in Chapter 9.

Chapter 10 addresses another issue that remains from Chapter 8: The formulation contains operations with discontinuous functions that are susceptible to numerical cancellation errors. These operations are prevented where possible with a suitable preprocessing method. The preprocessing also manages to simplify the whole equation for the symmetrically coupled eddy current problem.

Finally, Chapter 11 summarizes the thesis and gives an outlook on possible future work.

The appendix (Chapter 12) contains code snippets that show how the methods applied throughout the thesis are actually implemented using the BETL2 framework [21]. The snippets are referenced throughout the thesis but moved to the appendix for readability purposes.

2 Eddy current model problem

As stated in the introduction, eddy currents are caused by time-varying magnetic fields in conductors. In this model the magnetic field will be caused by an exciting current $\mathbf{j}_s$. The conductor Ω_c, the interior of its complement Ω_e and the boundary between them form the domain.

2.1 Domain

The domain consists of the conductor, a bounded open Ω_c in $\mathbb{R}^3$, the interior of the complement of the conductor, Ω_e, which is empty space and the boundary between the conductor and the air which is called Γ. A sketch of the situation for a model problem (the simple geometry, which will be discussed in Chapters 5 and 6) can be seen in Figure 2.1.
It is assumed that the conductor is simple, linear, homogeneous and isotropic with constant conductivity $\sigma > 0$ and constant permeability $\mu_c > 0$ (these approximations are from the model used in [14] which introduces them in Section 2). It is also assumed that the exciting current $\mathbf{j}_s$ is compactly supported outside the conductor.
Additionally it is assumed that the surface of Ω_c, Γ, is a smooth closed manifold. In practice it will consist of piecewise polygons (which may be curved), as the input for the numerical method will be a mesh of triangles.

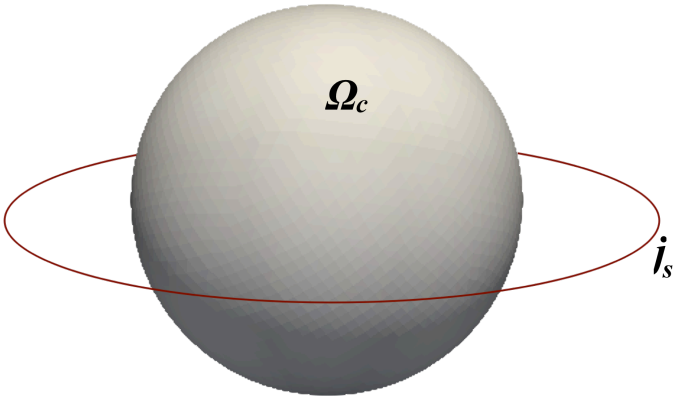

Figure 2.1: Sketch of the domain; the conductor Ω_c in gray and the exciting current $\mathbf{j}_s$ in red

2.2 Maxwell equations

The eddy current model is based on the Maxwell equations. The basic Maxwell equations (this version is taken from [12, p. 330]) are:

$$\begin{aligned} \operatorname{div} \mathbf{D} &= \rho_f \\ \operatorname{div} \mathbf{B} &= 0 \\ \mathbf{curl}\, \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \mathbf{curl}\, \mathbf{H} &= \mathbf{j}_f + \frac{\partial \mathbf{D}}{\partial t} \end{aligned} \tag{2.1}$$

$\mathbf{H}$ is understood to be $\frac{1}{\mu}\mathbf{B}$ (because of the linear material, ref. [12, p. 382]). $\mathbf{D}$ is understood to be $\varepsilon\mathbf{E}$. ρ_f and $\mathbf{j}_f$ are free charge and free current respectively. The conductivity of the material is denoted by σ.
From now on free charge will be assumed to be 0.

This version of the Maxwell equations is just one version; there are other versions that may differ by scaling (ref. [12, p. 326]). However, all describe the same physics.

In frequency domain (after a Fourier transform), the Maxwell equations are:

$$\begin{aligned} \operatorname{div} \mathbf{D} &= 0 \\ \operatorname{div} \mathbf{B} &= 0 \\ \mathbf{curl}\, \mathbf{E} &= -i\omega \mathbf{B} = -i\omega\mu \mathbf{H} \\ \mathbf{curl}\, \mathbf{H} &= \mathbf{j}_f + i\omega \mathbf{D} \end{aligned} \tag{2.2}$$

where ω is the frequency.

2.3 Approximations

Some simplifications are made to the Maxwell equations in the eddy current model. It is assumed that the field is quasi-stationary without displacement current (this is justified in e.g. [7, p. 151], [14, Chapter 2]).
This is a good approximation if $\omega\varepsilon_0 \ll \sigma$ (where ε_0 is the permittivity of vacuum). As [14, Chapter 1] states, this means that the timescale is long enough so that space charges do not need to be taken into account (this is the quasi-stationary part). This generally holds "for good conductors and frequencies which are not too high" [7, p. 151].
It is also necessary that the field is slowly varying, or as [7, (30)] puts it, $\frac{\omega L}{c} \ll 1$ with c the speed of light and L the maximum distance that can occur between two points in the conductor (this is the same requirement as [14, (1)]).

Omitting the displacement current turns (2.2) into:

$$\begin{aligned} \operatorname{div} \mathbf{D} &= 0 \\ \operatorname{div} \mathbf{B} &= 0 \\ \mathbf{curl}\, \mathbf{E} &= -i\omega\mu \mathbf{H} \\ \mathbf{curl}\, \mathbf{H} &= \mathbf{j}_f \end{aligned} \tag{2.3}$$

Outside the conductor the free current is just the exciting current $\mathbf{j}_s$. Inside the conductor (where there is conductivity) Ohm's law gives $\mathbf{j}_f = \sigma\mathbf{E}$ (ref. [12, p. 285]). Ohm's law is a simplification of the Lorentz force law if the velocity of the charges is so small that it can be disregarded (which is assumed here).

The final equations governing the model are thus:

$$\begin{aligned} \operatorname{div} \mathbf{E} &= 0 \\ \operatorname{div} \mathbf{B} &= 0 \\ \mathbf{curl}\, \mathbf{E} &= -i\omega\mu \mathbf{H} \\ \mathbf{curl}\, \mathbf{H} &= \begin{cases} \mathbf{j}_s & \text{in } \Omega_e \\ \sigma\mathbf{E} & \text{in } \Omega_c \end{cases} \end{aligned} \tag{2.4}$$

This corresponds to the model used in [14, p. 216].

2.4 Potentials

As usual in electromagnetics it is possible to define potentials for the electric and magnetic fields. These are introduced here so that they can be used later on.
Similar to [12, pp. 416] the way to define the magnetic potential $\mathbf{A}$ is such that $\mathbf{B} = \mathbf{curl}\,\mathbf{A}$. The gauge is chosen such that $\operatorname{div}\mathbf{A} = 0$.
Again similar to [12, pp. 416] the way to define the electric potential V is such that $\mathbf{E} = -\,\mathbf{grad}\,V - i\omega\mathbf{A}$. As there are no free charges it is possible to set $V = 0$. This gives $\mathbf{E} = -i\omega\mathbf{A}$ and is called temporal gauge (ref. e.g. [19, p. 676]).

This is consistent with the potentials as defined in [14, p. 217].

2.5 Exciting field

The exciting current can be any sufficiently regular current as long as $\operatorname{div}\mathbf{j}_s = 0$ (which means the current is closed). The exciting current creates the two exciting fields $\mathbf{E}_s$ and $\mathbf{H}_s$. By Maxwell clearly $\operatorname{div}\mathbf{H}_s = 0$ and as there are no free electric charges $\operatorname{div}\mathbf{E}_s = 0$. (2.3) together with the definition of the magnetic field then gives:

$$\begin{aligned} \mathbf{curl}\,\mathbf{E}_s &= -i\omega\mu_0\mathbf{H}_s \\ \mathbf{curl}\,\mathbf{H}_s &= \mathbf{j}_s \end{aligned} \tag{2.5}$$

It is possible to conclude that:

$$\mathbf{H}_s = -\frac{1}{i\omega\mu_0}\,\mathbf{curl}\,\mathbf{E}_s \tag{2.6}$$

This is consistent with [14, p. 217].

2.6 Reaction field

In Ω_e it makes sense to introduce the so-called reaction fields (ref. [14, p. 217]) $\mathbf{E}_r := \mathbf{E} - \mathbf{E}_s$ and $\mathbf{H}_r := \mathbf{H} - \mathbf{H}_s$. (2.5) then gives:

$$\begin{aligned} \mathbf{curl}\,\mathbf{H}_r &= 0 \qquad \text{in } \Omega_e \\ \mathbf{curl}\,\mathbf{curl}\,\mathbf{E}_r &= 0 \qquad \text{in } \Omega_e \end{aligned} \tag{2.7}$$

Thus outside the conductor (Ω_e) there is a double pair of fields: the *exciting fields* $\mathbf{H}_s$ and $\mathbf{E}_s$ and the *reaction fields* $\mathbf{H}_r$ and $\mathbf{E}_r$. Inside the conductor (Ω_c) the fields $\mathbf{H}$ and $\mathbf{E}$ will be called *total fields*, following the naming convention of [14].

2.7 Continuity along interfaces

There are multiple areas with different conductivity and permittivity in this model. Because of this, jumps of the fields along interfaces occur. Jumps across the surface Γ are written as $[\cdot]_\Gamma$.

For the magnetic field, [12, pp. 273] states (where $\perp$ denotes the perpendicular part of the field and $||$ denotes the parallel part of the field):

$$\begin{aligned} [\mathbf{B}^\perp]_\Gamma = 0 \quad \Rightarrow \quad \mu_0 \mathbf{H}_s^\perp + \mu_0 \mathbf{H}_r^\perp = \mu_c \mathbf{H}^\perp \\ \mathbf{H}_s^{||} + \mathbf{H}_r^{||} = \mathbf{H}^{||} + \mathbf{K}_f \times \mathbf{n} = \mathbf{H}^{||} \end{aligned} \tag{2.8}$$

($\mathbf{K}_f$, the free surface current, is 0)

For the electric field, [12, pp. 178] states:

$$\begin{aligned} \mathbf{E}_s^\perp + \mathbf{E}_r^\perp = \mathbf{E}^\perp \\ \frac{1}{\mu_0}\mathbf{E}_s^{||} + \frac{1}{\mu_0}\mathbf{E}_r^{||} = \frac{1}{\mu_c}\mathbf{E}^{||} \end{aligned} \tag{2.9}$$

These jump conditions can also be formulated using the trace operators from Chapter 3. They will be repeated in that chapter with updated notation.

3 Function spaces

In this chapter the various function spaces needed for the correct formulation of boundary integral equations are introduced, as well as the traces associated with them. The spaces are formulated similarly to [14].

3.1 Spaces

Definition 3.1 *The space for the magnetic field is (ref. [14, p. 218]):*

$$\boldsymbol{H}(\mathbf{curl};\Omega) := \{\boldsymbol{\Phi} \in \boldsymbol{L}^2(\Omega);\ \mathbf{curl}\,\boldsymbol{\Phi} \in \boldsymbol{L}^2(\Omega)\} \tag{3.1}$$

This can be justified with the physics of magnetic fields. As e.g. [12, p. 319] states the magnetic energy is an integral of the vector potential and the current, $\mathbf{A} \cdot \mathbf{j}$ (times a constant). As the current is the curl of the magnetic field (times a constant), the field and its curl should be in $\boldsymbol{L}^2(\Omega)$ in order to ensure that the energy is finite.
One needs to be more diligent with formulating a space for the electric field, but as this thesis only discusses methods based on the magnetic field it can be ignored for now.

Additionally, certain scalar traces formulated in the following section will need the standard Sobolev-Hilbert spaces:

$$H^1(\Omega) := \{\varphi \in L^2(\Omega);\ D\varphi \in L^2(\Omega)\} \tag{3.2}$$

and the associated $H^0(\Omega)$, $H^{\frac{1}{2}}(\Omega)$, $H^{-\frac{1}{2}}(\Omega)$. Theory for these can be found in standard PDE texts such as [9].
It is also necessary to introduce the following space (ref. e.g. [14, p. 219]):

$$H^1(\triangle,\Omega) := \{\varphi \in H^1(\Omega);\ \triangle\varphi \in L^2(\Omega)\} \tag{3.3}$$

Definition 3.2 *The different traces will map into different function spaces on the boundary. These are:*

$$\begin{aligned} \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma,\Gamma) &:= \{\boldsymbol{\Phi} \in \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\Gamma);\ \mathbf{curl}_\Gamma\boldsymbol{\Phi} \in \boldsymbol{H}^{-\frac{1}{2}}(\Gamma)\} \\ \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma,\Gamma) &:= \{\boldsymbol{\Phi} \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\Gamma);\ \mathrm{div}_\Gamma\boldsymbol{\Phi} \in H^{-\frac{1}{2}}(\Gamma)\} \end{aligned} \tag{3.4}$$

which are based on the Sobolev-Hilbert spaces $\boldsymbol{H}_{||}^{-\frac{1}{2}}(\Gamma)$ (functions with tangential continuity) and $\boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\Gamma)$ (functions with normal continuity) on the boundary (ref. [14, pp. 218]). According to e.g. [14, p. 219] or [4, Part II] the second space is the dual of the first with respect to the usual $\boldsymbol{L}^2(\Omega)$ duality pairing.

3.2 Traces

Definition 3.3 *The following trace is the Dirichlet-like trace for the space* $\boldsymbol{H}(\mathbf{curl};\Omega)$*:*

$$\gamma_{\mathbf{t}} : \boldsymbol{H}(\mathbf{curl};\Omega) \to \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_{\Gamma},\Gamma) \qquad \gamma_{\mathbf{t}}\boldsymbol{\Phi} := \mathbf{n}\times(\boldsymbol{\Phi}\times\mathbf{n})\big|_{\Gamma} \tag{3.5}$$

The definition on the right only holds for smooth functions, but is easily extended to all Sobolev functions in the respective space. This trace is called Dirichlet or tangential trace.

The vector $\mathbf{n}$ is understood as the *outward* normal vector of Ω_c. The definition coincides with the one from [14, p. 219].

Definition 3.4 *The following two traces are Neumann-like traces for the vectorial spaces:*

$$\begin{aligned} \gamma_N : \boldsymbol{H}(\mathbf{curl}\,\mathbf{curl},\Omega) \to \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_{\Gamma},\Gamma) \qquad & \gamma_N\boldsymbol{\Phi} := \mathbf{curl}\,\boldsymbol{\Phi}\times\mathbf{n}\big|_{\Gamma} \\ \gamma_{\mathbf{n}} : \boldsymbol{H}(\mathrm{div};\Omega) \to H^{-\frac{1}{2}}(\Gamma) \qquad & \gamma_{\mathbf{n}}\boldsymbol{\Phi} := \mathbf{n}\cdot\boldsymbol{\Phi}\big|_{\Gamma} \end{aligned} \tag{3.6}$$

The definitions on the right again hold for smooth functions, but are easily extended to all Sobolev functions in the respective space. They will be called Neumann and normal trace.

For the Neumann trace, [14] uses a Beppo-Levi-type space. As this thesis will not deal with the electric fields that live in this space, the normal curl space will suffice. The definitions coincide with [14, p. 220].

For non-vectorial functions there are the standard Dirichlet and Neumann traces (as for example used in [27]). They will be denoted by:

$$\begin{aligned} \gamma : H^1(\Omega) \to H^{\frac{1}{2}}(\Gamma) \\ \partial_{\mathbf{n}} : H^1(\triangle,\Omega) \to H^{-\frac{1}{2}}(\Gamma) \end{aligned} \tag{3.7}$$

According to [14, Chapter 3] these are the correct trace spaces.

With the notation for trace operators the jump conditions for the magnetic field formulated in (2.8) are:

$$\begin{aligned} \mu_0\gamma_{\mathbf{n}}^e\mathbf{H}_s + \mu_0\gamma_{\mathbf{n}}^e\mathbf{H}_r = \mu_c\gamma_{\mathbf{n}}^c\mathbf{H} \\ \gamma_{\mathbf{t}}^e\mathbf{H}_s + \gamma_{\mathbf{t}}^e\mathbf{H}_r = \gamma_{\mathbf{t}}^c\mathbf{H} \end{aligned} \tag{3.8}$$

where the superscript ^{e} denotes that the trace is taken from the outside and the superscript ^{c} denotes that the trace is taken from the inside of the conductor.

4 Boundary integral operators

In this chapter the boundary integral operators and their Calderon identities are introduced. These operators are later used to formulate boundary integral equations and solve the eddy current problem.
On the outside of the conductor, where there is no conductivity, the problem will be solved like a scalar potential problem. The theory for this is well-established and will be based on [29] in this thesis with notation from [14] for compatibility.
On the inside of the conductor the situation is more complicated. A vector-valued **curl curl** problem has to be solved. Theory for this will be taken from [5], with notation from [14].

4.1 Operators for scalar-valued functions

[29, p. 111] gives the following fundamental solution to the Laplace equation in 3D:

$$G_0(x,y) := \frac{1}{4\pi}\frac{1}{|x-y|} \tag{4.1}$$

With this fundamental solution the boundary integral operators can be defined:

Definition 4.1 *The* Laplace single layer operator V^0*, the* double *and* adjoint double layer operators K^0*,* $K^{0,*}$ *and the* hypersingular operator D^0 *are defined (ref. e.g. [29, pp. 118], [14, p. 225]) as following:*

$$\begin{aligned}
V^0 &:= \gamma^c \Psi_V^0 : H^{-\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma), \\
&\qquad (\Psi_V^0 \varphi)(x) := \int_\Gamma G_0(x,y)\varphi(y)dS(y),\ x \notin \Gamma \\
K^{0,*} &:= \frac{1}{2}(\partial_{\mathbf{n}}^c + \partial_{\mathbf{n}}^e)\Psi_V^0 : H^{-\frac{1}{2}}(\Gamma) \to H^{-\frac{1}{2}}(\Gamma) \\
K^0 &:= \frac{1}{2}(\gamma^c + \gamma^e)\Psi_K^0 : H^{\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma), \\
&\qquad (\Psi_K^0 \varphi)(x) := \int_\Gamma (\partial_{\mathbf{n}_y}^c G_0(x,y))\varphi(y)dS(y),\ x \notin \Gamma \\
D^0 &:= -\partial_{\mathbf{n}}^c \Psi_K^0
\end{aligned} \tag{4.2}$$

where the superscript ^{c} *means approaching from inside the conductor and the superscript* ^{e} *approaching from outside the conductor.* Ψ_V^0 *and* Ψ_K^0 *are called single and double layer potential respectively.*

The adjoint double layer operator can easily be expressed using the double layer operator as one is the adjoint of the other in the duality pairing of $H^{-\frac{1}{2}}(\Gamma)$ and $H^{\frac{1}{2}}(\Gamma)$ (ref. [29, p. 152]).
[29, Theorem 6.17] gives a more convenient representation for the hypersingular operator on closed surfaces with continuous functions:

$$\langle Du, v\rangle_\Gamma = \frac{1}{4\pi}\int_\Gamma\int_\Gamma \frac{\mathbf{curl}_\Gamma u(y)\cdot\mathbf{curl}_\Gamma v(x)}{|x-y|}dS(x)dS(y) \tag{4.3}$$

where $\langle\cdot,\cdot\rangle$ is the L^2 scalar product.

For Ψ_V^0 and Ψ_K^0, the following jump relations hold as of [29, pp. 118]:

$$\begin{aligned} [\gamma\Psi_V^0]_\Gamma &= 0 & [\partial_{\mathbf{n}}\Psi_V^0]_\Gamma &= -\operatorname{id} \\ [\gamma\Psi_K^0]_\Gamma &= \operatorname{id} & [\partial_{\mathbf{n}}\Psi_K^0]_\Gamma &= 0 \end{aligned} \tag{4.4}$$

These operators together with the jump relations and the representation formula [29, (6.1)] give the internal and external Calderon identity for the Laplace equation:

Lemma 4.2 *For $u \in H^1(\Gamma)$ solving the Laplace equation $-\triangle u = 0$ on $\Omega_c \cup \Omega_e$ it holds:*

$$\begin{pmatrix}\gamma^c u\\ \partial_{\mathbf{n}}^c u\end{pmatrix} = \begin{pmatrix}\frac{1}{2}\operatorname{id} - K^0 & V^0\\ D^0 & \frac{1}{2}\operatorname{id} + K^{0,*}\end{pmatrix}\begin{pmatrix}\gamma^c u\\ \partial_{\mathbf{n}}^c u\end{pmatrix} \tag{4.5}$$

$$\begin{pmatrix}\gamma^e u\\ \partial_{\mathbf{n}}^e u\end{pmatrix} = \begin{pmatrix}\frac{1}{2}\operatorname{id} + K^0 & -V^0\\ -D^0 & \frac{1}{2}\operatorname{id} - K^{0,*}\end{pmatrix}\begin{pmatrix}\gamma^e u\\ \partial_{\mathbf{n}}^e u\end{pmatrix} \tag{4.6}$$

Proof Ref. [29, p. 137] and [29, p. 182]. □

4.2 Operators for vector-valued functions

For the vectorial case, a representation formula is needed from which a vectorial Calderon identity is constructed. This formula is taken from [5, Chapter 4].
[5] works with the fundamental Helmholtz solution $E_k(x,y) = \frac{1}{4\pi|x-y|}e^{ik|x-y|}$ to solve a $\mathbf{curl}\,\mathbf{curl} - k^2$ type equation. As this thesis is concerned with solving a $\mathbf{curl}\,\mathbf{curl} + \kappa^2$ type equation, the fundamental solution is (from [14, p. 226] for compatibility):

$$G_\kappa(x,y) := \frac{1}{4\pi}\frac{e^{-\kappa|x-y|}}{|x-y|} \tag{4.7}$$

with $\kappa = -ik$, thus $\kappa^2 = -k^2$ and, as in [5] $\operatorname{Im}(k) > 0$ is required, $\operatorname{Im}(-\kappa/i) = \operatorname{Im}(i\kappa) = \operatorname{Re}(\kappa) > 0$.

The vectorial single layer potential is defined as in [5, p. 15]:

$$(\mathbf{\Psi}_V^\kappa \mathbf{\Phi})(x) := \int_\Gamma G_\kappa(x,y)\mathbf{\Phi}(y)dS(y) \tag{4.8}$$

The Maxwell single and double layer potentials are defined as in [5, p. 15] (for $\kappa \neq 0$ only):[1]

$$\begin{aligned} \mathbf{\Psi}_A^\kappa \mathbf{\Phi} &:= \mathbf{\Psi}_V^\kappa(\mathbf{\Phi}) - \frac{1}{\kappa^2}\,\mathbf{grad}\,\Psi_V^\kappa(\mathrm{div}_\Gamma \mathbf{\Phi}) \\ \mathbf{\Psi}_C^\kappa \mathbf{\Phi} &:= \mathbf{curl}\,\mathbf{\Psi}_V^\kappa(\mathbf{R}\mathbf{\Phi}) \end{aligned} \tag{4.9}$$

Ψ_V^κ is the scalar single layer potential with Helmholtz instead of Laplace fundamental solution. $\mathbf{R}$ denotes the rotation around the outward normal vector by $\frac{\pi}{2}$. It is needed in the formulation because [5] uses a rotated tangential trace instead of the normal tangential trace (ref. [14, p. 228]).

For $\mathbf{\Psi}_A^\kappa$ and $\mathbf{\Psi}_C^\kappa$, the following jump relations hold as of [5, Theorem 7]:

$$\begin{aligned} [\gamma_\mathbf{t}\mathbf{\Psi}_A^\kappa]_\Gamma &= 0 & [\gamma_N\mathbf{\Psi}_A^\kappa]_\Gamma &= -\,\mathrm{id} \\ [\gamma_\mathbf{t}\mathbf{\Psi}_C^\kappa]_\Gamma &= -\,\mathrm{id} & [\gamma_N\mathbf{\Psi}_C^\kappa]_\Gamma &= 0 \end{aligned} \tag{4.10}$$

With these operators the Stratton-Chu representation formula [5, p. 16] holds:

$$\mathbf{U} = \mathbf{\Psi}_C^\kappa \gamma_\mathbf{t}^c \mathbf{U} + \mathbf{\Psi}_A^\kappa \gamma_N^c \mathbf{U} \tag{4.11}$$

for a $\mathbf{U}$ such that $\mathbf{curl}\,\mathbf{curl}\,\mathbf{U} + \kappa^2\mathbf{U} = 0$.

