Shape Optimization with Boundary Elements

A dissertation presented
by

Charilaos Mylonas

to
The Department of Mathematics
in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of

Computational Science and Engineering

Federal Institute of Techology (ETH)
Zürich, Switzerland
August 2015

Thesis advisor

Prof. Ralf Hiptmair

Author

Charilaos Mylonas

Shape Optimization with Boundary Elements

Abstract

In the present work an optimization procedure for the fully coupled eddy current problem using the boundary element method is presented. The computational problem is modelling a conductive coil with prescribed current that surrounds a conducting sphere where eddy currents are inducted as described in [2].The coil is represented by a torus. In the setting of the current work the accessible shape configurations of the torus are generated by deformation along the surface normal of the torus. Explicit expressions for the shape sensitivities with respect to the chosen control function of the bilinear forms used to discretize the boundary layer operators are derived computed and validated. The solution of the optimization problem consists of retrieving the geometry that produces a prescribed magnetic field. All computations were conducted by implementing the necessary extensions to BETL2, a boundary element template library [6].

Contents

Acknowledgments

Above all I would like to thank my thesis supervisor Prof. Ralf Hiptmair for entrusting me with this challenging topic and for his close and decicevely helpful supervision and encouragement. In particular our frequent meetings motivated me in many levels to make this project happen in the best way possible. Our cooperation was more than I could ever have asked for.

I also have to note the contribution of Alberto Pagannini and Elke Spindler for mathematical aspects of this project and for practical advice regarding the setup and editing of BETL2 source.

I would also like to thank my friends Angeliki Kamoutsi for some suggestions on a very early draft and Maria Kourouni for offering to type some LaTeXformulas (although her help was not needed after all) and all my friends in Zürich reminding me that there is still life AFK!

This thesis is the latest milestone in a long exciting journey that started when I was very young and there are a lot of people to thank for that. Above all, I thank my parents for putting their trust in me and providing me with the means to develop academically. I have to thank them predominantly for raising me to have a certain set of character traits since it turned out that this was essential for the succesful completion of this project as many other challenging, on their own setting, projects before it.

I also should mention my first programming teacher Andreas Delis, my inspirational linear algebra lecturer Thanasis Kehagias, my first finite element professor Theodoros Chatzigogos, and of course my thesis supervisor from my previous degree, and of course the talented *"teacher"* (as he preferred to be called) Nikos Charalambakis. Finally from my studies at ETH Zürich although I had many excellent teachers in a variety of subjects, I would like to take this opportunity to express my gratitude for the exceptional and inspirational lectures to Prof. Matthias Troyer, Prof. Christian Hafner and Sascha Schnepp.

"There is a theory which states that if ever anyone
discovers exactly what the universe is for and why is
is here, it will instantly disappear and be replaced by
something even more bizare and inexplicable.
There is another which states that this has already happened."
(from Restaurant at the End of the Universe)

Chapter 1

Introduction and Summary

Chapter 1: Introduction

The ultimate objective of engineering is undeniably to estimate the best possible solution to a real world design problem. In order to take steps towards that goal physicists, along with mathematicians and engineers develop progressively more accurate (and complicated) *models* that make it possible to simulate physical phenomena that arise in real world design problems. More often than not, the models at hand even after realistic simplifications, cannot be solved, or it is very impractical to be solved, with analytical mathematical expressions but they require *numerical approximation techniques* to be solved. The development of accurate numerical techniques that simulate phenomena that are of interest in engineering design is of course a broad field of research by itself, and it is of great importance to technological progress. In the present work we assume that we have at hand a realistic model together with a sufficiently accurate numerical technique.

The next natural step in the engineering design procedure, that is after acquiring a reasonably realistic and calibrated model and a reasonably accurate estimation technique for the solution of the problem, is the selection of a *configuration* (that could mean *shape,*

position, material parameters) through comparison of the model solutions, that produces the *optimal* solution according to our engineering design objectives. The selection of the actual configurations that are tested depends on the engineers intuition and experience for the problem at hand.

The valuable contribution of ***shape optimization*** to engineering design is the potential to circumvent the, sometimes flawed, engineering intuition with respect to the choice of design configurations. This can be very helpful when the optimal configuration, that satisfies the engineering requirements, or even the underlying physical phenomenon, is too complicated to be tractable using the engineering intuition alone.

The purpose of numerical shape optimization is to develop techniques that consistently approximate design configurations approaching the optimal, that is, the configuration that best satisfies the design objectives.

In the present thesis the physical model at hand is the so-called ***eddy current model***. The eddy current model is used to estimate the induced electric currents due to the effect of slowly varying harmonic electromagnetic field. A numerical approximation to the solution of the eddy current problem is estimated through use of a numerical technique, the ***Boundary Element Method (BEM)***. A short presentation of the method is given in chapter 2. Parts of the technical report [3] and [2] are presented in order to make the exposition of the method complete and accessible in term of implementation. In BEM the numerical solution of the physical problem is only computed on a discretized boundary. The main advantage of this technique when applied in electromagnetics problems is that it more realistic in the sense that as the physical problem requires, the solution extends (but decays) to infinity whereas in other widely used numerical methods such as *finite volume method* or *finite element* method this is treated using special techniques (for example absorbing boundaries). Moreover, in boundary element methods it is necessary to integrate only

along the surface that represents the geometry.

Some of the technological applications where the eddy current model is of interest are the description of inductive heating, used also for inductive hardening of mechanical components, and the computation of inductances in power electronics. Eddy currents are also the main physical phenomenon behind regenerative and dissipative electric braking systems and of interest on the design of induction motors.

As a prototypical example in the present work the optimization method is validated in the form of an inverse problem where the reconstruction of a shape that produces a specific configuration of Neumann traces is attempted.

Structure of this thesis

The present thesis is organized as such:

- chapter 1 Introduction chapter,
- chapter 2 Boundary element method for eddy currents,
- chapter 3 Adjoint formulation and shape optimization,
- chapter 4 Computation on periodic surface and analytical shape derivatives,
- chapter 5 Numerical results from optimization runs,
- chapter 6 Implementation details,
- chapter 7 Conclusion and future work,
- Appendix A Validation of shape derivatives of operators,

In chapter 2 the computational method used is shortly described. The subtleties of the integration for boundary element operators were not considered in the present work.

In chapter 3 the derivation of the adjoint equations and the shape derivative of the objective function is derived for the BEM-BEM coupled eddy current problem with the so-called *A-field formulation*.

In chapter 4 a parametrization was chosen for the 3D surface we seek to optimize. The numerical integration of the problem is performed on a parameter domain that maps to the surface of a torus. In this part the boundary integral operators and the explicit expressions for their shape derivatives are presented.

In chapter 5 the numerical results of some representative optimization runs are presented. To the best of the knowledge of the author these results are original since there is a limited number of works on three dimensional shape optimization with the BEM and shape optimization for the fully coupled eddy current model has not been treated before.

In chapter 6 the most important implementation details are presented along with some propositions for future development. The thesis closes with a short chapter of conclusions and propositions for future work and a short chapter where the validation of the implementation of the analytical shape derivatives is discussed.

Chapter 2

Boundary Element Method - Eddy Current Computation

2.1 The Eddy Current Model

The eddy current model is used to describe the effect of harmonic electromagnetic field excitation in a conductor Ω_c and the non-conducting surrounding region Ω_e. The model is derived by assuming slow variation of the magnetic field $\mathbf{H}$ (the magnetoquasistatic approximation) that leads to neglecting the displacement current. In the following we denote the electric field by $\mathbf{E}$. The model equations then read,

$$\mathbf{curl}\,\mathbf{E} = -i\omega\mu\mathbf{H}, \quad \text{in} \quad \mathbb{R}^3, \quad \mathbf{curl}\,\mathbf{H} = \begin{cases} \sigma\mathbf{E} & \text{in} \quad \Omega_c \\ \\ \mathbf{j_s} & \text{in} \quad \Omega_e \end{cases}. \tag{2.1}$$

In the above μ is the permeability, constant and equal to μ_c in the conductor Ω_c and constant and equal to μ_0 in the surrounding non-conducting region Ω_e, σ is the constant conductivity in Ω_c and $\mathbf{j_S}$ an exciting current. The first equation is the Faraday's law of induction and

the second is Ampere's law. These equations are supplemented by the conditions

$$\mathbf{H} = O(|\mathbf{x}|^{-1}), \quad \mathbf{E} = O(|\mathbf{x}|^{-1}) \quad \text{uniformly for} \quad |\mathbf{x}| \to \infty \tag{2.2}$$

In the following a trimmed-down presentation of the derivation of the bilinear form for the eddy current problem will be given presenting parts from [2] in order to establish notation and arrive at the variational problem.

We define the tangential components trace $(\gamma_{\mathbf{t}}\mathbf{U}) \coloneqq \mathbf{n}(\mathbf{x}) \times (\mathbf{U} \times \mathbf{n}(\mathbf{x}))$ for $\mathbf{U} \in C^{\infty}(\bar{\Omega})$ and the twisted tangential tangential trace $(\gamma_{\times}) \coloneqq \mathbf{U}(\mathbf{x}) \times \mathbf{n}(\mathbf{x})$. We assume that the boundaries are piecewise smooth and we introduce the spaces $\mathbf{H}_{\perp}^{1/2}(\Gamma)$ and $\mathbf{H}_{\parallel}^{1/2}(\Gamma)$. The tangential traces become then continuous surjective operators $\gamma_{\mathbf{t}} : \mathbf{H}^1(\Omega) \mapsto \mathbf{H}_{\parallel}^{1/2}(\Gamma), \quad \gamma_{\times} : \mathbf{H}^1(\Omega) \mapsto \mathbf{H}_{\perp}^{1/2}(\Gamma)$. The space $\mathbf{H}_{\parallel}^{1/2}(\Gamma)$ can be seen as the space containing functions that are tangentially continuous across the edges and the space $\mathbf{H}_{\perp}^{1/2}(\Gamma)$ functions that are continuous along the normals of Γ. Due to the preceding, the integration by parts formula 2.3 holds. The associated dual spaces will be denoted by $\mathbf{H}_{\parallel}^{-1/2}(\Gamma)$ and $\mathbf{H}_{\perp}^{-1/2}(\Gamma)$.

$$\int_{\Omega} \mathbf{curl}\,\mathbf{V} \cdot \mathbf{U} - \mathbf{V} \cdot \mathbf{curl}\,\mathbf{U} d\mathbf{x} = \int_{\partial\Omega} \gamma_{\times}\mathbf{U} \cdot \gamma_{\mathbf{t}}\mathbf{V} dS \tag{2.3}$$

Using the integration by parts formula 2.3 we define for vector fields $\mathbf{U} \in \mathbf{H}(\mathrm{div}; \Omega) \coloneqq \{\mathbf{V} \in \mathbf{L}^2(\Omega), \mathrm{div}\mathbf{V} \in \mathbf{L}^2(\Omega)\}$ the weak normal trace by

$$\langle \gamma_{\mathbf{n}}\mathbf{U}, \gamma\Phi \rangle_{1/2,\Gamma} = \int_{\Omega} \mathrm{div}\mathbf{U}\bar{\Phi} + \mathbf{U} \cdot \mathbf{grad}\,\bar{\Phi} dx \quad \forall \quad \Phi \in H^1(\Omega), \tag{2.4}$$

with $\langle \cdot, \cdot \rangle_{1/2,\Gamma}$ as duality pairing between $H^{-1/2}(\partial\Omega)$ and $H^{1/2}(\partial\Omega)$. In the context of boundary value problems for the Laplacian $-\Delta$ the trace operator $\gamma : H^1(\Omega) \mapsto H^{1/2}(\Gamma)$ can be called the "Dirichlet trace" whereas $\partial_{\mathbf{n}} \coloneqq \gamma_{\mathbf{n}} \circ \mathbf{grad}$ provides the "Neumann trace". They are linked by

$$\langle \partial_{\mathbf{n}}\Psi, \gamma\Phi \rangle_{1/2,\Gamma} = \int_{\Omega} \Delta\Psi\bar{\Phi} + \mathbf{grad}\,\Psi \cdot \mathbf{grad}\,\bar{\Phi} d\mathbf{x} \quad \forall \quad \Phi \in H^1(\Omega). \tag{2.5}$$

However, in the eddy current problem we need corresponding Dirichlet and Neumann traces that can be used to represent the **curl curl** operator. Towards that goal, γ_N is defined for

$$\mathbf{U} \in \mathbf{W}(\mathbf{curl}^2, \Omega) \coloneqq \{\mathbf{V} \in \mathbf{W}(\mathbf{curl}, \Omega), \mathbf{curl}\,\mathbf{curl}\,\mathbf{V} \in \mathbf{L}^2(\Omega)\} \tag{2.6}$$

by demanding that for all $\mathbf{V} \in H(\mathbf{curl}; \Omega)$

$$\langle \gamma_N \mathbf{U}, \gamma_{\mathbf{t}} \mathbf{V} \rangle_\tau = \int_\Omega \mathbf{curl}\,\mathbf{U}\,\mathbf{curl}\,\bar{\mathbf{V}} - \mathbf{curl}\,\mathbf{curl}\,\mathbf{U} \cdot \bar{\mathbf{V}} d\mathbf{x} \tag{2.7}$$

where $\langle \cdot, \cdot \rangle_\tau$ is the sesquilinear duality pairing. Through the defined Neumann trace we have

$$\gamma_N : \mathbf{W}(\mathbf{curl}^2, \Omega) \mapsto \mathbf{H}_{\|}^{-1/2}(\mathrm{div}_\Gamma, \Gamma). \tag{2.8}$$

In the above $\mathbf{W}(\mathbf{curl}, \Omega) = \{\frac{\mathbf{V}(\mathbf{x})}{\sqrt{1+|\mathbf{x}|^2}} \in L^2(\Omega), \mathbf{curl}\,\mathbf{V} \in L^2\}$ is a weighted Beppo Levi space.

This short exposition closes with the essential *transmission conditions* that should hold across $\Gamma \coloneqq \partial\Omega_c$

$$[\gamma_{\mathbf{t}} \mathbf{E}]_\Gamma = 0 \quad \text{and} \quad [\gamma_{\mathbf{t}} \mathbf{H}]_\Gamma = 0 \tag{2.9}$$

where $[\cdot]_\Gamma$ defines the jump on a trace from the exterior (Ω_e) to the interior (Ω_c) domain. The exterior traces are denoted as $(\cdot)^+$ and the interior traces are denoted by $(\cdot)^-$.

2.2 Variational Formulation

Again in the following parts of [2] are presented. There are two different approaches to a variational formulation of Equation 2.1. We are going to follow the so-called **E** based approach where the unknown is the (fictitious) electric field. We proceed by substituting the expression for the magnetic field from Faraday's law to Ampere's law and testing with a function on $\mathbf{V} \in \mathbf{W}(\mathbf{curl}, \mathbb{R}^3)$. Then we have

$$\big(\frac{1}{\mu}\,\mathbf{curl}\,\mathbf{E}, \mathbf{curl}\,\mathbf{V}\big)_{L^2(\mathbb{R}^3)} + \mathrm{i}\omega\big(\sigma\mathbf{E}, \mathbf{V}\big)_{L^2(\Omega_c)} = -\mathrm{i}\omega\big(\mathbf{j_s}, \mathbf{V}\big)_{L^2(\mathbb{R}^3)} \tag{2.10}$$

The "*offset fields*" $\mathbf{E}_s$ and $\mathbf{H}_s$ are defined as

$$\begin{aligned} \mathbf{curl}\,\mathbf{curl}\,\mathbf{E}_s = -\mathrm{i}\omega\mu_0\mathbf{j}_s, \qquad & \mathbf{curl}\,\mathbf{H}_s = \mathbf{j}_s, \\ \mathrm{div}\mathbf{E}_s = 0, \qquad & \mathrm{div}\mathbf{H}_s = 0, \end{aligned} \quad \text{in} \quad \Omega_e. \tag{2.11}$$

The offset fields can be used in order to apply current excitation to the eddy current model. The total fields are then the sum of the offset and reaction currents

$$\begin{aligned} \mathbf{E} = \mathbf{E}_r + \mathbf{E}_s, \quad \mathbf{H} = \mathbf{H}_r + \mathbf{H}_s \\ \mathbf{curl}\,\mathbf{curl}\,\mathbf{E}_r = 0\,, \quad \mathbf{curl}\,\mathbf{H}_r = 0 \end{aligned} \tag{2.12}$$

where $\mathbf{E}$ and $\mathbf{H}$ are the total fields and $\mathbf{E}_r$ and $\mathbf{H}_r$ are the reaction fields. Taking into account Equation 2.12 we can formulate the variational problem as a transmission problem in the form of

$$\begin{aligned} \mathbf{curl}\,\mathbf{curl}\,\mathbf{E} + \kappa^2\mathbf{E} = \mathbf{0} \quad & \text{in } \Omega_c \\ \mathbf{div}\,\mathbf{E}_r = 0\,, \mathbf{curl}\,\mathbf{curl}\,\mathbf{E}_r = \mathbf{0} \quad & \text{in } \Omega_e \\ \gamma_{\mathbf{t}}^{+}\mathbf{E}_r - \gamma_{\mathbf{t}}^{-}\mathbf{E} = -\gamma_{\mathbf{t}}^{+}\mathbf{E}_s\,, & \\ \tfrac{1}{\mu_0}\gamma_N^{+}\mathbf{E}_r - \tfrac{1}{\mu_c}\gamma_N^{-}\mathbf{E} = -\tfrac{1}{\mu_0}\gamma_N^{+}\mathbf{E}_s \quad & \text{on } \Gamma \end{aligned} \tag{2.13}$$

with $\kappa^2 \coloneqq \mathrm{i}\omega\sigma\mu_c$ with i the imaginary unit, ω the angular frequency of the excitation, σ the conductivity and μ_c the permeability of the conductor. It is noted that there is an alternative approach for the derivation of a variational formulation for the eddy current problem where $\mathbf{H}$ is the unknown [2]. In the following the boundary operators and their Galerkin discretization will be elaborated.

