JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

An Application of Isogeometric Shape Optimization in Magnetostatics

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplomingenieurin

im Masterstudium

Industriemathematik

Eingereicht von:
Katharina Rafetseder

Angefertigt am:
Institut für Numerische Mathematik

Beurteilung:
A. Univ.-Prof. Dipl.-Ing. Dr. Walter Zulehner

Linz, September 2014

Abstract

Shape optimization is a widely used approach for improving the shape of the boundary of a structure in various disciplines. Classical applications originate from structural mechanics, but this techniques have also been effectively applied to problems in electrical engineering. The starting point for our consideration is an application in the field of electrical engineering, which consists of finding the optimal shape of electromagnets such that the generated magnetic field in a certain part is as homogeneous as possible in a given direction. We model this physical application in form of a shape optimization problem.

In this thesis we follow the concept "first discretize, then optimize", meaning that we first discretize the problem and perform the optimization on the discrete level. In shape optimization the common approach is to use Bézier curves, B-splines or non-uniform rational B-splines (NURBS) for the geometric description of the boundary. The idea of isogeometric analysis (IGA) enables us to use NURBS (or B-splines) also for the analysis, i.e., for computing the response of the structure for a given design. Therefore, we investigate in this thesis the concept of isogeometric analysis, which makes it possible to easily modify our design by adjusting the control points that determine the geometry. This technique allows us to take advantage of the exact and smooth geometry representation via NURBS (or B-splines) in the shape optimization process. In the implementation we use for the geometry description the G+SMO (Geometry + Simulation Modules) library, which is developed at the JKU in Linz.

For the optimization we employ for comparability reasons the MATLAB routine `fminunc` with a gradient-based quasi-Newton algorithm (BFGS method). For this reason, we give an overview of approaches used to perform a sensitivity analysis and derive in detail an analytical design sensitivity of the discrete problem in case of a B-spline discretization. Finally, numerical results for different settings, e.g., using an analytically computed gradient or approximate the gradient with finite differences, various initial shapes and different number of design variables, are presented and compared with each other.

Zusammenfassung

Formoptimierung ist ein in verschiedenen Disziplinen weit verbreiteter Zugang zur Verbesserung der Form des Randes eines Gebiets. Klassische Anwendungen sind im Bereich der Strukturmechanik zu finden, aber diese Techniken wurden auch erfolgreich auf Probleme aus der Elektrotechnik angewandt. Der Ausgangspunkt für unsere Überlegungen ist eine Anwendung aus der Elektrotechnik, welche darin besteht das optimale Design der Elektromagneten zu finden, sodass das erzeugte Magnetfeld in einem bestimmten Bereich so homogen als möglich in eine vorgegebene Richtung ist. Wir werden dieses praktische Problem mittels eines Formoptimierungsproblems modellieren.

In dieser Arbeit folgen wir dem Konzept „zuerst diskretisieren, dann optimieren". Mit anderen Worten wir diskretisieren zuerst das Problem und führen die Optimierung am diskreten Level durch. In der Formoptimierung ist es gängig Bézier Kurven, B-splines oder non-uniform rational B-splines (NURBS) zur geometrischen Beschreibung des Randes zu verwenden. Die Idee der Isogeometrischen Analyse (IGA) ermöglicht es uns NURBS (oder B-splines) auch für die Analyse, d.h. zur Berechnung der Antwort der Struktur für ein vorgegebenes Design, zu verwenden. Aus diesem Grund werden wir in dieser Arbeit das Konzept der Isogeometrischen Analyse einsetzten, welches uns eine einfache Möglichkeit bietet das Design zu modifizieren, indem wir die Kontrollpunkte verschieben, welche die Geometrie festlegen. Weiters können wir damit die exakte und glatte Geometrie Repräsentierung mittels NURBS in den Optimierungsprozess integrieren. In der Implementierung verwenden wir für die Geometrie Beschreibung die G+SMO (Geometry + Simulation Modules) Bibliothek, welche an der JKU in Linz entwickelt wird.

Zur besseren Vergleichbarkeit verwenden wir für die Optimierung die MATLAB Routine `fminunc` mit einem Gradienten basierten quasi-Newton Algorithmus (BFGS Methode). Aus diesem Grund geben wir einen Überblick zu verschiedenen Techniken, welche zur Durchführung einer Sensitivitätsanalyse verwendet werden und leiten im Detail die analytische Designsensitivität des diskreten Problems für den Fall einer B-spline Diskretisierung her. Abschließend präsentieren und vergleichen wir numerische Resultate für verschiedene Einstellungen, z.B. Verwendung des analytisch berechneten Gradienten oder Approximation des Gradienten mittels finiter Differenzen, unterschiedliche Ausgangsdesigns und verschiedene Anzahl von Design Variablen.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Walter Zulehner for his guidance and support during the preparation of my master thesis and for organising financial support. Especially, I am grateful for the numerous time-consuming discussions, even at short notice.

This work was supported by the Austrian Science Fund (FWF) under the grant S11702-N23 (NFN Project 2).