Taking the Dirichlet trace from inside yields (ref. [14, (36)] and [5, Chapter 5]):

$$\gamma_\mathbf{t}^c\mathbf{U} = \gamma_\mathbf{t}^c\mathbf{\Psi}_C^\kappa\gamma_\mathbf{t}^c\mathbf{U} + \gamma_\mathbf{t}^c\mathbf{\Psi}_A^\kappa\gamma_N^c\mathbf{U} = \left(\frac{1}{2}\,\mathbf{id} + \mathbf{C}^\kappa\right)\gamma_\mathbf{t}^c\mathbf{U} + \mathbf{A}^\kappa\gamma_N^c\mathbf{U} \tag{4.12}$$

Taking the Neumann trace from inside yields (ref. [14, (36)] and [5, Chapter 5]):

$$\gamma_N^c\mathbf{U} = \gamma_N^c\mathbf{\Psi}_C^\kappa\gamma_\mathbf{t}^c\mathbf{U} + \gamma_N^c\mathbf{\Psi}_A^\kappa\gamma_N^c\mathbf{U} = \mathbf{N}^\kappa\gamma_\mathbf{t}^c\mathbf{U} + \left(\frac{1}{2}\,\mathbf{id} + \mathbf{B}^\kappa\right)\gamma_N^c\mathbf{U} \tag{4.13}$$

where the following definitions were used:

Definition 4.3 *The* Maxwell single layer operator $\mathbf{A}^\kappa$*, the* double *and* adjoint double layer operators $\mathbf{C}^\kappa$*,* $\mathbf{B}^\kappa$ *and the* hypersingular operator $\mathbf{N}^\kappa$ *are defined as following (ref. [14, p. 229]):*

[1] Note that this is slightly different from the $\mathbf{\Psi}_{SL}^\kappa$ used in [5], as the Neumann trace there is the normal Neumann trace divided by $i\kappa$.

$$
\begin{aligned}
\mathbf{A}^\kappa &= \gamma_{\mathbf{t}}^c \boldsymbol{\Psi}_A^\kappa : \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma) \to \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \\
\mathbf{C}^\kappa &= \frac{1}{2}(\gamma_{\mathbf{t}}^c + \gamma_{\mathbf{t}}^e)\boldsymbol{\Psi}_C^\kappa : \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \to \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \\
\mathbf{B}^\kappa &= \frac{1}{2}(\gamma_N^c + \gamma_N^e)\boldsymbol{\Psi}_A^\kappa : \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \to \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \\
\mathbf{N}^\kappa &= \gamma_N^c \boldsymbol{\Psi}_C^\kappa : \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma) \to \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)
\end{aligned}
\tag{4.14}
$$

The adjoint double layer operator can be expressed using the double layer operator (ref. [14, Theorem 10]) as they are negative adjoints of each other with the respective duality pairing.
The hypersingular operator can be expressed using the vectorial single layer and single layer potentials and a rotation $\mathbf{R}$ by $\frac{\pi}{2}$ around the outward normal (ref. [14, (55)]):

$$
\begin{aligned}
\langle \mathbf{N}^\kappa \mathbf{u}, \mathbf{v} \rangle_{\boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma,\Gamma), \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma,\Gamma)} &= \kappa^2 \langle \gamma_{\mathbf{t}} \boldsymbol{\Psi}_V^\kappa \mathbf{R}\mathbf{u}, \mathbf{R}\mathbf{v} \rangle_{\boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma,\Gamma), \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma,\Gamma)} \\
&\quad + \langle \gamma \Psi_V^\kappa \operatorname{curl}_\Gamma \mathbf{u}, \operatorname{curl}_\Gamma \mathbf{v} \rangle_{H^{\frac{1}{2}}(\Gamma), H^{-\frac{1}{2}}(\Gamma)}
\end{aligned}
\tag{4.15}
$$

The calculations of (4.12) and (4.13) are summarized in the Maxwell Calderon identities (ref. [14, (36)]):

Lemma 4.4 *For* $\mathbf{U} \in \boldsymbol{H}(\mathbf{curl}; \Omega_c)$ *such that* $\mathbf{curl}\,\mathbf{curl}\,\mathbf{U} + \kappa^2 \mathbf{U} = 0$ *it holds:*

$$
\begin{pmatrix} \gamma_{\mathbf{t}}^c \mathbf{U} \\ \gamma_N^c \mathbf{U} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\,\mathbf{id} + \mathbf{C}^\kappa & \mathbf{A}^\kappa \\ \mathbf{N}^\kappa & \frac{1}{2}\,\mathbf{id} + \mathbf{B}^\kappa \end{pmatrix} \begin{pmatrix} \gamma_{\mathbf{t}}^c \mathbf{U} \\ \gamma_N^c \mathbf{U} \end{pmatrix} \tag{4.16}
$$

The Calderon identities together with the physics from Chapter 2 can now be used to solve the eddy current problem.

5 The method for simple geometries

The Calderon identities from Chapter 4 for the Laplace equation (outside) and the Maxwell equation (inside) can now be combined to form a boundary element method to solve the eddy current problem.

For this chapter it will be assumed that the geometry of Γ is simple, that means Ω_e has trivial first cohomology group. This will allow easy construction of scalar potentials in the outside region Ω_e.

5.1 Inside the conductor

Inside Ω_c, (2.4) states:

$$\begin{aligned} \mathbf{curl}\,\mathbf{E} &= -i\omega\mu_c\mathbf{H} \\ \mathbf{curl}\,\mathbf{H} &= \sigma\mathbf{E} \end{aligned} \tag{5.1}$$

This gives:

$$\begin{aligned} \mathbf{curl}\,\mathbf{curl}\,\mathbf{H} &= \sigma\,\mathbf{curl}\,\mathbf{E} = -i\omega\sigma\mu_c\mathbf{H} \\ \mathbf{curl}\,\mathbf{curl}\,\mathbf{H} + \kappa^2\mathbf{H} &= 0 \end{aligned} \tag{5.2}$$

with $\kappa = \sqrt{i\omega\sigma\mu_c}$. As the physical constants ω, σ, μ_c are all positive, $\operatorname{Re}\kappa > 0$.

This allows the application of the Calderon identity (4.16) to the interior total field $\mathbf{H}$ to obtain:

$$\begin{pmatrix} \gamma_{\mathbf{t}}^c\mathbf{H} \\ \gamma_N^c\mathbf{H} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\,\mathbf{id} + \mathbf{C}^\kappa & \mathbf{A}^\kappa \\ \mathbf{N}^\kappa & \frac{1}{2}\,\mathbf{id} + \mathbf{B}^\kappa \end{pmatrix} \begin{pmatrix} \gamma_{\mathbf{t}}^c\mathbf{H} \\ \gamma_N^c\mathbf{H} \end{pmatrix} \tag{5.3}$$

By (5.2), $\mathbf{H} = -\frac{1}{\kappa^2}\,\mathbf{curl}\,\mathbf{curl}\,\mathbf{H}$. Testing this with some $\mathbf{\Phi} \in \boldsymbol{H}(\mathbf{curl};\Omega)$ gives (with $\langle\cdot,\cdot\rangle_l$ the duality pairing between the div and **curl** spaces on the boundary):

$$\begin{aligned} \langle\mathbf{H},\mathbf{\Phi}\rangle_{\mathbf{L}^2(\Omega_c)} &= -\frac{1}{\kappa^2}\langle\mathbf{curl}\,\mathbf{curl}\,\mathbf{H},\mathbf{\Phi}\rangle_{\mathbf{L}^2(\Omega_c)} \\ &= -\frac{1}{\kappa^2}\left(\langle\mathbf{n}\times\mathbf{curl}\,\mathbf{H},\mathbf{\Phi}\rangle_l + \langle\mathbf{curl}\,\mathbf{H},\mathbf{curl}\,\mathbf{\Phi}\rangle_{\mathbf{L}^2(\Omega_c)}\right) \\ &= -\frac{1}{\kappa^2}\left(\langle\gamma_N^c\mathbf{H},\gamma_{\mathbf{t}}^c\mathbf{\Phi}\rangle_l + \langle\mathbf{curl}\,\mathbf{H},\mathbf{curl}\,\mathbf{\Phi}\rangle_{\mathbf{L}^2(\Omega_c)}\right) \end{aligned} \tag{5.4}$$

by integration by parts (integration by parts for curl follows from Gauss's divergence theorem as seen in [23, Chapter 12] and basic vector calculus identities as seen in [12, Back matter]).

5.2 Outside the conductor

In Ω_e, (2.4), (2.7) and the fact that $\operatorname{div}\mathbf{j}_s = 0$ give:

$$\begin{aligned} \operatorname{div}\mathbf{H}_r &= 0 \\ \mathbf{curl}\,\mathbf{H}_r &= 0 \end{aligned} \tag{5.5}$$

Now the assumption of trivial first cohomology group will be used:

Lemma 5.1 *Let $\mathbf{U} \in \boldsymbol{H}(\mathbf{curl}; M)$ where M is a smooth manifold with trivial first cohomology group. Then there is a potential function $u \in H^1(\triangle, M)$ such that $\mathbf{grad}\,u = \mathbf{U}$ as long as $\mathbf{curl}\,\mathbf{U} = 0$.*
The potential is not unique (ref. [14, Chapter 4]).

Proof If the first de Rham cohomology group of M is trivial, for every differential 1-form ω that is a cocycle (i.e. $d\omega = 0$) there is a function (a 0-form) u such that $du = \omega$.[1]

The de Rham theorem (ref. [3, p. 287]) states that de Rham and singular cohomology are isomorphic for smooth manifolds. This means the assumption of trivial first cohomology group suffices to have this property.

The curl in a differential geometric setting is associated with the exterior derivative of the corresponding covector 1-form (ref. [3, p. 269]), so if ω is the covector field corresponding to $\mathbf{U}$, it is a cocycle. The gradient in a differential geometric setting gives the vector field corresponding to the covector 1-form generated by the exterior derivative [3, p. 80]. This means $\mathbf{grad}\,u = \mathbf{U}$.

The potential is clearly not unique as any constant function can be added to it without changing its important properties. □

This carries over into the non-smooth setting sufficiently.
This means it is possible to introduce a potential function h_r such that:

$$\begin{aligned} \mathbf{grad}\,h_r &= \mathbf{H}_r \\ \triangle h_r &= \operatorname{div}\mathbf{H}_r = 0 \end{aligned} \tag{5.6}$$

This is the Laplace equation. This means the Calderon identity (4.6) applies:

$$\begin{pmatrix} \gamma^e h_r \\ \partial^e_{\mathbf{n}} h_r \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\operatorname{id} + K^0 & -V^0 \\ -D^0 & \frac{1}{2}\operatorname{id} - K^{0,*} \end{pmatrix} \begin{pmatrix} \gamma^e h_r \\ \partial^e_{\mathbf{n}} h_r \end{pmatrix} \tag{5.7}$$

[1] Terminology in this section is from [3] but should be standard.

By the discussion in Chapter 3 and Lemma 5.1, h_r should be in $H^1(\triangle, \Omega_e)$. It is possible to test this with the Dirichlet trace of some $\varphi \in H^1(\Omega_e)$ and apply integration by parts to obtain:

$$\begin{aligned} \langle \mathbf{grad}\, h_r, \mathbf{grad}\, \varphi \rangle_{\mathbf{L}^2(\Omega_e)} &= \langle \mathbf{n} \cdot \mathbf{grad}\, h_r, \varphi \rangle_d - \langle \triangle h_r, \varphi \rangle_{\mathbf{L}^2(\Omega_c)} \\ &= \langle \partial^e_{\mathbf{n}} h_r, \gamma^e \varphi \rangle_d \end{aligned} \tag{5.8}$$

where $\langle \cdot, \cdot \rangle_d$ denotes the duality pairing between the respective $-\frac{1}{2}$ and $+\frac{1}{2}$ Sobolev-Hilbert spaces on the boundary.

5.3 Combining inside and outside

At this point a decay condition is added that assumes that the magnetic field decays when going to infinity fast enough. This makes sense, as $\mathbf{H}$ should have finite energy. The two test functions $\mathbf{\Phi}$ in Ω_c and φ in Ω_e can be combined into a test function for all of $\mathbb{R}^3$, $\mathbf{V} \in \boldsymbol{H}(\mathbf{curl}; \mathbb{R}^3), \mathbf{curl}\, \mathbf{V}\big|_{\Omega_e} = 0$ by piecewise definition:

$$\begin{aligned} \mathbf{V} &:= \mathbf{\Phi} \quad \text{in } \Omega_c \\ \mathbf{V} &:= \mathbf{grad}\, \varphi \quad \text{in } \Omega_e \\ \gamma^c_{\mathbf{t}} \mathbf{\Phi} &= \gamma^e_{\mathbf{t}} \mathbf{grad}\, \varphi \end{aligned} \tag{5.9}$$

where the usual jump relations (2.8) must hold, but this time with no excitation field.

With the decay conditions, (2.4) and more integration by parts (ref. [14, (19)]):[2]

$$\begin{aligned} -i\omega\mu \langle \mathbf{H}, \mathbf{V} \rangle_{\mathbf{L}^2(\mathbb{R}^3)} &= \langle \mathbf{curl}\, \mathbf{E}, \mathbf{V} \rangle_{\mathbf{L}^2(\mathbb{R}^3)} = \langle \mathbf{E}, \mathbf{curl}\, \mathbf{V} \rangle_{\mathbf{L}^2(\mathbb{R}^3)} \\ &= \langle \mathbf{E}, \mathbf{curl}\, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} = \frac{1}{\sigma} \langle \mathbf{curl}\, \mathbf{H}, \mathbf{curl}\, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} \end{aligned} \tag{5.10}$$

To simplify notation, let $\mu_r = \mu_c/\mu_0$ and $\tau = \frac{1}{i\omega\mu_0\sigma} = \frac{\mu_r}{\kappa^2}$. Split into contributions of Ω_c and Ω_e this gives:

$$\begin{aligned} -\mu_r \langle \mathbf{H}, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} - \langle \mathbf{grad}\, h_r + \mathbf{H}_s, \mathbf{grad}\, \varphi \rangle_{\mathbf{L}^2(\Omega_e)} &= \tau \langle \mathbf{curl}\, \mathbf{H}, \mathbf{curl}\, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} \\ -\mu_r \langle \mathbf{H}, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} - \langle \mathbf{grad}\, h_r, \mathbf{grad}\, \varphi \rangle_{\mathbf{L}^2(\Omega_e)} - \langle \gamma^e_{\mathbf{n}} \mathbf{H}_s, \varphi \rangle_d &= \tau \langle \mathbf{curl}\, \mathbf{H}, \mathbf{curl}\, \mathbf{\Phi} \rangle_{\mathbf{L}^2(\Omega_c)} \end{aligned} \tag{5.11}$$

where in the last step integration by parts and $\operatorname{div} \mathbf{H}_s = 0$ are used.

Inserting (5.4) and (5.8) into above equation gives (ref. [14, (51)]):

$$\tau \langle \gamma^c_N \mathbf{H}, \gamma^c_{\mathbf{t}} \mathbf{\Phi} \rangle_l - \langle \partial^e_{\mathbf{n}} h_r, \gamma^e \varphi \rangle_d = \langle \gamma^e_{\mathbf{n}} \mathbf{H}_s, \gamma^e \varphi \rangle_d \tag{5.12}$$

[2] this derivation is also inspired by [2, Chapter 8]

Combining (5.3) and (5.7) with (5.12) and the tangential jump relations (2.8) gives (ref. [14, (54)]):

Theorem 5.2 *The full boundary integral equations needed to solve the eddy current problem with the exciting field* $\mathbf{H}_s$ *are:*

$$\begin{aligned}\tau \left\langle \mathbf{N}^\kappa \mathbf{grad}_\Gamma \gamma^e h_r, \mathbf{grad}_\Gamma \gamma^e \varphi \right\rangle_l + \left\langle D^0 \gamma^e h_r, \gamma^e \varphi \right\rangle_d + \tau \left\langle (0.5\,\mathbf{id} + \mathbf{B}^\kappa) \gamma_N^c \mathbf{H}, \mathbf{grad}_\Gamma \gamma^e \varphi \right\rangle_l \\ - \left\langle (0.5\,\mathrm{id} - K^{0,*}) \partial_{\mathbf{n}}^e h_r, \gamma^e \varphi \right\rangle_d = \left\langle \gamma_{\mathbf{n}}^e \mathbf{H}_s, \gamma^e \varphi \right\rangle_d - \tau \left\langle \mathbf{N}^\kappa \gamma_{\mathbf{t}}^e \mathbf{H}_s, \mathbf{grad}_\Gamma \gamma^e \varphi \right\rangle_l\end{aligned} \tag{5.13a}$$

$$\left\langle \gamma_N^c \mathbf{\Phi}, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\, \mathbf{grad}_\Gamma \gamma^e h_r \right\rangle_l + \left\langle \gamma_N^c \mathbf{\Phi}, \mathbf{A}^\kappa \gamma_N^c \mathbf{H} \right\rangle_l = \left\langle \gamma_N^c \mathbf{\Phi}, (0.5\,\mathbf{id} - \mathbf{C}^\kappa) \gamma_{\mathbf{t}}^e \mathbf{H}_s \right\rangle_l \tag{5.13b}$$

$$\left\langle \partial_{\mathbf{n}}^e \varphi, V^0 \partial_{\mathbf{n}}^e h_r \right\rangle_d + \left\langle \partial_{\mathbf{n}}^e \varphi, (0.5\,\mathrm{id} - K^0) \gamma^e h_r \right\rangle_d = 0 \tag{5.13c}$$

The operators are as defined in Chapter 4.
The trial/test spaces are:

$$\begin{aligned}\gamma^e h_r, \gamma^e \varphi &\in H^{\frac{1}{2}}(\Gamma) \\ \partial_{\mathbf{n}}^e h_r, \partial_{\mathbf{n}}^e \varphi &\in H^{-\frac{1}{2}}(\Gamma) \\ \gamma_N^c \mathbf{H}, \gamma_N^c \mathbf{\Phi} &\in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)\end{aligned}$$

Proof The first equation follows from the bottom line of (5.7), the bottom line of (5.3) and (5.12).
The second equation follows from the top line of (5.3).
The third equation follows from the top line of (5.7).

Throughout the calculations, $\gamma_{\mathbf{t}}^c \mathbf{H} = \mathbf{grad}_\Gamma \gamma^e h_r + \gamma_{\mathbf{t}}^e \mathbf{H}_s$ and $\gamma_{\mathbf{t}}^c \mathbf{\Phi} = \mathbf{grad}_\Gamma \gamma^e \varphi$ have to be used. □

6 Implementation of the method for simple geometries

This chapter is about the implementation of the boundary element method for simple geometries (i.e. the ones where the first cohomology group of Ω_e is trivial). First the discretization of the functional spaces is discussed, then the discretization of the boundary element operators and the boundary element equations. Subsequently the results are presented for the case where Γ is the 2-sphere and are compared with an analytical solution.

The framework used for the implementation that does all the discretization and setup of operators is BETL2 [21]. The actual construction of meshes is done with Gmsh [11] and the visualization is done with Paraview [22].

6.1 Discretization of spaces and operators in BETL2

BETL2 offers tools for the discretization of the boundary functional spaces with functions of order i with usually $i = 0, 1, 2$. The following is how the discretized BETL2-spaces will be called in this thesis:

Definition 6.1 $\tilde{\mathbf{H}}(0)$ *is the discretization of* $H^{-\frac{1}{2}}(\Gamma)$ *with piecewise continuous functions of order* 0*. It has dimension* n_0*.*
$\tilde{\mathbf{H}}(1)$ *is the discretization of* $H^{\frac{1}{2}}(\Gamma)$ *with continuous piecewise linear functions of order* 1*. It has dimension* n_1*.*
$\tilde{\mathbf{H}}(\mathrm{div})$ *is the discretization of* $\boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$ *with edge functions of order* 1*. It has dimension* n_{edge}*.*
$\tilde{\mathbf{H}}(\mathrm{curl})$ *is the discretization of* $\boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma)$ *with edge functions of order* 1*. It has dimension* n_{edge}*.*

These are all finite-dimensional vector spaces and the functions mapping between them are always understood as matrices. The way the spaces are actually implemented in BETL2 (using code) can be seen in the appendix in Section 12.1.

For all these spaces, BETL2 has the corresponding single and double layer operators on a trial space tested with a test space. These will be denoted by $\Lambda^\kappa_{SL}(\mathrm{test}, \mathrm{trial})$ and $\Lambda^\kappa_{DL}(\mathrm{test}, \mathrm{trial})$. The actual implementation of the operators in C++ code is discussed in the appendix in Section 12.5.

For $\mathbf{grad}_\Gamma$, $\mathbf{curl}_\Gamma$ and div_Γ, BETL2 offers the following discretizations respectively:

$$\begin{aligned} G_g &: \tilde{\mathbf{H}}(1) \to \tilde{\mathbf{H}}(\mathrm{curl}), \qquad && G_g \in \mathbb{C}^{n_{\mathrm{edge}} \times n_1} \\ G_c &: \tilde{\mathbf{H}}(1) \to \tilde{\mathbf{H}}(\mathrm{div}), \qquad && G_c \in \mathbb{C}^{n_{\mathrm{edge}} \times n_1} \\ G_d &: \tilde{\mathbf{H}}(\mathrm{div}) \to \tilde{\mathbf{H}}(0), \qquad && G_d \in \mathbb{C}^{n_0 \times n_{\mathrm{edge}}} \end{aligned} \tag{6.1}$$

These are matrices, their sizes correspond to the respective degrees of freedom of the element spaces' discretizations (where $\mathbb{C}^{n \times m}$ is the space of complex $n \times m$ matrices). The implementation of these operators in C++ code is discussed in the appendix in Section 12.2.

For purely notational purposes the rotation operator, which rotates around the outward unit normal by $\frac{\pi}{2}$ is introduced as R. In the implementation with BETL2 this is not actually needed, as BETL2 rotates automatically depending on which test and trial spaces are used so that the space $\tilde{\mathbf{H}}(\mathrm{curl})$ is always matched with its dual $\tilde{\mathbf{H}}(\mathrm{div})$.[1]

The scalar boundary integral operators from Chapter 4 are implemented as follows:

$$\begin{aligned} \tilde{V}^0 &= \Lambda^0_{SL}(\tilde{\mathbf{H}}(0), \tilde{\mathbf{H}}(0)), && \tilde{V}^0 \in \mathbb{C}^{n_0 \times n_0} \\ \tilde{K}^0 &= \Lambda^0_{DL}(\tilde{\mathbf{H}}(0), \tilde{\mathbf{H}}(1)), && \tilde{K}^0 \in \mathbb{C}^{n_0 \times n_1} \\ \tilde{K}^{0,*} &= (\tilde{K}^0)^T && \tilde{K}^{0,*} \in \mathbb{C}^{n_1 \times n_0} \\ \tilde{D}^0 &= G_c^T \Lambda^0_{SL}(\tilde{\mathbf{H}}(\mathrm{div}), \tilde{\mathbf{H}}(\mathrm{div})) G_c, && \tilde{D}^0 \in \mathbb{C}^{n_1 \times n_1} \end{aligned} \tag{6.2}$$

As before, these are matrices, their sizes correspond to the respective degrees of freedom of the element spaces' discretizations. The superscript ^{T} denotes the transpose of matrices.

The vectorial boundary integral operators are implemented as follows:

$$\begin{aligned} \tilde{A}^\kappa &= \Lambda^\kappa_{SL}(\tilde{\mathbf{H}}(\mathrm{div}), \tilde{\mathbf{H}}(\mathrm{div})) + \frac{1}{\kappa^2} G_d^T \Lambda^\kappa_{SL}(\tilde{\mathbf{H}}(0), \tilde{\mathbf{H}}(0)) G_d \\ \tilde{C}'^\kappa &= \Lambda^\kappa_{DL}(\tilde{\mathbf{H}}(\mathrm{div}), \tilde{\mathbf{H}}(\mathrm{curl})) \\ \tilde{B}^\kappa &= -(\tilde{C}'^\kappa)^T \\ \tilde{N}^\kappa &= \kappa^2 R^T \Lambda^\kappa_{SL}(\tilde{\mathbf{H}}(\mathrm{div}), \tilde{\mathbf{H}}(\mathrm{div})) R \\ & \qquad \tilde{A}^\kappa, \tilde{C}'^\kappa, \tilde{B}^\kappa, \tilde{N}^\kappa \in \mathbb{C}^{n_{\mathrm{edge}} \times n_{\mathrm{edge}}} \end{aligned} \tag{6.3}$$

As before, these are matrices, their sizes correspond to the respective degrees of freedom of the element spaces' discretizations. The implementation of $\mathbf{N}^\kappa$ lacks the curl_Γ parts from (4.15). This is irrelevant however: wherever $\mathbf{N}^\kappa$ will be used in the actual method it will always be composed with at least one G_g and $\mathrm{curl}_\Gamma \mathbf{grad}_\Gamma = 0$.

[1] In Section 12.2 it is mentioned that the gradient into the curl space is currently implemented incorrectly. This holds for the rotation as well: it happens in the wrong direction. This has to be fixed in code by simply adding a minus sign.

6.2 The discretized boundary integral equation

With all the discretized boundary integral operators defined as matrices, the discretized version of Theorem 5.2 is: Seek $u \in \tilde{\mathbf{H}}(1)$, $\eta \in \tilde{\mathbf{H}}(\mathrm{div})$ and $\psi \in \tilde{\mathbf{H}}(0)$ such that, for $\delta \in \tilde{\mathbf{H}}(\mathrm{curl})$ and $\nu \in \tilde{\mathbf{H}}(0)$:

$$\begin{pmatrix} \tau G_g^T \tilde{N}^\kappa G_g + \tilde{D}^0 & \tau G_g^T (0.5M + \tilde{B}^\kappa) & \tilde{K}^{0,*} - 0.5M \\ (\tilde{C}^\kappa - 0.5M) G_g & \tilde{A}^\kappa & 0 \\ 0.5M - \tilde{K}^0 & 0 & \tilde{V}^0 \end{pmatrix} \begin{pmatrix} u \\ \eta \\ \psi \end{pmatrix} = \begin{pmatrix} M\nu - \tau G_g^T \tilde{N}^\kappa \delta \\ (0.5M - C^\kappa)\delta \\ 0 \end{pmatrix} \tag{6.4}$$

where M is the respective mass matrix here with the appropriate test and trial space. The implementation of the mass matrices in C++ code is discussed in the appendix in Section 12.3.
In the notation of Chapter 5, u corresponds to $\gamma^e h_r$, η corresponds to $\gamma_N^c \mathbf{H}$ and ψ corresponds to $\partial_{\mathbf{n}}^e h_r$. The exciting field data δ corresponds to $\gamma_{\mathbf{t}}^e \mathbf{H}_s$, the exciting field data ν corresponds to $\gamma_{\mathbf{n}}^e \mathbf{H}_s$. The C++ code used for the incorporation of the boundary conditions is explained in the appendix in Section 12.6.

Using properties of the particular operator allows the simplification of the system to:

$$\begin{pmatrix} \mathbf{N} & -\mathbf{C}^T & -\mathbf{K}^T \\ \mathbf{C} & \tau \tilde{A}^\kappa & 0 \\ \mathbf{K} & 0 & \tilde{V}^0 \end{pmatrix} \begin{pmatrix} u \\ \eta \\ \psi \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \\ 0 \end{pmatrix} \tag{6.5}$$

where some notations have been simplified:

$$\begin{aligned} \mathbf{N} &:= \tau G_g^T \tilde{N}^\kappa G_g + \tilde{D}^0 \\ \mathbf{C} &:= \tau(\tilde{C}^\kappa - 0.5M) G_g \\ \mathbf{K} &:= 0.5M - \tilde{K}^0 \\ \mathbf{f} &= M\nu - \tau G_g^T \tilde{N}^\kappa \delta \\ \mathbf{g} &= \tau(0.5M - \tilde{C}^\kappa)\delta \end{aligned} \tag{6.6}$$

Because of the gauging freedom of scalar potentials there is no unique solution u to this system. For this to happen the additional constraint that $\gamma^e h_r$ integrates to 0 over Γ has to be incorporated. The way this is done is discussed in the appendix in Section 12.4.

6.3 Results

The equation (6.5) has been solved for a sphere with radius $a = 0.05$ sitting inside a simple wire coil of radius $b = 0.065$ on which a constant current $j_0 = 10^6$ is the exciting

number of bdry elements	rel. error u	rel. error η	rel. error ψ
128	0.335	0.232	0.234
512	0.163	0.111	0.0785
2048	0.080	0.052	0.022
8192	0.040	0.025	0.007

Table 6.1: $\mathbf{L}^2$ errors of the method described in this chapter. u is compared to the exact solution, ψ is compared to an interpolation of the exact solution in $\tilde{\mathbf{H}}(0)$ and η is not compared to the exact Neumann trace of the interior magnetic field, but to minus its conjugate.

current. The calculation of the corresponding magnetic field given this current can be found in [18, pp. 181]. The analytical solution to compare with is taken from [26]. Further parameters are $\omega = 2\pi \cdot 10^4$, $\sigma = 2 \cdot 10^6$, $\mu_r = 10$. The linear system was solved with Eigen's `partialPivLu` [8].