2.3 Galerkin Boundary Element Method for Eddy Current Computation

A valuable resource for the present part of the thesis was [3] where the Galerkin discretization of the BEM operators is elaborated in practical terms. When a distribution $\mathbf{U}$ solves the homogeneous equation $\mathbf{curl}\,\mathbf{curl}\,\mathbf{U} + \kappa^2\mathbf{U} = 0$ in $\Omega_c \cup \Omega_e$ then according to theorem 6 of [2] for $\kappa \neq 0$

$$\mathbf{U} = -\boldsymbol{\Psi}^{\kappa}_{\mathbf{A}}([\gamma_N \mathbf{U}]_\Gamma) - \boldsymbol{\Psi}^{\kappa}_{M}([\gamma_{\mathbf{t}} \mathbf{U}]_\Gamma) - \mathbf{grad}\,\Psi^{\kappa}_{V}([\gamma_{\mathbf{n}} \mathbf{U}]_\Gamma) \tag{2.14}$$

where

$$\boldsymbol{\Psi}^{\kappa}_{A}(\boldsymbol{\lambda})(\mathbf{x}) \coloneqq \int_\Gamma G^{\kappa}(\mathbf{x},\mathbf{y})\,\boldsymbol{\lambda}(\mathbf{y})\,ds_{\mathbf{y}} \tag{2.15}$$

is the *Vectorial Single Layer Potential*,

$$\boldsymbol{\Psi}^{\kappa}_{M}(\mathbf{v})(\mathbf{x}) \coloneqq \mathbf{curl}\,\boldsymbol{\psi}^{\kappa}_{A}(\mathbf{R}\mathbf{v})\,, \quad \mathbf{R}\mathbf{u} \coloneqq \mathbf{n} \times \mathbf{u} \tag{2.16}$$

is the *Maxwell Double Layer Potential*,

$$\boldsymbol{\Psi}^{\kappa}_{V}(\varphi)(\mathbf{x}) \coloneqq \int_\Gamma G^{\kappa}(\mathbf{x},\mathbf{y})\,\varphi(\mathbf{y})\,ds_{\mathbf{y}} \tag{2.17}$$

is the *Scalar Single Layer Potential* and

$$G^{\kappa}(\mathbf{x},\mathbf{y}) \coloneqq \frac{1}{4\pi}\frac{\exp(-\kappa|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|} \tag{2.18}$$

the *Helmholtz Kernel*. It also holds that

$$\gamma^{\pm}_{\mathbf{n}}\mathbf{U} = -\frac{1}{\kappa^2}\mathrm{div}_\Gamma(\gamma^{\pm}_{N}\mathbf{U}). \tag{2.19}$$

and defining the *Maxwell Single Layer Potential* as

$$\boldsymbol{\Psi}^{\kappa}_{S}(\boldsymbol{\lambda})(\mathbf{x}) \coloneqq \boldsymbol{\Psi}^{\kappa}_{A}(\boldsymbol{\lambda}) - \tfrac{1}{\kappa^2}\,\mathbf{grad}\,\Psi^{\kappa}_{V}(\mathbf{div}_\Gamma\,\boldsymbol{\lambda}) \tag{2.20}$$

we can rewrite Equation 2.14 as

$$\mathbf{U} = -\mathbf{\Psi}_S^\kappa([\gamma_N \mathbf{U}]) - \mathbf{\Psi}_M^\kappa([\gamma_\mathbf{t} \mathbf{U}]). \tag{2.21}$$

The expression 2.14 and its simplification for $k \neq 0$, 2.21 is the **Straton - Chu representation formula**. When $k = 0$ we have to use Equation 2.14 as a representation formula. For future reference we also need to define the scalar double layer potential

$$\mathbf{\Psi}_K^\kappa(u)(\mathbf{x}) \coloneqq \int_\Gamma \frac{\partial}{\partial \mathbf{n}} G^\kappa(\mathbf{x}, \mathbf{y}) u(\mathbf{y}) d\mathbf{y} \tag{2.22}$$

2.3.1 Bilinear forms, E - Based Model

We proceed to present the bilinear forms needed for the solution of Equation 2.13. According to [2] theorem 9, for $\kappa \neq 0$ the following operators on Γ are well defined and continuous

$$\begin{aligned}
\mathsf{A}^\kappa &\coloneqq \gamma_\mathbf{t} \mathbf{\Psi}_S^\kappa & \mathsf{A}^0 &\coloneqq \gamma_\mathbf{t} \mathbf{\Psi}_A^0 \\
\mathsf{B}^\kappa &\coloneqq \frac{1}{2}(\gamma_N^- + \gamma_N^+) \mathbf{\Psi}_S^\kappa & \mathsf{B}^0 &\coloneqq \frac{1}{2}(\gamma_N^- + \gamma_N^+) \mathbf{\Psi}_A^0 \\
\mathsf{C}^\kappa &\coloneqq \frac{1}{2}(\gamma_\mathbf{t}^- + \gamma_\mathbf{t}^+) \mathbf{\Psi}_M^\kappa & \mathsf{K}^\kappa &\coloneqq \frac{1}{2}(\gamma^- + \gamma^+) \mathbf{\Psi}_K^\kappa \\
\mathsf{N}^\kappa &\coloneqq \gamma_N \mathbf{\Psi}_M^\kappa. & \mathsf{V}^\kappa &\coloneqq \gamma \mathbf{\Psi}_V^\kappa
\end{aligned} \tag{2.23}$$

For $\mathbf{u}, \mathbf{v} \in \mathbf{H}_\parallel^{-1/2}(\mathbf{div}_\Gamma, \Gamma)$, and $\boldsymbol{\mu}, \boldsymbol{\lambda} \in \mathbf{H}_\perp^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$.

After applying trace operators to the representation formulas 2.14 and 2.21 we acquire the *Calderon identities*.

$$\begin{aligned}
\gamma_\mathbf{t}^- \mathbf{E} &= \mathsf{A}^\kappa(\gamma_N^- \mathbf{E}) + \big(\frac{1}{2} Id + \mathsf{C}^\kappa\big)(\gamma_\mathbf{t}^- \mathbf{E}), \\
\gamma_N^- \mathbf{E} &= \big(\frac{1}{2} Id + \mathsf{B}^\kappa\big)(\gamma_N^- \mathbf{E}) + \mathsf{N}^\kappa(\gamma_\mathbf{t}^- \mathbf{E})
\end{aligned} \tag{2.24}$$

where $\kappa = \frac{1}{2}\sqrt{2}(1 + \mathrm{i})\sqrt{\omega \sigma \mu_c}$ for the conductor (interior) domain Ω_c and

$$\begin{aligned}
\gamma_{\mathbf{t}}^{+}\mathbf{E}_r &= -\mathsf{A}^0(\gamma_N^{+}\mathbf{E}_r) + (\frac{1}{2}Id - \mathsf{C}^0)(\gamma_{\mathbf{t}}^{+}\mathbf{E}_r) - \mathbf{grad}_\Gamma\, \mathsf{V}^0(\gamma_N^{+}\mathbf{E}_r),\\
\gamma_N^{+}\mathbf{E}_r &= (\frac{1}{2}Id - \mathsf{B}^0)(\gamma_N^{+}\mathbf{E}_r) - \mathsf{N}^0(\gamma_N^{+}\mathbf{E}_r),\\
\gamma_{\mathbf{n}}^{+}\mathbf{E}_r &= -\gamma_{\mathbf{n}}^{+}\mathbf{\Psi}_A^0(\gamma_N^{+}\mathbf{E}_r) - \gamma_{\mathbf{n}}^{+}\mathbf{\Psi}_M^0(\gamma_{\mathbf{t}}^{+}\mathbf{E}_r) - (\frac{1}{2}Id - \mathsf{K}^0)(\gamma_{\mathbf{n}}^{+}\mathbf{E}_r)
\end{aligned} \tag{2.25}$$

for the exterior domain Ω_e. In order to couple the equations it is necessary to use the transmission conditions from Equation 2.13. The dependence on $\gamma_{\mathbf{n}}^{+}$ is disappearing on the weak form when seeking solutions on $\mathbf{H}_{\parallel}^{-1/2}(\mathrm{div}_\Gamma 0, \Gamma)$. For more details again the reader is referred to [2]. By using the transmission conditions we arrive at the following variational problem: Seek $\mathbf{u} \in \mathbf{H}_{\perp}^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$, $\varphi \in \mathbf{H}_{\parallel}^{1/2}(\mathrm{div}_\Gamma 0, \Gamma)$ such that

$$\begin{aligned}
\langle \tilde{\mathsf{N}}\mathbf{u}, \mathbf{v}\rangle + \langle \tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma\,\varphi, \mathbf{v}\rangle &= \langle \gamma_N \mathbf{E}_s, \mathbf{v}\rangle\\
\langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_\Gamma\,\psi\rangle + \langle \tilde{\mathsf{A}}\,\mathbf{curl}_\Gamma\,\varphi, \mathbf{curl}_\Gamma\,\psi\rangle &= \langle \gamma_{\mathbf{t}}\mathbf{E}_s, \mathbf{curl}_\Gamma\,\psi\rangle
\end{aligned} \tag{2.26}$$

for all $\mathbf{v} \in \mathbf{H}_{\perp}^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$, $\psi \in \mathbf{H}_{\parallel}^{1/2}(\mathrm{div}_\Gamma 0, \Gamma)$. The boundary integral operators $\tilde{(\cdot)}$ are

$$\begin{aligned}
\tilde{\mathsf{N}} &\coloneqq \mathsf{N}^0 + \tfrac{1}{\mu_r}\mathsf{N}^\kappa\\
\tilde{\mathsf{B}} &\coloneqq \mathsf{B}^0 + \mathsf{B}^\kappa\\
\tilde{\mathsf{C}} &\coloneqq \mathsf{C}^0 + \mathsf{C}^\kappa\\
\tilde{\mathsf{A}} &\coloneqq \mathsf{A}^0 + \mu_r\mathsf{A}^\kappa
\end{aligned} \tag{2.27}$$

with $\mu_r = \frac{\mu_c}{\mu_0}$ the relative permeability. The unknowns $\mathbf{u} = \gamma_{\mathbf{t}}^{-}\mathbf{E}$ and $\mathbf{curl}_\Gamma\,\varphi = \frac{1}{\mu_r}\gamma_N^{-}\mathbf{E}$ are the interior traces of the electric and magnetic fields. According to [3] the bilinear forms for the interior domain are

$$\begin{aligned}
\langle \mathsf{A}^\kappa\mathbf{u}, \mathbf{v}\rangle &= \langle \mathbf{\Psi}_s^\kappa\mathbf{u}, \mathbf{v}\rangle = \langle \mathbf{\Psi}_A^\kappa\mathbf{u}, \mathbf{v}\rangle + \tfrac{1}{\kappa^2}\langle \Psi_V^\kappa\,\mathbf{div}_\Gamma\,\mathbf{u}, \mathbf{div}_\Gamma\,\mathbf{v}\rangle\\
\langle \mathsf{C}^\kappa\boldsymbol{\mu}, \mathbf{v}\rangle &= \langle \mathbf{\Psi}_M^\kappa\boldsymbol{\mu}, \mathbf{v}\rangle = -\langle \mathsf{B}^\kappa\mathbf{v}, \boldsymbol{\mu}\rangle\\
\langle \mathsf{N}^\kappa\boldsymbol{\mu}, \boldsymbol{\lambda}\rangle &= \kappa^2\langle \mathbf{\Psi}_A^\kappa(\mathbf{R}\boldsymbol{\mu}), \mathbf{R}\boldsymbol{\lambda}\rangle + \langle \Psi_V^\kappa\,\mathrm{curl}_\Gamma\,\boldsymbol{\mu}, \mathrm{curl}_\Gamma\,\boldsymbol{\lambda}\rangle = \kappa^2\langle \mathsf{A}^\kappa\mathbf{u}, \mathbf{v}\rangle\ .
\end{aligned} \tag{2.28}$$

In addition, for the exterior domain we have

$$\begin{aligned} \langle \mathsf{A}^0 \mathbf{u}, \mathbf{v} \rangle &= \langle \boldsymbol{\psi}_A^0 \mathbf{u}, \mathbf{v} \rangle \\ \langle \mathsf{N}^0 \boldsymbol{\mu}, \boldsymbol{\lambda} \rangle &= \langle \psi_V^0 \operatorname{curl}_\Gamma \boldsymbol{\mu}, \operatorname{curl}_\Gamma \boldsymbol{\lambda} \rangle \end{aligned} \tag{2.29}$$

This was used as the starting point for the computational work of the present thesis[1].

2.3.2 Galerkin Discretization of Bilinear Forms

Basis functions - transformations

In the following we turn our attention on the discretization of the required boundary operators. Coordinates on the the reference element are denoted by $\hat{x}_i$, the a point on the reference element by $\hat{\mathbf{x}} = \{\hat{x}_1, \hat{x}_2\}$. The basis functions for the *"left-facing"* unit reference triangle for the discretization of tangential functions in $\mathbf{H}^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$ we use the 1st order Raviart-Thomas basis functions. More explicitly they read

$$\hat{\mathbf{u}}_{12} = \begin{bmatrix} 1 - \hat{x}_2 \\ \hat{x}_1 - 1 \end{bmatrix}, \ \hat{\mathbf{u}}_{23} = \begin{bmatrix} -\hat{x}_2 \\ \hat{x}_1 \end{bmatrix}, \ \hat{\mathbf{u}}_{31} = \begin{bmatrix} -\hat{x}_2 \\ \hat{x}_1 - 1 \end{bmatrix} \tag{2.30}$$

where $\hat{u}_{ij}$ denotes the basis function on the edge between nodes i and j. The local to global mapping for these basis functions reads

$$\mathbf{u}_{ij}(\mathbf{x}) = \mathbf{J}\mathbf{G}^{-1}\hat{\mathbf{u}}_{ij}(\hat{\mathbf{x}}) \tag{2.31}$$

where $\mathbf{J}$ is the jacobian of the local to global transformation and $\mathbf{G} \coloneqq \mathbf{J}^\mathbf{T}\mathbf{J}$ is the Gram matrix. The $\operatorname{div}_\Gamma$ conforming basis functions are constructed by considering the rotation of the $\mathbf{curl}_\Gamma$ conforming basis functions. We have for $\mathbf{v} \in \mathbf{H}^{-1/2}(\operatorname{div}_\Gamma, \Gamma)$ that $\mathbf{v} = \mathbf{R}\mathbf{u}$ where $u \in \mathbf{H}_\perp^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$. The $\operatorname{div}_\Gamma$ conforming elements transform according to

$$\mathbf{v}_{ij}(\mathbf{x}) = -\frac{1}{\sqrt{|\mathbf{G}|}}\mathbf{J}\hat{\mathbf{v}}_{ij}(\hat{\mathbf{x}}) \tag{2.32}$$

[1] Between [5] and [2] there are different definitions of the "Maxwell double-layer potential" but also the transmission problem is posed differently. BETL2 developers should be aware that the fundamental solution used in the code is the same as in [2] with a sign change on κ.

where $|\cdot|$ signifies the determinant. Equation 2.32 will also be useful in the following chapters when we transform the bilinear forms discretized with div-conforming elements from the parametric square to the torus. The rotation $\mathbf{R}$ of the $\mathbf{curl}_\Gamma$ conforming elements in the reference coordinate system, to div_Γ conforming elements is done internally on BETL2 using the rotation matrix

$$H = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.33}$$

In order to construct the space $\mathbf{H}_{\|}^{-1/2}(\mathrm{div}_\Gamma 0, \Gamma)$ we make use of a scalar basis $\phi \in \mathbf{H}^{1/2}(\Gamma)$ since it holds that

$$\mathbf{curl}_\Gamma \, \phi \in \mathbf{H}_{\|}^{-1/2}(\mathrm{div}_\Gamma 0, \Gamma) \tag{2.34}$$

The surface curl, necessary for the above technique is defined as $\mathbf{curl}_\Gamma \, \phi \coloneqq \mathbf{grad} \, \phi \times \mathbf{n}$. In local coordinates the surface curl is computed by

$$\mathbf{curl}_\Gamma \, \phi(\mathbf{x}) = \frac{1}{\sqrt{|\mathbf{G}|}} \mathbf{J} \, \mathbf{H}\widehat{\mathbf{grad}}\hat{\phi}(\hat{(}\mathbf{x})). \tag{2.35}$$