Special thanks go to Dr. Clemens Hofreither and Dr. Angelos Mantzaflaris for their friendly assistance that has been really helpful for getting started with the G+SMO (Geometry + Simulation Modules) library.

Moreover, particular thanks are due to my parents and my family, especially, my grandmother, for their support throughout my studies. Finally, I would like to thank my boyfriend and colleague Christoph Hofer for all his aid, various interesting discussions and the proof-reading.

Katharina Rafetseder
Linz, September 2014

Contents

Chapter 1

Introduction

In *structural optimization* we aim at finding the structure that performs a certain task in the best possible way. In mathematical terms, the performance measure is given by a function, called cost or objective function. Classical examples for cost functions in structural optimization are, e.g., weight, stiffness, compliance and material cost of the considered structure. Structural optimization has been originally applied to problems in the field of structural mechanics. This means mechanical structures whose major task is to sustain loads, like a bridge or a cantilever beam, have been considered. However, this concept has been effectively used to treat problems from various other disciplines, e.g., electrical engineering, as we will do in this thesis.

Structural optimization problems are divided into the following three main classes: sizing optimization, shape optimization and topology optimization. Typically, *sizing optimization* is used to optimize truss structures. In this case the design variable that describes the design is the cross-section area of the bars in the truss. In *shape optimization* the design variable is given by the whole or a part of the boundary. *Topology optimization* is the most general class of structural optimization, where also the topology of the structure can be modified. In case of a truss structure this would mean that bars can also be removed from the truss.

The starting point for our considerations is an application in the field of electrical engineering. The goal is to design the form of the electromagnets such that the generated magnetic field in a certain part of the domain is as homogeneous as possible in a given direction. We will model this physical application by means of a shape optimization problem. In this thesis we apply the approach "first discretize, then optimize". In other words, we first discetize our problem and perform the optimization on the discrete level, meaning our design variable and state variable (describing the reply of the structure) are finite dimensional vectors rather than some functions.

At the beginning of shape optimization the design variables were defined by boundary nodes of the finite element discretization. However, this approach leads to wiggly unrealistic shapes as optimized designs, as you can see in Figure 1.1.

Instead, the common approach is to distinguish between a design model consisting of a (coarse) geometry description and an analysis model with a different geometric

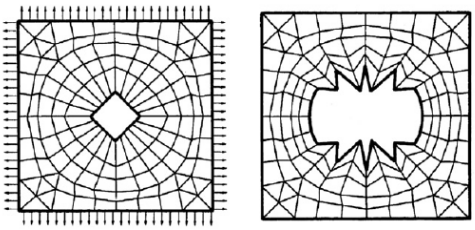


Figure 1.1: Initial design (left) and optimized shape (right) of a hole in a plate such that the weight is minimized subject to a constraint on the stress at certain points. Figures are taken from [2].

description to compute the solution of the state problem. Note that the solution of the state problem represents the response of the structure for a given design, e.g., displacement, stress or force. For the geometric description Bézier curves, B-splines and non-uniform rational B-splines (NURBS) are widely used (see e.g. [2] in combination with the design element technique). There are several reasons for this choice. First, with a small number of control points as design variables a comprehensive set of shapes can be represented. Moreover, by means of splines we gain a smooth and regular boundary representation. Note that in most industrial applications the description of the geometry is given via a computed aided design (CAD) system that is based on NURBS. If we employ classical finite elements for the analysis, finite element meshes have to be generated from the CAD data. In this context, note that the necessary data exchange between the systems during the design process is a considerable drawback.

The concept of isogeometric analysis (IGA), introduced in [9], enables us to use NURBS (or B-splines) not only for the geometry description but also as basis for the analysis, i.e., for solving the state problem. In other words, by means of this approach we can merge design and analysis model. Keep in mind, this idea allows us to take advantage of the exact and smooth geometry representation via NURBS (or B-splines) in the shape optimization process. Finally, note that using IGA makes the communication with a finite element mesh generator unnecessary.

The remainder of this thesis is organized as follows:

In Chapter 2 we introduce in detail the physical application of this thesis, which consists of finding the optimal design of electromagnets. In order to apply mathematical techniques, we have to formulate a mathematical model for our considered application. For this purpose, we first take a closer look at the general framework of a structural optimization problem and its elements and derive the 2D magnetostatic formulation, which acts as the governing state problem in our application. Finally, we are able to establish a mathematical model of our physical problem in form of a shape optimization problem.

In Chapter 3 we present the discretization technique we are going to apply, the concept of isogeometric analysis (IGA). For this end, we first introduce the needed geometrical background, i.e., we define B-spline basis functions and B-spline geometries. The next step is to find a geometry description of our considered geometry of

electromagnets and give a definition of the design variables. Moreover, we derive a discrete variational formulation of the state equation, where we incorporate the Dirichlet boundary conditions in a weak sense by means of Nitsche's method, and show existence and uniqueness of the solution.

Chapter 4 focuses on the numerical method used for the optimization, the MATLAB routine `fminunc` with a quasi-Newton algorithm (BFGS method). Since quasi-Newton methods need gradient information of the objective function, we give an overview of approaches used to perform a sensitivity analysis and derive in more detail an analytical sensitivity of the already discretized problem.