A picture of the method's results can be seen in Figure 6.1, a table with error analysis can be seen in Table 6.1. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8.

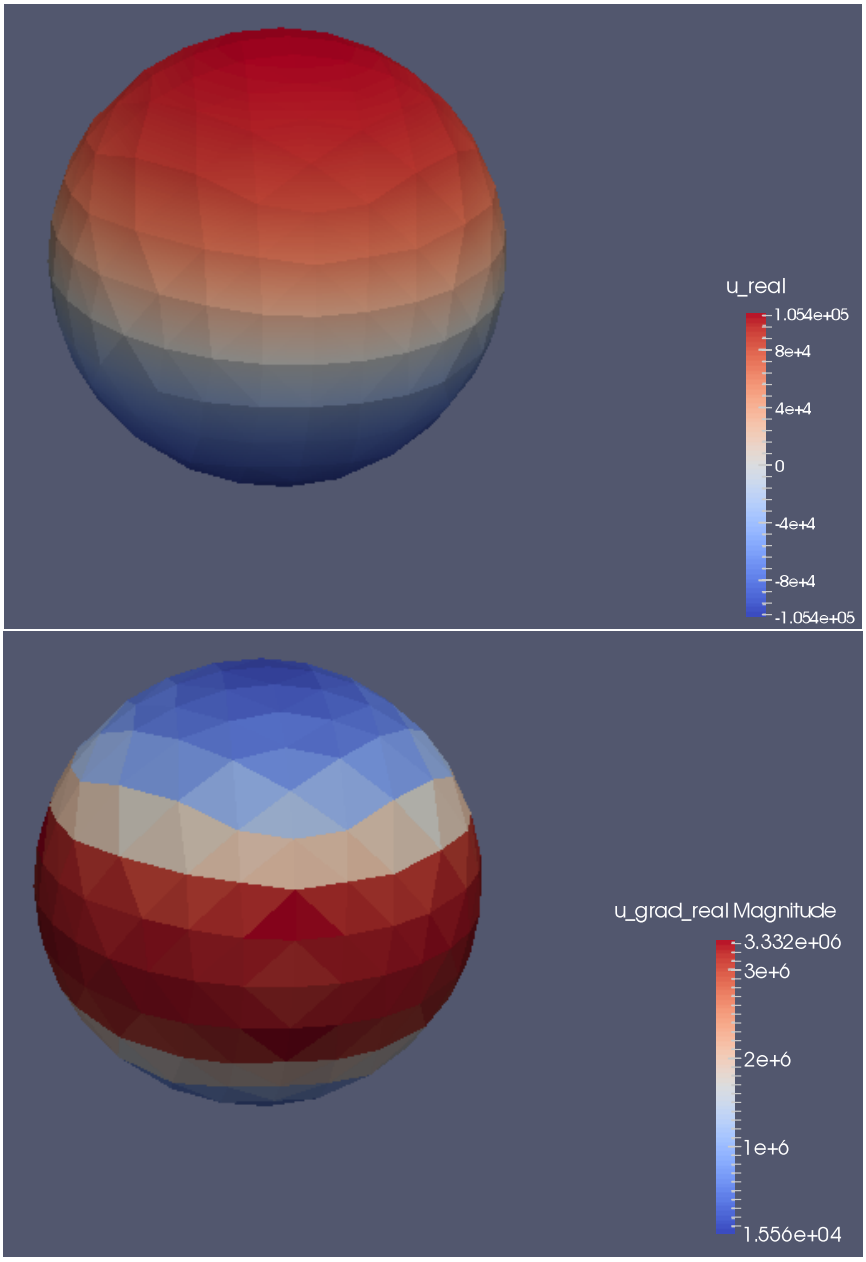

Figure 6.1: A visualization of the real parts of u and the real part of the magnitude of $\mathbf{grad}_\Gamma\, u$, approximations of $\gamma^e h_r$ and $\gamma_{\mathbf{t}}^e\, \mathbf{grad}\, h_r$ on a sphere with 512 triangular elements

7 Derivation of the method for complicated geometries

In Chapter 5 it was assumed that Ω_e has trivial first cohomology group in order to find a scalar potential for the outside Laplace problem. As this restriction excludes many interesting geometries it is necessary to find a method that also works if Ω_e has nontrivial first cohomology. This chapter discusses the reformulation of the method for such cases.

The chapter takes its inspiration from [14] but reformulates many equations.

7.1 Inside the conductor

Inside Ω_c the model remains the same as in Chapter 5 as the model does not change with a more complicated geometry. The standard vectorial Calderon identity (5.3) holds:

$$\begin{pmatrix} \gamma_{\mathbf{t}}^c \mathbf{H} \\ \gamma_N^c \mathbf{H} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\,\mathbf{id} + \mathbf{C}^\kappa & \mathbf{A}^\kappa \\ \mathbf{N}^\kappa & \frac{1}{2}\,\mathbf{id} + \mathbf{B}^\kappa \end{pmatrix} \begin{pmatrix} \gamma_{\mathbf{t}}^c \mathbf{H} \\ \gamma_N^c \mathbf{H} \end{pmatrix} \tag{7.1}$$

Weakly tested with $\mu \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$, the first line of (7.1) gives:

$$\langle \mu, \gamma_{\mathbf{t}}^c \mathbf{H} \rangle_\Gamma = \langle \mu, (\mathbf{C}^\kappa + 0.5\,\mathbf{id})\, \gamma_{\mathbf{t}}^c \mathbf{H} \rangle_\Gamma + \langle \mu, \mathbf{A}^\kappa \gamma_N^c \mathbf{H} \rangle_\Gamma \tag{7.2}$$

By the argumentation from Chapter 5 it makes sense to test the second line of (7.1) with $\mathbf{V} \in \boldsymbol{H}_{\perp}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma)$, $\mathbf{curl}_\Gamma \mathbf{V} = 0$:

$$\langle \mathbf{V}, \gamma_N^c \mathbf{H} \rangle = \langle \mathbf{V}, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \rangle_\Gamma + \langle \mathbf{V}, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^c \mathbf{H} \rangle_\Gamma \tag{7.3}$$

7.2 Outside the conductor

Outside the conductor, in Ω_e, the model is more complicated. The calculation from Chapter 5 used the assumption that Ω_e had trivial first cohomology. This assumption is now dropped.

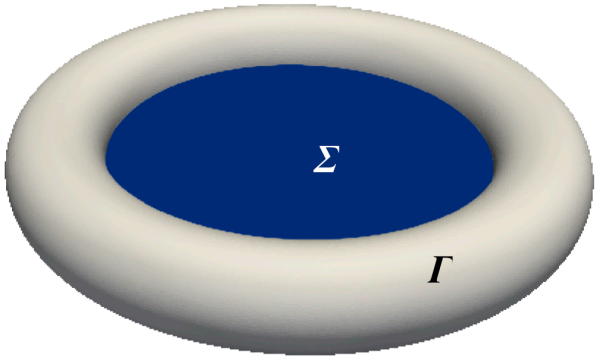

Figure 7.1: Sketch of a torus with the cutting surface Σ in blue

7.2.1 Finding the correct function space

As Γ is a smooth closed surface embedded in $\mathbb{R}^3$, the classification theorem of compact orientable surfaces [17, p. 204] states that it is a k-torus. For the k-torus there is a simple method to trivialize the first cohomology group of Ω_e by introducing cutting surfaces:

Definition 7.1 *If* Γ *is a* k*-torus forming the boundary between* Ω_c *and* Ω_e*, let* $\Sigma_1, ..., \Sigma_k$ *be compact oriented surfaces with trivial reduced cohomology (not intersecting each other) that fill the* k *holes of the torus* Γ*. These surfaces are called* Seifert surfaces *(ref. [24, pp.15]).*
Define as well:

$$\Omega_e' := \Omega_e \backslash (\Sigma_1 \cup ... \cup \Sigma_k)$$
$$\Gamma' := \Gamma \cup \Sigma_1 \cup ... \cup \Sigma_k$$

(ref. [14, Chapter 4])

An example for cutting surfaces for a torus can be seen in Figure 7.1.

Ω_e' from Definition 7.1 has trivial first cohomology group by construction and $\mathbf{curl}\,\mathbf{H}_r = 0$ there. This means it is possible to find a potential $h_r \in H^1(\triangle, \Omega_e')$ such that:

$$\begin{aligned} \widetilde{\mathbf{grad}}\, h_r &= \mathbf{H}_r \\ \widetilde{\triangle} h_r &= \operatorname{div} \mathbf{H}_r = 0 \end{aligned} \tag{7.4}$$

where $\widetilde{\mathbf{grad}}$ is the gradient in Ω_e', $\widetilde{\triangle}$ is the Laplacian in Ω_e', if (2.7) holds.

The space $H^1(\triangle, \Omega_e')$ however is too large to look for a solution in. A few properties of the solution h_r summarized in Lemma 7.2 can be used to find a smaller space:

Lemma 7.2 *The solution h_r to (7.4) fulfills the following properties:*
(i) The gradient $\widetilde{\mathbf{grad}}\, h_r$ is continuous along all the cutting surfaces.
(ii) The function h_r has a constant jump across each cutting surface.

Proof (i) The continuity of $\widetilde{\mathbf{grad}}\, h_r$ along cutting surfaces follows from the fact that $\widetilde{\mathbf{grad}}\, h_r = \mathbf{H}_r$. The discussion in Chapter 2 of $\mathbf{H}_r$ as a physical quantity gives the continuity of $\mathbf{H}_r$ in all of Ω_e (not just Ω_e').
This also means that although $h_r \in H^1(\triangle, \Omega_e')$ it holds that $\widetilde{\mathbf{grad}}\, h_r$ can be seen as a map into $\boldsymbol{H}(\mathbf{curl}; \Omega_e)$. From now on $\widetilde{\mathbf{grad}}\, h_r$ will always mean the extended, continuous version in Ω_e.

(ii) The constant jump of h_r follows from Ampère's law of magnetic fields. As of (2.7) it holds in all of Ω_e:

$$\mathbf{curl}\, \widetilde{\mathbf{grad}}\, h_r = \mathbf{curl}\, \mathbf{H}_r = 0 \tag{7.5}$$

This means by Stokes that for a closed curve C not passing through a cutting surface:

$$\int_C \widetilde{\mathbf{grad}}\, h_r(x) \cdot \mathbf{t}(x)\, dl(x) = 0 \tag{7.6}$$

where $\mathbf{t}(x)$ is an appropriate tangent field. This is evident, as by the fundamental theorem of line integrals [23, Chapter 5.3] it should hold that:

$$h_r(x_2) - h_r(x_1) = \int_G \widetilde{\mathbf{grad}}\, h_r(x) \cdot \mathbf{t}(x)\, dl(x) \tag{7.7}$$

where G is a curve from x_1 to x_2; thus at any point x where h_r is continuous it must hold $h_r(x) - h_r(x) = 0$.

Consider now the points x_ε^+ and x_ε^- which are at a distance of ε from the same point x_0 of the cutting surface, just above and just below. In the limit $\varepsilon \to 0$ the curve C from x_ε^- to x_ε^+ (inside Ω_e', not along the cutting surface) is a closed curve in Ω_e.
Because C passes a cutting surface, Stokes and (7.5) do not hold, so the integral will be some quantity depending on the point of discontinuity:

$$\int_C \widetilde{\mathbf{grad}}\, h_r(x) \cdot \mathbf{t}(x)\, dl(x) = I(x_0) \tag{7.8}$$

This function I signifies the jump of h_r across the cutting surface, as (7.7) gives $h_r(x_\varepsilon^+) - h_r(x_\varepsilon^-) = I(x_0)$. In the electromagnetic context this is the current flowing inside the torus.
However $I(x_0)$ is independent of the chosen x_0 on the cutting surface. Consider a second point x_0' somewhere on the cutting surface. Then it holds:

$$I(x_0') = I(x_0) + \int_{G^+} \widetilde{\mathbf{grad}}\, h_r(x) \cdot \mathbf{t}(x)\, dl(x) + \int_{G^-} \widetilde{\mathbf{grad}}\, h_r(x) \cdot \mathbf{t}(x)\, dl(x) \tag{7.9}$$

where G^+ is a path from $x_\varepsilon^{+\prime}$ to x_ε^+ and G^- is a path from x_ε^- to $x_\varepsilon^{-\prime}$. As $\widetilde{\mathbf{grad}}\, h_r$ is continuous across the cutting surface, the two additional integrals must cancel in the limit $\varepsilon \to 0$.
This means $I(x_0') = I(x_0)$. Thus the jump of h_r across any cutting surface is constant.

□

This is consistent with [14, Chapter 4].

Lemma 7.2 motivates Definition 7.3:

Definition 7.3

$$H^1_\Sigma(\Omega_e') := \{\varphi \in H^1(\triangle, \Omega_e') \mid [\gamma\varphi]_{\Sigma_i} = const,\ (\partial_{\mathbf{n}}\varphi)_{\Sigma_i^+} = -(\partial_{\mathbf{n}}\varphi)_{\Sigma_i^-}\ \ \forall i\}$$

$[f]_{\Sigma_i}$ is the jump of f across Σ_i.
Σ_i^+ is the "above part" of Σ_i oriented upwards and Σ_i^- is the "below part" of Σ_i oriented downwards.
The corresponding Dirichlet and Neumann trace spaces are:

$$H^{\frac{1}{2}}_\Sigma(\Gamma') := \{v \in H^{\frac{1}{2}}(\Gamma \setminus \cup_i \partial\Sigma_i) \mid v \in H^{\frac{1}{2}}(\Sigma_i^+), v \in H^{\frac{1}{2}}(\Sigma_i^-), [v]_{\Sigma_i} = const \quad \forall i\}$$
$$H^{-\frac{1}{2}}_\Sigma(\Gamma') := \{v \in H^{-\frac{1}{2}}(\Gamma \setminus \cup_i \partial\Sigma_i) \mid v \in H^{-\frac{1}{2}}(\Sigma_i^+), v \in H^{-\frac{1}{2}}(\Sigma_i^-), v|_{\Sigma_i^+} = -v|_{\Sigma_i^-} \quad \forall i\}$$

Lemma 7.2 shows that h_r must be in $H^1_\Sigma(\Omega_e')$: the constant jump in Dirichlet trace as well as the continuity in the gradient (which leads to opposite Neumann traces due to the opposite orientation of normal vectors) come from Lemma 7.2. It is enough to test with the appropriate test functions from this space in a weak formulation. This is beyond the scope of this thesis; theory on this can be found in e.g. [1, Section 4], [14, Theorem 2].

7.2.2 Calderon identities

As discussed in Section 7.2, inside the conductor the normal Calderon identities hold. Outside the conductor however they do not hold; a modified version has to be used which is derived from Green's third identity (the representation formula).
The calculations in this subsection are done just for the case of the torus (where there is *one* cutting surface, Σ), but they easily extend to the general case.

Consider the cross-section of the torus with infinitesimally thin cutting surface displayed in Figure 7.2. This model provides a closed, oriented surface $\Gamma' = \Gamma \cup \Sigma^+ \cup \Sigma^-$ on which integral equations can be formulated.

The first Calderon identity in this model takes the form of Lemma 7.4:

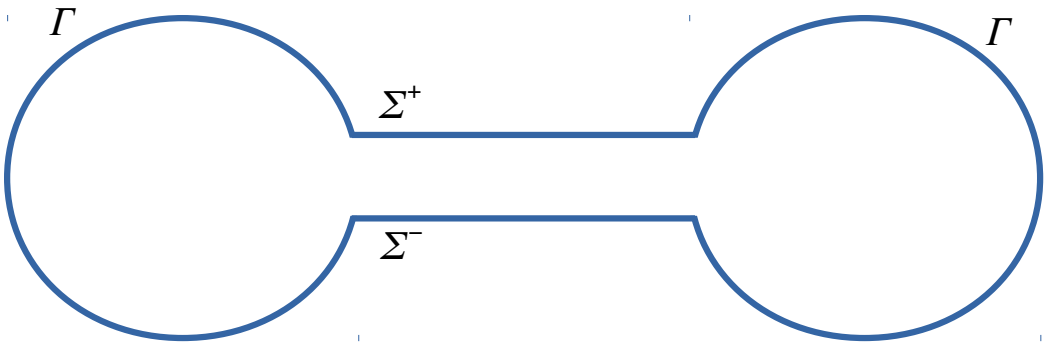

Figure 7.2: Cross-section of a torus where the cutting surface Σ has been expanded to be infinitesimally thick. Its above part is Σ^+, its below part is Σ^-. The whole surface is $\Gamma' = \Gamma \cup \Sigma^+ \cup \Sigma^-$.

Lemma 7.4 *Let h_r be the solution to (7.4).*
Let V_S be the single layer operator over the surface S and K_S the double layer operator over the surface S. Then it holds:

$$\gamma^e h_r = -V_\Gamma \partial^e_{\mathbf{n}} h_r + \left(K_\Gamma + \frac{1}{2}\right)\gamma^e h_r + [h_r]_\Sigma K_{\Sigma^+} 1 \tag{7.10a}$$

$$\gamma^+ h_r = -V_\Gamma \partial^e_{\mathbf{n}} h_r + K_\Gamma \gamma^e h_r + [h_r]_\Sigma \left(K_{\Sigma^+} 1 + \frac{1}{2}\right) \tag{7.10b}$$

$$\gamma^- h_r = -V_\Gamma \partial^e_{\mathbf{n}} h_r + K_\Gamma \gamma^e h_r + [h_r]_\Sigma \left(K_{\Sigma^+} 1 - \frac{1}{2}\right) \tag{7.10c}$$

Proof Consider the single and double layer potentials as defined in [29, p. 124], on the torus with infinitesimally thin cutting surface from Figure 7.2:

$$\begin{aligned}(Vw)(x) &:= \int_{\Gamma'} G(x,y) w(y) dS(y) \\ (Ww)(x) &:= \int_{\Gamma'} \mathbf{n}(y) \cdot \mathbf{grad}_y G(x,y) w(y) dS(y)\end{aligned} \tag{7.11}$$

Depending on the region x is in the potential shall be called W^+, W^0 or W^- (as seen in Figure 7.3).

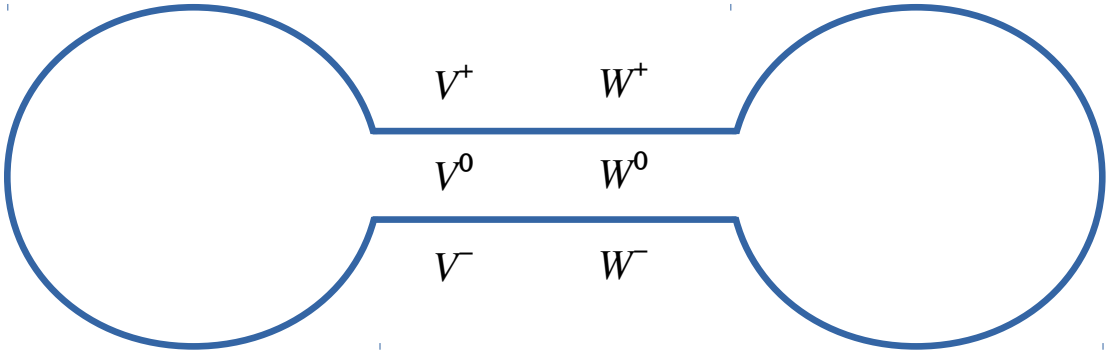

Figure 7.3: The surface from Figure 7.2 with regional potentials delineated

[29, Lemma 6.11] gives:

$$\begin{aligned}
\gamma^+ W^+ h_r &= \left(K_{\Sigma^+} + \frac{1}{2}\right)\gamma^+ h_r \\
\gamma^+ W^0 h_r &= \left(K_{\Sigma^+} - \frac{1}{2}\right)\gamma^+ h_r \\
\gamma^- W^0 h_r &= \left(K_{\Sigma^-} - \frac{1}{2}\right)\gamma^- h_r \\
\gamma^- W^- h_r &= \left(K_{\Sigma^-} + \frac{1}{2}\right)\gamma^- h_r
\end{aligned} \tag{7.12}$$

because the limit is taken from inside (Ω_c) and outside (Ω_e) respectively. γ^+ is the Dirichlet trace at Σ^+ and γ^- is the Dirichlet trace at Σ^-. As of [29, Lemma 6.7] the Dirichlet trace of the single layer potential is just continuously equal to the single layer operator.

Now it is possible to derive the first Calderon identity in a calculation similar to [29]. At the beginning is Green's third identity for $x \in \Omega_e, x \notin \Gamma'$ (ref. [9, p.712], notice the changed signs, as in this formulation the normal vectors point outwards):

$$
\begin{aligned}
h_r(x) &= -\int_{\Gamma'} G(x,y)\ \mathbf{n}_y \cdot \mathbf{grad}\, h_r(y) dS(y) + \int_{\Gamma'} h_r(y)\ \mathbf{n}_y \cdot \mathbf{grad}_y\, G(x,y) dS(y) \\
&= -\int_{\Gamma} G(x,y)\ \mathbf{n}_y \cdot \mathbf{grad}\, h_r(y) dS(y) + \int_{\Gamma} h_r(y)\ \mathbf{n}_y \cdot \mathbf{grad}_y\, G(x,y) dS(y) \\
&\quad - \int_{\Sigma^+} G(x,y)\ \mathbf{n}_y^+ \cdot \mathbf{grad}\, h_r(y) dS(y) + \int_{\Sigma^+} h_r(y)\ \mathbf{n}_y^+ \cdot \mathbf{grad}_y\, G(x,y) dS(y) \\
&\quad - \int_{\Sigma^-} G(x,y)\ \mathbf{n}_y^- \cdot \mathbf{grad}\, h_r(y) dS(y) + \int_{\Sigma^-} h_r(y)\ \mathbf{n}_y^- \cdot \mathbf{grad}_y\, G(x,y) dS(y)
\end{aligned}
\tag{7.13}
$$

Taking the Dirichlet trace for $x \in \Gamma$ the limit properties of the single and double layer operator give:

$$
\begin{aligned}
\gamma^e h_r(x) = &-(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + \left(K_\Gamma + \frac{1}{2}\right)\gamma^e h_r(x) \\
&- (V_{\Sigma^+}\partial_{\mathbf{n}}^+ h_r)(x) + K_{\Sigma^+}\gamma^+ h_r(x) - (V_{\Sigma^-}\partial_{\mathbf{n}}^- h_r)(x) + K_{\Sigma^-}\gamma^- h_r(x)
\end{aligned}
\tag{7.14}
$$

By the definition of the single layer operator, $V_{\Sigma^+} = V_{\Sigma^-}$. Because of the opposite orientation of normal vectors it holds that $K_{\Sigma^+} = -K_{\Sigma^-}$ and $\partial_{\mathbf{n}}^+ h_r = -\partial_{\mathbf{n}}^- h_r$. Finally, because of Lemma 7.2 it holds that $\gamma^- h_r = \gamma^+ h_r - [h_r]_\Sigma$ where $h_r]_\Sigma$ is the constant jump of the scalar potential across the cutting surface. Together these properties give:

$$
\begin{aligned}
\gamma^e h_r(x) &= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + \left(K_\Gamma + \frac{1}{2}\right)(\gamma^e h_r)(x) \\
&\quad + K_{\Sigma^+}\gamma^+ h_r(x) - K_{\Sigma^+}(\gamma^+ h_r(x) - [h_r]_\Sigma) \\
&= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + \left(K_\Gamma + \frac{1}{2}\right)(\gamma^e h_r)(x) + [h_r]_\Sigma K_{\Sigma^+} 1
\end{aligned}
\tag{7.15}
$$

Taking the Dirichlet trace for $x \in \Sigma^+$ (from above the cutting surface) the limit properties of the single layer operator and the double layer operator described in (7.12) give (the integral over Σ^+ corresponds to $\gamma^+ W^+ h_r$, the integral over Σ^- corresponds to $\gamma^- W^0 h_r$):

$$
\begin{aligned}
\gamma^+ h_r(x) &= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) - (V_{\Sigma^+}\partial_{\mathbf{n}}^+ h_r)(x) + \left(K_{\Sigma^+} + \frac{1}{2}\right)\gamma^+ h_r(x) \\
&\quad - (V_{\Sigma^-}\partial_{\mathbf{n}}^- h_r)(x) + \left(K_{\Sigma^-} - \frac{1}{2}\right)\gamma^- h_r(x) \\
&= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) + \left(K_{\Sigma^+} + \frac{1}{2}\right)\gamma^+ h_r(x) \\
&\quad + \left(-K_{\Sigma^+} - \frac{1}{2}\right)(\gamma^+ h_r(x) - [h_r]_\Sigma) \\
&= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) + [h_r]_\Sigma \left(K_{\Sigma^+} 1 + \frac{1}{2}\right)
\end{aligned}
\tag{7.16}
$$

Taking the Dirichlet trace for $x \in \Sigma^-$ (from below the cutting surface) the limit properties of the single layer operator and the double layer operator described in (7.12) give (the integral over Σ^+ corresponds to $\gamma^+ W^0 h_r$, the integral over Σ^- corresponds to $\gamma^- W^- h_r$):

$$\begin{aligned}
\gamma^- h_r(x) &= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) - (V_{\Sigma^+} \partial_{\mathbf{n}}^+ h_r)(x) + \left(K_{\Sigma^+} - \frac{1}{2}\right) \gamma^+ h_r(x) \\
&\quad - (V_{\Sigma^-} \partial_{\mathbf{n}}^- h_r)(x) + \left(K_{\Sigma^-} + \frac{1}{2}\right) \gamma^- h_r(x) \\
&= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) + \left(K_{\Sigma^+} - \frac{1}{2}\right) \gamma^+ h_r(x) \\
&\quad + \left(-K_{\Sigma^+} + \frac{1}{2}\right) (\gamma^+ h_r(x) - [h_r]_\Sigma) \\
&= -(V_\Gamma \partial_{\mathbf{n}}^e h_r)(x) + (K_\Gamma \gamma^e h_r)(x) + [h_r]_\Sigma \left(K_{\Sigma^+} 1 - \frac{1}{2}\right)
\end{aligned} \tag{7.17}$$

□

The second Calderon identity in this model takes the form of Lemma 7.5

Lemma 7.5 *Let h_r be the solution to (7.4).*
Let D_S be the hypersingular operator over the surface S and K_S^* the adjoint double layer operator over the surface S. Then it holds:

$$\partial_{\mathbf{n}}^e h_r = -\left(K_\Gamma^* - \frac{1}{2}\right) \partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r] D_{\Sigma^+} 1 \tag{7.18a}$$

$$\partial_{\mathbf{n}}^+ h_r = -K_\Gamma^* \partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r]_\Sigma D_{\Sigma^+} 1 \tag{7.18b}$$

$$\partial_{\mathbf{n}}^- h_r = -K_\Gamma^* \partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r]_\Sigma D_{\Sigma^+} 1 \tag{7.18c}$$

Proof Consider the single layer potential defined on the regions of Figure 7.3 from Lemma 7.4. Then [29, Lemma 6.8] gives:

$$\begin{aligned}
\partial_{\mathbf{n}}^+ V^+ (\mathbf{n}^+ \cdot \mathbf{grad}\, h_r) &= \left(K_{\Sigma^+}^* - \frac{1}{2}\right) (\mathbf{n}^+ \cdot \mathbf{grad}\, h_r) = \left(K_{\Sigma^+}^* - \frac{1}{2}\right) \partial_{\mathbf{n}}^+ h_r \\
\partial_{\mathbf{n}}^+ V^0 (\mathbf{n}^+ \cdot \mathbf{grad}\, h_r) &= \left(K_{\Sigma^+}^* + \frac{1}{2}\right) (\mathbf{n}^+ \cdot \mathbf{grad}\, h_r) = K_{\Sigma^+}^* \partial_{\mathbf{n}}^+ h_r + \frac{1}{2} \partial_{\mathbf{n}}^- h_r \\
\partial_{\mathbf{n}}^- V^0 (\mathbf{n}^- \cdot \mathbf{grad}\, h_r) &= \left(K_{\Sigma^-}^* + \frac{1}{2}\right) (\mathbf{n}^- \cdot \mathbf{grad}\, h_r) = K_{\Sigma^-}^* \partial_{\mathbf{n}}^- h_r + \frac{1}{2} \partial_{\mathbf{n}}^+ h_r \\
\partial_{\mathbf{n}}^- V^- (\mathbf{n}^- \cdot \mathbf{grad}\, h_r) &= \left(K_{\Sigma^-}^* - \frac{1}{2}\right) (\mathbf{n}^- \cdot \mathbf{grad}\, h_r) = \left(K_{\Sigma^-}^* - \frac{1}{2}\right) \partial_{\mathbf{n}}^- h_r
\end{aligned} \tag{7.19}$$

The two different Neumann traces for the V^0 traces are because the limits are taken from the other side than the natural definition of the trace (which is defined as the external

trace and thus has the normal vector in the opposite direction). This is intrinsic to the operators $K^*_{\Sigma^{+/-}}$ and $V_{\Sigma^{+/-}}$ – they change the direction of the normal vector, depending on whether they are evaluated on Σ^+ or Σ^-.