The $\mathrm{div}_\Gamma = 0$ constraint is realized by assembling the element-wise matrices T_e for the gradient to a global sparse matrix T and applying the constraint to the relevant block of the matrix. In the same spirit, we need a combinatorial operator for the divergence on the computational surface. This is realized by assembling the element-wise matrices D_e

$$\widehat{\mathbf{div}}\hat{\mathbf{v}}_{ij} = D_e\hat{\psi} \quad \Leftrightarrow \quad \begin{bmatrix} \widehat{\mathbf{div}}\hat{\mathbf{v}}_{12} \\ \widehat{\mathbf{div}}\hat{\mathbf{v}}_{23} \\ \widehat{\mathbf{div}}\hat{\mathbf{v}}_{31} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \begin{bmatrix} \hat{\psi} \end{bmatrix} \tag{2.36}$$

where ψ is piecewise constant Lagrangian function, with $i = \{1, 2, 3\}$, $j = (i+1) mod 3$. The global assembly of D_e is denoted D. The edge-element relation that realizes the gradient

of the scalar field is

$$\widehat{\mathbf{grad}}\hat{\varphi}_i = T_e \hat{\mathbf{u}}_{ij} \Leftrightarrow \begin{bmatrix} \widehat{\mathbf{grad}}\hat{\varphi}_1 \\ \widehat{\mathbf{grad}}\hat{\varphi}_2 \\ \widehat{\mathbf{grad}}\hat{\varphi}_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & +1 \\ +1 & -1 & 0 \\ 0 & +1 & -1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{12} \\ \hat{\mathbf{u}}_{23} \\ \hat{\mathbf{u}}_{31} \end{bmatrix} . \tag{2.37}$$

2.3.3 Linear System

This section continues to present parts of [3]. For convenience in the notation we define

$$\begin{aligned} Q_h^{\kappa,1}[i,j] &\coloneqq \langle \mathsf{A}^\kappa \mathbf{u}_j, \mathbf{u}_i \rangle , \qquad \mathbf{u} \in \mathbf{H}_{\|}^{-1/2}(\mathbf{div}_\Gamma, \Gamma) \\ Q_h^{\kappa,2}[i,j] &\coloneqq \int_{\hat{T}} \int_{\hat{T}} G^\kappa(\mathbf{x}(\hat{\mathbf{x}}), \mathbf{y}(\hat{\mathbf{y}}))\, \psi_j(\hat{\mathbf{y}}) \psi_i(\hat{\mathbf{x}}) \sqrt{|\mathbf{G}(\hat{\mathbf{x}})|} \sqrt{|\mathbf{G}(\hat{\mathbf{y}})|}\, d\hat{\mathbf{y}} d\hat{\mathbf{x}} , \quad \psi \in H^{-1/2}(\Gamma). \end{aligned} \tag{2.38}$$

Single Layer Operator

The bilinear form $\frac{1}{\kappa^2}\langle \mathbf{\Psi}_V^\kappa \mathrm{div}_\Gamma \mathbf{u}, \mathrm{div}_\Gamma \mathbf{v} \rangle$ can be discretized with lagrangian constant basis functions. The surface divergence is realized through the use of the D operator. The bilinear form $\langle \mathbf{\Psi}_A^\kappa \mathbf{u}, \mathbf{v} \rangle$ is discretized with div conforming elements. The $\mathrm{div}_\Gamma = 0$ constraint is realized separately through the use of the sparse gradient operator. Finally the unconstrained operator as it enters the computation reads,

$$\begin{aligned} \tilde{A}_h &\coloneqq A_h^0 + A_h^\kappa \\ A_h^0 &\coloneqq Q_h^{0,1} \\ A_h^\kappa &\coloneqq Q_h^{\kappa,1} + \tfrac{1}{\kappa_2} D Q_h^{\kappa,2} D^\top \end{aligned} \tag{2.39}$$

The interior part of $\tilde{N}_h$ as also seen in 2.28 is simply

$$N_h^\kappa \coloneqq \kappa^2 A_h^\kappa. \tag{2.40}$$

The exterior trace is discretized again with constant lagrangian elements and the combinatorial divergence is taken. The exterior domain term reads

$$N_h^0 \coloneqq D\, Q_h^{0,2}\, D^T. \tag{2.41}$$

Double Layer Operator

The discretization of the double layer operator can be performed using functions belonging only to $\mathbf{H}_{\|}^{-1/2}(\mathbf{curl}_\Gamma, \Gamma)$ by using an appropriate form of the integral. The operator is discretized as

$$\begin{aligned} B_h^k &\coloneqq \langle \mathsf{B}^\kappa \mathbf{u}_i, \mathbf{v}_j \rangle \\ &= \int_\Gamma \mathbf{u}_i \int_\Gamma \mathbf{curl}_\Gamma\, G^\kappa(\mathbf{x},\mathbf{y}) \mathbf{R}\mathbf{v}_i d\mathbf{x} d\mathbf{y} \\ &\quad \text{with} \quad \mathbf{v_i} \in \mathbf{H}_{\perp}^{-1/2}(\mathbf{curl}_\Gamma, \Gamma) \quad \mathbf{u_i} \in \mathbf{H}_{\|}^{-1/2}(\mathrm{div}_\Gamma, \Gamma). \end{aligned} \tag{2.42}$$

Using the definition $\mathbf{curl}_\Gamma\, \varphi \coloneqq \mathbf{grad}\, \varphi \times \mathbf{n}$ the local representation can be shown to be computable as

$$\begin{aligned} B_h^k[i,j] &= \int_\Gamma \int_\Gamma (\mathbf{grad}_\mathbf{x}\, G^\kappa(\mathbf{x},\mathbf{y}) \times \mathbf{J}_\mathbf{y}\mathbf{u}_i \cdot \mathbf{J}_\mathbf{x}\mathbf{u}_j) d\mathbf{x} d\mathbf{y} \\ &\quad \text{with} \quad \mathbf{u_i} \in \mathbf{H}_{\|}^{-1/2}(\mathrm{div}_\Gamma, \Gamma). \end{aligned} \tag{2.43}$$

Excitation - Boundary Conditions

In non-simply connected domains, as it is the case for the torus that we are considering in the present work, it is necessary to define a cut along a circle of the torus bounding relative to the non-conducting domain Ω_e. In this cut the scalar field $\varphi \in H^{1/2}(\Gamma)$ should have a prescribed fixed jump in order to take into account non-local inductive excitation.

The computational mesh of the torus comprises of a rolled-up square mesh. Thus it was also necessary to impose periodicity along the edges of the domain for the vectorial degrees of freedom , and periodicity for the scalar degrees of freedom along the "small" circle

of the torus (non-bounding w.r.t. the exterior domain) and the aforementioned constant "jump" on the scalar field. All constraints were implemented with a penalty method.

We denote the constraint matrices that correspond to circles non-bounding with respect to the external domain as $P_c^{\cdot}$ and $P_e^{\cdot}$ the circle bounding with respect to the external domain. The penalty matrices that are used to constrain the scalar degrees of freedom are written as P_c^s and P_e^s while the vectorial constraints are noted P_c^v and P_e^v. Finally we denote the right hand side needed for the non-homogeneous jump boundary condition as f_e^s. The linear system then reads,

$$\begin{bmatrix} \tilde{N}_h + P_c^v + P_e^v & -\tilde{B}_h^\top T^\top \\ T\tilde{B}_h & T\tilde{A}_h T^\top + P_c^s + P_e^s + \underline{\alpha}\underline{\alpha}^\top \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \varphi \end{bmatrix} = \begin{bmatrix} 0 \\ f_e^s \end{bmatrix} . \tag{2.44}$$

The term $\underline{\alpha}$ is a vector included in order to constrain the average of φ to zero making the solution for φ unique. More precisely $a_i = \langle \varphi_i, 1 \rangle$. Since the geometry changes this term has a dependence on the control and should be taken into account on the shape derivative. However, the scalar field φ enters the computation only through its **curl** thus enforcing this constraint is not critical for calculating the shape derivative in our case. An alternative approach to constrain the φ space in order for the solution to be unique would simply be constraining the value of one degree of freedom. Both approaches were tested and they produced almost identical results at all the levels of computations including the critical step of calculating the shape derivative.

The structure of the linear system calls for taking advantage of the Schur complement method.

Chapter 3

The Shape Optimization Problem

We seek to optimize the geometry according to an objective function j_{obj} that depends on the solution of the eddy current boundary value problem 2.26.

$$J_{\text{obj}} = \int j_{\text{obj}}(\mathbf{u}, \phi) dS \tag{3.1}$$

A straightforward approach in order to calculate shape sensitivities would be to vary the configuration according to a set of directions and calculate the effect of the variation to the solution of the problem. That, of course, requires the solution of a linear system for each possible direction and this approach is computationally intractable.

A technique that circumvents that issue is based on the solution of only two full computational problems, the *adjoint* and the *forward problem*, and requires only a numerical integration for each possible direction[1]. This technique seems to date back to 1974 [4] [1] and has been applied to a large variety of inverse problems. A presentation of this method for the eddy current problem is given in the following pages. The computation of the shape gradient of the objective function is performed through the minimization of a lagrangian

[1] In the case of BEM this integration is not particularly cheap if approximation methods such as panel clustering are not used.

functional $\mathcal{L}$ where the forward problem (2.26) also referred to as the *state problem* enters as a constraint

3.1 Adjoint Computation of Shape Derivatives

3.1.1 The adjoint approach for general PDEs

We define Q the control space and V the state space. The optimization problem is defined,

$$\begin{aligned} \text{find} \quad q \in Q \quad \text{such that} \quad : \quad J(u;q) \to min \quad \text{subject to} \quad a(q;u,v) = l(v), \\ u \in V, \quad \forall v \in V. \end{aligned} \tag{3.2}$$

Where q is the control function. The lagrangian for this problem is defined as

$$\mathcal{L}(u,q,w) = J(u;q) + (a(u,q,w) - l(w)) \tag{3.3}$$

where w is called the adjoint state variable. Here, intuitively, one can argue that the PDE has entered as a constraint to the Lagrangian and the adjoint state variable is acting as a Lagrange multiplier enforcing the constraint. Differentiating the above w.r.t. the state variable u we have

$$\begin{aligned} &\langle \frac{\partial \mathcal{L}}{\partial u}(u,q,w), u' \rangle = 0 \\ &\Leftrightarrow a(q;u',w) = -\langle D_u J(u;q), u' \rangle \quad \forall u' \in V \end{aligned} \tag{3.4}$$

where $D_u \cdot$ denotes the sensitivity w.r.t. u. This is called the *adjoint state equation* of the problem. Now we consider the derivative of the Lagrangian w.r.t. the control q as

$$\begin{aligned} \langle \frac{\partial \mathcal{L}}{\partial q}(u,q,w), q' \rangle = \underbrace{\langle D_q J(u,q), q' \rangle + \langle \frac{\partial a}{\partial q}(q;u,w), q' \rangle}_{= \text{Gradient } \langle D_q \tilde{J}(q), q' \rangle} \\ \text{with } \tilde{J}(q) = J(u(q);q) \end{aligned} \tag{3.5}$$

for u solving the state equation and w solving the adjoint state equation. The final equation depends only on the solution of the adjoint and the state equation and it is the shape gradient of the objective function.

3.1.2 Adjoint formulation for the coupled eddy current problem

For $\mathbf{u} \in \mathbf{H}_{\perp}^{-1/2}(\mathbf{curl}_{\Gamma}, \Gamma)$, $\varphi \in H^{1/2}(\Gamma)$ and q a scalar control function, we define the lagrangian of the optimization problem as

$$\begin{aligned}\mathcal{L}(q; \mathbf{u}, \varphi, \mathbf{v}, \psi) = &\langle \tilde{\mathsf{N}}\mathbf{u}, \mathbf{v}\rangle + \langle \tilde{\mathsf{B}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{v}\rangle \\ &+\langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_{\Gamma}\,\psi\rangle + \langle \tilde{\mathsf{A}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{curl}_{\Gamma}\,\psi\rangle \\ &+\int_{\Gamma_q} j_{\mathrm{obj}}(\mathbf{u}, \phi) dS.\end{aligned} \tag{3.6}$$

The loading terms were neglected since we consider non local excitation that enters as a constraint on φ. along the loop and this is implemented by constraining the solution space. We focus on the effect of the variations of 3.6 with respect to its dependencies. The variation with respect to $\mathbf{v}$ and ψ should not affect the lagrangian functional when $\mathbf{u}$ and φ are solutions of the eddy current problem (2.26). Indeed, by considering their variations due to the linearity of all the involved operators we get 3.7 and 3.8 that are zero for any $\omega \in H^{1/2}$ or $\vec{\omega} \in H^{-1/2}$ respectively.

$$\begin{aligned}\langle \frac{\partial \mathcal{L}(q; \mathbf{u}, \mathbf{v}, \varphi, \psi)}{\partial \psi}, \omega\rangle = &\\ &= \lim_{h\to 0} \frac{L(q; \mathbf{u}, \mathbf{v}, \varphi, \psi + h\omega) - L(q; \mathbf{u}, \mathbf{v}, \varphi, \psi)}{h} \\ &= \lim_{h\to 0} \frac{1}{h}\Big\{\langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_{\Gamma}(\psi + h\omega)\rangle + \langle \tilde{\mathsf{A}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{curl}_{\Gamma}(\psi + h\omega)\rangle \\ &\quad - \langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_{\Gamma}\,\psi\rangle - \langle \tilde{\mathsf{A}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{curl}_{\Gamma}\,\psi\rangle\Big\} \\ &= \lim_{h\to 0} \frac{1}{h}\Big\{\langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_{\Gamma}\,h\omega\rangle + \langle \tilde{\mathsf{A}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{curl}_{\Gamma}\,h\omega\rangle\Big\} \\ &= \langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_{\Gamma}\,\omega\rangle + \langle \tilde{\mathsf{A}}\,\mathbf{curl}_{\Gamma}\,\varphi, \mathbf{curl}_{\Gamma}\,\omega\rangle = 0\end{aligned} \tag{3.7}$$

$$
\begin{aligned}
\langle \frac{\partial \mathcal{L}(q;\mathbf{u},\mathbf{v},\phi,\psi)}{\partial \mathbf{v}}, \vec{\omega}\rangle = & \\
&= \lim_{h\to 0} \frac{L(q;\mathbf{u},\mathbf{v}+h\vec{\omega},\phi,\psi) - L(q;\mathbf{u},\mathbf{v},\phi,\psi)}{h} \\
&= \lim_{h\to 0} \frac{1}{h}\Big\{ \langle \tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma(\varphi), \mathbf{v}+h\vec{\omega}\rangle + \langle \tilde{\mathsf{N}}\mathbf{u}, \mathbf{v}+h\vec{\omega}\rangle \\
&\quad - \langle \tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma(\varphi), \mathbf{v}\rangle + \langle \tilde{\mathsf{N}}\mathbf{u}, \mathbf{v}\rangle \Big\} \\
&= \langle \tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma\,\varphi, \vec{\omega}\rangle + \langle \tilde{\mathsf{N}}\mathbf{u}, \vec{\omega}\rangle = 0
\end{aligned}
\tag{3.8}
$$

Since we have established that the lagrangian is independent from variations of the test functions of the underlying PDE when we have the solution of 2.26 we remove them from the notation for the rest of the text. The functional derivatives (sensitivities) with respect to the unknowns of the variational problem are

$$
\begin{aligned}
\langle \frac{\partial L(q;\mathbf{u},\phi)}{\partial \phi}, \omega\rangle &= \lim_{h\to 0} \frac{L(q;\mathbf{u},\phi+h\omega) - L(q;\mathbf{u},\phi)}{h} \\
\langle \frac{\partial L(q;\mathbf{u},\phi)}{\partial \mathbf{u}}, \vec{\omega}\rangle &= \lim_{h\to 0} \frac{L(q;\mathbf{u}+h\vec{\omega},\phi) - L(q;\mathbf{u},\phi)}{h}.
\end{aligned}
\tag{3.9}
$$

By considering again the linearity of the involved operators, this translates to

$$
\begin{aligned}
&\langle \frac{\partial L(q;\mathbf{u},\varphi)}{\partial \mathbf{u}}, \vec{\omega}\rangle = \\
&= \lim_{h\to 0} \frac{1}{h}\Big\{ \langle \tilde{\mathsf{N}}(\mathbf{u}+h\vec{\omega}), \mathbf{v}\rangle + \langle \tilde{\mathsf{C}}(\mathbf{u}+h\vec{\omega}), \mathbf{curl}_\Gamma\,\psi\rangle + \int_S j_{\text{obj}}(\mathbf{u}+h\vec{\omega},\varphi)ds \\
&- \langle \tilde{\mathsf{N}}\mathbf{u}, \mathbf{v}\rangle - \langle \tilde{\mathsf{C}}\mathbf{u}, \mathbf{curl}_\Gamma\,\psi\rangle - \int_S j_{\text{obj}}(\mathbf{u},\varphi)ds \Big\} \\
&= \lim_{h\to 0} \frac{1}{h} h\Big\{ \langle \tilde{\mathsf{N}}\vec{\omega}, \mathbf{v}\rangle + \langle \tilde{\mathsf{C}}\vec{\omega}, \mathbf{curl}_\Gamma\,\psi\rangle \Big\} + \int_S \partial_{\mathbf{u}}(j_{\text{obj}}(\mathbf{u},\varphi))\vec{\omega}ds \\
&= \langle \tilde{\mathsf{N}}\vec{\omega}, \mathbf{v}\rangle + \langle \tilde{\mathsf{C}}\vec{\omega}, \mathbf{curl}_\Gamma\,\psi\rangle + \int_S \partial_{\mathbf{u}}(j_{\text{obj}}(\mathbf{u},\varphi))\vec{\omega}ds \\
&= \langle \tilde{\mathsf{N}}\mathbf{v}, \vec{\omega}\rangle - \langle \tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma\,\psi, \vec{\omega}\rangle + \int_S \partial_{\mathbf{u}}(j_{\text{obj}}(\mathbf{u},\varphi))\vec{\omega}ds
\end{aligned}
\tag{3.10}
$$

for the vectorial unknowns and

$$\begin{aligned}
&\langle \frac{\partial L(q;\mathbf{u},\varphi)}{\partial \varphi}, \omega \rangle = \\
&= \lim_{h\to 0} \frac{1}{h} \Big\{ \langle \tilde{\mathsf{A}}\, \mathbf{curl}_\Gamma(\varphi + h\omega), \mathbf{curl}_\Gamma\, \psi \rangle + \langle \tilde{\mathsf{B}}\, \mathbf{curl}_\Gamma(\varphi + h\omega), \mathbf{v} \rangle + \\
&- \langle \tilde{\mathsf{A}}\, \mathbf{curl}_\Gamma(\varphi), \mathbf{curl}_\Gamma\, \psi \rangle - \langle \tilde{\mathsf{B}}\, \mathbf{curl}_\Gamma(\varphi), \mathbf{v} \rangle \Big\} + \int_S \partial_\varphi (j_{\text{obj}}(\mathbf{u},\varphi)) \omega ds \qquad (3.11) \\
&= \langle \tilde{\mathsf{A}}\, \mathbf{curl}_\Gamma\, \omega, \mathbf{curl}_\Gamma\, \psi \rangle + \langle \tilde{\mathsf{B}}\, \mathbf{curl}_\Gamma\, \omega, \mathbf{v} \rangle + \int_S \partial_\varphi (j_{\text{obj}}(\mathbf{u},\varphi)) \omega ds \\
&= \langle \tilde{\mathsf{A}}\, \mathbf{curl}_\Gamma\, \psi, \mathbf{curl}_\Gamma\, \omega \rangle - \langle \tilde{\mathsf{C}}\mathbf{v}, \mathbf{curl}_\Gamma\, \omega \rangle + \int_S \partial_\varphi (j_{\text{obj}}(\mathbf{u},\varphi)) \omega ds
\end{aligned}$$

for the scalar unknowns.

These are the **adjoint equations** of the eddy current problem. For ease of implementation the L^2 norm of the difference of the A - field traces on the computational surface with respect to numerically computed values from a known shape were used as an objective function.

This amounts to a simple loading condition for the adjoint problem. Namely we have

$$\int_S j_{\text{obj}} ds = \frac{1}{2} \int_S ||\mathbf{A}_{opt} - \mathbf{A}_h||^2 ds \qquad (3.12)$$

Through the solution of the adjoint problem 3.10, 3.11 and the forward problem 2.26 the shape sensitivity of the objective function can be calculated for multiple directions without having to solve a linear system for each direction. Note that due to 2.28 there is a sign change on $\langle \tilde{\mathsf{B}}\, \mathbf{curl}_\Gamma\, \omega, \mathbf{v} \rangle$. It turns out that in the case of the adjoint equations the total system matrix is simply the transpose of that of the forward problem.

Assuming the geometry has a continuous dependence on the control q and the lagrangian has a continuous dependence on the geometry of our problem, assumptions that are quite general, we can proceed by taking the first derivative of $\mathcal{L}$ with respect to the control function q along direction δq.

$$\langle\frac{\partial\mathcal{L}}{\partial q},\delta q\rangle = \langle\langle\frac{\partial\tilde{\mathsf{A}}}{\partial q}\,\mathbf{curl}_\Gamma\,\psi,\mathbf{curl}_\Gamma\,\varphi\rangle,\delta q\rangle + \langle\langle\frac{\partial\tilde{\mathsf{C}}}{\partial q}\mathbf{v},\mathbf{curl}_\Gamma\,\varphi\rangle,\delta q\rangle$$
$$+\langle\langle\frac{\partial\tilde{\mathsf{N}}}{\partial q}\mathbf{v},\mathbf{u}\rangle,\delta q\rangle + \langle\langle\frac{\partial\tilde{\mathsf{B}}}{\partial q}\,\mathbf{curl}_\Gamma\,\psi,\mathbf{u}\rangle,\delta q\rangle$$
$$+\langle\langle\tilde{\mathsf{A}}\,\mathbf{curl}_\Gamma\,\psi,\mathbf{curl}_\Gamma\,\omega\rangle,\delta q\rangle - \langle\langle\tilde{\mathsf{C}}\mathbf{v},\mathbf{curl}_\Gamma\,\omega\rangle,\delta q\rangle$$
$$+\langle\langle\tilde{\mathsf{N}}\mathbf{v},\vec{\omega}\rangle,\delta q\rangle - \langle\langle\tilde{\mathsf{B}}\,\mathbf{curl}_\Gamma\,\psi,\vec{\omega}\rangle,\delta q\rangle$$
$$+\langle\int_S\partial_{\mathbf{u}}(j_{\text{obj}}(\mathbf{u},\varphi))\vec{\omega}ds,\delta q\rangle + \langle\int_S\partial_\varphi(j_{\text{obj}}(\mathbf{u},\varphi))\omega ds,\delta q\rangle + \langle\int_S\partial_q(j_{\text{obj}}(\mathbf{u},\varphi))\vec{\omega}ds,\delta q\rangle \quad (3.13)$$

For $\mathbf{u}$ and φ solutions of the forward problem and $\mathbf{v}$ and ψ solutions of the adjoint problem, all terms except the shape derivatives of the operators cancel giving Equation 3.14. The shape derivatives of the involved operators will be discussed in the following chapter. The physical meaning of the operator shape derivatives is the effect of a change in the problem configuration that is quantified by δq on the operators. For a specific direction δq the shape derivative of an operator is represented by a matrix the same size as the operator. In the case of local shape functions this matrix is sparse (but not banded). In the general case of non-local base functions, as the ones considered in this work, this matrix is dense. For all possible directions the shape derivative becomes a three-way tensor. For any discretization of δq by choosing a finite set of N basis functions the shape derivatives of the operators is a set of N matrices. Following the common nomenclature, we change the notation for the solution of adjoint problem from $\mathbf{v}$ to $\mathbf{u}^*$ and from ψ to ϕ^*.

$$\langle\frac{\partial\mathcal{L}}{\partial q},\delta q\rangle = \langle\langle\frac{\partial\tilde{\mathsf{A}}}{\partial q}\,\mathbf{curl}_\Gamma\,\varphi^*,\mathbf{curl}_\Gamma\,\varphi\rangle,\delta q\rangle + \langle\langle\frac{\partial\tilde{\mathsf{C}}}{\partial q}\mathbf{u}^*,\mathbf{curl}_\Gamma\,\varphi\rangle,\delta q\rangle$$
$$+\langle\langle\frac{\partial\tilde{\mathsf{N}}}{\partial q}\mathbf{u}^*,\mathbf{u}\rangle,\delta q\rangle + \langle\langle\frac{\partial\tilde{\mathsf{B}}}{\partial q}\,\mathbf{curl}_\Gamma\,\varphi^*,\mathbf{u}\rangle,\delta q\rangle \quad (3.14)$$

3.2 Linear system for the adjoint equation

We are using the same discretization for the adjoint problem as for the forward problem. The system matrix used for the adjoint problem turns out to be simply the transpose of the system matrix of the forward problem. The loading due to the inhomogeneous term does not exist. By simply omitting the penalty term for the scalar unknowns φ along the "*cut*" direction the unknowns are left to vary.

The loading as commented earlier is simply the difference between the tangential trace of the solution u_{opt} and the tangential trace from the solution of the state equation u_h. The adjoint linear system then reads,

$$\begin{bmatrix} \tilde{N}_h + P_c^v + P_e^v & \tilde{B}_h^\top T^\top \\ -T\tilde{B}_h & T\tilde{A}_h T^\top + P_c^s + \underline{\alpha}\underline{\alpha}^\top \end{bmatrix} \begin{bmatrix} \mathbf{u}^* \\ \varphi^* \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{opt} - \mathbf{u}_h \\ 0 \end{bmatrix} . \tag{3.15}$$

The schur complement method is employed again for the solution of the system above.

Chapter 4

Computation on the Periodic Surface

4.1 Analytical Shape Derivative Formulas

4.1.1 Parametrization of the Torus

The undeformed configuration and the parametric plane

The surface of the torus Γ, can be seen as a rolled-up 2π periodic plane on $\mathbb{R}^2$. We chose a parametrization for the surface of the torus. The transformation from the parameter domain to the torus is

$$\Gamma_0 \coloneqq \left\{ \mathbf{x} : \mathbf{\Phi}(\alpha, \varphi) = \begin{bmatrix} \cos\alpha(r\cos\varphi + R) \\ r\sin\varphi \\ \sin\alpha(r\cos\varphi + R) \end{bmatrix} \quad \varphi, \alpha \in [0, 2\pi[\right\}. \tag{4.1}$$

where R is the large radius of the torus and r the small radius. The normal of the undeformed torus is

$$\mathbf{n}(\alpha,\varphi) = \frac{\partial_\varphi \mathbf{\Phi} \times \partial_\alpha \mathbf{\Phi}}{||\partial_\varphi \mathbf{\Phi} \times \partial_\alpha \mathbf{\Phi}||} = \begin{bmatrix} \cos\alpha\cos\varphi \\ \sin\varphi \\ \sin\alpha\cos\varphi \end{bmatrix} \tag{4.2}$$

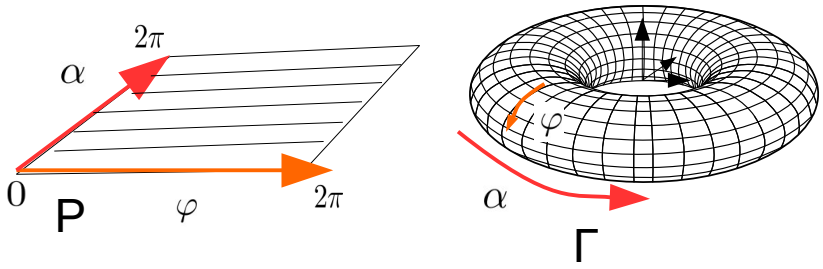
Control Function

We define a scalar control function $q(\bar{\mathbf{x}})$ where $\bar{\mathbf{x}} = [\alpha,\varphi]^\top$ is a point on the periodic parameter plane P. We denote with $\bar{(\cdot)}$ coordinates on P. The control function quantifies a displacement along the normal of Γ_0.

$$q \in H^1(\Gamma_0) \quad : \quad \Gamma_0 \mapsto \Gamma \tag{4.3}$$

where Γ_0 is the reference torus and Γ is the deformed configuration such that

$$\begin{aligned} \Gamma \quad &\coloneqq \quad \Big\{ \mathbf{x} = \hat{\mathbf{x}} + q(\hat{\mathbf{x}})\,\mathbf{n}(\hat{\mathbf{x}}) \qquad , \quad \hat{\mathbf{x}} \in \Gamma_0 \Big\} \\ &= \quad \Big\{ \mathbf{x} = \boldsymbol{\chi}(q;\bar{\mathbf{x}}) \qquad , \quad \bar{\mathbf{x}} \in P \Big\}, \\ &\text{with} \quad \boldsymbol{\chi}(q;\bar{\mathbf{x}}) \coloneqq \mathbf{\Phi}(\bar{\boldsymbol{x}}) + q(\bar{x})\mathbf{n}(\bar{x}). \end{aligned} \tag{4.4}$$



We also define the jacobian of the transformation from the parametric domain to the torus

$$\mathbf{D}\boldsymbol{\chi}(\bar{\mathbf{x}}; q(\bar{\mathbf{x}}))_{ij} = \frac{\partial \boldsymbol{\chi}(\bar{\mathbf{x}}; q(\bar{\mathbf{x}}))_i}{\partial \bar{\mathbf{x}}_j} \tag{4.5}$$

. given explicitly at Equation 4.29.

4.1.2 Single Layer Scalar Potential

The general scalar BEM integral operator reads $\langle \mathsf{K}u, v\rangle$. In what follows we denote that simply as $\mathcal{K}$. The directional derivative of the operator $\mathcal{K}$ along the direction δq is

$$\langle \frac{\partial \mathcal{K}}{\partial q}, \delta q\rangle = \lim_{h\to 0} \frac{\mathcal{K}_{(q+h\delta q)} - \mathcal{K}_{(q)}}{h} \tag{4.6}$$

where h is a scalar and δq is a direction of deformation. A Galerkin discretization of the operator $\mathcal{K}$, is a double integral over the surface of the domain of integration (4.7).

$$\begin{aligned}\mathcal{K} &:= \int_\Gamma \int_\Gamma G^\kappa(\mathbf{x},\mathbf{y})\varphi(\mathbf{y})\psi(\mathbf{x}) ds_\mathbf{y} ds_\mathbf{x} \\ &= \int_\Gamma \int_\Gamma G^\kappa(r(\mathbf{x},\mathbf{y}))\varphi(\mathbf{y})\psi(\mathbf{x}) ds_\mathbf{y} ds_\mathbf{x}\end{aligned} \tag{4.7}$$

with $r = |\mathbf{y} - \mathbf{x}|$.

We seek to perform the integration of 4.7 which is defined on surface Γ on P. In order to achieve that we employ the jacobian of the mapping χ_q denoted as $D\boldsymbol{\chi}(\cdot; q_\cdot)$ and the square root of the Gram determinant which is $\sqrt{|D\boldsymbol{\chi}(\cdot; q_\cdot)^T D\boldsymbol{\chi}(\cdot; q_\cdot)|}$. The transformed integral for the single layer scalar potential is

$$\begin{aligned}\mathcal{K} = \int_P \int_P G^\kappa(r(\bar{\mathbf{x}},\bar{\mathbf{y}}))&\sqrt{|D\boldsymbol{\chi}(\bar{\mathbf{x}}; q_{\bar{\mathbf{x}}})^T D\boldsymbol{\chi}(\bar{\mathbf{x}}; q_{\bar{\mathbf{x}}})|} \\ &\sqrt{|D\boldsymbol{\chi}(\bar{\mathbf{y}}; q_{\bar{\mathbf{y}}})^T D\boldsymbol{\chi}(\bar{\mathbf{y}}; q_{\bar{\mathbf{y}}})|}\varphi(\bar{\mathbf{x}})\psi(\bar{\mathbf{y}}) ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}}.\end{aligned} \tag{4.8}$$

In the following, for brevity we define $\sqrt{|D\boldsymbol{\chi}(\cdot; q_\cdot)^T D\boldsymbol{\chi}(\cdot; q_\cdot)|} = \sqrt{g(\cdot)}$. Then assuming the control function q affects the mapping χ continuously, and the mapping affects the value of the integral continuously as well, the chain rule can be applied to the total derivative of $\mathcal{K}$ with respect to control q. The control affects the value of the integral also through its derivatives (the gradient of the control function)

$$\frac{d\mathcal{K}}{dq} = \frac{\partial \mathcal{K}}{\partial q(\bar{\mathbf{x}})} dq(\bar{\mathbf{x}}) + \frac{\partial \mathcal{K}}{\partial q(\bar{\mathbf{y}})} dq(\bar{\mathbf{y}}) + \frac{\partial \mathcal{K}}{\partial(\partial_\phi q(\bar{\mathbf{x}}))} d(\partial_\phi q(\bar{\mathbf{x}})) + \frac{\partial \mathcal{K}}{\partial(\partial_\alpha q(\bar{\mathbf{y}}))} d(\partial_\alpha q(\bar{\mathbf{y}})) \tag{4.9}$$

We are considering a specific δq function, such that $q' = q + \epsilon\delta q$ with ϵ small

$$\frac{d\mathcal{K}}{dq} = \frac{\partial \mathcal{K}}{\partial q(\bar{\mathbf{x}})} \delta q(\bar{\mathbf{x}}) + \frac{\partial \mathcal{K}}{\partial q(\bar{\mathbf{y}})} \delta q(\bar{\mathbf{y}}) + \frac{\partial \mathcal{K}}{\partial(\partial_\phi q(\bar{\mathbf{x}}))} \delta(\partial_\phi q(\bar{\mathbf{x}})) + \frac{\partial \mathcal{K}}{\partial(\partial_\alpha q(\bar{\mathbf{y}}))} \delta(\partial_\alpha q(\bar{\mathbf{y}})). \tag{4.10}$$

Since the kernel function $G^\kappa(r)$ depends directly only on the distance $r = |\chi(\bar{\mathbf{y}}; q_{\bar{\mathbf{y}}}) - \chi(\bar{\mathbf{x}}; q_{\bar{\mathbf{x}}})|$ we can apply the chain rule differentiating first with respect to r. The derivative of a kernel function w.r.t. the control function reads

$$\begin{aligned}
&\frac{\partial G^\kappa(r(\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}}),\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})))}{\partial q(\bar{\mathbf{y}})} = \\
&= \frac{\partial G^\kappa(r)}{\partial r}\frac{\partial r}{\partial \boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}\frac{\partial \boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}{\partial q(\bar{\mathbf{y}})} \\
&= \frac{\partial G^\kappa(r)}{\partial r}\frac{(\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})-\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}}))}{r}\cdot\frac{\partial \boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}{\partial q(\bar{\mathbf{y}})}.
\end{aligned} \tag{4.11}$$