Again a trace operator is applied to Green's third identity (7.13). This time the Neumann trace is taken for $x \in \Gamma$:

$$\begin{aligned}\partial^e_{\mathbf{n}} h_r(x) = &-\left(K^*_\Gamma - \frac{1}{2}\right)(\partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) \\ &- (K^*_{\Sigma^+}\partial^+_{\mathbf{n}} h_r)(x) - (D_{\Sigma^+}\gamma^+ h_r)(x) - (K^*_{\Sigma^-}\partial^-_{\mathbf{n}} h_r)(x) - (D_{\Sigma^-}\gamma^- h_r)(x)\end{aligned} \tag{7.20}$$

By the definition of the adjoint double layer operator, $K^*_{\Sigma^+} = K^*_{\Sigma^-}$. Because of the opposite orientation of normal vectors, $D_{\Sigma^+} = -D_{\Sigma^-}$. These properties give:

$$\begin{aligned}\partial^e_{\mathbf{n}} h_r(x) &= -\left(K^*_\Gamma - \frac{1}{2}\right)(\partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) \\ &\quad - (D_{\Sigma^+}\gamma^+ h_r)(x) + (D_{\Sigma^+}\gamma^+ h_r(x) - [h_r]_\Sigma) \\ &= -\left(K^*_\Gamma - \frac{1}{2}\right)(\partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - [h_r] D_{\Sigma^+} 1\end{aligned} \tag{7.21}$$

Taking the Neumann trace for $x \in \Sigma^+$ (infinitesimally close above the cutting surface) the limit properties of the single layer operator described in (7.19) and the double layer operator give:

$$\begin{aligned}\partial^+_{\mathbf{n}} h_r(x) &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - \left(K^*_{\Sigma^+} - \frac{1}{2}\right)(\partial^+_{\mathbf{n}} h_r)(x) - (D_{\Sigma^+}\gamma^+ h_r)(x) \\ &\quad - \left((K^*_{\Sigma^-}\partial^-_{\mathbf{n}} h_r)(x) + \frac{1}{2}(\partial^+_{\mathbf{n}} h_r)(x)\right) - (D_{\Sigma^-}\gamma^- h_r)(x) \\ &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - (D_{\Sigma^+}\gamma^+ h_r)(x) \\ &\quad + D_{\Sigma^+}(\gamma^+ h_r(x) - [h_r]_\Sigma) \\ &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - [h_r]_\Sigma D_{\Sigma^+} 1\end{aligned} \tag{7.22}$$

Taking the Neumann trace for $x \in \Sigma^-$ (infinitesimally close below the cutting surface) gives:

$$\begin{aligned}\partial^-_{\mathbf{n}} h_r(x) &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - \left((K^*_{\Sigma^+}\partial^+_{\mathbf{n}} h_r)(x) + \frac{1}{2}(\partial^-_{\mathbf{n}} h_r)(x)\right) \\ &\quad - (D_{\Sigma^+}\gamma^+ h_r)(x) - \left(K^*_{\Sigma^-} - \frac{1}{2}\right)(\partial^-_{\mathbf{n}} h_r)(x) - (D_{\Sigma^-}\gamma^- h_r)(x) \\ &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - (D_{\Sigma^+}\gamma^+ h_r)(x) \\ &\quad + D_{\Sigma^+}(\gamma^+ h_r(x) - [h_r]_\Sigma) \\ &= -(K^*_\Gamma \partial^e_{\mathbf{n}} h_r)(x) - (D_\Gamma \gamma^e h_r)(x) - [h_r]_\Sigma D_{\Sigma^+} 1\end{aligned} \tag{7.23}$$

□

7.2.3 Weak formulation

It remains to find a weak formulation for the Calderon identities described in the last subsection. As the spaces are dual to each other, the first Calderon identity from Lemma 7.4 is tested with a function φ from $H_{\Sigma}^{-\frac{1}{2}}(\Gamma')$:

$$\begin{aligned}
\langle \varphi, \gamma^e h_r \rangle_{\Gamma'} = &\left\langle \varphi, -V_\Gamma \partial_{\mathbf{n}}^e h_r + \left(K_\Gamma + \frac{1}{2}\right)\gamma^e h_r + [h_r]_\Sigma K_{\Sigma^+} 1 \right\rangle_\Gamma \\
&+ \left\langle \varphi, -V_\Gamma \partial_{\mathbf{n}}^e h_r + K_\Gamma \gamma^e h_r + [h_r]_\Sigma \left(K_{\Sigma^+} 1 + \frac{1}{2}\right) \right\rangle_{\Sigma^+} \\
&+ \left\langle \varphi, -V_\Gamma \partial_{\mathbf{n}}^e h_r + K_\Gamma \gamma^e h_r + [h_r]_\Sigma \left(K_{\Sigma^+} 1 - \frac{1}{2}\right) \right\rangle_{\Sigma^-} \\
= &- \langle \varphi, V_\Gamma \partial_{\mathbf{n}}^e h_r \rangle_\Gamma + \left\langle \varphi, \left(K_\Gamma + \frac{1}{2}\right)\gamma^e h_r \right\rangle_\Gamma \\
&+ [h_r]_\Sigma \langle \varphi, K_{\Sigma^+} 1 \rangle_\Gamma + [h_r]_\Sigma \langle \varphi, 1 \rangle_{\Sigma^+}
\end{aligned} \tag{7.24}$$

because φ is exactly opposite on Σ^+ and Σ^-.

On the left side of the equation it holds:

$$\begin{aligned}
\langle \varphi, \gamma^e h_r \rangle_{\Gamma'} &= \langle \varphi, \gamma^e h_r \rangle_\Gamma + \langle \varphi, \gamma^+ h_r \rangle_{\Sigma^+} + \langle \varphi, \gamma^- h_r \rangle_{\Sigma^-} \\
&= \langle \varphi, \gamma^e h_r \rangle_\Gamma + \langle \varphi, \gamma^+ h_r \rangle_{\Sigma^+} - \langle \varphi, \gamma^+ h_r - [h_r]_{\Sigma^+} \rangle_{\Sigma^+} \\
&= \langle \varphi, \gamma^e h_r \rangle_\Gamma + [h_r]_\Sigma \langle \varphi, 1 \rangle_{\Sigma^+}
\end{aligned} \tag{7.25}$$

Combined this gives:

$$\langle \varphi, \gamma^e h_r \rangle_\Gamma = - \langle \varphi, V_\Gamma \partial_{\mathbf{n}}^e h_r \rangle_\Gamma + \left\langle \varphi, \left(K_\Gamma + \frac{1}{2}\right)\gamma^e h_r \right\rangle_\Gamma + [h_r]_\Sigma \langle \varphi, K_{\Sigma^+} 1 \rangle_\Gamma \tag{7.26}$$

By the same duality reasoning, the second Calderon identity from Lemma 7.5 is tested with a function v from $H_{\Sigma}^{\frac{1}{2}}(\Gamma')$:

$$\begin{aligned}
\langle v, \partial_{\mathbf{n}}^e h_r \rangle_{\Gamma'} = &\left\langle v, -\left(K_\Gamma^* - \frac{1}{2}\right)\partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r] D_{\Sigma^+} 1 \right\rangle_\Gamma \\
&+ \langle v, -K_\Gamma^* \partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r]_\Sigma D_{\Sigma^+} 1 \rangle_{\Sigma^+} \\
&+ \langle v, -K_\Gamma^* \partial_{\mathbf{n}}^e h_r - D_\Gamma \gamma^e h_r - [h_r]_\Sigma D_{\Sigma^+} 1 \rangle_{\Sigma^-} \\
= &- \left\langle v, \left(K_\Gamma^* - \frac{1}{2}\right)\partial_{\mathbf{n}}^e h_r \right\rangle_\Gamma - \langle v, D_\Gamma \gamma^e h_r \rangle_\Gamma - [h_r] \langle v, D_{\Sigma^+} 1 \rangle_\Gamma \\
&- \langle v, K_\Gamma^* \partial_{\mathbf{n}}^e h_r \rangle_{\Sigma^+} - \langle v, D_\Gamma \gamma^e h_r \rangle_{\Sigma^+} - [h_r]_\Sigma \langle v, D_{\Sigma^+} 1 \rangle_{\Sigma^+} \\
&+ \langle v - [v]_\Sigma, K_\Gamma^* \partial_{\mathbf{n}}^e h_r \rangle_{\Sigma^+} + \langle v - [v]_\Sigma, D_\Gamma \gamma^e h_r \rangle_{\Sigma^+} + [h_r]_\Sigma \langle v - [v]_\Sigma, D_{\Sigma^+} 1 \rangle_{\Sigma^+} \\
= &- \left\langle v, \left(K_\Gamma^* - \frac{1}{2}\right)\partial_{\mathbf{n}}^e h_r \right\rangle_\Gamma - [v]_\Sigma \langle 1, K_\Gamma^* \partial_{\mathbf{n}}^e h_r \rangle_{\Sigma^+} \\
&- \langle v, D_\Gamma \gamma^e h_r + [h_r]_\Sigma D_{\Sigma^+} 1 \rangle_\Gamma - [v]_\Sigma \langle 1, D_\Gamma \gamma^e h_r + [h_r]_\Sigma D_{\Sigma^+} 1 \rangle_{\Sigma^+}
\end{aligned} \tag{7.27}$$

On the left side of the equation it holds:

$$\begin{aligned}\langle v, \partial^e_{\mathbf{n}} h_r\rangle_{\Gamma'} &= \langle v, \partial^e_{\mathbf{n}} h_r\rangle_{\Gamma} + \left\langle v, \partial^+_{\mathbf{n}} h_r\right\rangle_{\Sigma^+} + \left\langle v, \partial^-_{\mathbf{n}} h_r\right\rangle_{\Sigma^-} \\ &= \langle v, \partial^e_{\mathbf{n}} h_r\rangle_{\Gamma} + \left\langle v, \partial^+_{\mathbf{n}} h_r\right\rangle_{\Sigma^+} - \left\langle v - [v]_\Sigma, \partial^+_{\mathbf{n}} h_r\right\rangle_{\Sigma^+} \\ &= \langle v, \partial^e_{\mathbf{n}} h_r\rangle_{\Gamma} + [v]_\Sigma \left\langle 1, \partial^+_{\mathbf{n}} h_r\right\rangle_{\Sigma^+}\end{aligned} \tag{7.28}$$

Combined this gives:

$$\begin{aligned}\langle v, \partial^e_{\mathbf{n}} h_r\rangle_{\Gamma} &= -\left\langle v, \left(K^*_\Gamma - \frac{1}{2}\right)\partial^e_{\mathbf{n}} h_r\right\rangle_\Gamma - [v]_\Sigma \langle 1, K^*_\Gamma \partial^e_{\mathbf{n}} h_r\rangle_{\Sigma^+} \\ +[v]_\Sigma \left\langle 1, \partial^+_{\mathbf{n}} h_r\right\rangle_{\Sigma^+} &\quad - \langle v, D_\Gamma \gamma^e h_r + [h_r]_\Sigma D_{\Sigma^+} 1\rangle_\Gamma - [v]_\Sigma \langle 1, D_\Gamma \gamma^e h_r + [h_r]_\Sigma D_{\Sigma^+} 1\rangle_{\Sigma^+}\end{aligned} \tag{7.29}$$

7.2.4 Partial integration of the hypersingular operator

A minor issue which has to be addressed here is the formulation of the hypersingular operator. As the hypersingular operator involves two derivatives it can not be implemented the way it is defined numerically. For the case of the method with simple geometries the hypersingular operator has been reduced to the single layer operator for vectors in the bilinear setting (ref. (4.3) where it was assumed that the surface is closed and the functions continuous) using [29, Lemma 6.16] and [29, Theorem 6.17].
However these do not hold for the case of discontinuous functions and non-closed surfaces. In order to be able to implement the hypersingular operator these issues have to be addressed.

The suitable modification for [29, Lemma 6.16] is Lemma 7.6:

Lemma 7.6 *Γ', Γ, $\Sigma^\pm$ as used in this chapter before. $u \in H^{\frac{1}{2}}_\Sigma(\Gamma')$ and $\mathbf{v} \in \boldsymbol{H}_\perp^{-\frac{1}{2}}(\mathbf{curl}_\Gamma, \Gamma)$. Then it holds:*

$$\int_\Gamma \mathbf{curl}_\Gamma u(x) \cdot \mathbf{v}(x) dS(x) = -\int_\Gamma u(x) \operatorname{curl}_\Gamma \mathbf{v}(x) dS(x) - [u]_\Sigma \int_{\Sigma^+} \operatorname{curl}_{\Sigma^+} \mathbf{v}(x) dS(x)$$

Proof For reasonable extensions $\tilde{u}$ and $\tilde{\mathbf{v}}$ to Ω_e, by the reasoning of [29, Lemma 6.16] it holds

$$\begin{aligned}\int_\Gamma \mathbf{curl}_\Gamma u(x) \cdot \mathbf{v}(x) dS(x) &= \int_\Gamma (\mathbf{curl}(\tilde{u}(x)\tilde{\mathbf{v}}(x)) - \tilde{u}(x)\, \mathbf{curl}\, \tilde{\mathbf{v}}(x)) \cdot \mathbf{n}(x) dS(x) \\ \int_\Gamma u(x) \operatorname{curl}_\Gamma \mathbf{v}(x) dS(x) &= \int_\Gamma \mathbf{curl}(\tilde{u}(x)\tilde{\mathbf{v}}(x)) \cdot \mathbf{n}(x) dS(x) - \int_\Gamma \mathbf{curl}_\Gamma u(x) \cdot \mathbf{v}(x) dS(x)\end{aligned} \tag{7.30}$$

Because of the discontinuity at the boundary of the cutting surface, applying Stokes gives:

$$\begin{aligned}\int_\Gamma \mathbf{curl}(\tilde{u}(x)\tilde{\mathbf{v}}(x))\cdot\mathbf{n}(x)dS(x) &= -\int_{\partial\Sigma^+}\tilde{u}(x)\tilde{\mathbf{v}}(x)\cdot\mathbf{t}_{\partial\Sigma^+}(x)dS(x) - \int_{\partial\Sigma^-}\tilde{u}(x)\tilde{\mathbf{v}}(x)\cdot\mathbf{t}_{\partial\Sigma^-}(x)dS(x)\\ &= -[u]_\Sigma\int_{\partial\Sigma^+}\tilde{\mathbf{v}}(x)\cdot\mathbf{t}_{\partial\Sigma^+}(x)dS(x)\\ &= -[u]_\Sigma\int_{\Sigma^+}\operatorname{curl}_{\Sigma^+}\mathbf{v}(x)dS(x)\end{aligned} \tag{7.31}$$

where $\mathbf{t}$ is an appropriate tangent field. The change in sign comes from the fact that the cutting surfaces' boundaries are oriented opposite to the "discontinuity boundaries" on Γ.
This fact combined with (7.30) gives the statement of the lemma. □

The following version of the lemma is also necessary:

Lemma 7.7 *Γ', Γ, $\Sigma^\pm$ as used in this chapter before. $u \in H^{\frac{1}{2}}(\Gamma'), f \in H^{\frac{1}{2}}_\Sigma(\Gamma')$ and $\mathbf{v} \in \boldsymbol{H}^{-\frac{1}{2}}_\perp(\mathbf{curl}_\Gamma, \Gamma)$.*
Then it holds:

$$\int_\Gamma \mathbf{curl}_\Gamma u(x)\cdot(f(x)\mathbf{v}(x))dS(x) = -\int_\Gamma u(x)\operatorname{curl}_\Gamma(f(x)\mathbf{v}(x))dS(x) - [f]_\Sigma\int_{\Sigma^+}\operatorname{curl}_{\Sigma^+}(u(x)\mathbf{v}(x))dS(x)$$

Proof Following the proof of Lemma 7.7 it is possible to arrive at the following identity:

$$\begin{aligned}\int_\Gamma u(x)\operatorname{curl}_\Gamma(f(x)\mathbf{v}(x))dS(x) = &\int_\Gamma \mathbf{curl}\left(\tilde{u}(x)\tilde{f}(x)\tilde{\mathbf{v}}(x)\right)\cdot\mathbf{n}(x)dS(x)\\ &-\int_\Gamma \mathbf{curl}_\Gamma u(x)\cdot(f(x)\mathbf{v}(x))dS(x)\end{aligned} \tag{7.32}$$

Because of the discontinuity, this time of f, at the boundary (as in Lemma 7.6) it holds:

$$\int_\Gamma \mathbf{curl}\left(\tilde{u}(x)\tilde{f}(x)\tilde{\mathbf{v}}(x)\right)\cdot\mathbf{n}(x)dS(x) = -[f]_\Sigma\int_{\Sigma^+}\operatorname{curl}_{\Sigma^+}(u(x)\mathbf{v}(x))dS(x) \tag{7.33}$$

which gives the statement of the lemma. □

The suitable modification for [29, Theorem 6.17] is Theorem 7.8:

Theorem 7.8 *Γ', Γ, $\Sigma^{\pm}$ as used in this chapter before. $u \in H_{\Sigma}^{\frac{1}{2}}(\Gamma')$. Then it holds:*

$$\begin{aligned}
&\langle v, D_\Gamma u + [u]_\Sigma D_{\Sigma^+} 1\rangle_\Gamma + [v]_\Sigma \langle 1, D_\Gamma \gamma^e u + [u]_\Sigma D_{\Sigma^+} 1\rangle_{\Sigma^+} \\
&\quad = \int_\Gamma \int_\Gamma \mathbf{curl}_\Gamma v(x) \cdot \mathbf{curl}_\Gamma u(y) G(x,y) dS(x) dS(y) \\
&\quad = \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma u\rangle_\Gamma
\end{aligned}$$

where D_S is the scalar hypersingular operator and $\mathbf{A}_S$ is the vectorial single layer operator (for the Laplace equation) on the surface S.

Proof Let $w(\tilde{x}) := \int_\Gamma u(y) \partial_{\mathbf{n}}^e G(\tilde{x}, y) dS(y) + [u]_\Sigma \int_{\Sigma^+} \partial_{\mathbf{n}}^+ G(\tilde{x}, y) dS(y)$ where $\tilde{x} \in \Omega_e$, $\tilde{x} \notin \Gamma$.
By the reasoning of [29, Theorem 6.17] it holds:

$$\begin{aligned}
\frac{\partial}{\partial \tilde{x}_i} w(\tilde{x}) = &\int_\Gamma u(y) \mathrm{curl}_{\Gamma y} \left(\mathbf{e}_i \times \mathbf{grad}_y G(\tilde{x}, y) \right) dS(y) \\
&+ [u]_\Sigma \int_{\Sigma^+} \mathrm{curl}_{\Sigma^+ y} \left(\mathbf{e}_i \times \mathbf{grad}_y G(\tilde{x}, y) \right) dS(y)
\end{aligned} \tag{7.34}$$

where the $\mathbf{e}_i$ are the unit vectors in $\mathbb{R}^3$.

Using Lemma 7.6 it then holds:

$$\frac{\partial}{\partial \tilde{x}_i} w(\tilde{x}) = -\int_\Gamma \mathbf{curl}_\Gamma u(y) \cdot \left(\mathbf{e}_i \times \mathbf{grad}_y G(\tilde{x}, y) \right) dS(y) \tag{7.35}$$

Now it is again possible to follow the proof from [29, Theorem 6.17] to get the following limit $\tilde{x} \to x \in \Gamma$:

$$\left(D_\Gamma u + [u]_\Sigma D_{\Sigma^+} 1 \right)(x) = -\lim_{\varepsilon \to 0} \int\limits_{\substack{y \in \Gamma \\ |x-y| \geq \varepsilon}} \mathbf{curl}_\Gamma u(y) \cdot \mathbf{curl}_{\Gamma x} G(x,y) dS(y) \tag{7.36}$$

When weakly testing with some $v \in H_{\Sigma}^{\frac{1}{2}}(\Gamma)$, the following holds:

$$\begin{aligned}
\langle v, D_\Gamma u + [u]_\Sigma D_{\Sigma^+} 1\rangle_\Gamma &= -\int_\Gamma v(x) \lim_{\varepsilon \to 0} \int\limits_{\substack{y \in \Gamma \\ |x-y| \geq \varepsilon}} \mathbf{curl}_\Gamma u(y) \cdot \mathbf{curl}_{\Gamma x} G(x,y) dS(y) dS(x) \\
&= -\int_\Gamma \lim_{\varepsilon \to 0} \int\limits_{\substack{x \in \Gamma \\ |x-y| \geq \varepsilon}} (v(x) \mathbf{curl}_\Gamma u(y)) \cdot \mathbf{curl}_{\Gamma x} G(x,y) dS(x) dS(y)
\end{aligned} \tag{7.37}$$

Using Lemma 7.7 gives:

$$\begin{aligned}\langle v, D_\Gamma u + [u]_\Sigma D_{\Sigma^+}1\rangle_\Gamma = \int_\Gamma \lim_{\varepsilon\to 0}\Bigg(&\int\limits_{\substack{x\in\Gamma\\ |x-y|\geq\varepsilon}} \operatorname{curl}_{\Gamma x}\left(v(x)\mathbf{curl}_\Gamma u(y)\right) G(x,y)dS(x) \\ &+ [v]_\Sigma \int\limits_{\substack{x\in\Sigma^+\\ |x-y|\geq\varepsilon}} \operatorname{curl}_{\Sigma^+ x}(G(x,y)\mathbf{curl}_\Gamma u(y))dS(x)\Bigg)dS(y)\end{aligned} \tag{7.38}$$

At the same time it holds:

$$\begin{aligned}\langle 1, D_\Gamma\gamma^e u + [u]_\Sigma D_{\Sigma^+}1\rangle_{\Sigma^+} &= -\int_\Gamma \lim_{\varepsilon\to 0} \int\limits_{\substack{x\in\Sigma^+\\ |x-y|\geq\varepsilon}} \mathbf{curl}_\Gamma u(y)\cdot\mathbf{curl}_{\Sigma^+ x}G(x,y)dS(x)dS(y) \\ &= -\int_\Gamma \lim_{\varepsilon\to 0} \int\limits_{\substack{x\in\Sigma^+\\ |x-y|\geq\varepsilon}} \mathbf{curl}_x\left(G(x,y)\mathbf{curl}_\Gamma u(y)\right)\cdot\mathbf{n}(x)dS(x)dS(y)\end{aligned} \tag{7.39}$$

by the same reasoning as in the proofs of Lemmas 7.6 and 7.7. This time however, there is no sign flip when integrating over $\partial\Sigma^+$ as the boundary is approached from inside Σ^+ itself and not from Γ.

Combining the last two equations gives:

$$\begin{aligned}&\langle v, D_\Gamma u + [u]_\Sigma D_{\Sigma^+}1\rangle_\Gamma + [v]_\Sigma \langle 1, D_\Gamma\gamma^e u + [u]_\Sigma D_{\Sigma^+}1\rangle_{\Sigma^+} \\ &\quad= \int_\Gamma \lim_{\varepsilon\to 0} \int\limits_{\substack{x\in\Gamma\\ |x-y|\geq\varepsilon}} \operatorname{curl}_{\Gamma x}\left(v(x)\mathbf{curl}_\Gamma u(y)\right) G(x,y)dS(x)dS(y)\end{aligned} \tag{7.40}$$

Together with the identity $\operatorname{curl}_{\Gamma x}\left(v(x)\mathbf{curl}_{\Gamma y}u(y)\right) = \mathbf{curl}_{\Gamma x}v(x)\cdot\mathbf{curl}_{\Gamma y}u(y)$ from [29, p. 136] this proves the theorem. □

Theorem 7.8 can now be used to get a simpler statement of (7.29):

$$\begin{aligned}\langle\mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma\mathbf{curl}_\Gamma\gamma^e h_r\rangle_\Gamma = -\left\langle v, \left(K_\Gamma^* + \frac{1}{2}\right)\partial_{\mathbf{n}}^e h_r\right\rangle_\Gamma - [v]_\Sigma \langle 1, K_\Gamma^*\partial_{\mathbf{n}}^e h_r\rangle_{\Sigma^+} \\ - [v]_\Sigma \left\langle 1, \partial_{\mathbf{n}}^+ h_r\right\rangle_{\Sigma^+}\end{aligned} \tag{7.41}$$

7.3 Combining inside and outside

As in Chapter 5, the equations for inside the conductor, (7.2) and (7.3), and the equations for outside the conductor, (7.26) and (7.29), are now combined using the coupling property (5.12). This can be accomplished in two ways: using only the first Calderon identity (the non-symmetric way) or using both Calderon identities (the symmetric way, as it is done in Chapter 5). Both ways will be investigated here.

7.3.1 The non-symmetric way

Inserting $\gamma_{\mathbf{t}}^c \mathbf{H} = \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r + \gamma_{\mathbf{t}}^e \mathbf{H}_s$ into (7.3) using the coupling property (5.12) gives (where $v \in H_\Sigma^{\frac{1}{2}}(\Gamma)$):

$$\begin{aligned}
\frac{1}{\tau} \langle \partial_{\mathbf{n}}^e h_r, v \rangle_d + \frac{1}{\tau} \langle \gamma_{\mathbf{n}}^e \mathbf{H}_s, v \rangle_d &= \langle \mathbf{V}, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \rangle_l + \langle \mathbf{V}, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^c \mathbf{H} \rangle_l \\
\langle \partial_{\mathbf{n}}^e h_r, v \rangle_d + \langle \gamma_{\mathbf{n}}^e \mathbf{H}_s, v \rangle_d &= \tau \langle \mathbf{V}, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \rangle_l \\
&\quad + \tau \left\langle \mathbf{V}, \mathbf{N}^\kappa \left(\widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r + \gamma_{\mathbf{t}}^e \mathbf{H}_s \right) \right\rangle_l \\
\langle \gamma_{\mathbf{n}}^e \mathbf{H}_s, v \rangle_d - \tau \langle \mathbf{V}, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^e \mathbf{H}_s \rangle_l &= \tau \left\langle \mathbf{V}, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l \\
&\quad + \tau \langle \mathbf{V}, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \rangle_l - \langle \partial_{\mathbf{n}}^e h_r, v \rangle_d \\
\langle v, \gamma_{\mathbf{n}}^e \mathbf{H}_s \rangle_d - \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^e \mathbf{H}_s \right\rangle_l &= \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l \\
&\quad + \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \right\rangle_l - \langle v, \partial_{\mathbf{n}}^e h_r \rangle_d
\end{aligned} \tag{7.42}$$

where in the last line it was used that $\mathbf{V} = \widetilde{\mathbf{grad}_\Gamma} v$, $v \in H_\Sigma^{\frac{1}{2}}(\Gamma)$.

Inserting $\gamma_{\mathbf{t}}^c \mathbf{H} = \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r + \gamma_{\mathbf{t}}^e \mathbf{H}_s$ into (7.2) gives:

$$\begin{aligned}
0 &= \left\langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id}) \left(\widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r + \gamma_{\mathbf{t}}^e \mathbf{H}_s \right) \right\rangle_l + \langle \mu, \mathbf{A}^\kappa \gamma_N^c \mathbf{H} \rangle_l \\
\langle \mu, (0.5\,\mathbf{id} - \mathbf{C}^\kappa)\, \gamma_{\mathbf{t}}^e \mathbf{H}_s \rangle_l &= \left\langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\, \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l + \langle \mu, \mathbf{A}^\kappa \gamma_N^c \mathbf{H} \rangle_l
\end{aligned} \tag{7.43}$$

with $\mu \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$ as in (7.2).

Finally, (7.26) gives:

$$0 = \left\langle \varphi, \left(K_\Gamma - \frac{1}{2} \right) \gamma^e h_r \right\rangle_d + [h]_\Sigma \langle \varphi, K_{\Sigma^+} 1 \rangle_d - \langle \varphi, V_\Gamma \partial_{\mathbf{n}}^e h_r \rangle_d \tag{7.44}$$

where $\varphi \in H^{-\frac{1}{2}}(\Gamma)$.

7.3.2 The symmetric way

The symmetric version is obtained by using the second scalar Calderon identity (7.29) on the scalar Neumann trace in (7.42). The result is summed up in Theorem 7.9:

Theorem 7.9 *The weak formulation of the symmetric boundary integral equations for the eddy current problem are:*

$$\begin{aligned}\langle v, \gamma_{\mathbf{n}}^e \mathbf{H}_s\rangle_d - \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^e \mathbf{H}_s \right\rangle_l &= \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l \\ +[v]_\Sigma \left\langle 1, \gamma_{\mathbf{n}}^+ \mathbf{H}_s \right\rangle_{d(\Sigma^+)} &\quad + \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \right\rangle_l \\ &\quad + \left\langle \left(K_\Gamma - \frac{1}{2}\right) v, \partial_{\mathbf{n}}^e h_r \right\rangle_d \\ &\quad + [v]_\Sigma \left\langle K_{\Sigma^+} 1, \partial_{\mathbf{n}}^e h_r \right\rangle_d - [v]_\Sigma \frac{\tau \mu_0}{\mu_c} L \gamma_N^c \mathbf{H} \\ &\quad + \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma h_r \rangle_l \end{aligned} \tag{7.45a}$$

$$\langle \mu, (0.5\,\mathbf{id} - \mathbf{C}^\kappa)\, \gamma_{\mathbf{t}}^e \mathbf{H}_s \rangle_l = \left\langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\, \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l + \langle \mu, \mathbf{A}^\kappa \gamma_N^c \mathbf{H} \rangle_l \tag{7.45b}$$

$$0 = \left\langle \varphi, \left(\frac{1}{2} - K_\Gamma\right) \gamma^e h_r \right\rangle_\Gamma - [h]_\Sigma \left\langle \varphi, K_{\Sigma^+} 1 \right\rangle_\Gamma + \langle \varphi, V_\Gamma \partial_{\mathbf{n}}^e h_r \rangle_\Gamma \tag{7.45c}$$

where the unknowns are $\gamma^e h_r \in H_\Sigma^{\frac{1}{2}}(\Gamma)$, $\partial_{\mathbf{n}}^e h_r \in H^{-\frac{1}{2}}(\Gamma)$ *and* $\gamma_N^c \mathbf{H} \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$. *and the test functions are* $v \in H_\Sigma^{\frac{1}{2}}(\Gamma)$, $\mu \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$ *and* $\varphi \in H^{-\frac{1}{2}}(\Gamma)$.

Proof Inserting (7.29) into (7.42) gives the first equation:

$$\begin{aligned}\langle v, \gamma_{\mathbf{n}}^e \mathbf{H}_s\rangle_d - \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \gamma_{\mathbf{t}}^e \mathbf{H}_s \right\rangle_l &= \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l \\ &\quad + \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \right\rangle_l \\ &\quad + \left\langle v, \left(K_\Gamma^* - \frac{1}{2}\right) \partial_{\mathbf{n}}^e h_r \right\rangle_d \\ &\quad + [v]_\Sigma \left\langle 1, K_\Gamma^* \partial_{\mathbf{n}}^e h_r \right\rangle_{d(\Sigma^+)} + [v]_\Sigma \left\langle 1, \partial_{\mathbf{n}}^+ h_r \right\rangle_{d(\Sigma^+)} \\ &\quad + \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma h_r \rangle_l \\ &= \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma} \gamma^e h_r \right\rangle_l \\ &\quad + \tau \left\langle \widetilde{\mathbf{grad}_\Gamma} v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \gamma_N^c \mathbf{H} \right\rangle_l \\ &\quad + \left\langle \left(K_\Gamma - \frac{1}{2}\right) v, \partial_{\mathbf{n}}^e h_r \right\rangle_d \\ &\quad + [v]_\Sigma \left\langle K_{\Sigma^+} 1, \partial_{\mathbf{n}}^e h_r \right\rangle_d + [v]_\Sigma \left\langle 1, \partial_{\mathbf{n}}^+ h_r \right\rangle_{d(\Sigma^+)} \\ &\quad + \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma h_r \rangle_l \end{aligned} \tag{7.46}$$

The quantity $\langle 1, \partial^+_{\mathbf{n}} h_r \rangle_{d(\Sigma^+)}$ should not turn up in the boundary integral equations as this would mean introducing a substantial new quantity and performing complicated integrals on the cutting surface. Fortunately the quantity represents the flow of the magnetic field through the cutting surface and can be removed using the physics of the system.

Maxwell's laws (2.4) give

$$\begin{aligned} \mathbf{curl}\,\mathbf{E} &= -i\omega\mu\mathbf{H} \qquad \text{in } \Omega_e \\ \mathbf{curl}\,\mathbf{H} &= \sigma\mathbf{E} \qquad \text{in } \Omega_c \end{aligned} \tag{7.47}$$

Together with Stokes' law and the transmission properties of the electric field across boundaries (2.9) it is possible to conclude for the flow of the magnetic field:

$$\begin{aligned} &\int_{\Sigma^+} \mathbf{H}_r(x)\cdot\mathbf{n}(x)dS(x) + \int_{\Sigma^+} \mathbf{H}_s(x)\cdot\mathbf{n}(x)dS(x) \\ &\qquad = -\sigma\tau \int_{\Sigma^+} \mathbf{curl}\,\mathbf{E}_r(x)\cdot\mathbf{n}(x)\; dS(x) - \sigma\tau \int_{\Sigma^+} \mathbf{curl}\,\mathbf{E}_s(x)\cdot\mathbf{n}(x)\; dS(x) \\ &\qquad = -\sigma\tau \int_{\partial\Sigma^+} \mathbf{E}_r(x)\cdot\mathbf{t}(x)\; dS(x) - \sigma\tau \int_{\partial\Sigma^+} \mathbf{E}_s(x)\cdot\mathbf{t}(x)\; dS(x) \\ &\qquad = -\frac{\sigma\tau\mu_0}{\mu_c} \int_{\partial\Sigma^+} \mathbf{E}(x)\cdot\mathbf{t}(x)\; dS(x) = -\frac{\tau\mu_0}{\mu_c} \int_{\partial\Sigma^+} \mathbf{curl}\,\mathbf{H}(x)\cdot\mathbf{t}(x)dS(x) \end{aligned} \tag{7.48}$$

where $\mathbf{t}$ is an appropriate tangent vector field to the boundary of the cutting surface. This leads to the following result (which uses Definition 7.10):

$$\left\langle 1, \partial^+_{\mathbf{n}} h_r \right\rangle_{d(\Sigma^+)} = -\left\langle 1, \gamma^+_{\mathbf{n}} \mathbf{H}_s \right\rangle_{d(\Sigma^+)} - \frac{\tau\mu_0}{\mu_c} L \gamma^c_N \mathbf{H} \tag{7.49}$$

Inserting this into the main equation gives:

$$\begin{aligned} \langle v, \gamma^e_{\mathbf{n}}\mathbf{H}_s\rangle_d - \tau\left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \gamma^e_{\mathbf{t}}\mathbf{H}_s \right\rangle_l &= \tau\left\langle \widetilde{\mathbf{grad}_\Gamma} v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma}\gamma^e h_r \right\rangle_l \\ +[v]_\Sigma \left\langle 1, \gamma^+_{\mathbf{n}}\mathbf{H}_s\right\rangle_{d(\Sigma^+)} &\quad + \tau\left\langle \widetilde{\mathbf{grad}_\Gamma} v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\,\gamma^c_N\mathbf{H}\right\rangle_l \\ &\quad + \left\langle \left(K_\Gamma - \frac{1}{2}\right) v, \partial^e_{\mathbf{n}} h_r \right\rangle_d \\ &\quad + [v]_\Sigma \left\langle K_{\Sigma^+} 1, \partial^e_{\mathbf{n}} h_r\right\rangle_d - [v]_\Sigma \frac{\tau\mu_0}{\mu_c} L\gamma^c_N\mathbf{H} \\ &\quad + \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma h_r\rangle_l \end{aligned} \tag{7.50}$$

The last two equations of the theorem are just (7.43) and (7.44). □

The following definition is used in the theorem:

Definition 7.10 *The operator $L : \boldsymbol{H}(\mathrm{div};\Omega) \to \mathbb{R}$ is defined such that it performs the following line integral:*

$$L\gamma_N^c \mathbf{H} = \int_{\partial\Sigma^+} \mathbf{curl}\,\mathbf{H}(x) \cdot \mathbf{t}(x) dS(x) \tag{7.51}$$

Notice that the Neumann trace of the magnetic field is not the curl but the rotated curl. This has to be addressed in an implementation of the line integral operator L.

Also note that the symmetric formulation is not truly symmetric (as it is the case with more simple cohomology groups). However it is largely symmetric with only a small non-symmetric part caused by the line integral operator.

7.3.3 Dependence on the choice of cutting surface

It should be noted here that both formulations are completely independent of the choice of the cutting surface Σ. The only relevant choice is the boundary of the cutting surface. This is because both formulations contain Σ only in integrals of the following type:

$$\langle \varphi, K_{\Sigma^+} 1 \rangle_\Gamma = \int_\Gamma \varphi(x) \int_{\Sigma^+} \partial^e_{\mathbf{n}y} G(x,y) dS(y) dS(x) \tag{7.52}$$

As the divergence of $\mathbf{grad}_y G(x,y)$ is 0 (except on a null set where Γ intersects Σ^+) for any other cutting surface $\Sigma^{+\prime}$ with the same boundary oriented in the same way it holds by Gauss's divergence theorem (ref. e.g. [23, Chapter 12]):

$$\begin{aligned} \int_{\Sigma^+} \partial^e_{\mathbf{n}y} G(x,y) dS(y) - \int_{\Sigma^{+\prime}} \partial^e_{\mathbf{n}y} G(x,y) dS(y) = 0 \\ \int_{\Sigma^+} \partial^e_{\mathbf{n}y} G(x,y) dS(y) = \int_{\Sigma^{+\prime}} \partial^e_{\mathbf{n}y} G(x,y) dS(y) \end{aligned} \tag{7.53}$$

where the minus in the formulation comes from the fact that $\Sigma^{+\prime}$ has to be oriented in the context of Gauss's theorem: the two surfaces have to form the closed boundary of a volume.

Thus the formulation is not dependent on how the cutting surface is chosen, only its boundary matters. This is consistent with [14, pp. 242].

8 Implementation of the method for complicated geometries

This chapter describes the implementation of the boundary element method for complicated geometries outlined in Chapter 7. First the discretization of the spaces is discussed, then the Calderon identities from Lemmas 7.4 and 7.5 are tested with a real-world example on a torus. At last the full eddy current problem on the torus is implemented.

As in Chapter 6 the actual implementations are done with BETL2 [21], meshing is done with Gmsh [11] and visualization is done with Paraview [22].

In this section the domain in question will always be one where the conductor is a torus Γ with just one cutting surface Σ.

8.1 Discretization of the jump spaces

The discretization of the normal trace spaces $H^{\frac{1}{2}}(\Gamma)$ and $H^{-\frac{1}{2}}(\Gamma)$ are discussed in Chapter 6. However if the boundary integral equations from Lemmas 7.4 and 7.5 are to be implemented it is necessary to discretize $H_{\Sigma}^{\frac{1}{2}}(\Gamma')$ restricted to Γ. The space $H_{\Sigma}^{-\frac{1}{2}}(\Gamma')$ does not have to be implemented as it is discontinuous by nature and all parts on the actual cutting surface Σ vanish in the formulation of the Calderon identities.

Consider Definition 8.1:

Definition 8.1 *Let ρ be a function in $H_{\Sigma}^{\frac{1}{2}}(\Gamma')$ with jump of 1 over the cutting surface that integrates to 0.*
This function is called the ansatz function.

An example of such a function can be seen in Figure 8.1. The value of ρ on Σ is not relevant, as this only enters the formulation via the jump of ρ over Σ. The value of ρ on Γ is important in the sense that the solution will depend on it; but the physical quantities do not depend on the choice of ρ as will be apparent by construction.

These facts motivate the following ansatz (hence the name ansatz function for ρ):

$$\gamma^e h_r = u + \alpha\rho \qquad u \in H^{\frac{1}{2}}(\Gamma),\ \alpha \in \mathbb{C} \tag{8.1}$$

With this formulation u can be used with BETL2's usual discretization of boundary spaces and an additional degree of freedom, the jump α over the cutting surface, is introduced (ρ is not an unknown).

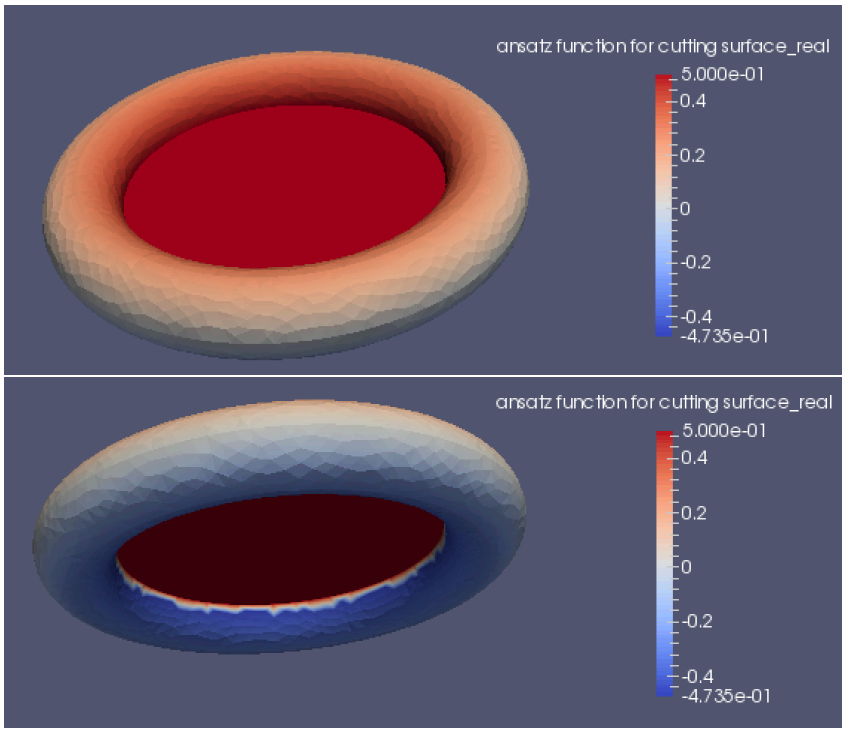

Figure 8.1: An example for a possible choice of the ansatz function ρ. As the chosen visualization method has some difficulties representing discontinuities, the picture of the bottom shows artifacts that stem from this discontinuity. The calculation itself in BETL2 is not affected by this.

8.2 Testing the Calderon identities

The weak formulations of the Calderon identities applied to the ansatz (8.1) produce a linear system of equations for both identities.

8.2.1 Testing the first Calderon identity

(7.26) becomes:

$$\begin{aligned}\langle \varphi, V_\Gamma \psi \rangle_d &= \left\langle \varphi, \left(K_\Gamma - \frac{1}{2} \right) (u + \alpha\rho) \right\rangle_d + \alpha \langle \varphi, K_{\Sigma^+} 1 \rangle_d \\ &= \left\langle \varphi, \left(K_\Gamma - \frac{1}{2} \right) u \right\rangle_d + \alpha \left(\left\langle \varphi, \left(K_\Gamma - \frac{1}{2} \right) \rho \right\rangle_d + \langle \varphi, K_{\Sigma^+} 1 \rangle_d \right)\end{aligned} \tag{8.2}$$

where $u \in H^{\frac{1}{2}}(\Gamma)$ is discussed above and $\psi \in H^{-\frac{1}{2}}(\Gamma)$ represents the Neumann trace of h_r on Γ. The test function is $\varphi \in H^{-\frac{1}{2}}(\Gamma)$.

As BETL2 also offers integrals over discontinuous, piecewise linear spaces, (8.2) can be implemented in the same framework as before. Using the notation from Chapter 6, the following matrices are formulated:

$$\begin{aligned}\tilde{V}^0 &= \Lambda^0_{SL}(\tilde{\mathbf{H}}(0_\Gamma), \tilde{\mathbf{H}}(0_\Gamma)), & \tilde{V}^0 &\in \mathbb{C}^{n_0 \times n_0} \\ \tilde{K}^0 &= \Lambda^0_{DL}(\tilde{\mathbf{H}}(0_\Gamma), \tilde{\mathbf{H}}(1_\Gamma)), & \tilde{K}^0 &\in \mathbb{C}^{n_0 \times n_1} \\ \tilde{K}^0_d &= \Lambda^0_{DL}(\tilde{\mathbf{H}}(0_\Gamma), \tilde{\mathbf{H}}(1^{disc}_\Gamma)), & \tilde{K}^0_d &\in \mathbb{C}^{n_0 \times n_{\text{disc}}} \\ \tilde{K}^0_s &= \Lambda^0_{DL}(\tilde{\mathbf{H}}(0_\Gamma), \tilde{\mathbf{H}}(1_{\Sigma^+})), & \tilde{K}^0_s &\in \mathbb{C}^{n_0 \times n_{1,s}}\end{aligned} \tag{8.3}$$

where a subscript denotes the surface on which the operator is formulated and a superscript 1^{disc} means the space does not consist of continuous piecewise linears but discontinuous piecewise linears. All the discontinuous discretizations have n_{disc} degrees of freedom. $n_{1,s}$ is the dimension of the space $\tilde{\mathbf{H}}(1_{\Sigma^+})$.

This gives the following matrix equation for (8.3):

$$\tilde{V}^0 \psi = \left(\tilde{K}^0 - 0.5M \right) u + \alpha \left(\left(\tilde{K}^0_d - 0.5M \right) \rho + \tilde{K}^0_s 1 \right) \tag{8.4}$$

where M is the respective mass matrix and 1 is a constant vector.

Consider now the physical setup of a constant current j_0 along a wire loop on the inside of the torus (such that the loop has constant distance to Γ) with constant minimal distance from the torus surface Γ. This will produce a magnetic field in Ω_e which will have a potential h_r in Ω'_e obeying the Calderon identities.

The formula for such a magnetic field is known from [18, pp. 181] which gives the Neumann trace. The potential h_r along Γ is found using numerical quadrature. This

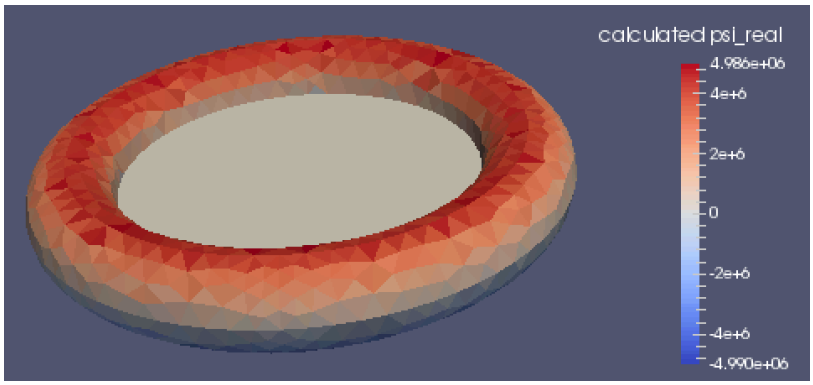

Figure 8.2: A visualization of the calculated Neumann trace of the potential ψ on a torus with 2766 boundary elements.

number of bdry elements	rel. error ψ
232	0.359274
784	0.226869
2766	0.0845532
6150	0.0633667
10106	0.0538717

Table 8.1: $\mathbf{L}^2$ Errors of the method described in this chapter. ψ is compared to the exact Neumann trace known from [18, pp.181].

means there is an exact solution to try (8.4) on.
The method produces the correct solution and converges for an implementation of (8.4) with $\alpha = -j_0 = -10^6$, where the torus has a large radius of $r_1 = 0.5$ and a small radius of $r_2 = 0.1$. A picture of the solution can be seen in Figure 8.2, a convergence table can be seen in Figure 8.1. The linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8.
However it can be seen that there are small artifacts that make the solution look erratic. This behavior gets worse when the grid is refined or the cutting surface is disturbed. This will be discussed later on.

8.2.2 Testing the second Calderon identity

For the second identity that is tested with $H^{\frac{1}{2}}_{\Sigma}(\Gamma)$ the setup is more intricate. In line with the ansatz above it is sufficient to test with functions from $H^{\frac{1}{2}}(\Gamma)$ as well as the

ansatz function ρ.

(7.29) tested with $v \in H^{\frac{1}{2}}(\Gamma)$, becomes:

$$\begin{aligned} \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma (u + \alpha\rho) \rangle_l &= -\left\langle v, \left(K_\Gamma^* + \frac{1}{2}\right)\psi \right\rangle_d \\ \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma u \rangle_l + \alpha \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma \rho \rangle_l &= -\left\langle \left(K_\Gamma + \frac{1}{2}\right) v, \psi \right\rangle_d \end{aligned} \tag{8.5}$$

(7.29) tested with ρ, becomes:

$$\begin{aligned} \langle \mathbf{curl}_\Gamma \rho, \mathbf{A}_l \mathbf{curl}_\Gamma (u + \alpha\rho) \rangle_\Gamma &= -\left\langle \rho, \left(K_\Gamma^* + \frac{1}{2}\right)\psi \right\rangle_d - \langle 1, K_\Gamma^* \psi \rangle_{d(\Sigma^+)} - \beta \\ \langle \mathbf{curl}_\Gamma \rho, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma u \rangle_l + \alpha \langle \mathbf{curl}_\Gamma \rho, \mathbf{A}_\Gamma \mathbf{curl}_\Gamma \rho \rangle_l &= -\left\langle \left(K_\Gamma + \frac{1}{2}\right) \rho, \psi \right\rangle_d - \langle K_{\Sigma^+} 1, \psi \rangle_d - \beta \end{aligned} \tag{8.6}$$

where in both equations the flow of the magnetic field through the cutting surface, β, was used.

The double layer operator is implemented as usual. The hypersingular operator is replaced by the vectorial single layer operator (ref. Theorem 7.8) and implemented as follows:

$$\begin{aligned} \tilde{D}^0 &= G_c^T \Lambda^0_{SL}(\tilde{\mathbf{H}}(\mathrm{div}_\Gamma), \tilde{\mathbf{H}}(\mathrm{div}_\Gamma)) G_c, & \tilde{D}^0 &\in \mathbb{C}^{n_1 \times n_1} \\ \tilde{D}^0_d &= G_c^T \Lambda^0_{SL}(\tilde{\mathbf{H}}(\mathrm{div}_\Gamma), \tilde{\mathbf{H}}(\mathrm{div}^{disc}_\Gamma)) G_c^{disc} & \tilde{D}^0_d &\in \mathbb{C}^{n_1 \times n_{\mathrm{disc}}} \\ \tilde{D}^0_{dd} &= G_c^{disc\,T} \Lambda^0_{SL}(\tilde{\mathbf{H}}(\mathrm{div}^{disc}_\Gamma), \tilde{\mathbf{H}}(\mathrm{div}^{disc}_\Gamma)) G_c^{disc} & \tilde{D}^0_{dd} &\in \mathbb{C}^{n_{\mathrm{disc}} \times n_{\mathrm{disc}}} \end{aligned} \tag{8.7}$$

where G_c is the combinatorial curl as used in Chapter 6 and G_c^{disc} its version on discontinuous spaces. Those are also matrices and their sizes correspond to the respective degrees of freedom of the element spaces.