It also holds that

$$\begin{aligned}
\frac{\partial r}{\partial \boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})} &= \frac{\partial ||\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})-\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})||}{\partial \boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})} \\
&= \frac{1}{r}\big(\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})-\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})\big) \\
&= -\frac{\partial r}{\partial \boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})}.
\end{aligned} \tag{4.12}$$

The expression of the directional derivative for the kernel then reads

$$\langle\frac{\partial G^\kappa(q;\mathbf{x},\mathbf{y})}{\partial q},\delta q\rangle := \frac{\partial G^\kappa(r)}{\partial r}\frac{\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})-\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}{r}\cdot\langle\frac{\partial \boldsymbol{\chi}}{\partial q}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})-\frac{\partial \boldsymbol{\chi}}{\partial q}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}}),\delta q\rangle. \tag{4.13}$$

For the case of the torus we have

$$\langle\frac{\partial \boldsymbol{\chi}}{\partial q}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})-\frac{\partial \boldsymbol{\chi}}{\partial q}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}}),\delta q\rangle = \mathbf{n}(\bar{\mathbf{y}})\delta q(\bar{\mathbf{y}})-\mathbf{n}(\bar{\mathbf{x}})\delta q(\bar{\mathbf{x}}) \tag{4.14}$$

Which is of order $\mathcal{O}(|\bar{\mathbf{x}}-\bar{\mathbf{y}}|)$. Hence taking the derivative of the bilinear form we do not see the singularity of the kernel get stronger due to that term and the $(\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})-\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}}))$ term which is also $\mathcal{O}(r)$. Taking the partial derivative of the product of the gramians we obtain

$$\begin{aligned}
\frac{\partial \mathcal{K}}{\partial q(\bar{x})} = \int_P\int_P\Big(\Big(\frac{\partial G^\kappa(r)}{\partial r}\frac{1}{r}(\boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})-\boldsymbol{\chi}(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}}))\frac{\partial \boldsymbol{\chi}(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})}{\partial q_{\bar{\mathbf{x}}}}\sqrt{g(\bar{\mathbf{y}};q(\bar{\mathbf{y}}))}\sqrt{g(\bar{\mathbf{x}};q(\bar{\mathbf{x}}))}\Big) \\
+ G^\kappa(r)\Big(\frac{\partial\sqrt{g(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})}}{\partial q_{\bar{\mathbf{x}}}}\sqrt{g(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}\Big)\Big)\varphi(\bar{\mathbf{x}})\varphi(\bar{\mathbf{y}})ds_{\bar{\mathbf{x}}}ds_{\bar{\mathbf{y}}}.
\end{aligned} \tag{4.15}$$

The derivative of the bilinear form with respect to the gradients of q reads

$$\frac{\partial \mathcal{K}}{\partial(\partial_\alpha q(\bar{\mathbf{x}}))}\delta\partial_\alpha q(\bar{\mathbf{x}}) + \frac{\partial \mathcal{K}}{\partial(\partial_\varphi q(\bar{\mathbf{x}}))}\delta\partial_\varphi q(\bar{\mathbf{x}}) = \int_P\int_P G^\kappa(r)\Big(\Big(\frac{\partial\sqrt{g(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})}}{\partial(\partial_\alpha q_{\bar{\mathbf{x}}})}\delta(\partial_\alpha q(\bar{\mathbf{x}})) + \frac{\partial\sqrt{g(\bar{\mathbf{x}};q_{\bar{\mathbf{x}}})}}{\partial(\partial_\varphi q_{\bar{\mathbf{x}}})}\delta(\partial_\varphi q(\bar{\mathbf{x}}))\Big)\sqrt{g(\bar{\mathbf{y}};q_{\bar{\mathbf{y}}})}\Big)$$
$$\varphi(\bar{\mathbf{x}})\varphi(\bar{\mathbf{y}})ds_{\bar{\mathbf{x}}}ds_{\bar{\mathbf{y}}}. \quad (4.16)$$

The directional derivative then is

$$\begin{aligned}\langle\frac{\partial\mathcal{K}}{\partial q},\delta q\rangle = \int_P\int_P \Big(\Big(\frac{\partial G^\kappa(r)}{\partial r}\frac{1}{r}(\boldsymbol{\chi}_{(\bar{\mathbf{y}})}-\boldsymbol{\chi}_{(\bar{\mathbf{x}})})\cdot\Big(\frac{\partial\boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}\delta q_{(\bar{\mathbf{y}})} - \frac{\partial\boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}\delta q_{(\bar{\mathbf{x}})}\Big)\sqrt{g(\bar{\mathbf{y}})}\sqrt{g(\bar{\mathbf{x}})}\Big)\\ + G^\kappa(r)\Big(\Big(\frac{\partial\sqrt{g(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}\delta q_{(\bar{\mathbf{x}})} + \frac{\partial\sqrt{g(\bar{\mathbf{x}})}}{\partial(\partial_\alpha q_{(\bar{\mathbf{x}})})}\delta(\partial_\alpha q_{(\bar{\mathbf{x}})}) + \frac{\partial\sqrt{g(\bar{\mathbf{x}})}}{\partial(\partial_\varphi q_{\bar{\mathbf{x}}})}\delta(\partial_\varphi q_{(\bar{\mathbf{x}})})\Big)\sqrt{g(\bar{\mathbf{y}})}\\ + \Big(\frac{\partial\sqrt{g(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}\delta q_{(\bar{\mathbf{y}})} + \frac{\partial\sqrt{g(\bar{\mathbf{y}})}}{\partial(\partial_\alpha q_{(\bar{\mathbf{y}})})}\delta(\partial_\alpha q_{(\bar{\mathbf{y}})}) + \frac{\partial\sqrt{g(\bar{\mathbf{y}})}}{\partial(\partial_\varphi q_{(\bar{\mathbf{y}})})}\delta(\partial_\varphi q_{(\bar{\mathbf{y}})})\Big)\sqrt{g(\bar{\mathbf{x}})}\Big)\Big)\varphi(\bar{\mathbf{x}})\varphi(\bar{\mathbf{y}})ds_{\bar{\mathbf{x}}}ds_{\bar{\mathbf{y}}}.\end{aligned} \quad (4.17)$$

For readability the term $\boldsymbol{\chi}(\bar{\cdot};q_{\bar{\cdot}})$ was replaced with $\chi_{(\cdot)}$ and $\sqrt{g(\cdot;q(\cdot))}$ with $\sqrt{g(\cdot)}$. Since the basis functions $\varphi(\cdot)$ are already on the parameter domain there is no need to take derivatives over them. It is noted that since there is an analytical scalar expression for $\sqrt{g(\cdot)}$and we can straightforwardly derive the partial derivatives that appear in the previous expressions. This is given in Equation 4.36

4.1.3 Single Layer Vector Potential

Now we turn our attention to the Vector Single potential 2.15 and its Galerkin discretization 4.19. The basis functions used for the Galerkin discretization of the operator (the bilinear form) are vector fields tangential to the surface of integration. That means that they will transform according to

$$\boldsymbol{\lambda}(\mathbf{x}) = \frac{D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\sqrt{g_{(\bar{\mathbf{x}})}}}\hat{\boldsymbol{\lambda}}(\bar{\mathbf{x}}) \quad (4.18)$$

which in some texts is refered to as the *"Contravariant Piola Mapping"*. In the following we denote again for brevity $\langle \mathsf{W}\boldsymbol{\lambda}, \boldsymbol{\mu} \rangle$ as $\mathcal{W}$.

$$\langle \mathsf{W}\boldsymbol{\lambda}, \boldsymbol{\mu} \rangle \coloneqq \int_\Gamma \int_\Gamma G^\kappa(\mathbf{x}, \mathbf{y}) \boldsymbol{\lambda}(\mathbf{x}) \boldsymbol{\mu}(\mathbf{y}) ds_\mathbf{y} ds_\mathbf{x} \tag{4.19}$$

The transformed bilinear form to the parameter domain is

$$\begin{aligned} \mathcal{W} &= \int_P \int_P G^\kappa(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \frac{D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\cancel{\sqrt{g_{(\bar{\mathbf{x}})}}}} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \frac{D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\cancel{\sqrt{g_{(\bar{\mathbf{y}})}}}} \boldsymbol{\mu}(\bar{\mathbf{y}}) \cancel{\sqrt{g_{(\bar{\mathbf{x}})}}} \cancel{\sqrt{g_{(\bar{\mathbf{y}})}}} ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}} \\ &= \int_P \int_P G^\kappa(\bar{\mathbf{x}}, \bar{\mathbf{y}}) D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}} \end{aligned} \tag{4.20}$$

The derivative of the jacobian is

$$\langle \frac{\partial D\boldsymbol{\chi}_{(q;\bar{\mathbf{x}})}}{\partial q}, \delta q \rangle = \mathbf{n}(\bar{\mathbf{x}}) \nabla \delta q(\bar{\mathbf{x}})^\top + D\mathbf{n}(\bar{\mathbf{x}}) \delta q(\bar{\mathbf{x}}) \tag{4.21}$$

Where $D\mathbf{n}(\bar{\mathbf{x}})$ denotes the jacobian of the transformation for the normal and

$$\nabla \delta q(\bar{\mathbf{x}}) = \begin{bmatrix} \delta(\partial_\alpha q(\bar{\mathbf{x}})) \\ \\ \delta(\partial_\varphi q(\bar{\mathbf{x}})) \end{bmatrix} \tag{4.22}$$

is the gradient of the deformation. It should be noted that with consistent use of Equation 4.10 we arive at the same result as with Equation 4.21. In order to keep this section compact, the analytical expression is given at Equation 4.35. The total formula reads

$$\begin{aligned} \langle \frac{d\mathcal{W}}{dq}, \delta q \rangle = \\ \int_P \int_P G^\kappa(r) \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \cdot \langle \frac{\partial D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}, \delta q \rangle \boldsymbol{\mu}(\bar{\mathbf{y}}) + \langle \frac{\partial D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}, \delta q \rangle \boldsymbol{\lambda}(\bar{\mathbf{x}}) \cdot D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) \\ \frac{\partial G^\kappa}{\partial r} \frac{1}{r} (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(\frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}} \delta q_{(\bar{\mathbf{y}})} - \frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}} \delta q_{(\bar{\mathbf{x}})} \Big) D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}} \end{aligned} \tag{4.23}$$

At this point, it is worth to be noted that in the single-layer vectorial operator the basis functions cannot be factored out from the first part of the integral. Due to that fact implementation difficulties arise (see subsection 6.5.2).

4.1.4 Double Layer Potential

In the case of the double layer potential, $\langle \mathsf{M}\boldsymbol{\lambda}, \boldsymbol{\mu} \rangle$ or $\mathcal{M}$ for brevity, the bilinear form is 4.24.

$$\begin{aligned}
&\langle \mathsf{M}\boldsymbol{\lambda}, \boldsymbol{\mu} \rangle = \\
&= \int_P \int_P \mathbf{grad}_y \, G^\kappa \cdot \Big(\frac{1}{\cancel{\sqrt{g(\bar{\mathbf{x}})}}} D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times \frac{1}{\cancel{\sqrt{g(\bar{\mathbf{y}})}}} D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) \cancel{\sqrt{g(\bar{\mathbf{y}})}} \cancel{\sqrt{g(\bar{\mathbf{x}})}} d\bar{\mathbf{x}} d\bar{\mathbf{y}} \\
&= \int_P \int_P \mathbf{grad}_y \, G^\kappa \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) d\bar{\mathbf{x}} d\bar{\mathbf{y}} \\
&= \int_P \int_P \frac{\partial G^\kappa}{\partial r} \frac{1}{r} (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) d\bar{\mathbf{x}} d\bar{\mathbf{y}}
\end{aligned} \tag{4.24}$$

The directional derivative of the first part reads

$$\begin{aligned}
\langle \frac{\partial}{\partial q} \Big(\frac{1}{r} \frac{\partial G^\kappa(r)}{\partial r} (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \Big), \delta q \rangle = \\
\langle \frac{\partial}{\partial q} \Big(\frac{1}{r} \frac{\partial G^\kappa(r)}{\partial r} \Big), \delta q \rangle (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) + \frac{1}{r} \frac{\partial G^\kappa}{\partial r} \langle \frac{\partial}{\partial q} (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}), \delta q \rangle = \\
\frac{\partial}{\partial r} \Big(\frac{1}{r} \frac{\partial G^\kappa(r)}{\partial r} \Big) (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \langle \frac{\partial \boldsymbol{\chi}}{\partial q}(q; \bar{\mathbf{y}}) - \frac{\partial \boldsymbol{\chi}}{\partial q}(q; \bar{\mathbf{x}}), \delta q \rangle \cdot (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})})
\end{aligned} \tag{4.25}$$

$$\begin{aligned}
&\frac{\partial \mathcal{M}}{\partial q_{(\bar{\mathbf{y}})}} = \\
&\int_P \int_P \frac{\partial}{\partial r} \Big(\frac{\partial G^\kappa}{\partial r} \frac{1}{r} \Big) \frac{1}{r} \Big\{ (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(\frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}} \delta q_{(\bar{\mathbf{y}})} \Big) \\
&\qquad (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) \Big\} \\
&\quad + \frac{1}{r} \frac{\partial G^\kappa}{\partial r} \Big\{ \Big(\frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}} \delta q_{(\bar{\mathbf{y}})} \Big) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})} \boldsymbol{\mu}(\bar{\mathbf{y}}) \Big) \\
&\quad + \Big(\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})} \Big) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})} \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times \frac{\partial D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}} \boldsymbol{\mu}(\bar{\mathbf{y}}) \delta q_{(\bar{\mathbf{y}})} \Big) \Big\} ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}}.
\end{aligned} \tag{4.26}$$

Again we take note of 4.12 so the final expression for the shape derivative of the double layer operator is

$$
\begin{aligned}
\langle \frac{\partial \mathcal{M}}{\partial q}, \delta q \rangle = \\
&\int_P \int_P \frac{\partial}{\partial r}\Big(\frac{\partial G^\kappa}{\partial r}\frac{1}{r}\Big)\frac{1}{r}\Big\{(\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(\frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}\delta q_{(\bar{\mathbf{y}})} - \frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}\delta q_{(\bar{\mathbf{x}})}\Big) \\
&\quad (\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}\boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}\boldsymbol{\mu}(\bar{\mathbf{y}})\Big)\Big\} \\
&\quad + \frac{1}{r}\frac{\partial G^\kappa}{\partial r}\Big\{\Big(\frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}\delta q_{(\bar{\mathbf{y}})} - \frac{\partial \boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}\delta q_{(\bar{\mathbf{x}})}\Big) \cdot \Big(D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}\boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}\boldsymbol{\mu}(\bar{\mathbf{y}})\Big) \\
&\quad + \Big(\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}\Big) \cdot \Big(\langle \frac{\partial D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}}{\partial q_{(\bar{\mathbf{x}})}}, \delta q\rangle \boldsymbol{\lambda}(\bar{\mathbf{x}}) \times D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}\boldsymbol{\mu}(\bar{\mathbf{y}}) \\
&\quad - D\boldsymbol{\chi}_{(\bar{\mathbf{x}})}\boldsymbol{\lambda}(\bar{\mathbf{x}}) \times \langle \frac{\partial D\boldsymbol{\chi}_{(\bar{\mathbf{y}})}}{\partial q_{(\bar{\mathbf{y}})}}, \delta q\rangle \boldsymbol{\mu}(\bar{\mathbf{y}})\Big)\Big\} ds_{\bar{\mathbf{x}}} ds_{\bar{\mathbf{y}}}
\end{aligned}
\tag{4.27}
$$

Where the shape derivatives $\langle \frac{\partial D\boldsymbol{\chi}(\cdot)}{\partial q(\cdot)}, \delta q\rangle$ are given by Equation 4.21. It should be noted that the order of the singularity of the kernel of the first term of 4.27 is reduced by two due to the $\boldsymbol{\chi}_{(\bar{\mathbf{y}})} - \boldsymbol{\chi}_{(\bar{\mathbf{x}})}$ terms, therefore no special integration issues arise due to the singularity.

4.2 Explicit expressions for Pullbacks, Gramians and their derivatives

In order for the derivation of the shape derivatives of the boundary integral operators to be more straightforward it was deemed beneficial to define the computational problem on a periodic parameter domain in $\mathbb{R}^2$.