Written as a matrix equation, (8.6) becomes:

$$\begin{pmatrix} \tilde{D}^0 & \tilde{D}^0_d \rho \\ \rho^T \tilde{D}^{0T}_d & \rho^T \tilde{D}^0_{dd} \rho \end{pmatrix} \begin{pmatrix} u \\ \alpha \end{pmatrix} = \begin{pmatrix} -(\tilde{K}^{0T} + 0.5M)\psi \\ -\rho^T (\tilde{K}^{0T}_d + 0.5M)\psi - \psi^T \tilde{K}^0_s 1 - \beta \end{pmatrix} \tag{8.8}$$

as the hypersingular operator is self-adjoint (this can be seen from its definition in [29, Chapter 6.5]).

The solution on Γ is reclaimed as $u + \alpha\rho$.

The same test as in the last subsection can be performed here. As before, a stabilization condition is needed so that u integrates to 0. A visualization of the result can be seen in Figure 8.3. A table with error values can be seen in Table 8.2. As before, the linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8.

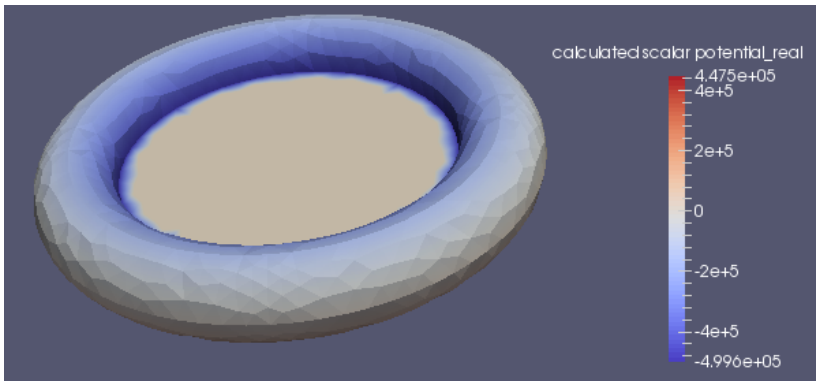

Figure 8.3: A visualization of the calculated Dirichlet trace of the potential on a torus with 2766 boundary elements.

number of bdry elements	rel. error u	rel. error α
232	0.0809576	0.000946661
784	0.0234041	0.00119306
2766	0.00807363	0.0007195
6150	0.00339069	0.000514334
10106	0.00167479	0.000200209

Table 8.2: $\mathbf{L}^2$ Errors of the method described in this chapter. The errors in u and α are listed separately.

8.3 The eddy current problem

No convenient exact solution is available for the full eddy current problem. However, the correctness of the Calderon identities checked in the last section instills confidence in the correctness of the method for eddy current problems as well.

8.3.1 The non-symmetric way

This subsection investigates the implementation of the non-symmetric equations (7.42), (7.43) and (7.44).

Consider the usual ansatz (the C++ code used for the implementation of the ansatz function is explained in the appendix in Section 12.6):

$$\begin{aligned}
\gamma^e h_r &= u + \alpha\rho, \quad u \in H^{\frac{1}{2}}(\Gamma) \\
\partial^e_{\mathbf{n}} h_r &= \psi, \quad \psi \in H^{-\frac{1}{2}}(\Gamma) \\
\gamma^c_N \mathbf{H} &= \eta, \quad \eta \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)
\end{aligned}$$

Denote the boundary conditions from the exciting field by (the C++ code used for the incorporation of the boundary conditions also is explained in the appendix in Section 12.6):

$$\begin{aligned}
\delta &= \gamma^e_{\mathbf{t}} \mathbf{H}_s \\
\nu &= \gamma^e_{\mathbf{n}} \mathbf{H}_s
\end{aligned}$$

(7.42) becomes, tested with $v \in H^{\frac{1}{2}}(\Gamma)$:

$$\begin{aligned}
&\langle v, \nu \rangle_d - \tau \langle \mathbf{grad}_\Gamma v, \mathbf{N}^\kappa \delta \rangle_l = \\
&\qquad \tau \langle \mathbf{grad}_\Gamma v, \mathbf{N}^\kappa \, \mathbf{grad}_\Gamma u \rangle_l + \alpha\tau \left\langle \mathbf{grad}_\Gamma v, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma \rho} \right\rangle_l \\
&\qquad + \tau \langle \mathbf{grad}_\Gamma v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \eta \rangle_l - \langle v, \psi \rangle_d
\end{aligned} \tag{8.9}$$

Tested with ρ it becomes:

$$\begin{aligned}
&\langle \rho, \nu \rangle_d - \tau \left\langle \widetilde{\mathbf{grad}_\Gamma \rho}, \mathbf{N}^\kappa \delta \right\rangle_l = \\
&\qquad \tau \left\langle \widetilde{\mathbf{grad}_\Gamma \rho}, \mathbf{N}^\kappa \, \mathbf{grad}_\Gamma u \right\rangle_l + \alpha\tau \left\langle \widetilde{\mathbf{grad}_\Gamma \rho}, \mathbf{N}^\kappa \widetilde{\mathbf{grad}_\Gamma \rho} \right\rangle_l \\
&\qquad + \tau \left\langle \widetilde{\mathbf{grad}_\Gamma \rho}, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\, \eta \right\rangle_l - \langle \rho, \psi \rangle_d
\end{aligned} \tag{8.10}$$

(7.43) becomes, tested with $\mu \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$:

$$\begin{aligned}\langle \mu, (0.5\,\mathbf{id} - \mathbf{C}^\kappa)\,\delta\rangle_l = \langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\,\mathbf{grad}_\Gamma\, u\rangle_l + \alpha \left\langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\,\widetilde{\mathbf{grad}_\Gamma}\rho \right\rangle_l \\ + \langle \mu, \mathbf{A}^\kappa \eta\rangle_l\end{aligned} \tag{8.11}$$

(7.44) becomes, tested with $\varphi \in H^{-\frac{1}{2}}(\Gamma)$:

$$0 = \left\langle \varphi, \left(K_\Gamma - \frac{1}{2}\right) u \right\rangle_d + \alpha \left\langle \varphi, \left(K_\Gamma - \frac{1}{2}\right)\rho \right\rangle_d + \alpha\,\langle \varphi, K_{\Sigma^+} 1\rangle_d - \langle \varphi, V_\Gamma \psi\rangle_d \tag{8.12}$$

As a matrix equation this becomes:

$$\begin{pmatrix} \mathbf{N} & -\tau\mathbf{C}^T & -M & \tilde{\mathbf{N}} \\ \mathbf{C} & \tilde{A}^\kappa & 0 & \tilde{\mathbf{C}} \\ -\mathbf{K} & 0 & -\tilde{V}^0 & -\tilde{\mathbf{K}} \\ \tilde{\mathbf{N}}^T & -\tau\tilde{\mathbf{C}}^T & -\rho^T M & \tilde{\mathbf{N}}' \end{pmatrix} \begin{pmatrix} u \\ \eta \\ \psi \\ \alpha \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \\ 0 \\ \tilde{\mathbf{f}} \end{pmatrix} \tag{8.13}$$

with M the appropriate mass matrix and the bold letters defined as following:

$$\begin{aligned} \mathbf{N} &:= \tau G_g^T \tilde{N}^\kappa G_g \\ \tilde{\mathbf{N}} &:= \tau G_g^T \tilde{N}^\kappa (\widetilde{\mathbf{grad}_\Gamma}\rho) \\ \tilde{\mathbf{N}}' &:= \tau (\widetilde{\mathbf{grad}_\Gamma}\rho)^T \tilde{N}^\kappa (\widetilde{\mathbf{grad}_\Gamma}\rho) \\ \mathbf{C} &:= (\tilde{C}^\kappa - 0.5M) G_g \\ \tilde{\mathbf{C}} &:= (\tilde{C}^\kappa - 0.5M)(\widetilde{\mathbf{grad}_\Gamma}\rho) \\ \mathbf{K} &:= 0.5M - \tilde{K}^0 \\ \tilde{\mathbf{K}} &:= (0.5M - \tilde{K}_d^0)\rho - \tilde{K}_s^0 1 \\ \mathbf{f} &= M\nu - \tau G_g^T \tilde{N}^\kappa \delta \\ \tilde{\mathbf{f}} &= \rho^T M\nu - \tau (\widetilde{\mathbf{grad}_\Gamma}\rho)^T \tilde{N}^\kappa \delta \\ \mathbf{g} &= (0.5M - \tilde{C}^\kappa)\delta \end{aligned} \tag{8.14}$$

where the BETL2 operators have either been defined in Chapter 6 or in this chapter.

The results can be seen in Figure 8.4. The linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8. The parameters are wire coil radius $b = 0.1$, current along wire coil $j_0 = 10^6$, torus large radius $r_1 = 0.05$, torus small radius $r_2 = 0.01$.

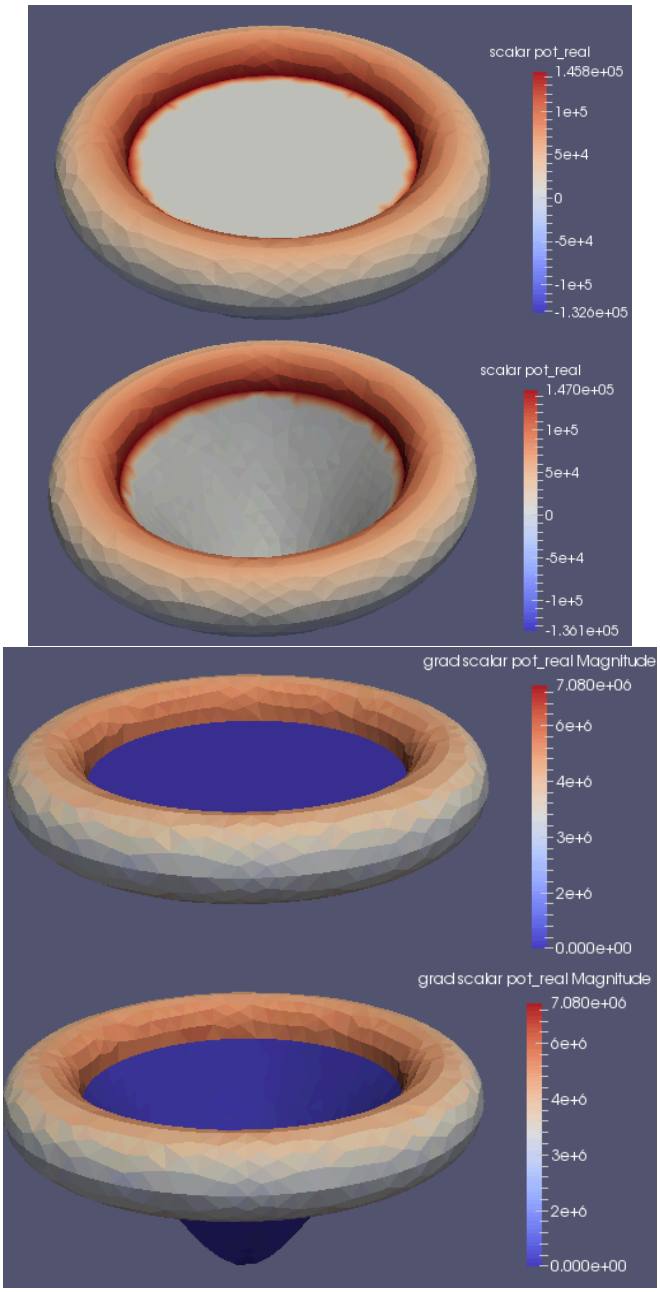

Figure 8.4: The top picture shows the real part of the scalar potential on the torus with two different cutting surfaces, the bottom picture shows the real part of the reaction field $\mathbf{H}_r$ on the torus with two different cutting surfaces (as calculated with the non-symmetric method). The results are independent of the choice of cutting surface.

Note that the result is independent of the choice of cutting surface, as the two different cutting surfaces of Figure 8.4 demonstrate. This is consistent with the physics of the problem, as the cutting surface has no physical significance but is just a mathematical tool to solve the problem. This mathematical fact is shown in Subsection 7.3.3. However they are slightly different and certain artifacts occur. This is for the same reason as in Subsection 8.2.1 and will be discussed later.

8.3.2 The symmetric way

This subsection describes the implementation of the symmetric equations from Theorem 7.9.

The first equation, tested with $v \in H^{\frac{1}{2}}(\Gamma)$, becomes:

$$\begin{aligned}\langle v,\nu\rangle_d - \tau\,\langle \mathbf{grad}_\Gamma\, v, \mathbf{N}^\kappa\delta\rangle_l = \tau\,\langle \mathbf{grad}_\Gamma\, v, \mathbf{N}^\kappa\,\mathbf{grad}_\Gamma\, u\rangle_l + \alpha\tau\left\langle \mathbf{grad}_\Gamma\, v, \mathbf{N}^\kappa\widetilde{\mathbf{grad}_\Gamma}\rho\right\rangle_l \\ + \tau\,\langle \mathbf{grad}_\Gamma\, v, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\,\eta\rangle_l + \left\langle v, \left(K_\Gamma^* - \frac{1}{2}\right)\psi\right\rangle_d \\ + \langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma\mathbf{curl}_\Gamma u\rangle_l + \alpha\left\langle \mathbf{curl}_\Gamma v, \mathbf{A}_\Gamma\widetilde{\mathbf{curl}_\Gamma}\rho\right\rangle_l\end{aligned} \tag{8.15}$$

The first equation, tested with ρ, becomes:

$$\begin{aligned}\langle \rho,\nu\rangle_d - \tau\left\langle \widetilde{\mathbf{grad}_\Gamma}\rho, \mathbf{N}^\kappa\delta\right\rangle_l = \tau\left\langle \widetilde{\mathbf{grad}_\Gamma}\rho, \mathbf{N}^\kappa\,\mathbf{grad}_\Gamma\, u\right\rangle_l + \alpha\tau\left\langle \widetilde{\mathbf{grad}_\Gamma}\rho, \mathbf{N}^\kappa\widetilde{\mathbf{grad}_\Gamma}\rho\right\rangle_l \\ + \left\langle 1, \gamma_{\mathbf{n}}^+\mathbf{H}_s\right\rangle_{d(\Sigma^+)} \qquad + \tau\left\langle \widetilde{\mathbf{grad}_\Gamma}\rho, (\mathbf{B}^\kappa + 0.5\,\mathbf{id})\,\eta\right\rangle_l + \left\langle \rho, \left(K_\Gamma^* - \frac{1}{2}\right)\psi\right\rangle_d \\ + \left\langle \widetilde{\mathbf{curl}_\Gamma}\rho, \mathbf{A}_\Gamma\mathbf{curl}_\Gamma u\right\rangle_l + \alpha\left\langle \widetilde{\mathbf{curl}_\Gamma}\rho, \mathbf{A}_\Gamma\widetilde{\mathbf{curl}_\Gamma}\rho\right\rangle_l \\ + \langle K_{\Sigma^+}1, \psi\rangle_d - \frac{\tau\mu_0}{\mu_c}L\gamma_N^c\eta\end{aligned} \tag{8.16}$$

The second equation, tested with $\mu \in \boldsymbol{H}_{||}^{-\frac{1}{2}}(\mathrm{div}_\Gamma, \Gamma)$, becomes:

$$\begin{aligned}\langle \mu, (0.5\,\mathbf{id} - \mathbf{C}^\kappa)\,\delta\rangle_l = \langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\,\mathbf{grad}_\Gamma\, u\rangle_l + \alpha\left\langle \mu, (\mathbf{C}^\kappa - 0.5\,\mathbf{id})\,\widetilde{\mathbf{grad}_\Gamma}\rho\right\rangle_l \\ + \langle \mu, \mathbf{A}^\kappa\eta\rangle_l\end{aligned} \tag{8.17}$$

The third equation, tested with $\varphi \in H^{-\frac{1}{2}}(\Gamma)$, becomes:

$$0 = \left\langle \varphi, \left(\frac{1}{2} - K_\Gamma\right) u \right\rangle_\Gamma + \alpha \left\langle \varphi, \left(\frac{1}{2} - K_\Gamma\right) \rho \right\rangle_\Gamma - \alpha \langle \varphi, K_{\Sigma^+} 1 \rangle_\Gamma + \langle \varphi, V_\Gamma \psi \rangle_\Gamma \quad (8.18)$$

As a matrix equation this becomes:

$$\begin{pmatrix} \mathbf{N} & -\mathbf{C}^T & -\mathbf{K}^T & \tilde{\mathbf{N}} \\ \mathbf{C} & \tau \tilde{A}^\kappa & 0 & \tilde{\mathbf{C}} \\ \mathbf{K} & 0 & \tilde{V}^0 & \tilde{\mathbf{K}} \\ \tilde{\mathbf{N}}^T & -\tilde{\mathbf{C}}^T - \frac{\tau\mu_0}{\mu_c} L & -\tilde{\mathbf{K}}^T & \tilde{\mathbf{N}}' \end{pmatrix} \begin{pmatrix} u \\ \eta \\ \psi \\ \alpha \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \\ 0 \\ \tilde{\mathbf{f}} \end{pmatrix} \quad (8.19)$$

with L the line integral operator (its implementation is discussed in Section 12.7 in the appendix), M the appropriate mass matrix and the bold letters defined as following:

$$\begin{aligned} \mathbf{N} &:= \tau G_g^T \tilde{N}^\kappa G_g + \tilde{D}^0 \\ \tilde{\mathbf{N}} &:= \tau G_g^T \tilde{N}^\kappa (\widetilde{\mathbf{grad}_\Gamma \rho}) + \tilde{D}_d^0 \rho \\ \tilde{\mathbf{N}}' &:= \tau (\widetilde{\mathbf{grad}_\Gamma \rho})^T \tilde{N}^\kappa (\widetilde{\mathbf{grad}_\Gamma \rho}) + \rho^T \tilde{D}_{dd}^0 \rho \\ \mathbf{C} &:= \tau (\tilde{C}^\kappa - 0.5M) G_g \\ \tilde{\mathbf{C}} &:= \tau (\tilde{C}^\kappa - 0.5M) (\widetilde{\mathbf{grad}_\Gamma \rho}) \\ \mathbf{K} &:= 0.5M - \tilde{K}^0 \\ \tilde{\mathbf{K}} &:= (0.5M - \tilde{K}_d^0)\rho - \tilde{K}_s^0 1 \\ \mathbf{f} &= M\nu - \tau G_g^T \tilde{N}^\kappa \delta \\ \tilde{\mathbf{f}} &= \rho^T M \nu - \tau (\widetilde{\mathbf{grad}_\Gamma \rho})^T \tilde{N}^\kappa \delta + \beta_s \\ \mathbf{g} &= \tau (0.5M - \tilde{C}^\kappa) \delta \end{aligned} \quad (8.20)$$

where β_s is the flow of the exciting field through the cutting surface.

The results can be seen in Figure 8.5. The linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8. The parameters are wire coil radius $b = 0.1$, current along wire coil $j_0 = 10^6$, torus large radius $r_1 = 0.05$, torus small radius $r_2 = 0.01$.

It is worth noting that the results are slightly different for plain and distorted cutting surfaces. Again certain artifacts occur. This is for the same reason as in Subsection 8.2.1 and will be discussed later.

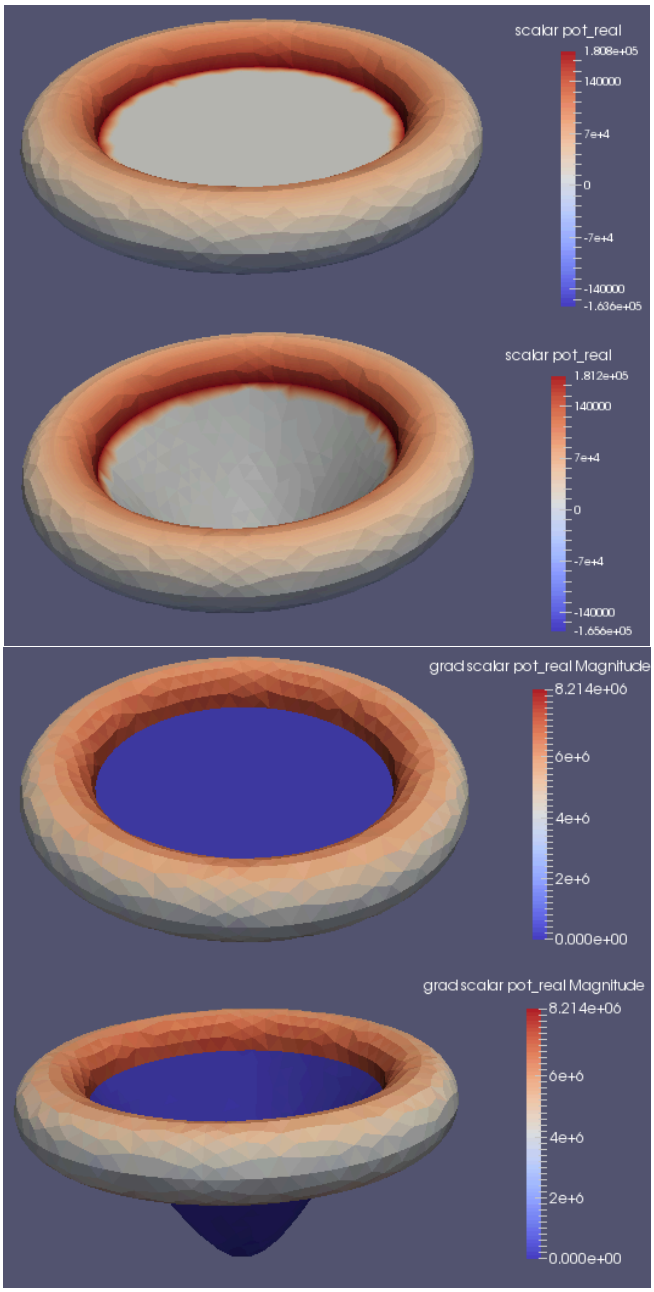

Figure 8.5: The top picture shows the real part of the scalar potential on the torus with two different cutting surfaces, the bottom picture shows the real part of the reaction field $\mathbf{H}_r$ on the torus with two different cutting surfaces (as calculated with the symmetric method). The results are independent of the choice of cutting surface.

9 Removing the cutting surfaces

In Chapters 7 and 8 a method for general geometries was formulated. However, for the case with nontrivial first cohomology group of Ω_e the cutting surface Σ has to be introduced. It serves no physical purpose and there is a lot of freedom in choosing it – it is arbitrary to a certain degree as shown in Subsection 7.3.3.
This suggests that they can somehow be avoided completely in the formulation of the method. This chapter deals with removing integrals over Σ from the formulation in Chapters 7 and 8 entirely.

9.1 Theory

The integral that has to be simplified is:

$$\left\langle u, K^0_{\Sigma^+} 1 \right\rangle_d = \int_\Gamma \int_{\Sigma^+} \partial^+_{\mathbf{n}_y} G_0(x,y) u(x) dS(y) dS(x) \tag{9.1}$$

As they will be frequently used in this chapter, boundary cycles are defined here:

Definition 9.1 *Let Σ^+ be the cutting surface oriented upwards as used in previous chapters. Define:*

$$c := \partial \Sigma^+ \tag{9.2}$$

Those are the boundaries of the cutting surfaces. They are boundary cycles in the first homology group of Γ of which the cohomology group is the dual. As the cutting surfaces can be varied, the cycles can also be replaced by any other homologous cycle.

The double layer potential can not be reduced to c using Stokes's theorem, as $\mathbf{grad}_y G_0(x,y)$ has no vector potential but only a scalar potential. Let it be at first assumed that $x \notin \Sigma$. This holds almost everywhere, except for the points where Σ is connected to Γ.

The gradient of the kernel $\mathbf{grad}_y G_0(x,y)$ is just the electric field at y generated by a point charge of 1 or -1 at x, depending on the orientation of the surface. This allows the following theorem to be used:

Theorem 9.2 *Let e be an electrostatic point charge and $\mathbf{E}(x)$ the electric field it generates.*

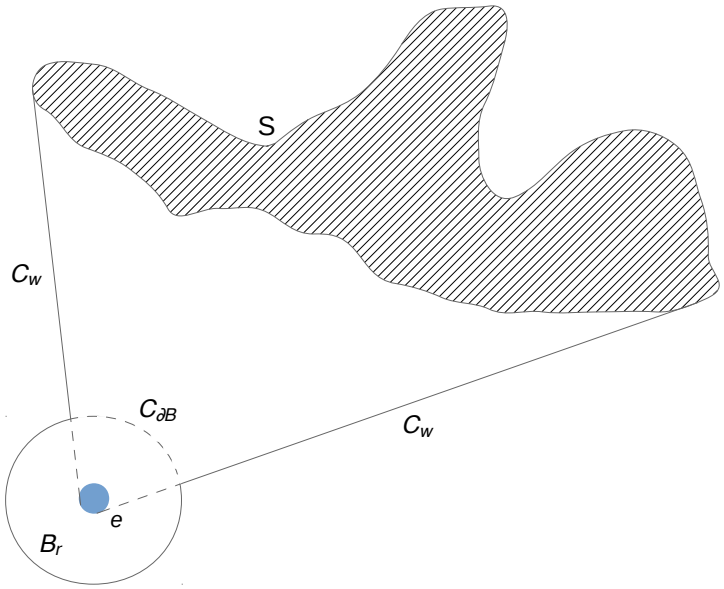

Figure 9.1: A sketch of the cone construction used in the proof of Theorem 9.2

Let S be a surface separated from the charge e by a hyperplane.[1]
Let Ψ be the solid angle of the boundary of S over the point of the charge. The solid angle is a generalization of the 2D angle to $\mathbb{R}^3$. It is the area on the unit sphere covered by the projection of a surface onto the unit sphere (ref. [30]), or as in this case, the area circumscribed by the projection of a curve onto the unit sphere.
Then it holds:

$$\int_S \mathbf{n}(x) \cdot \mathbf{E}(x)\, ds_x = \frac{\Psi}{4\pi} e$$

Proof Assume the charge and the surface are separated by some hyperplane $H \in \mathbb{R}^3$.

Consider a ball B_r with radius $r > 0$ centered around the charge e fully contained in the charge's half-space. There is a cone C of ∂S over the position of the charge e. Without loss of generality, the cone does not intersect S except at ∂S (the same argument holds even with intersections, but is more cumbersome). Where the cone intersects the ball B_r it forms a surface $C_{\partial B}$ with area Ψr^2. The side of the cone up to the intersection with B_r is called C_w. A sketch of the situation can be seen in Figure 9.1.

For notational convenience, the volume contained by $C_{\partial B}$, C_w and S is called V.

[1] This is not a harsh restriction. In most cases it should be possible to choose the surface like that.

Now, Gauss's electric field law (ref. [12, pp. 67] which uses different scaling) states that the electric flux surface integral over a closed surface is just the sum of all charges enclosed by the surface. This gives:

$$\int_{\partial B_r} \mathbf{n}(x) \cdot \mathbf{E}(x)\, dS(x) = e \tag{9.3}$$

Thus, because of symmetry:

$$\int_{C_{\partial B}} \mathbf{n}(x) \cdot \mathbf{E}(x)\, dS(x) = \frac{\Psi r^2}{4\pi r^2} e = \frac{\Psi}{4\pi} e \tag{9.4}$$

Because V contains no charges, clearly the flux integral over ∂V is 0. Also, because of how the cone was constructed, all field lines of $\mathbf{E}$ are parallel to C_w. This leads to the flux integral of the field over the surface C_w being 0. This gives:

$$\begin{aligned} 0 = \int_{\partial V} \mathbf{n}(x) \cdot \mathbf{E}(x)\, ds_x &= \left(-\int_{C_{\partial B}} + \int_{C_w} + \int_S \right) \mathbf{n}(x) \cdot \mathbf{E}(x)\, dS(x) \\ \int_S \mathbf{n}(x) \cdot \mathbf{E}(x)\, dS(x) &= \int_{C_{\partial B}} \mathbf{n}(x) \cdot \mathbf{E}(x)\, dS(x) = \frac{\Psi}{4\pi} e \end{aligned} \tag{9.5}$$

(note: the first minus sign is due to different orientation of $C_{\partial B}$ depending on which volume it bounds)

This proves the theorem.[2]

□

This reduces the integral in (9.1) to the following integral:

$$\left\langle u, K^0_{\Sigma^+} 1 \right\rangle_d = \frac{1}{4\pi} \int_\Gamma \mp \Psi_c(x) u(x) dS(x) \tag{9.6}$$

where $\Psi_L(x)$ is the solid angle created by the loop L over the point x and the $\mp$ depends on the orientation of Σ^+ in relation to x. If x is below Σ the situation corresponds to a negative electrostatic charge, if it is above the situation corresponds to a positive electrostatic charge. This is well-defined as the theorem assumes the point charge and the surface are separated by a hyperplane, so x is either above or below the surface.

It remains to handle the cases where $x = y$ can not be disregarded. This will only occur on the intersection of Σ^+ and Γ. This is a null set, so it will not matter in the second integration.

[2]Some ideas for this proof come from a conversation with Nicolaus Heuer [13].

With these simplifications, the whole method has been reduced to just evaluations on Γ and the boundary of the cutting surface, c, whose dual cocycles generate the first cohomology group of Ω_e. This is highly relevant; there are efficient algorithms for the computation of these cycles, (as can be seen e.g. in [16]). Those algorithms are less expensive than constructing the whole cutting surface.

The same properties as in Chapter 8 still hold, as all the operators are still the same mathematically. There might be numerical issues which depend on how the solid angle is implemented.

9.2 Algorithm

The only part of last section's method of removing cutting surfaces that requires an algorithm that is not already used in this thesis is the calculation of the solid angle of a point over a cutting surface. To that end, a simple algorithm will be proposed in this section.

It is assumed that the input corresponding to the boundary cycle c is a polygon `c` approximating c given in the form of a list of vertices. It is also assumed that the point over which the solid angle is formed is 0 (otherwise a simple translation has to be made). Then the polygon can be projected onto the sphere with the algorithm from Code Snippet 9.1.

```
1 for each vertex v in c do:
2   v := v/norm(v)
3 end for
```

Code Snippet 9.1: Projecting the boundary of the cutting surface to the unit sphere

It remains to calculate the area enclosed by the projected polygon. This is the solid angle. Multiple approaches exist for this. If the polygon is convex there is a simple algorithm depending only on the enclosed angles (where the formula is taken from [31]). It can be seen in Code Snippet 9.2.

```
1 angles := 0
2 nVertices := 0
3 for each vertex v in c do:
4   angles += enclosedAngleAt(v)
5   nVertices += 1
6 end for
7 area := (angles - (nVertices-2)*Pi)
```

Code Snippet 9.2: Calculating the area of the projected cutting surface

The algorithm is however very unstable as two large numbers are subtracted from each other and the result is small. If Boost is used for the implementation, its function [10] can be used to calculate the area of an arbitrary spherical polygon. In general the approximative spherical polygon area formula from [6, pp. 6] can be used – it is rather stable and quick to compute as its compute time is linear in the number of edges of the polygon.

Additional care has to be taken on the intersection of Σ^+ and Γ (which was disregarded in the theorem due to it being a null set). The calculation of the solid angle relying on polygon area will not be possible as the boundary is infinitesimally close to the evaluation point. It is however possible to just calculate the limits of the solid angle by hand. In the torus example used in Chapter 8 and others where the tangent spaces of Σ^+ and Γ are orthogonal at the intersection the limit is just π. This can be seen by imagining a small ball around the evaluation point approaching c from straight above or straight below: Looking at the situation from infinitesimally close, Σ^+ looks like a planar half-space orthogonal to the direction of approach. Its solid angle will cover a quarter of the surface area of the small ball.

Due to the $\mp$ in (9.6) this will lead to a discontinuity. This discontinuity will be discussed in Chapter 10.

10 Preprocessing

The method described in Chapter 8 works, but has one flaw: As mentioned in Subsection 8.2.1, the visible artifacts in Figure 8.2 suggest there is a stability problem. This carries over to the eddy current problem: Small disturbances in the grid will produce different results.

This chapter is about remedying this problem.

10.1 Identifying the problem

The reason for the instabilities lie in the formulation (8.4) which is reprinted here for convenience:

$$\tilde{V}^0\psi = \left(\tilde{K}^0 - 0.5M\right) u + \alpha \left(\left(\tilde{K}_d^0 - 0.5M\right)\rho + \tilde{K}_s^0 1\right) \tag{10.1}$$

As the single layer operator V maps into $H^{\frac{1}{2}}(\Gamma)$ (ref. [29, p. 119]) $u \in H^{\frac{1}{2}}(\Gamma)$ and $K : H^{\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma)$ (ref. [29, p. 149]) by choice this means the following expression must also be in $H^{\frac{1}{2}}(\Gamma)$:

$$(K_\Gamma - 0.5)\,\rho + K_{\Sigma^+} 1 \tag{10.2}$$

However when actually calculating this in a numerical setting, evaluating this expression involves the subtraction of two discontinuous functions, ρ and $K_{\Sigma^+}1$ where the result is continuous. Cancellation errors will occur here and are probably what lead to instabilities in Chapter 8.

There is still ample freedom in choosing ρ though that wasn't used at all in Chapter 8. The only conditions that ρ must fulfil are $\rho \in H_\Sigma^{\frac{1}{2}}(\Gamma')$ and that ρ integrates to 0.

It suggests itself to use this freedom to construct a new ansatz function ρ_p that solves the stability issue:

Definition 10.1 *The new ansatz function $\rho_p \in H_\Sigma^{\frac{1}{2}}(\Gamma')$ is defined such that:*

$$\rho_p = -\left(K_\Gamma - 0.5\right)^{-1} K_{\Sigma^+} 1 \tag{10.3}$$

With this new function the formulation (8.4) reduces to:

$$\tilde{V}^0\psi = \left(\tilde{K}^0 - 0.5M\right)u \tag{10.4}$$

Thus the only discontinuities that turn up are in the construction of the solution $u + \alpha\rho_p$ after solving the boundary element equations.

The ansatz function ρ_p itself is calculated by testing (10.2) with functions from $H_{\Sigma}^{\frac{1}{2}}(\Gamma')$ leading to a second-kind formulation. In the language of Chapter 8 this leads to the following matrix equation:

$$\begin{pmatrix} \tilde{K}_2^0 - 0.5M & \left(\tilde{K}_{2,d}^0 - 0.5M\right)\rho \\ \rho^T\left(\tilde{K}_{d,2}^0 - 0.5M\right) & \rho^T\left(\tilde{K}_{d,2,d}^0 - 0.5M\right)\rho \end{pmatrix} \begin{pmatrix} r \\ * \end{pmatrix} = \begin{pmatrix} \tilde{K}_{2s}^0 1 \\ \rho^T \tilde{K}_{d,2s}^0 1 \end{pmatrix} \tag{10.5}$$

$* \in \mathbb{C}$ is a dummy variable that is discarded after the calculation (the exact solution is 1), $r \in \tilde{\mathbf{H}}(1)$ and $\rho_p = r + \rho$. A stabilizer is needed to make sure that r integrates to 0.

The following matrices were used:

$$\begin{aligned}
\tilde{K}_2^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma), \tilde{\mathbf{H}}(1_\Gamma)), & \tilde{K}_2^0 &\in \mathbb{C}^{n_1 \times n_1} \\
\tilde{K}_{2,d}^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma), \tilde{\mathbf{H}}(1_\Gamma^{disc})), & \tilde{K}_{2,d}^0 &\in \mathbb{C}^{n_1 \times n_{\text{disc}}} \\
\tilde{K}_{d,2}^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma^{disc}), \tilde{\mathbf{H}}(1_\Gamma)), & \tilde{K}_{d,2}^0 &\in \mathbb{C}^{n_{\text{disc}} \times n_1} \\
\tilde{K}_{d,2,d}^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma^{disc}), \tilde{\mathbf{H}}(1_\Gamma^{disc})), & \tilde{K}_{d,2,d}^0 &\in \mathbb{C}^{n_{\text{disc}} \times n_{\text{disc}}} \\
\tilde{K}_{2s}^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma), \tilde{\mathbf{H}}(1_{\Sigma^+})), & \tilde{K}_s^0 &\in \mathbb{C}^{n_1 \times n_{1,s}} \\
\tilde{K}_{d,2s}^0 &= \Lambda_{DL}^0(\tilde{\mathbf{H}}(1_\Gamma^{disc}), \tilde{\mathbf{H}}(1_{\Sigma^+})), & \tilde{K}_s^0 &\in \mathbb{C}^{n_{\text{disc}} \times n_{1,s}}
\end{aligned} \tag{10.6}$$

and M is the appropriate mass matrix. The matrices defined on Σ^+ are constructed with [6]'s method from Chapter 9.

This newly constructed ansatz function is independent of any boundary conditions; it only depends on the geometry of the problem. Thus it can be used to simplify the methods from Chapter 9 without adding much to the computation time; it can be calculated ahead of time.

10.2 Testing the first Calderon identity

The new matrix equation for testing the first Calderon identity is (10.4). Consider again the physical setup of a constant current j_0 along a wire loop on the inside of the torus (such that the loop has constant distance to Γ) with constant minimal distance from the torus surface Γ. The method was calculated for $\alpha = -j_0 = -10^6$, where the torus has

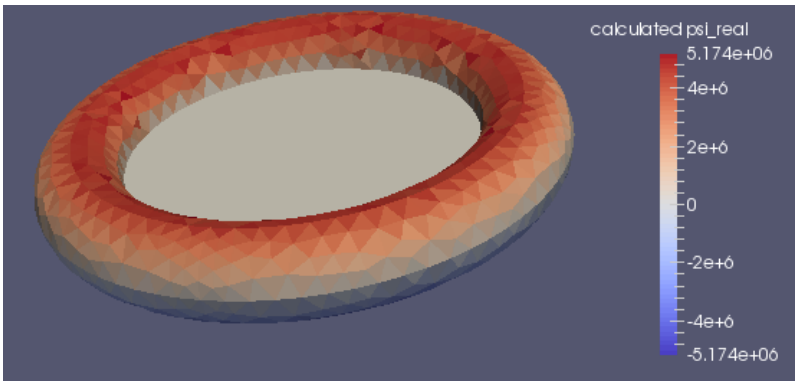

Figure 10.1: A visualization of the calculated Neumann trace of the potential ψ on a torus with 2766 boundary elements and the new ansatz function.

number of bdry elements	rel. error ψ
232	0.364728
784	0.153359
2766	0.11491
6150	0.0987014

Table 10.1: $\mathbf{L}^2$ Errors of the method with the new ansatz function described in this chapter. ψ is compared to the exact Neumann trace known from [18, pp.181].

a large radius of $r_1 = 0.5$ and a small radius of $r_2 = 0.1$. A picture of the solution can be seen in Figure 10.1, a convergence table can be seen in Table 10.1. The linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8.
As can be seen the convergence in L^2 norm is slightly worse, however there are less artifacts when refining the mesh or disturbing the cutting surface.

10.3 Solving the eddy current problem the symmetric way

The eddy current problem is massively simplified as well using the new ansatz obtained with preprocessing. Using ρ_p instead of ρ makes $\tilde{K}$ in (8.20) disappear. The new matrix equation for the symmetric solution to the eddy current system is:

$$\begin{pmatrix} \mathbf{N} & -\mathbf{C}^T & -\mathbf{K}^T & \tilde{\mathbf{N}} \\ \mathbf{C} & \tau\tilde{A}^{\kappa} & 0 & \tilde{\mathbf{C}} \\ \mathbf{K} & 0 & \tilde{V}^0 & 0 \\ \tilde{\mathbf{N}}^T & -\tilde{\mathbf{C}}^T - \frac{\tau\mu_0}{\mu_c}L & 0 & \tilde{\mathbf{N}}' \end{pmatrix} \begin{pmatrix} u \\ \eta \\ \psi \\ \alpha \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \\ 0 \\ \tilde{\mathbf{f}} \end{pmatrix} \tag{10.7}$$

where in all definitions ρ has been replaced with ρ_p.

The results can be seen in Figure 10.2. The linear system was solved with Eigen's `partialPivLu` [8]. The C++ code that is used to produce the visualization is explained in the appendix in Section 12.8. The parameters are wire coil radius $b = 0.1$, current along wire coil $j_0 = 10^6$, torus large radius $r_1 = 0.05$, torus small radius $r_2 = 0.01$.

Here no exact solution is available but it can be noted that the solution is much stabler to changes in the grid and the cutting surface than with the previous ansatz function.

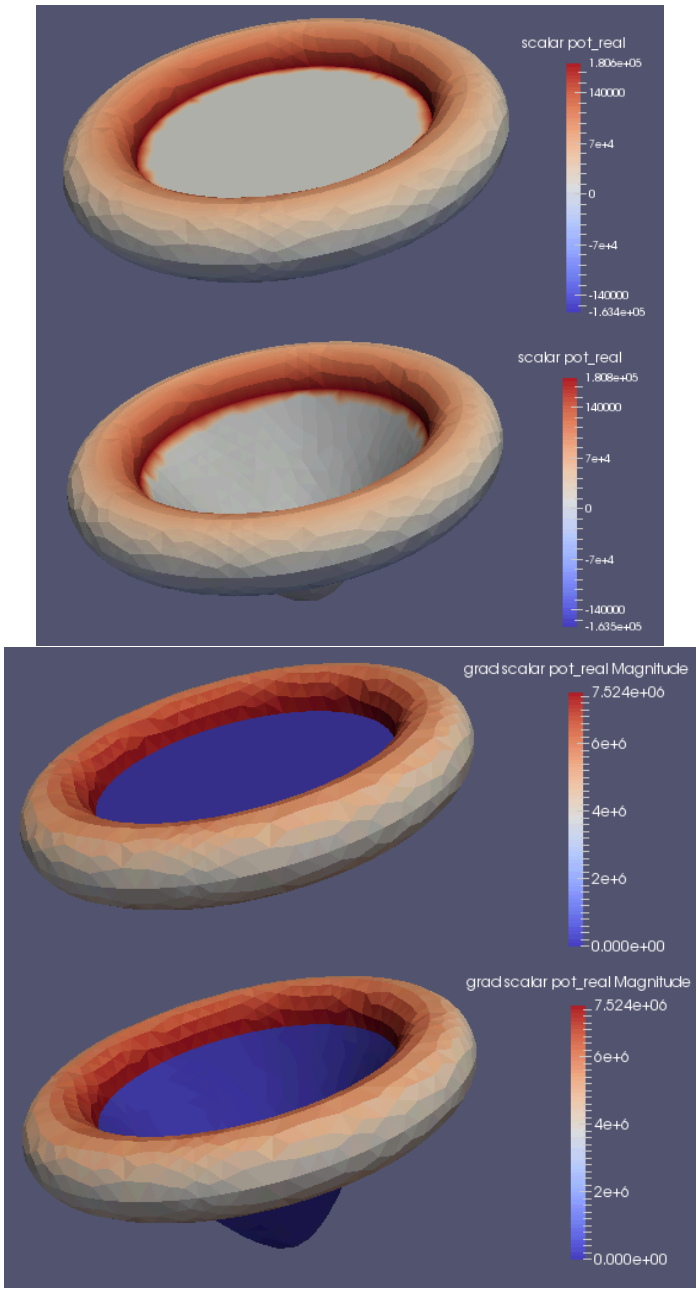

Figure 10.2: The top picture shows the real part of the scalar potential on the torus with two different cutting surfaces, the bottom picture shows the real part of the reaction field $\mathbf{H}_r$ on the torus with two different cutting surfaces (as calculated with the symmetric method with the new ansatz). The results are independent of the choice of cutting surface.

11 Conclusion

This chapter will wrap up the master's thesis, repeating its most important points and produce an outlook on what further work can be done.

This master's thesis introduced the magnetic field scalar potential approach for solving quasi-stationary eddy current problems outlined in [14]. The approach solves the internal problem using the normal vectorial Maxwell equations and vectorial boundary element methods. The external problem is solved as a scalar Laplace equation using scalar boundary element methods.

The theory was formulated for objects with simple geometries using standard scalar potentials and then for more intricate geometries using discontinuous scalar potentials. The scalar potentials depend on cutting surfaces for more intricate geometries which are used to trivialize the first cohomology group of the nonconductive region Ω_e.

Then boundary integral equations were formulated and simulated using BETL2 [21] for the sphere as well as for the torus. For the sphere a symmetric coupling was used, for the torus a symmetric and a non-symmetric coupling were implemented.
The results conform to the correct exact solutions where available. However the calculations suffer from a lack of stability because of cancellation caused by the handling of discontinuous functions.

Additionally, it was shown that the method is not only independent of the chosen cutting surfaces for complicated geometries, but that the method can be formulated to not need any calculation on the cutting surfaces whatsoever. Chapter 9 reduces the whole method to boundaries of cutting surfaces only. These boundaries can be calculated in reasonable time using established algorithms.

At last the instability problems from Chapter 8 were addressed and a preprocessing method was introduced using some freedoms in the choice of ansatz functions that were not yet exploited. This preprocessing makes the method more stable and also makes the matrix in the symmetric eddy current approach less dense.

12 Appendix

The appendix consists mostly of code snippets that illustrate the way the operators used in this thesis are actually implemented in BETL2 using C++ code. The appendix itself provides only little context, but the different code snippets are referenced throughout the main body of the thesis.

The BETL2 revision used for the code is 73355M.[1]

All codes in this section are heavily influenced by the examples included in BETL2 [21] and often directly taken from included applications. The examples are written either by Dr. Lars Kielhorn [20] or Elke Spindler [28]. Calculus and vector calculus identities needed are common knowledge from [23] or [12, Back matter].

12.1 Implementation of finite element spaces

BETL2 offers four basic finite element space types that are used in this thesis. Linear and constant Lagrangian elements (scalar) and curl- and div-type edge elements (vectorial). All the piecewise linear finite element spaces can be implemented as continuous (which assigns only one degree of freedom for each edge or vertex) or discontinuous (which assigns each edge or vertex multiple degrees of freedom, one for each adjacent triangle).

The finite element spaces are set up by first defining a finite element basis type, then defining a type for the `DofHandler` (where Dof stands for degrees of freedom), instantiating the `DofHandler` and then distribute the dofs for some grid structure. A code example for this is provided in Code Snippet 12.1.

```
1 //set up element spaces
2 typedef fe::FEBasis< fe::Linear, fe::FEBasisType::Lagrange >
      fe_basis_lagrange1_t;
3 typedef fe::FEBasis< fe::Constant, fe::FEBasisType::Lagrange >
      fe_basis_lagrange0_t;
4 typedef fe::FEBasis< fe::Linear, fe::FEBasisType::Div >
      fe_basis_div_t;
5 typedef fe::FEBasis< fe::Linear, fe::FEBasisType::Curl >
      fe_basis_curl_t;
6
```

[1] In the kernel integration routine, the scalar multiplication function of vectors has to be set to do a normal scalar multiplication, not a complex one.

```
7  //set up dofhandlers
8  typedef fe::DofHandler< fe_basis_lagrange1_t,
       fe::FESContinuity::Continuous, grid_factory_t >
       dofhandler_lagrange1_t;
9  typedef fe::DofHandler< fe_basis_lagrange1_t,
       fe::FESContinuity::Discontinuous, grid_factory_t >
       dofhandler_lagrange1_discontin_t;
10 typedef fe::DofHandler< fe_basis_lagrange0_t,
       fe::FESContinuity::Discontinuous, grid_factory_t >
       dofhandler_lagrange0_t;
11 typedef fe::DofHandler< fe_basis_div_t,
       fe::FESContinuity::Continuous, grid_factory_t > dofhandler_div_t;
12 typedef fe::DofHandler< fe_basis_div_t,
       fe::FESContinuity::Discontinuous, grid_factory_t >
       dofhandler_div_discontin_t;
13 typedef fe::DofHandler< fe_basis_curl_t,
       fe::FESContinuity::Continuous, grid_factory_t > dofhandler_curl_t;
14 typedef fe::DofHandler< fe_basis_curl_t,
       fe::FESContinuity::Discontinuous, grid_factory_t >
       dofhandler_curl_discontin_t;
15
16 //instantiate dofhandlers
17 dofhandler_lagrange1_t dofhandler_lagrange1;
18 dofhandler_lagrange1_discontin_t dofhandler_lagrange1_discontin;
19 dofhandler_lagrange0_t dofhandler_lagrange0;
20 dofhandler_div_t dofhandler_div;
21 dofhandler_div_discontin_t dofhandler_div_discontin;
22 dofhandler_curl_t dofhandler_curl;
23 dofhandler_curl_discontin_t dofhandler_curl_discontin;
24
25 //distribute dofs
26 dofhandler_lagrange1.distributeDofs(grid_factory);
27 dofhandler_lagrange1_discontin.distributeDofs(grid_factory);
28 dofhandler_lagrange0.distributeDofs(grid_factory);
29 dofhandler_div.distributeDofs(grid_factory);
30 dofhandler_div_discontin.distributeDofs(grid_factory);
31 dofhandler_curl.distributeDofs(grid_factory);
32 dofhandler_curl_discontin.distributeDofs(grid_factory);
```

Code Snippet 12.1: The process needed to set up finite element spaces and distribute the degrees of freedom in BETL2 for a grid with structure saved in `grid_factory`

To formulate certain operators only on parts of the grid (as it is done when using cutting surfaces in Chapter 8), those surfaces have to be marked in the mesh file using Gmsh physical surfaces. The class `MultiInterface` handles the reading of the Gmsh markings, and the extracting of the actual FE subspaces is shown in Code Snippet 12.2.

```
//surface dof markers & constrfespaces
typedef fe::SurfaceDofMarker<dofhandler_lagrange1_t::fespace_t,
    MultiItf_t, grid_factory_t> surf_marker_lagrange1_t;
surf_marker_lagrange1_t
    surf_marker_lagrange1(dofhandler_lagrange1.fespace(), MultiItf,
    grid_factory);
surf_marker_lagrange1.mark();
const auto& itf_fespace_map_lagrange1 =
    surf_marker_lagrange1.FESpaceMap();

//get handlers to fespaces
typedef dofhandler_lagrange1_t::fespace_t fespace_lagrange1_all_t;
fespace_lagrange1_all_t& fespace_lagrange1_all =
    dofhandler_lagrange1.fespace();

typedef surf_marker_lagrange1_t::constrFESpace_t
    fespace_lagrange1_constr_t;
fespace_lagrange1_constr_t& fespace_lagrange1_gamma =
    *itf_fespace_map_lagrange1.at(index_gamma);
fespace_lagrange1_constr_t& fespace_lagrange1_sigma =
    *itf_fespace_map_lagrange1.at(index_sigma);
```

Code Snippet 12.2: How to extract finite element subspaces in BETL2
The grid has already been read in and the indices for Γ and Σ^+ have been saved in `index_gamma` and `index_sigma`.
The process is only shown for `fe_basis_lagrange1_t` but is exactly the same for the other spaces. The results are the three finite element spaces `fespace_lagrange1_all`, `fespace_lagrange1_gamma` and `fespace_lagrange1_sigma`.