4.2.1 Details on pull-back, transformation and Gram determinant

We defined a transformation for the undeformed torus at Equation 4.1 and the parametrization of a deformed torus as deformation along the normal in Equation 4.4. The function $q(\bar{\mathbf{x}})$, for $\mathbf{x} = \{\alpha, \varphi\}$ coordinates on the parametric plane, is a scalar valued function that parametrizes a deformation along the the small radius of the torus which coincides with the normal of the torus. From now on q will be referred to as the control

function. The analytical expression for the jacobian of the transformation for a torus with small radius r is

$$D\mathbf{\Phi}\begin{pmatrix} r \\ \phi \\ \alpha \end{pmatrix} = \begin{bmatrix} \cos\alpha\cos\varphi & -r\cos\alpha\sin\varphi & -\sin\alpha(r\cos\varphi + R) \\ \sin\varphi & r\cos\varphi & 0 \\ \sin\alpha\cos\varphi & -r\sin\alpha\cos\varphi & \cos\alpha(r\cos\varphi + R) \end{bmatrix}. \tag{4.28}$$

The jacobian of the transformation $D\boldsymbol{\chi}(\alpha, \phi)$ is

$$D\boldsymbol{\chi}(\varphi, \alpha) = D\mathbf{\Phi}\begin{pmatrix} q(\alpha,\varphi) \\ \varphi \\ \alpha \end{pmatrix}\begin{bmatrix} \frac{\partial q}{\partial \alpha} & \frac{\partial q}{\partial \varphi} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.29}$$

The previous expression is straightforward to calculate explicitly. It is noted the control function q and the partial derivatives $\partial_\alpha q$, $\partial_\varphi q$ exist and are known. The analytical expression for the gram matrix of the transformation with arbitrary function $q(\alpha, \varphi)$ reads

$$\begin{aligned} D\boldsymbol{\chi}(\varphi,\alpha)^T D\boldsymbol{\chi}(\varphi,\alpha) &= \begin{bmatrix} \frac{\partial q}{\partial \alpha} & 0 & 1 \\ \frac{\partial q}{\partial \varphi} & 1 & 0 \end{bmatrix} D\mathbf{\Phi}^T D\mathbf{\Phi} \begin{bmatrix} \frac{\partial q}{\partial \alpha} & \frac{\partial q}{\partial \alpha} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial q}{\partial \alpha} & 0 & 1 \\ \frac{\partial q}{\partial \varphi} & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & & \\ & q(\varphi,\alpha)^2 & \\ & & (qcos\varphi + R)^2 \end{bmatrix} \begin{bmatrix} \frac{\partial q}{\partial \alpha} & \frac{\partial q}{\partial \alpha} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} (\partial_\varphi q)^2 + q^2 & \partial_\varphi q \partial_\alpha q \\ \partial_\alpha q \partial_\varphi q & (\partial_\alpha q)^2 + (qcos\varphi + R)^2 \end{bmatrix}. \end{aligned} \tag{4.30}$$

The square root of the gram matrix determinant is simply

$$\sqrt{|D\boldsymbol{\chi}(\varphi,\alpha)^T D\boldsymbol{\chi}(\varphi,\alpha)|} = \sqrt{q^2(\partial_\varphi)^2 + (qcos\varphi + R)^2(\partial_\varphi q)^2 + q^2(qcos\varphi + R)^2} \tag{4.31}$$

4.2.2 Derivatives with respect to q

In order to calculate the shape derivatives we need to calculate the derivatives of the transformation, the jacobian and the gram determinant with respect to the control q. The derivative of the transformation is

$$\frac{\partial \boldsymbol{\chi}(\alpha,\varphi)}{\partial q} = \begin{bmatrix} cos\alpha cos\varphi \\ sin\varphi \\ sin\alpha cos\varphi. \end{bmatrix} \tag{4.32}$$

For the derivative of the jacobian w.r.t. the control we need

$$D\frac{\partial \boldsymbol{\chi}}{\partial q} = D\mathbf{n} = \begin{bmatrix} -sin\alpha cos\varphi & -cos\alpha sin\varphi \\ 0 & cos\varphi \\ \underbrace{cos\alpha cos\varphi}_{\frac{\partial \mathbf{n}}{\partial \alpha}} & \underbrace{-sin\alpha sin\varphi}_{\frac{\partial \mathbf{n}}{\partial \varphi}} \end{bmatrix}. \tag{4.33}$$

We also need the quantity

$$\begin{aligned} &\frac{\partial D\boldsymbol{\chi}}{\partial(\partial_\alpha q)}\delta(\partial_\alpha q) + \frac{\partial D\boldsymbol{\chi}}{\partial(\partial_\varphi q)}\delta(\partial_\varphi q) = \\ &\begin{bmatrix} cos\alpha cos\varphi & 0 \\ sin\varphi & 0 \\ sin\alpha cos\varphi & 0 \end{bmatrix}\delta(\partial_\alpha q) + \begin{bmatrix} 0 & cos\alpha cos\varphi \\ 0 & sin\varphi \\ 0 & sin\alpha cos\varphi \end{bmatrix}\delta(\partial_\varphi q) = \begin{bmatrix} cos\alpha cos\varphi \\ sin\varphi \\ sin\alpha cos\varphi \end{bmatrix}\begin{bmatrix}\delta(\partial_\alpha q) & \delta(\partial_\varphi q)\end{bmatrix} \\ &= \mathbf{n}\nabla\delta q^\top \end{aligned} \tag{4.34}$$

The derivative of the jacobian finally reads

$$\begin{aligned} \langle\frac{\partial D\boldsymbol{\chi}}{\partial q}, \delta q\rangle = \mathbf{n}\nabla\delta q^\top + D\mathbf{n}\delta q = \\ \begin{bmatrix} cos\alpha cos\varphi \\ sin\varphi \\ sin\alpha cos\varphi \end{bmatrix}\begin{bmatrix}\delta(\partial_\alpha q) & \delta(\partial_\varphi q)\end{bmatrix} + \begin{bmatrix} -sin\alpha cos\varphi & -cos\alpha sin\varphi \\ 0 & cos\varphi \\ cos\alpha cos\varphi & -sin\alpha sin\varphi \end{bmatrix}\cdot\delta q \end{aligned} \tag{4.35}$$

the derivative of the square root of the gram matrix is

$$\frac{\partial\sqrt{g}}{\partial q} = \frac{1}{\sqrt{g}} cos\varphi(qcos\varphi + R)((\partial_\varphi q)^2 + q^2) + q((\partial_\alpha q)^2 + (qcos\varphi + R)^2) \tag{4.36}$$

and the derivatives of the grammian w.r.t. the gradients of q are

$$\frac{\partial\sqrt{g}}{\partial(\partial_\alpha q)} = \frac{1}{\sqrt{g}}((\partial_\alpha q)q^2 \quad \text{and} \quad \frac{\partial\sqrt{g}}{\partial(\partial_\varphi q)} = \frac{1}{\sqrt{g}}(\partial_\varphi q)(qcos\varphi + R)^2 \tag{4.37}$$

4.3 Integrating for the shape derivatives

The shape derivatives of each operator quantify the effect of an infinitestimal deformation along a specific direction $\delta q\mathbf{n}$. Thus in order to have a representation of the combined effect of a finite directional deformation to the objective function we should consistently apply to the shape derivatives the same surface operators that we have applied for the construction of the original system in the spirit of 4.39, 2.40, 2.41. This leaves us with

$$\mathsf{L}_{\delta q_i} = (\mathbf{u_q}^*)^\top \tilde{\mathsf{N}}^{sd}_{q;\delta q_i}\mathbf{u_q} + (\varphi_q{}^*)^\top \tilde{\mathsf{B}}^{sd}_{q;\delta q} T^\top \mathbf{u_q} - (\mathbf{u_q}^*)^\top T^\top \tilde{\mathsf{B}}^{sd}_{q;\delta q}\varphi_q + (\varphi_q^*)^\top \tilde{\mathsf{A}}^{sd}_{q;\delta q}\mathbf{u_q} \tag{4.38}$$

with

$$
\begin{aligned}
\tilde{B}^{sd}_{q;\delta q_m} &\coloneqq B^{\kappa}_{q;\delta q_m} + B^{0}_{q;\delta q_m} \\
\tilde{A}^{sd}_{q;\delta q_m} &\coloneqq A^{0}_{q;\delta q_m} + A^{\kappa}_{q;\delta q_m} \\
\tilde{\mathsf{N}}^{sd}_{q;\delta q_m} &\coloneqq N^{\kappa}_{q;\delta q_m} + N^{0}_{q;\delta q_m} \\
N^{\kappa}_{q;\delta q} &\coloneqq \kappa^2 A^{\kappa}_{q;\delta q_m} \\
A^{0}_{q;\delta q_m} &\coloneqq Y^{0,1}_{q;\delta q_m} \\
A^{\kappa}_{q;\delta q_m} &\coloneqq Y^{\kappa,1}_{q;\delta q_m} + \tfrac{1}{\kappa^2} D Y^{\kappa,2}_{q;\delta q_m} D^{\top} \\
N^{0}_{q;\delta q} &\coloneqq D Y^{0,2}_{q;\delta q_m} D^{\top}
\end{aligned}
\tag{4.39}
$$

$$
\begin{aligned}
B^{\kappa}_{q;\delta q_m}[i,j] &\coloneqq \langle \frac{\partial \mathcal{M}}{\partial q}(q;\mathbf{u}_i,\mathbf{u}_j),\delta q_m\rangle \quad \mathbf{u_i},\mathbf{u_j} \in \mathbf{H}_{\parallel}^{-1/2}(\mathrm{div}_\Gamma,\Gamma) \quad (\text{using } 4.27) \\
Y^{\kappa,1}_{q;\delta q_m}[i,j] &\coloneqq \langle \frac{\partial \mathcal{K}^{\kappa}}{\partial q}(q;\mathbf{u}_i,\mathbf{u}_j),\delta q_m\rangle \quad \mathbf{u_i},\mathbf{u_j} \in \mathbf{H}_{\parallel}^{-1/2}(\mathrm{div}_\Gamma,\Gamma) \quad (\text{using } 4.23) \\
Y^{\kappa,2}_{q;\delta q_m}[i,j] &\coloneqq \langle \frac{\partial \mathcal{W}^{\kappa}}{\partial q}(q;\psi_i,\psi_j),\delta q_m\rangle \quad \psi_i,\psi_j \in H^{-1/2}(\Gamma) \qquad (\text{using } 4.17)
\end{aligned}
\tag{4.40}
$$

which is the discrete version of 3.14. The discretization of the control is discussed in chapter 5.

Chapter 5

Numerical Results

5.1 Optimization problem set-up

The basis functions that were used to represent the deformation were different versions of truncated Fourier series for all examples since the problem considered is periodic. The forward problem is calculated once with the deformed geometry and the result is saved on the disk. Then the optimization run consists of starting from an undeformed torus shape and attempting to achieve the previously computed configuration of discrete Neumann traces on the surface by minimizing Equation 3.12 through the solution of the adjoint problem and the use of the analytical shape derivative formulas. In the following examples the gradients are normalized according to their L^2 norms.

In the following sections some representative computational examples are presented. Only a small number of basis functions were considered. The reason was that computationally heavy exact boundary element matrix assembly was employed. In order to reach the following results no approximation technique was used. An approximation technique at least for the assembly of the integrals is essential for the practical usefulness of the method.

5.1.1 Line search strategy

Initially exact line search was employed but it was found to be quite inefficient.Therefore an inexact line-search was employed. The strategy was simply the bisection of the step length in case the proposed step was leading to worse results but in case this process was repeated more than 4 times the latest proposed step would be accepted and the optimization would continue with computing a new descent direction.

5.1.2 Sphere enclosed by torus

Throughout the computation dimensionless units were considered. In the first computational example the initial shape is an undeformed torus with large radius $R = 0.01$ and small radius $r = 2 * 10^{-}3$. Non-local excitation is produced by constraining the scalar trace to have a unit jump along the cut as it was elaborated in section 2.3.3. In the present example the target shape is a torus with deformation

$$q^a_{opt} = \frac{r}{3}(sin(\alpha) + 0.5sin(2\alpha))sin(2\phi) \tag{5.1}$$

along the normal where α, ϕ are angles as elaborated in section 4.1. Two examples are presented with the torus/sphere configuration. In the first one the integration for the loading of the adjoint is performed only on the enclosed sphere, and in the second one the integration is performed on the entire computational domain. The basis functions that span the control space are

$$\delta q^a_{kl} = \sum_{k=1}^{2}\sum_{l=1}^{3} sin(k\alpha)sin(l\phi) \tag{5.2}$$

Integration on the sphere

As seen in Figure 5.1 the objective function diminishes quite consistently. The figure on the right of 5.1 is the evolution of the contribution of the basis functions (the

optimization parameters). A cross section of the configuration the optimization process produced with an outline of the target configuration boundary are given in Figure 5.3. Observing Figure 5.2 we can argue that the loading of the adjoint problem, or equivalently the objective function, decays with the procedure quite fast and quite consistently for visually indistinguishable variations on the torus surface. However, the shape of the torus was not recovered with this example possibly due to the fact that the problem is severely ill posed.

In the third and fourth examples an alternative configuration of measurement domain and torus is presented that allows for reconstruction of the original shape of the torus by data on a separate domain alone.

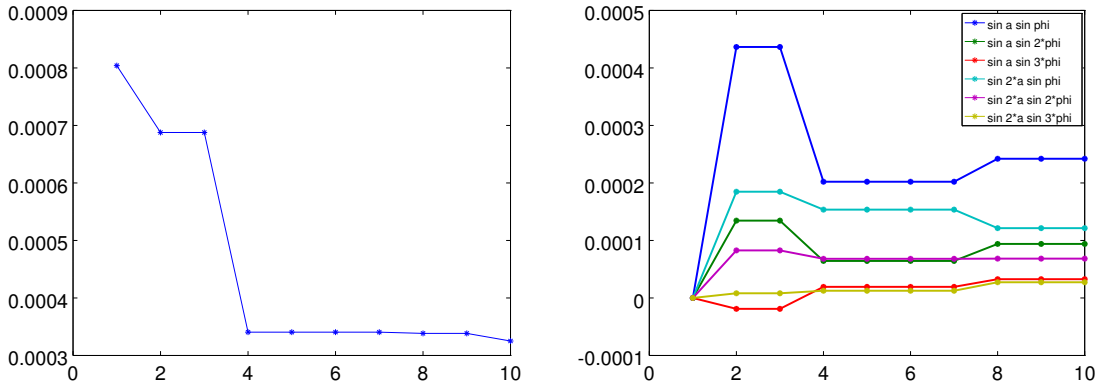


Figure 5.1: *Left*: Objective function evolution - example 1 (line-search steps) *Right*:Evolution of design vector - example 1 (line-search steps).

Integration on the entire domain

As validation of the implementation tests were run where the integration domain was the entire computational domain. The rationale behind these tests was that when the integration is the entire domain the shape reconstruction problem ceases to be so ill-posed

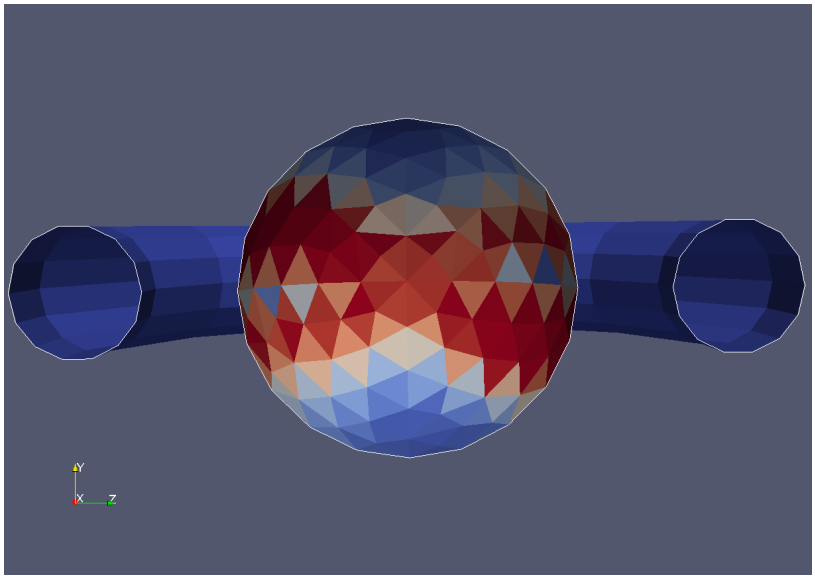


Figure 5.2: Sphere enclosed by torus - integration only on sphere. Initial adjoint loading. Color represents the cell averaged local contribution of the adjoint loading.

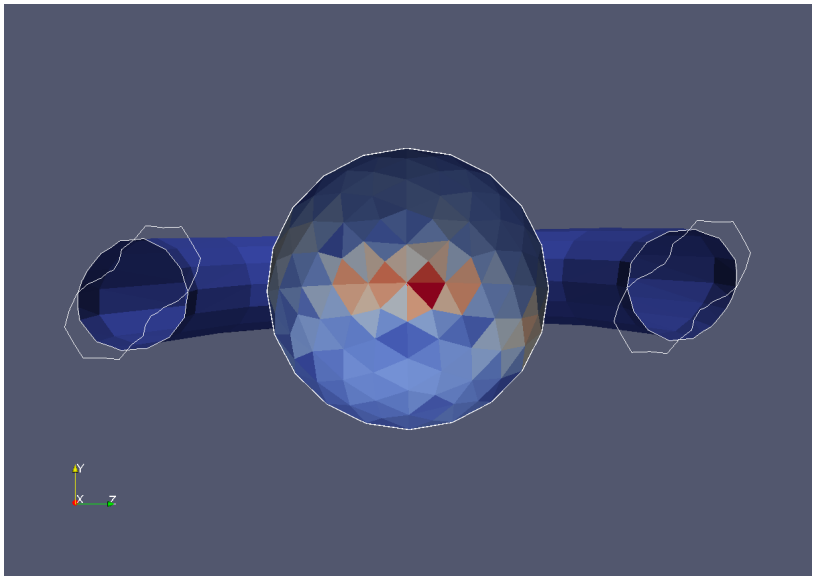


Figure 5.3: Sphere enclosed by torus - integration only on sphere. Color is the local contribution to the loading of the adjoint problem. Scale is identical to Figure 5.2

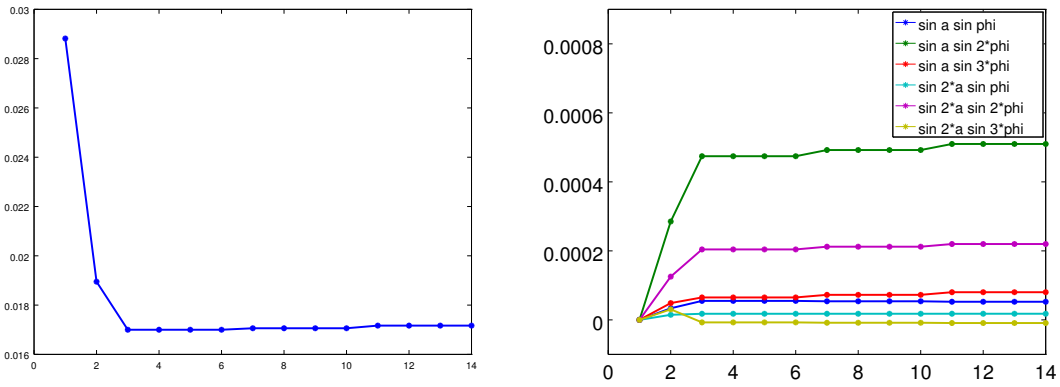


Figure 5.4: Left: *Objective function evolution - example 2 (x axis is line-search steps)* Right : *Evolution of design vector - example 2 (x-axis is line-search steps).*

as in the first example and reconstruction should be observed in a much smaller number of steps. In Figure 5.5 we can observe visually that the shape is indeed approximately reconstructed by the algorithm.

In Figure 5.4 the evolution of the objective function and of the design vector is presented. The full shape is not reconstructed due to the small number of steps but we can observe that the basis functions δq_{12} and δq_{22} assume quite fast values close to their optimal and continue to approach them while the contributions of all the other basis functions to the shape are and remain small as they are expected.

5.1.3 Integration on external plate probe

Finally optimization runs were executed where the integration was performed on a coarsely meshed external plate probe.

Optimization with target shape defined by q^a_{opt}

As seen in Figure 5.6 the shape is approximately reconstructed. In this example the very small number of line search steps (4) led to divergence of the method in few

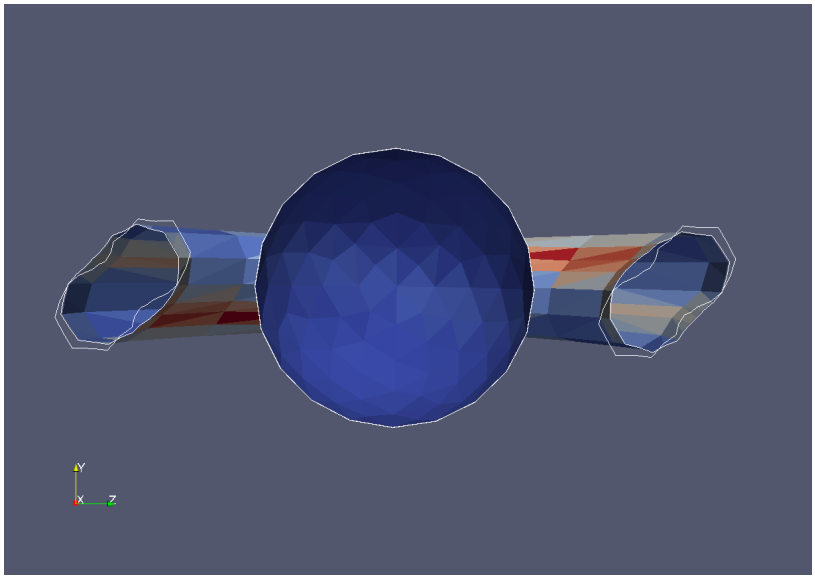


Figure 5.5: Sphere enclosed by torus - integration on both domains. The target shape is sketched in the cross section with a white the white line does not coincide with the outline of the deformed mesh.

steps. Due to time restrictions the test could not be run with different parameters. Instead a different example presented in the following subsection with an interesting, yet easier to reconstruct, shape was preferred to validate the developed method but also with integration on two plate probes.

Optimization with target shape defined by q^b_{opt}

As a final example a target shape was investigated with deformation

$$q^b_{opt} = 0.3r\cos(\alpha) - 0.3r\cos(2\alpha) + 0.1r\cos(3\alpha) + 0.2r\sin(2\varphi) \tag{5.3}$$

and with control discretization

$$\delta q^b_{kl} = \sum_{k=0}^{3} cos(k\alpha) + \sum_{l=1}^{2} \sin(l\varphi). \tag{5.4}$$

In this example the integration was performed on two plate probes were employed for integration enclosing the torus from two sides. It was observed that there is fast convergence to

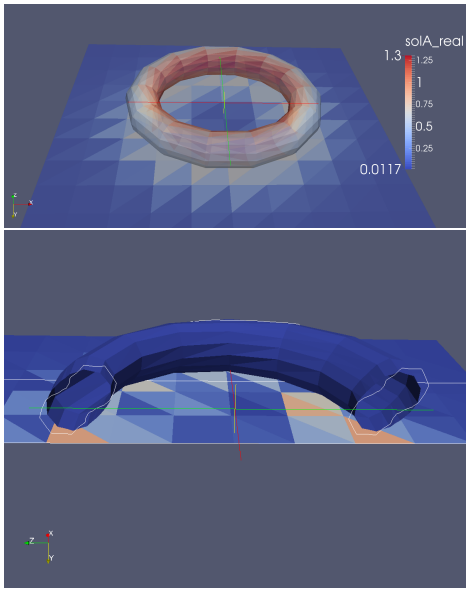


Figure 5.6: top:*Plate probe under torus - initial shape, forward solution*, bottom:*final shape for* q^a_{opt} *target deformation.*

the target configuration as seen in Figure 5.1.3. The objective function decreases accordingly as seen in Equation 5.1.3.

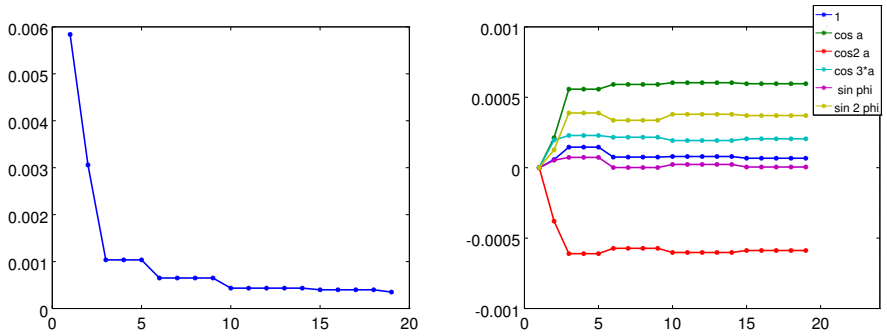


Figure 5.7: Left : *Evolution of objective function with linesearch step for* q^b_{opt}. Right: *Evolution of shape parameters with linesearch step for* q^b_{opt}.

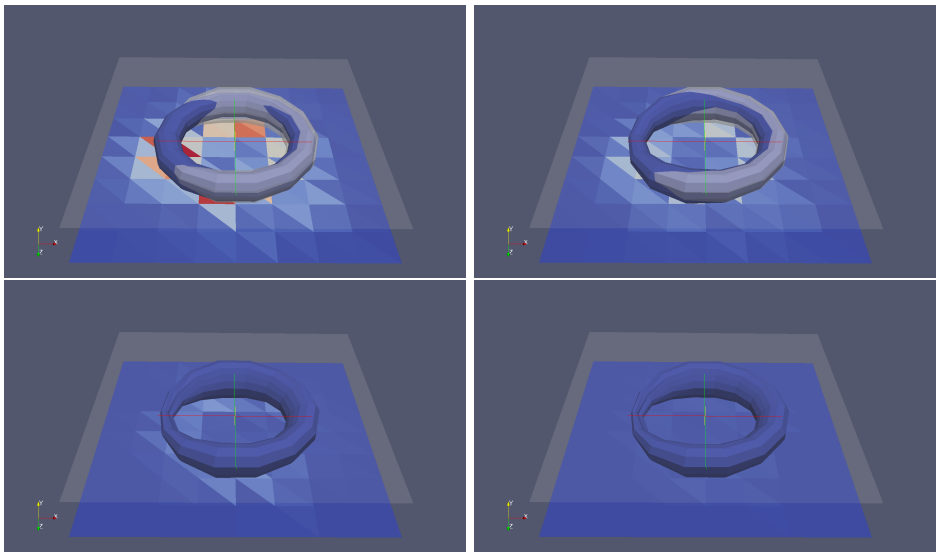


Figure 5.8: Left to right and top to bottom: *Torus shape - initial, step 1, step 2, step 6 for target* q^b_{opt}. *The gray outline is the target shape. The colors represent the local contributions to the loading of the adjoint and they diminish. The target shape practically coincides with the shape reconstructed at step 6.*

Chapter 6

Implementation

The implementation of the proposed method without causing bugs to the rest of BETL2 posed various interesting challenges. In the following the presentation will be focused on usage of the implemented classes and the extended parts of the code. Specific implementation details will be presented only when they possess some value to possible extensions and improvements of the code. The future developers of this work are strongly encouraged to read this part.

6.1 Parsing a mapped mesh

In order to facilitate the validation of the implementation of the "mapped" BEM operators special constructors of the mesh input interface and the internally used mesh parser were created that accept a functor that realizes the mapping. On input the nodes are mapped according to the functor thus there is a direct correspondence of the degrees of freedom computed on the parameter plane with the degrees of freedom computed on the torus. A usage example of the instantiation of such an input interface is given below.

```
1 namespace big = betl2::input::gmsh
2 // Where basename_par, the parameter plane mesh, and transf_func(...) a ↩
    functor used for the mapping
3 big::Input input_tor_minus( basename_par, transformation::transf_func(r_, R_↩
    , current_q_functions.q0_));
```

6.2 The `ParamatricDiffeomorphism` class

The first and most important challenge was to implement efficiently the computation of the additional jacobians and gramians used for the computation of the BEM operators on the parameter domain. The boundary element method, especially when used without acceleration techniques for the integration, was found to be rather unforgiving with respect to inefficient parts of code due to the underlying double integration. The initial approach was to implement this through the use of modified kernel functions. This approach was abandoned due to performance issues but also due to the fact that it complicates unnecessarily the implementation of the shape derivative integrators as it will be elaborated later. The most crucial performance consideration that led to the abandonment of this approach was the absence of facilities for caching the fundamental solution evaluations. If the additional quantities were to be computed along with the relatively cheap fundamental solution evaluation an increase on operation count would occur that would be both unnecessary and significantly detrimental to the performance. Of course caching the fundamental solution evaluations would have been too memory costly with small to non-existent performance gains for the usual operators and this is apparently why such an approach is not followed in the code. On the other hand there are caching facilities for the `Geometry` class and these facilities were extended to accommodate the high operation count computations for the additional jacobians and grammians.

Key to the management of these additional quantities is the `ParametricDiffeomorphism` class. This class accepts a set of functions for the additional quantities needed to implement the diffeomorphism and the modified `Geometry` class and the `FSLayerTraits` specializations that were implemented use these additional quantities. This had the advantage that since the `FSLayerTraits` and the `RuntimeCache` are requesting the geometry related quantities through the `Geometry` class they required minor modifications. However in order to implement caching for quantities relevant only to the shape derivatives the `RuntimeCache` class was extended. As a representative example of how the geometry class that was implemented works is given in Listing 6.1.

```
1
2
3  template< eth::base::RefElType RET, int NUM_NODES,
4            int DIM_TO , int NUM_LOCAL_POINTS>
5  class GeometryImplParam :
6    public GeometryImplBaseTemplate<GeometryImplParam<...>,
7                                                    ... > //CRTP base ↩
                                                        class
8  {
9    .
10   .
11   .
12   inline
13   matrix_t< dimFrom, dimTo*NUM_LOCAL_POINTS >
14   jacobianTransposed( const matrix_t<dimFrom,NUM_LOCAL_POINTS>& local )
15     const
16   {
17     // compute gradients
```

```
18      const matrix_t< NUM_NODES, dimFrom*NUM_LOCAL_POINTS > gradients =
19        geometryImplTraits::GradientShapeFun_t::Eval( local );
20      // instantiate J^T
21      matrix_t< dimFrom, dimTo*NUM_LOCAL_POINTS > JT;
22
23      //carefull - here I need the points un-transformed
24      //to the parametric surface! This is what the
25      //additional jacobian fcn uses
26      matrix_t<dimTo, NUM_LOCAL_POINTS> param_global = nodal_coords_ *
27                      geometryImplTraits::ShapeFun_t::template Eval<↩
                            NUM_LOCAL_POINTS>( local );
28
29      // go through all local points
30      for( int i = 0; i < NUM_LOCAL_POINTS; ++i ) {
31        //we want the jacobian returned to include the one
32        //introduced due to the parameter domain -> torus mapping
33        JT.template block<dimFrom,dimTo>(0, i*dimTo )
34          = gradients.template block<NUM_NODES,dimFrom>(0,i*dimFrom).↩
                transpose() *
35            nodal_coords_.transpose() *
36              const_cast<ParametricDiffeomorphism<DIM_TO,DIM_TO>& >(↩
                    par_diff_).getJacob(
37                 param_global.template block<dimTo,1>(0,i)).transpose();
38
39      }
40      return  JT;
41    }
42  }
```

Listing 6.1: GeometryImplParam

6.3 The `OperatorFactory` class

This class manages through template argument deduction and a simple *traits* class consistently the types for various interconnected objects in order to compute any of the operators needed for our problem. The object keeps a `RuntimeCache` object internally in order to avoid recomputing cached quantities for operators defined on the same mesh. Though some quantities are re-initialized, their initialization is computationally trivial for the problems considered and the main code becomes much simpler and easier to debug. The most important method is given bellow. Use of such encapsulation utilities is essential to the academically oriented usage of the library.

```
1  class OperatorFactory{
2  protected:
3    const grid_factory_t& grid_factory;
4    cache_t cache;
5    bool cache_contains_pdiff=false;
6    //singularity_detector_t singularity_detector;
7  public:
8    OperatorFactory(const grid_factory_t& grid_factory_)
9     : grid_factory(grid_factory_),
10      cache(grid_factory_)
11   {
12   };
13
14   template< class GALERKIN_KERNEL_T>
15   Eigen::Matrix<typename GALERKIN_KERNEL_T::numeric_t,Eigen::Dynamic,Eigen::↩
         Dynamic> build_operator(GALERKIN_KERNEL_T& galerkin_kernel)
16   {
17     singularity_detector_t singularity_detector(grid_factory);
```

```
18    typedef typename GALERKIN_KERNEL_T::testBasis_t feb_t;
19
20    typedef betl2::fe::DofHandler<feb_t,
21                                  FETypeLocalTraits<feb_t>::continuity,
22                                  grid_factory_t > dh_febasis_t;
23    dh_febasis_t dh_febasis;
24    dh_febasis.distributeDofs(grid_factory);
25
26    //integrator:
27    typedef bem::GalerkinIntegrator< GALERKIN_KERNEL_T , typename ↩
          FETypeLocalTraits<feb_t>::integration_traits > integrator_t;
28    integrator_t integrator(galerkin_kernel, singularity_detector, cache);
29
30    typedef BemOperator< integrator_t,
31                         typename dh_febasis_t::fespace_t > bem_op_t;
32
33    bem_op_t bem_op(integrator,dh_febasis.fespace() );
34    bem_op.compute();
35    return bem_op.matrix();
36  }
```

Listing 6.2: OperatorFactory

6.4 Implementation of Penalty method

Since BETL2 does not have any boundary condition enforcing capabilities a design approach that could be generalized in the future was chosen. The rationale behind the periodic boundary conditions is that the degrees of freedom that lie on corresponding faces that have to be constrained have to be identified separately. Then corresponding pairs

have to be identified in order to build the relevant vectors and matrices for the constraints. This task is achieved with the `CorrespondanceDeductor` class. This class operates through functors that represent the geometric conditions that define if a degree of freedom is on a face to be set to periodic conditions and a simple boolean denoting in which side. Keeping track of which side a degree of freedom lies is essential for the non-homogeneous scalar constraint. For non-homogeneous constraints for vector elements it might be important to keep track of the orientation but this issue does not occur in our case. The `nonHomogPenalty` function is presented bellow where the left hand side and right hand side of the penalty contributions are returned. It accepts a simple vector tuple that contains two integers denoting the matched DOFs and a boolean that denotes whether the orientation is conforming or not between vectorial degrees of freedom.

```
1  std::pair<systemMatrix_t, systemMatrix_t> nonHomogPenalty(std::vector<std↩
       ::tuple< int , int , bool> > mfc, double val, int numdofs){
2    //iterate through the map
3    //set diagonals to 1 off-diags to -1
4    systemMatrix_t lhs(numdofs, numdofs);
5    lhs.setZero();
6    systemMatrix_t rhs(numdofs, 1);
7    rhs.setZero();
8    for(int ind = 0 ; ind < mfc.size() ; ind++){
9      int dof1 = std::get<0>(mfc[ind]);
10     int dof2 = std::get<1>(mfc[ind]);
11     lhs(dof1,dof1) = 1;
12     lhs(dof2,dof2) = 1;
13     //the No2 position of the tuple contains a bool that tells us if the ↩
           orientation is
14     //consistent:
```

```
15        if(std::get<2>(mfc[ind]) == false){
16          std::cout<<"spotted orientation dissagreement for pair :"<<dof1 << "↩
                , " << dof2<<std::endl;
17        }
18        lhs(dof1, dof2) = std::get<2>(mfc[ind])? -1 : 1;
19        lhs(dof2, dof1) = std::get<2>(mfc[ind])? -1 : 1;
20        rhs(dof1) = val;
21        rhs(dof2) = -val;
22      }
23      return std::pair<systemMatrix_t, systemMatrix_t >(lhs, rhs);
24    }
25  };
```

Listing 6.3: nonHomogPenalty

6.5 Implementation of Shape Derivative Integrators

6.5.1 Implementation through the `Geometry`

For each case of shape derivative integrator (single layer scalar, single layer vector, double layer vector) a separate `FSLayerTraits` specialization had to be created. As a basis for the implementation of the shape derivative integrators the corresponding operators were used. For each of these cases the code needed to be edited in 3 parts that were quite similar. A representative example of the editing of the double layer integrator (or more precisely the `FSLayerTraits` is given bellow. The quantities `glo_dfi, dJyi R` all depend on computations from a `ParametricDiffeomorphism` instance. These quantities are not available through the BETL2 default geometry class and this was the reason the `ParametricDiffeomorphism` class had to be created and the `Geometry` class had to

be edited. One reason for this choice was that it was not necessary for the integrators for the regular BEM operators to be edited. Simply by making the geometry object contained in each element to return the jacobian or gramian "augmented" with the terms from `ParametricDiffeomorphism` when it contains such an object was sufficient for correct integration.