To embed the continuous finite element spaces into the discontinuous finite element spaces, special sparse matrices for continuous embeddings are constructed. Those are not built into BETL2 at the moment and they have to be implemented using BETL2's `sparse_base_operator` with the continuous and the discontinuous space respectively. An example of this can be seen in Code Snippet 12.3.

```
//THE FILE "continuous_embedding.hpp" WHICH IMPLEMENTS THE
    CONTINUOUS EMBEDDING CLASS

//! @file    continuous_embedding.hpp
//! @author Oded Stein
//! @date    2015

#ifndef BETL2_CONTINUOUS_EMBEDDING_HPP
#define BETL2_CONTINUOUS_EMBEDDING_HPP

```

```
10 // own includes
   ------------------------------------------------------------------------
11 #include "sparse_base_operator.hpp"
12
13 namespace betl2 {
14
15     template< typename CONTINUOUS_SPACE_T, typename
          DISCONTINUOUS_SPACE_T >
16     class ContinuousEmbedding : public SparseBaseOperator<double,
          linalg::sparse::Assign >
17     {
18     private:
19         typedef double
                numeric_t;
20         typedef SparseBaseOperator<numeric_t, linalg::sparse::Assign
              >   sparseBase_t;
21         typedef int
                index_t;
22
23         const CONTINUOUS_SPACE_T&        fe_contin_;
24         const DISCONTINUOUS_SPACE_T&     fe_discontin_;
25
26     public:
27         explicit ContinuousEmbedding( const CONTINUOUS_SPACE_T&
              fe_contin, const DISCONTINUOUS_SPACE_T& fe_discontin )
28         : sparseBase_t( )
29         , fe_contin_( fe_contin )
30         , fe_discontin_( fe_discontin )
31         {
32             /* empty */
33         }
34
35         /// forbid copies
36         ContinuousEmbedding( const ContinuousEmbedding& ) = delete;
37
38         /// forbid assignments
39         ContinuousEmbedding& operator=( const ContinuousEmbedding& )
              = delete;
40
41
42
43
44         class ElementFunctor
45         {
46         private:
47             typedef Eigen::Matrix< double, 3, 3 > value_t;
48
```

```
49          public:
50              ElementFunctor()
51              { /* empty */ }
52
53
54              template< typename ELEMENT_ENTITY_T >
55              value_t operator()( const ELEMENT_ENTITY_T& E ) const
56              {
57                  value_t result;
58                  result.setIdentity();
59
60                  return result;
61              }
62
63          };
64
65      public:
66          template< typename RUNTIME_CACHE_T >
67          void compute( const RUNTIME_CACHE_T& runtime_cache )
68          {
69              const ElementFunctor ef;
70              this -> assembleTriplets_( fe_contin_, fe_discontin_, ef
                    );
71              this -> make_sparse( );
72          }
73      }; // end class ContinuousEmbedding
74
75  } // end namespace betl2
76
77  #endif // BETL2_CONTINUOUS_EMBEDDING_HPP
78
79
80
81
82  //IMPLEMENTATION OF THE CONTINUOUS EMBEDDING
83  typedef ContinuousEmbedding< fespace_lagrange1_constr_t,
        fespace_lagrange1_discontin_constr_t >
        lagrange1_continuous_embedding_gamma_op_t;
84  lagrange1_continuous_embedding_gamma_op_t
        lagrange1_continuous_embedding_gamma_op( fespace_lagrange1_gamma,
        fespace_lagrange1_discontin_gamma );
85  lagrange1_continuous_embedding_gamma_op.compute(cache);
86  const auto& lag1_contin_emb_gamma =
        lagrange1_continuous_embedding_gamma_op.matrix();
```

Code Snippet 12.3: The class for the continuous embedding operator and an example for its implementation using the space of piecewise linear scalar functions on Γ
Its layout is inspired form BETL2's implementation of the combinatorial gradient.
The Eigen classes used here are from the Eigen matrix library [8].

12.2 Combinatorial gradient, curl and divergence

BETL2 offers combinatorial gradient, combinatorial curl (which is just combinatorial gradient into the div space instead of the curl space) and combinatorial divergence that map from the different finite element spaces to each other. An example for the implementation is given in Code Snippet 12.4.

```
1  //combinatorial gradient
2  typedef CombinatorialGradient< fespace_lagrange1_constr_t ,
       fespace_curl_constr_t > grad_1_curl_op_t;
3  grad_1_curl_op_t grad_1_curl_op(fespace_lagrange1_gamma ,
       fespace_curl_gamma);
4  grad_1_curl_op.compute(cache);
5  const auto& grad_1_curl = grad_1_curl_op.matrix();
6
7  //combinatorial curl
8  typedef CombinatorialGradient< fespace_lagrange1_constr_t ,
       fespace_div_constr_t > grad_1_div_op_t;
9  grad_1_div_op_t grad_1_div_op(fespace_lagrange1_gamma ,
       fespace_div_gamma);
10 grad_1_div_op.compute(cache);
11 const auto& grad_1_div = grad_1_div_op.matrix();
12
13 //combinatorial divergence
14 typedef CombinatorialDivergence< fespace_div_constr_t ,
       fespace_lagrange0_constr_t > div_div_0_op_t;
15 div_div_0_op_t div_div_0_op(fespace_div_gamma ,
       fespace_lagrange0_gamma);
16 div_div_0_op.compute(cache);
17 const auto& div_div_0 = div_div_0_op.matrix();
```

Code Snippet 12.4: An example for the implementation of the combinatorial gradient, combinatorial curl and combinatorial divergence matrices in BETL2
The combinatorial curl is just the combinatorial gradient into the div space instead of the curl space, as this automatically causes a rotation by $\frac{\pi}{2}$.

The BETL2 version used at the moment constructs the negative gradient instead of the gradient in the case of `grad_1_curl`. This has to be corrected in the code.

12.3 Mass matrices

Mass matrices are very simple to implement in BETL2. They can be set up for any two dual finite element spaces. An example for the setup of a mass matrix in BETL2 can be seen in Code Snippet 12.5.

```
1 typedef IdentityOperator< fespace_lagrange0_constr_t ,
      fespace_lagrange1_constr_t > identity_operator_lag0_lag1_t;
2 identity_operator_lag0_lag1_t
      identity_operator_lag0_lag1(fespace_lagrange0_gamma ,
      fespace_lagrange1_gamma);
3 identity_operator_lag0_lag1.compute();
4 const auto& identity_operator_lag0_lag1_mat =
      identity_operator_lag0_lag1.matrix();
5
6 matrix_t M_lag0_lag1 = identity_operator_lag0_lag1_mat;
```

Code Snippet 12.5: An example for the implementation of a mass matrix in BETL2 The mass matrix shown is for the two spaces $H^{-\frac{1}{2}}(\Gamma)$ and $H^{\frac{1}{2}}(\Gamma)$.

12.4 Adding constraints to functions

At one point in the method it is necessary to add a constraint that a certain function in $\tilde{\mathbf{H}}(1)$ integrates to 0. This is done with the help of the mass matrices from Section 12.2. An example is shown in Code Snippet 12.6. The approach described there is taken from other BETL2 implementations by Lars Kielhorn [20].

```
1 matrix_t stabilizer = matrix_t::Ones(1,
      M_lag0_lag1.rows())*M_lag0_lag1;
2 d += stabilizer.transpose()*stabilizer;
```

Code Snippet 12.6: An example of ensuring that the solution of a matrix equation integrates to zero. The stabilizer matrix evaluates to zero when multiplied with a trial vector for a function integrating to zero. The condition is then added to the actual matrix.

12.5 BEM operators

More work is needed to implement the BEM operators in BETL2. First a fundamental solution object has to be constructed (which can be a Laplace or a Helmholtz fundamental solution). Then the appropriate Galerkin kernel object has to be set up, which can be single layer (`SL`) or double layer (`DL`). The Galerkin kernel object is then used to construct a Galerkin integrator object (which can be constructed to use the Gram determinant during integration or not; it should not be used for operations on the edge spaces) which is used to construct a BEM operator object operating on the correct spaces. The matrix of the operator can then be extracted from the BEM operator object. An example of this is shown in Code Snippet 12.7.

```
1  // quadrature scheme
2  const int numQuadPoints = 12;
3  typedef bem::GalerkinQuadratureRule< numQuadPoints, numQuadPoints,
       numQuadPoints > quad_rule_t;
4
5  // cache
6  typedef bem::Cache< grid_factory_t, numQuadPoints, numQuadPoints >
       cache_t;
7  cache_t cache(grid_factory);
8
9  // singularity detector
10 typedef bem::GalerkinSingularityDetector< grid_factory_t >
       singularity_detector_t;
11 singularity_detector_t singularity_detector(grid_factory);
12
13 // integration traits
14 const bool withGram = true;
15 typedef bem::IntegrationTraits< quad_rule_t, grid_factory_t,
       singularity_detector_t, cache_t, withGram, withGram >
       integration_traits;
16
17 // fundamental solution
18 typedef bem::FundSol< bem::FSType::LAPLACE, 3 > laplace_fundsol_t;
19 laplace_fundsol_t laplace_fundsol;
20
21
22 //laplace_lag_sl on gamma
23 typedef bem::GalerkinKernel< laplace_fundsol_t, bem::FSLayer::SL,
       fespace_lagrange0_constr_t::fe_basis_t,
       fespace_lagrange0_constr_t::fe_basis_t > laplace_lag_sl_kernel_t;
24 laplace_lag_sl_kernel_t laplace_lag_sl_kernel(laplace_fundsol);
25
26 typedef bem::GalerkinIntegrator< laplace_lag_sl_kernel_t,
       integration_traits > laplace_lag_sl_integrator_t;
```

```
27 laplace_lag_sl_integrator_t
       laplace_lag_sl_integrator(laplace_lag_sl_kernel,
       singularity_detector, cache);
28
29 typedef BemOperator< laplace_lag_sl_integrator_t,
       fespace_lagrange0_constr_t, fespace_lagrange0_constr_t >
       laplace_lag_sl_bemop_t;
30 laplace_lag_sl_bemop_t
       laplace_lag_sl_bemop(laplace_lag_sl_integrator,
       fespace_lagrange0_gamma, fespace_lagrange0_gamma);
31 laplace_lag_sl_bemop.compute();
32
33 matrix_t laplace_lag_sl = (laplace_lag_sl_bemop.matrix()).template
       cast<std::complex<double>>();
```

Code Snippet 12.7: An example for the implementation of BEM operators in BETL2 The example shown is for the Laplace single layer operator on the space $H^{-\frac{1}{2}}(\Gamma)$. The parts before the definition of the fundamental solution define properties used by the integration objects.

12.6 Incorporating boundary conditions and the ansatz function

Given analytical functions are implemented in BETL2 using a specific class for projecting analytical functions onto the finite element spaces. An object of this class has to return the function value (or its gradient, or its curl, depending on the type of boundary conditions that are constructed) at every point on the grid. Then the `DofCreator()` function is used to construct a vector out of the analytical function object on a specific fespace. An example of this can be seen in Code Snippet 12.8.

```
1 typedef analytical::CircularCurrentLoopExcitationHfield
      coil_functor_t;
2 typedef bem::AnalyticalGridFunction< grid_factory_t, coil_functor_t,
      Trace::Dirichlet > coil_dirichlet_t;
3
4 const coil_functor_t coil_functor(j0, coil_radius, rel_delta);
5 const coil_dirichlet_t coil_dirichlet(grid_factory, coil_functor);
6
7 matrix_t coeff_dirichlet( DofCreator()(coil_dirichlet,
      fespace_curl_gamma) );
```

Code Snippet 12.8: An example for the incorporation of boundary conditions in BETL2 In this example, the Dirichlet trace of the excitation field projected onto the curl fespaye is obtained. The class `analytical :: CircularCurrentLoopExcitationHfield` is used to construct an object returning $\mathbf{H}_s$ at each point of the grid. The vector `coeff_dirichlet` is $\mathbf{H}_s$ projected onto `fespace_curl_gamma`.

The ansatz function ρ used in Chapter 8 is also implemented in that way. Is is constructed as the Dirichlet trace of a scalar function as described in Definition 8.1.

12.7 Calculating line integrals on the inner equator of a torus

The symmetric method for complicated geometries needs to calculate the line integral defined in Definition 7.10:

$$L\gamma_N^c\mathbf{H} = \int_{\partial\Sigma^+} \mathbf{curl}\,\mathbf{H}(x)\cdot\mathbf{t}(x)dS(x) \tag{12.1}$$

BETL2's implementation of the space $\boldsymbol{H}(\mathrm{div};\Omega)$ assigns one degree of freedom to one edge of the mesh that corresponds to a basis function perpendicular to that edge (the direction depends on the edge's global orientation). Thus the line integral over the inner equator of a torus can be computed defining a restricted subspace for only those edges on the inner equator, creating a covector with entries 1 or -1 (depending on whether the global orientation of the mesh matches the orientation of the inner equator of the torus) and then embedding the covector in the larger fespace where it can be multiplied with any element to compute its line integral.

The reason that the basis functions perpendicular to the edge are used to calculate the line integral and not the ones parallel is the fact that L operates on $\gamma_N^c\mathbf{H}$ but calculates the line integral of $\mathbf{curl}\,\mathbf{H}$. $\gamma_N^c\mathbf{H}$ corresponds to $\mathbf{curl}\,\mathbf{H}$ on the boundary rotated by $\frac{\pi}{2}$.

The implementation can be seen in Code Snippet 12.9.

```
1  //THE FILE "equator_dof_marker.hpp" WHICH IMPLEMENTS THE EQUATOR
       SUBSPACE CLASS
2
3  //! @file    equator_dof_marker.hpp
4  //! @author Oded Stein
5  //! @date    2015
6
7  #ifndef BETL2_FE_EQUATOR_DOF_MARKER_HPP
8  #define BETL2_FE_EQUATOR_DOF_MARKER_HPP
9
10 // system includes
```

```
11 #include <vector>
12
13 // own includes
14 #include <fe/dof.hpp>
15 #include <fe/constrained_fespace.hpp>
16 #include <fe/elemwise_constrained_fespace.hpp>
17 #include <fe/condition_dof.hpp>
18
19 //-----------------------------------
20
21 namespace betl2 {
22     namespace fe {
23
24         template< typename FESPACE_T, typename GRID_FACTORY_T >
25         class EquatorDofMarker
26         {
27
28         public:
29             enum EqOrNot { NotEquator = 0, Equator = 1 };
30
31             template<enum EqOrNot E>
32             struct Condition_equator
33             {
34                 bool operator()( const Dof& d ) const
35                 {
36                     return ( d.constraint() ).contains(E);
37                 }
38             };
39
40         public:
41             typedef eth::base::unsigned_t unsigned_t;
42
43         public:
44             typedef ConstrainedFESpace< FESPACE_T,
                    Condition_equator<Equator> > equator_fespace_t;
45         private:
46             typedef Eigen::Matrix< double, Eigen::Dynamic, 1 >
                    vector_t;
47             equator_fespace_t equator_fespace_;
48             vector_t ones_edge_function_;
49
50
51         private:
52             FESPACE_T& fespace_;
53             const GRID_FACTORY_T& grid_factory_;
54             const double r1_;
55             const double r2_;
56
57             int old_fespace_size;
```

```
58
59          public:
60              EquatorDofMarker( FESPACE_T& fespace, const
                    GRID_FACTORY_T& grid_factory, const double r1, const
                    double r2 )
61              : fespace_(fespace), grid_factory_(grid_factory),
                    r1_(r1), r2_(r2), equator_fespace_(fespace)
62              {
63                  /* empty */
64              }
65
66
67              /// move constructor
68              EquatorDofMarker( EquatorDofMarker&& other )
69              : fespace_(std::move(other.fespace_)),
                    grid_factory_(std::move(other.grid_factory_)),
                    r1_(other.r1_), r2_(other.r2_),
                    equator_fespace_(std::move(other.equator_fespace_)),
                    ones_edge_function_(std::move(other.ones_edge_function_))
70              {
71                  /* empty */
72              }
73
74              /// destructor
75              ~EquatorDofMarker( )
76              {
77                  /* empty */
78              }
79
80              /// forbid copies
81              EquatorDofMarker( const EquatorDofMarker& ) = delete;
82
83              /// forbid assignments
84              EquatorDofMarker& operator=( const EquatorDofMarker& ) =
                    delete;
85
86              /// access private variables
87              const equator_fespace_t& equator_fespace() const {
                    return equator_fespace_; }
88              const vector_t ones_edge_function() const { return
                    ones_edge_function_; }
89
90              /// mark function
91              void mark();
92              void restoreFespace();
93
94          }; // end class EquatorDofMarker
95
96
```

```
97          template< typename FESPACE_T, typename GRID_FACTORY_T >
98          void EquatorDofMarker< FESPACE_T, GRID_FACTORY_T >::mark( )
99          {
100             const double interior_rad = r1_-r2_;
101             const double tol = 1e-12;
102
103             auto& permutated_equator_dofs =
                    equator_fespace_.permutations();
104             std::unordered_set< Dof* > equator_dofs;
105
106             //Go through all elements and extract equator elements
107             for(const auto& elem:fespace_) {
108                 const auto& intersections =
                        grid_factory_.getView().intersections(elem);
109
110                 //go through all intersections
111                 for(const auto& intersection:intersections) {
112                     const auto& corner1 =
                            intersection.geometry().mapCorner(0);
113                     const auto& corner2 =
                            intersection.geometry().mapCorner(1);
114                     bool is_inner_equator = false;
115
116                     if(fabs(corner1(2)) < tol && fabs(corner2(2)) <
                            tol && fabs(corner1.norm() - interior_rad) <
                            tol && fabs(corner2.norm() - interior_rad) <
                            tol)
117                         is_inner_equator = true;
118
119                     auto filtered_dofs = fespace_.filterAll( elem,
                            intersection.indexInInside() );
120
121                     for( Dof* dof:filtered_dofs) {
122                         //Add inner equator flag if not already there
123                         auto& constraint = dof->constraint();
124                         if(is_inner_equator) {
125                             if(!constraint.contains(Equator)) {
126                                 constraint.add(Equator);
127                                 equator_dofs.insert(dof);
128                             }
129                         } else {
130                             if(!constraint.contains(NotEquator)) {
131                                 constraint.add(NotEquator);
132                             }
133                         }
134                     }
135                 }
136             }
137
```

```
138             //create permutation
139             old_fespace_size = fespace_.numDofs();
140             permutated_equator_dofs.resize(fespace_.numDofs());
141
142             //std::cout << "dof indices: ";
143             int cntr=0;
144             for(auto& dof:equator_dofs) {
145                 int dof_index = dof->index();
146                 permutated_equator_dofs[dof_index] = cntr++;
147
148                 //std::cout << dof_index << ", ";
149             }
150             //std::cout << std::endl;
151
152             //create subspace
153             equator_fespace_.setSize(equator_dofs.size());
154
155
156             //create ones edge function
157             ones_edge_function_ =
                    vector_t::Zero(equator_dofs.size());
158             for(auto& elem:equator_fespace_) {
159                 int center_above = 0;
160                 if(elem.geometry().center()(2) > 0) center_above = 1;
161                 else if(elem.geometry().center()(2) < 0)
                        center_above = -1;
162
163                 const vector_t orientation =
                        equator_fespace_.orientations(elem);
164
165                 const auto& intersections =
                        grid_factory_.getView().intersections(elem);
166                 for(const auto& intersection:intersections) {
167                     const auto insideIndex =
                            intersection.indexInInside();
168                     auto filtered_dofs = fespace_.filterAll( elem,
                            insideIndex );
169
170                     for( Dof* dof:filtered_dofs) {
171                         if(equator_dofs.find(dof) !=
                                equator_dofs.end() && center_above!=0) {
172                             if(center_above > 0) {
173                                 if(orientation(insideIndex)==1) {
174                                     ones_edge_function_
175                                       (permutated_equator_dofs[dof->index()])
                                            = -1.;
176                                 } else {
177                                     ones_edge_function_
```

```
178                                         (permutated_equator_dofs[dof->index()])
                                                = 1.;
179                                     }
180                                 }
181
182                             }
183                         }
184                     }
185                 }
186
187             }
188
189             template< typename FESPACE_T, typename GRID_FACTORY_T >
190             void EquatorDofMarker< FESPACE_T, GRID_FACTORY_T
                    >::restoreFespace( ) {
191
192                 auto& permutated_equator_dofs =
                        equator_fespace_.permutations();
193                 permutated_equator_dofs.resize(0);
194
195                 ones_edge_function_ = vector_t();
196
197                 equator_fespace_.setSize(old_fespace_size);
198                 old_fespace_size = 0;
199
200                 for(const auto& elem:fespace_) {
201                     for(auto dof=fespace_.begin(elem);
                            dof<fespace_.end(elem); ++dof) {
202                         auto& constraint = dof->constraint();
203                         for(const auto& cond:constraint) {
204                             constraint.remove(cond);
205                         }
206                     }
207                 }
208
209
210             }
211         } // end namespace fe
212     } // end namespace betl2
213
214
215     #endif // BETL2_FE_EQUATOR_DOF_MARKER_HPP
216
217
218
219     //IMPLEMENTATION OF THE ACTUAL LINE INTEGRAL OPERATOR IN THE MAIN
            FILE
220
221     //inner equator integrator
```

```
222     typedef fe::EquatorDofMarker<fespace_div_all_t, grid_factory_t>
            equator_marker_t;
223     equator_marker_t equator_marker(fespace_div_all, grid_factory,
            r1, r2);
224     equator_marker.mark();
225
226     matrix_t equator_ones =
            equator_marker.ones_edge_function().template
            cast<std::complex<double>>();
227     matrix_t equator_ones_all = equator_marker.equator_fespace()
228       .mapToGrid<std::complex<double>>(equator_ones);
229     matrix_t inner_equator_integrator =
            fespace_div_gamma.mapLocToGrid2(equator_ones_all).transpose();
230
231     equator_marker.restoreFespace();
```

Code Snippet 12.9: The class that is used to extract the fespace containing only the inner equator edges and that returns a vector with 1 and -1, corresponding to the global orientation of the edges respectively. In the implementation, after the construction of the line integrator, the function `equator_marker.restoreFespace()` has to be called as the construction of the subspace modifies the fespace.

12.8 Visualization

The visualization is done using Paraview [22]. BETL2 offers classes that visualize elements of the finite element spaces from Section 12.1 in Paraview. An example of this can be seen in Code Snippet 12.10.

```
1 //interpolate
2 typedef InterpolationGridFunction< grid_factory_t,
      fespace_lagrange1_all_t, std::complex<double> >
      interpolation_function_lagrange1_all_t;
3 interpolation_function_lagrange1_all_t interp_u(grid_factory,
      fespace_lagrange1_all, fespace_lagrange1_gamma.mapLocToGrid(u));
4
5 //export
6 const std::string export_name = basename +
      "_results/h_method_result";
7 vtu::Exporter<grid_factory_t> exporter(grid_factory, export_name);
8 exporter("calculated scalar potential", interp_u,
      vtu::Entity::Point);
```

Code Snippet 12.10: An example of visualization using BETL2's Paraview classes
The example is about visualizing the solution u to the eddy current problem. As the solution u is only defined on the subsurface Γ, but the Paraview classes need data on the whole mesh, u on Γ has to be embedded into the larger fespace using `mapLocToGrid`. The argument `vtu :: Entity :: Point` in the exporter function means that the degrees of freedom of u live on the vertices of the triangles. This makes sense for piecewise linear functions. Another option that can be used instead is `vtu :: Entity :: Cell` which is needed for piecewise constant functions.

13 Bibliography

[1] C. Amrouche et al. "Vector potentials in three-dimensional non-smooth domains". In: *Mathematical Methods in the Applied Sciences* 21.9 (1998), pp. 823–864.

[2] Alain Bossavit. *Computational Electromagnetism*. Academic Press, 1998.

[3] Glen Bredon. *Topology and Geometry*. Springer New York, 1993.

[4] A. Buffa and P. Jr. Ciarlet. "On traces for functional spaces related to Maxwell's Equations". In: *Math. Meth. Appl. Sci* 24 (2001).

[5] A. Buffa and R. Hiptmair. "Computational Methods in Wave Propagation: Galerkin Boundary Element Methods for Electromagnetic Scattering". In: ed. by M. Ainsworth. Vol. 31. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2003, pp. 85–126.

[6] Robert G. Chamberlain and William H. Duquette. "Some algorithms for polygons on a sphere". In: *JPL Publication* 07-03 (2007).

[7] H. K. Dirks. "Quasi-stationary fields for microelectronic applications". In: *Electrical engineering* 79.2 (1996), pp. 145–155.

[8] *Eigen, a C++ template library for linear algebra*. URL: `http://eigen.tuxfamily.org/` (visited on 03/24/2015).

[9] Lawrence C. Evans. *Partial Differential Equations*. Second Edition. American Mathematical Society, 2010.

[10] Barend Gehrels et al. *"Chapter 1. Geometry." Boost C++ Libraries*. URL: `http://www.boost.org/doc/libs/1_58_0/libs/geometry/doc/html/index.html` (visited on 06/02/2015).

[11] Christophe Geuzaine and Jean-François Remacle. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. URL: `http://geuz.org/gmsh/` (visited on 01/07/2015).

[12] David J. Griffiths. *Introduction to Electrodynamics*. Third edition. Prentice Hall, 1999.

[13] Nicolaus Heuer. *Personal conversation at ETH Zurich*.

[14] Ralf Hiptmair. "Boundary Element Methods for Eddy Current Computation". In: *Boundary Element Analysis*. Ed. by Martin Schanz and Olaf Steinbach. Vol. 29. Springer Berlin Heidelberg, 2007, pp. 213–248.

[15] Ralf Hiptmair and Jörg Ostrowski. "Coupled boundary-element scheme for eddy-current computation". In: *Journal of Engineering Mathematics* 51.3 (2005), pp. 231–250.

[16] Ralf Hiptmair and Jörg Ostrowski. "Generators of H1(H; Z) for triangulated surfaces: Construction And Classification". In: *SIAM J. Comput.* 21.5 (2002), pp. 1405–1423.

[17] Morris W. Hirsch. *Differential Topology*. 4th ed. Springer, 1991.

[18] John D. Jackson. *Classical electrodynamics*. Third edition. Wiley New York, 1999.

[19] John D. Jackson and Lew B. Okun. "Historical roots of gauge invariance". In: *Rev. Mod. Phys.* 73.3 (2001), pp. 663–680.

[20] Dr. Lars Kielhorn. *Personal conversation at ETH Zurich*.

[21] Lars Kielhorn. *BETL2: Boundary Element Template Library*. URL: `http://www.sam.math.ethz.ch/betl/` (visited on 01/07/2015).

[22] Kitware, ASC, and others (see `http://www.paraview.org/participants/`). *ParaView*. URL: `http://www.paraview.org/` (visited on 01/07/2015).

[23] Konrad Königsberger. *Analysis 2*. Springer Berlin Heidelberg, 2004.

[24] W.B. Raymond Lickorish. *An Introduction to Knot Theory*. Springer Science+Business Media New York, 1997.

[25] Salim Meddahi and Virginia Selgas. "A mixed–FEM and BEM coupling for a three-dimensional eddy current problem". In: *ESAIM: M2AN* 37.2 (2003), pp. 291–318.

[26] Jörg Ostrowski. "Boundary Element Methods for Inductive Hardening". PhD thesis. Universität Tübingen, 2002.

[27] Stefan A. Sauter and Christoph Schwab. *Boundary Element Methods*. Springer Berlin Heidelberg, 2011.

[28] Elke Spindler. *Personal conversation at ETH Zurich*.

[29] Olaf Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems*. Springer Science+Business Media, LLC, 2008.

[30] Eric W. Weisstein. *"Solid Angle." From MathWorld–A Wolfram Web Resource*. URL: `http://mathworld.wolfram.com/SolidAngle.html` (visited on 04/08/2015).

[31] Eric W Weisstein. *"Spherical Polygon." From MathWorld–A Wolfram Web Resource*. URL: `http://mathworld.wolfram.com/SphericalPolygon.html` (visited on 04/07/2015).