It is noted that a template specialization of the geometry would not have been sufficient and conceptually not integratable to BETL2 since the geometry has nothing to do with the nature of the operator (if it is a shape derivative operator or a regular operator). In the author's opinion the concept of BEM operator in BETL2 is insufficient to accommodate for the concept of the shape derivative of a BEM operator. However it is apparent that conceptually the object that should provide encapsulation for additional geometric quantities should be related somehow to geometry.

6.5.2 Implementation with fundamental solutions

The kernel functions return a number to the integrator given the global points. Neglecting for a second the serious performance issues related to the (sound design choice of) absence of caching for fundamental solution evaluations we briefly consider an implementation of the shape derivative operators through only the fundamental solutions (a template specialization of the `FundSol` class). First of all, we have to observe that all shape derivative integrals consist of two separate kernels. The implementation of the single-layer vector operator on the first part of the integral in Equation 4.23 contains inner products of the shape derivatives of the jacobians of the parametrization with the local to global jacobians. One possible way of dealing with this would be passing the local to global jacobians to the fundamental solution object for computation of their inner products with the derivatives of the parameter domain to torus jacobians. This approach would violate the

concept of fundamental solutions that are implemented simply to return a scalar value and was rejected.

6.5.3 A rough implementation proposition for shape derivatives on BETL2 with finite elements

The proposed way for general shape optimization (and partially implemented but not documented since it did not produce any results for the final project) would be an operator that accepts not two but three `FESpace` objects. The two are regular finite element spaces used for the BEM integration as BETL already implements but the third one should be a separate finite element space used to discretize the shape derivative. The return type for such an operator would be a sparse (but not banded) matrix for every degree of freedom of the control space. The shape derivative integrator has to implement even a naive way to identify the basis functions that have common support in the resulting triple integral since looping through all the elements as in dense BEM operators would be inefficient and unnecessary. It has to be noted that since all BETL2 BEM related classes are designed for dense operators, as BEM operators essentially are if approximation techniques for the integration are not used, their matrix representations are hard-coded on template parameters. A suitable data structure that would accommodate the shape derivative matrices of a discretization of the BEM operators with shape derivatives would have been something in the lines of a vector of sparse matrices. Thus it seems that in order for shape optimization with BEM to be feasible in BETL2 there are many extensions of BETL2 that have to be implemented.

```
1 for( int fx = 0; fx < NUM_ROWS; ++fx ) {
2     for( int fy = 0; fy < NUM_COLS; ++fy ) {
```

```
3         const dmatrix_t< dimFrom,1 >& Fy = evalY.template block< dimFrom, 1 ↩
              >( fy*dimFrom,gpY );
4         const int idx = utils::majorCol< NUM_ROWS, NUM_COLS >( fx, fy );
5         Eigen::Matrix<double,3,1> dFFy = (dJyi * Fy).eval();
6         Eigen::Matrix<double,3,1> FFy  = (Jyi  * Fy).eval();
7
8         auto crossProd_A = FFy.cross(glo_fi.template block< dimTo, 1>(0,fx))↩
              ;
9         auto crossProd_dx = FFy.cross(glo_dfi.template block< dimTo, 1>(0,fx↩
              ));
10        auto crossProd_dy = dFFy.cross(glo_fi.template block< dimTo, 1>(0,fx↩
              ));
11
12        result ( idx, 0 ) = result( idx, 0 )
13            + ddU_eval * R_dxy * R.dot( crossProd_A )
14            + dU_eval * ( dxy.dot( crossProd_A)
15            + R.dot(crossProd_dx + crossProd_dy) );
16    }
17};
```

Listing 6.4: "part of`GalerkinKernelLayerTraits_DL_sd`"

6.6 Function Bundle

It was very convenient to define the basis functions and their gradients in the present implementation as `structs` that contain lambda functions that are scaled according to the design vector. Addition subtraction and scaling are defined for these objects with operator overloading. A static factory method is provided that permits the easy construction of `ParametricDiffeomorphism` objects from `FunctionBundle` objects.

6.7 Basic Multithreading Control

When using the mapping from the parametric domain to the torus (for non-shape derivative operators) no threading issues arise. However, the setting and un-setting of a flag that controls the behavior of `ParametricDiffeomorphism` that was intended to be removed in later stages of development caused race conditions. BETL2 does not have high level facilities for threading management. Thus the code was edited to compute the mapped operators with multiple threads (used for the assembly of the linear system) and the shape derivative operators with only one thread. This was achieved by a trivial traits class. The cleanest way to implement the single threading was to instruct METIS to create a single cluster and adding a `const bool thread_safe` flag in the integrator. Future developers are referred to file *bem_operator_dense_impl_mt.hpp*.

```
1  namespace betl2 {
2    namespace bem {
3
4      template <FSLayer T>
5      struct ThreadSafeTraits{static const bool is_thread_safe=true;};
6
7      template <> struct  ThreadSafeTraits<FSLayer::SL_sd>{ static const bool ↩
           is_thread_safe=false; };
8      template <> struct  ThreadSafeTraits<FSLayer::DL_sd>{ static const bool ↩
           is_thread_safe=false; };
9      ...
10   }
11 }
```

Listing 6.5: ThreadSafeTraits

6.8 Eddy Current Solver

This is the main class used for the computation of the adjoint and the forward problem. In `main` it is used as shown bellow.

```
1  const ParametricDiffeomorphism<3,3> par_diff(FuncBundle::CreateParDiff(↩
       current_q_functions,FuncBundle::Zero(),true));
2  //instantiate a finite element space only to find the degrees of freedom↩
        easilly!
3  const grid_ptr_t grid_ptr_dofs(new grid_t(inpInterface_cyl, ↩
       inpInterface_par,
4                                    transformation::no_transf_pt(),
5                                    transformation::no_transf_jacob() ,
6                                    transformation::no_gram( )) );
7  const grid_ptr_t grid_ptr_param( new grid_t( inpInterface_cyl, ↩
       inpInterface_par, par_diff));
8  const grid_factory_t grid_factory_par(grid_ptr_param);
9  const grid_factory_t grid_factory_dofs(grid_ptr_dofs);
10 //initializing the solver:
11 EddyCurrentSolver eddy_current_solver(grid_factory_par, ↩
       grid_factory_dofs);
12
13 eddy_current_solver.initializeMatrices();
14 SolutionPair solution_fw  = eddy_current_solver.solve_forward(); //↩
       solves with  B and all constraints
15 //...
16 SolutionPair solution_adj = eddy_current_solver.solve_adjoint(); //↩
       solves with -B and missing the jump constraint, replaced with free ↩
       scalar values along the same cut.
```

Listing 6.6: `EddyCurrentSolver` usage

6.9 Usage

Except when the `--help` flag is used the first two arguments should be the mesh input files. The first input file is a 2π periodic plane mesh with $z = 0$ and the second argument is a mesh where no additional diffeomorphism is applied. A small set of flags were implemented in order to run various tests without having to recompile since compilation is really demanding in terms of system requirements.

See the following command :

```
1  BEM optimization for coupled eddy current simulation.
2
3  options:
4    --validate_shape_deriv [OP] [DELTA]           =
5                      run validation for shape derivative operator.
6                      [OP]   = B0 | A0_edge | N0_lagr
7    --calculate_target [c0][c1][c2][c3][s1][s2]    =
8                      compute and save a solution for a speciffic set of ↩
                           parameters.
9                      [cj] - cosines, [sj] - sines. Need to provide exactly 6.
10   --optimization_options [dom] [ofstring] [grad_rep] [max_ls_steps] =
11                     perform optimization run. Possible parameters:
12                     [dom]           : both-domains|probe
13                     [ofstring]      : phi|alpha|both
14                     [grad_rep]      : L2|H1|H1_2
15                     [max_ls_steps]  : 3,4...
```

```
16  --continue [dom] [ofstring] [grad_rep] [max_ls_steps] [c0][c1][c2][c3][s1↩
        ][s2]  =
17                      Same as the previous option, but with the choice of ↩
                            defining an initial design vector.
18
19  examples:
20
21
22    ./betl2_deformed_torus_operators msh0.45 asquare_coarse --↩
          calculate_target 0 0.3 -0.3 0.1 0 0.2
23    ./betl2_deformed_torus_operators inp_parametric inp_nonparametric --↩
          optimization_options both-domains alpha L2 10
```

In order to modify the set of basis functions it is necessary to edit the executable on the setting of the `directions` vector of functors. More flexibility was not deemed necessary since the natural track of this project is to achieve BEM optimization with surface Lagrange basis functions.

Chapter 7

Conclusion and future work

7.0.1 Conclusion

An optimization technique was presented and implemented for the **E** formulation of the BEM coupled eddy current problem, using analytical shape derivatives and the adjoint method. A parametrization of the shape was considered in a way that the shape derivative formulas can be derived and computed in a straightforward manner. The results of the performance of the method are satisfactory. The shape derivatives of the operators presented are not limited to the eddy current model. The same analytical shape derivatives can be used to compute gradients for other optimization problems with BEM.

7.0.2 Outlook

On BETL2 development

It is apparent that there are limitations that have not been dealt with and they can be covered in future projects. In the author's opinion future development in BEM with BETL2 should be performed straight away with approximation techniques, at least for the integration. Full integration for BEM operators is operation intensive on a scale that not

only it obscures the real power of BEM, but also it hinders debugging and development.

As discussed in subsection 6.5.3 a general discretization of the control space should be implemented and the relevant template classes should be extended to support it or a different set of classes should be put in place for that. The second option seems more viable and flexible but also probably less maintainable.

As for the coupled eddy current problem future works might investigate more general loading than non-local current excitation and the **H** based formulation. Of course the present work also paves the way for optimizing the shape of a coil for optimal inductive hardening of components of critical importance for energy saving and performance.

Bibliography

[1] Chavent G. Identification of function parameters in partial differential equations. *Identification of parameter distributed systems*, 1974.

[2] Ralf Hiptmair. Boundary element methods for eddy current computation. In Martin Schanz and Olaf Steinbach, editors, *Boundary Element Analysis*, volume 29 of *Lecture Notes in Applied and Computational Mechanics*, pages 213–248. Springer Berlin Heidelberg, 2007.

[3] L. Kielhorn. Implementing the bem-bem coupling for linear eddy currents ... in a nutshell. *BETL2 Documentation notes*, 2014.

[4] R.-E. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal*, 167(2):495–503, 2006.

[5] P.Meuri. *Stable Finite Element Boundary Element Galerkin Schemes for Acoustic and Electromagnetic Scattering.* PhD thesis, ETH Zürich, 2007.

[6] L. Kielhorn R. Hiptmair. A generic boundary element template library. *Technical Report, ETH Zürich*, page 36, 2012.

Appendix A

Validation of Mapping and Shape Derivatives

A.0.3 Boundary Integral Operators

The validation was performed by directly comparing the discrete version of the operators computed on the parametric plane and on an actual 3D mesh as it is represented from a speciffic control function

$$q = cos(2\alpha) + sin(3\varphi). \tag{A.1}$$

In order to circumvent a possible interpolation step the three dimensional geometry where the reference operators are computed is constructed by mapping the nodes from the two dimensional plane to the point they correspond to on the 3D space.

This approach introduced an interesting inconsistency for the validation of the matrix for the double layer potential. We keep in mind that the points are mapped from the parameter plane to the surface of a torus that possesses curvature and continuous normal vector. A plane triangle parametrization of the torus introduces discontinuities on

3-Noded Mesh-h	#elements	$err(DL)$	$err(SLv)$	$err(SLs)$
0.40	512	13.07%	0.86%	2.089%
0.36	648	12.56%	0.70%	1.731%
0.25	1352	11.25%	0.36%	1.008%
6-Noded Mesh-h	#elements	$err(DL)$	$err(SLv)$	$err(SLs)$
0.40	512	0.21%	0.146%	0.04%
0.36	648	0.17%	0.115%	0.05%
0.25	1352	0.10%	0.054%	0.02%

Table A.1: Validation of BEM operators

the normals along the elements and this had a signifficant impact on the validation. The validation passes with reasonably good results when curved elements were used. That effect was indistinguisable on the single-layer vector potential calculation, very limited in the single-layer scalar operator but very pronounced on the double layer potential that depends on the normal of the boundary elements (or equivalently on the cross product of surface vectors). We denote matrices computed on the parameter domain as M^P_{pot} and matrices computed directly on the 3D torus as M^T_{pot}. The subscript *pot* is $\{SLs, SLv, DL\}$ denoting the single layer scalar, single layer vectorial and double layer operators respectively. The results of the validation are given in the following table. $err(\cdot)$ denotes the norm $\frac{||M^P_\cdot - M^T_\cdot||}{||M^P_\cdot + M^T_\cdot||}$ The quadrature points are 12 for all different cases of integration (edge adjacent, vertex adjacent and regular).

A.0.4 Shape Derivatives of Operators

In order to validate the numerically computed shape derivatives from the analytical formulas a central difference of the matrices was calculated for a deformation δq along the

3-Noded Mesh-h	#elements	$err(DL)$	$err(SLv)$	$err(SLs)$
0.40	512	7.60E-6%	3.81E-6%	1.60E-6%
0.36	648	9.30E-6%	3.88E-6%	1.70E-6%
0.25	1352	1.64E-6%	4.09E-6%	1.90E-6%
6-Noded Mesh-h	#elements	$err(DL)$	$err(SLv)$	$err(SLs)$
0.40	512	7.50E-6%	3.81E-6%	1.60E-6%
0.36	648	9.20E-6%	3.88E-6%	1.70E-6%
0.25	1352	1.63E-6%	4.08E-6%	1.90E-6%

Table A.2: Validation of shape derivatives

normal vector of the undeformed torus surface.

$$\langle \frac{\partial \mathcal{K}}{\partial q}, \delta q \rangle := \lim_{\epsilon \to 0} \frac{\mathcal{K}(q + 1/2\,\epsilon\,\delta q) - \mathcal{K}(q - 1/2\,\epsilon\,\delta q)}{\epsilon} \tag{A.2}$$

The analytical formulas for the shape derivatives show excellent agreement with the formulas calculated by finite differences. However it should be noted that the finite difference approximation is particularly sensitive to the choice of ϵ and this approach breaks down for $\epsilon \to 0$ possibly due to quantities that occur during the evaluation that cannot be described with machine precision. The following results were acquired for a deformation along the direction A.1 and $\epsilon = 10^{-8} r$ where r is the small radius of the torus. It is worth noting that although the validation of the actual bilinear operators (not their shape derivatives) is not giving encouraging results for 3-noded triangular elements this trend is not followed from the shape derivatives of the bilinear operators.

The validation gives very good results for all the shape derivatives. There is a striking agreement of the shape derivatives regardless of the geometric order of the elements used. There is a slight trend of deterioration of the results with $h-$refinement but the order of the errors along with the trend being quite insignifficant can be attributed to the

numerical accuracy of the computations. We should take into account that pure numerical approximation innacuracy is expected to accumulate in larger computations. In light of these results one can argue that the shape derivative in our case seems to follow the order of approximation of the corresponding bilinear form [1].

[1]However, it must be noted that we compute on a smooth domain and we take account smooth variations of the domain. This argument might not generalize with sharp variations δq.