Chapter 5 presents and discusses the numerical results we received for the considered physical problem.

Concluding, in Chapter 6 we give an overview of the work realized in this thesis and suggest possible next steps.

Chapter 2

Problem Formulation

This chapter is dedicated to the mathematical formulation of the considered physical problem. In the first section the physical problem is introduced. Section 2.2 provides the mathematical framework of a general structural optimization problem. In Section 2.3 the two-dimensional magnetostatic formulation is derived, which will act as governing state problem in our application. Finally, in Section 2.4 we are able to formulate a mathematical model in form of a shape optimization problem that describes our physical application.

2.1 Physical Problem

The starting point for our considerations is a shape optimization problem in the field of electrical engineering, which is taken from the PhD thesis "Optimal Shape Design in Magnetostatics" by D. Lukáš [13]. Keep in mind, Lukáš uses classical finite elements (with Bézier curves as boundary representation), and in the present work we take advantage of isogeometric analysis.

The considered geometry of electromagnets, called *Maltese Cross* geometry, is displayed in Figure 2.1. It is composed of a ferromagnetic yoke and four poles with coils, which are pumped with direct electric current. This geometry is used for measurements of the so-called *magneto-optic Kerr effect*. This effect describes the change in polarization and intensity of light when reflected from a magnetized surface. For the measurements a magnetic material is positioned in the magnetization area Ω_m in the middle, see Figure 2.2. For a detailed description of the experiment we refer to [13]. Since the results depend considerably on the orientation of the sample, the tests have to be carried out for different directions of the magnetic field. With the above described geometry, *homogeneous*, which means constant, magnetic fields in eight (four vertical and four diagonal) directions can be produced by switching on and off the current in particular coils or change its flow direction. The important aspect for the further considerations is that the measurements require the magnetic field in the magnetization area as homogeneous as possible in the respective direction. As a practical application of the magneto-optic Kerr effect one should have in mind high capacity

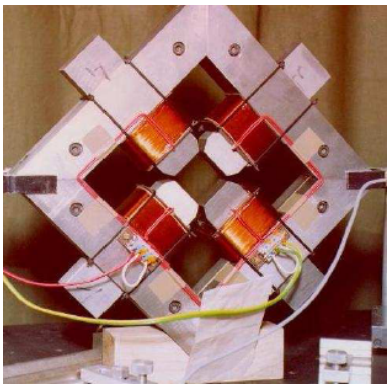


Figure 2.1: The Maltese Cross geometry of electromagnets. Photo is taken from [13].

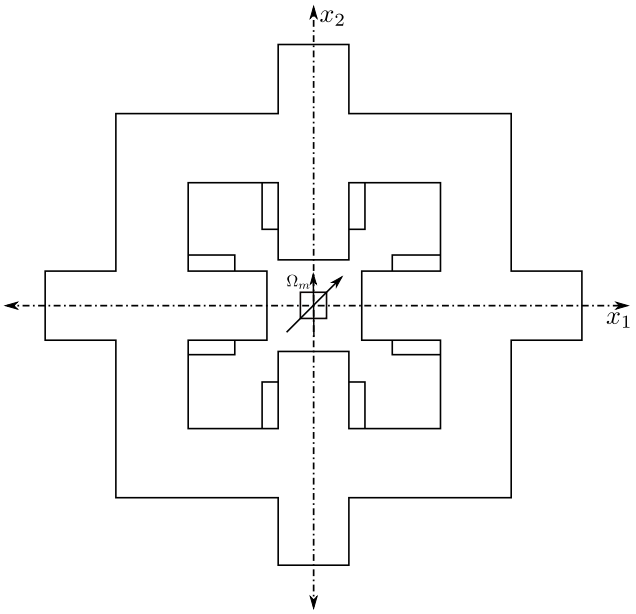


Figure 2.2: Cross section of the Maltese Cross geometry of electromagnets

data storage media like magnetic or compact discs.

Summing up, our goal is to design the shapes of the pole heads such that in the magnetization area the inhomogeneity of the eight magnetic fields in the inherent direction is minimized. On the other hand, the magnitude of the magnetic fields should be high enough to make the magneto-optic Kerr effect possible.

2.2 Structural Optimization Problem

In this section first a general mathematical form of a continuum structural optimization problem is introduced. Since in this thesis we apply the approach "first discretize, then optimize", the next step is to perform a discretization and obtain a discrete problem. This section is mainly based on [5] and [4].

A general continuum structural optimization problem reads

$$\min_{p,u} \ \mathcal{I}(p,u) \tag{2.1a}$$
$$s.t. \ u \text{ solves PDE} \tag{2.1b}$$
$$g(p,u) \leq 0, \tag{2.1c}$$

where $\mathcal{I}$ is called *cost function* or *objective function*, p *design variable* and u *state variable*.

- The cost function $\mathcal{I}$ is a measure of the quality of the design, and usually $\mathcal{I}$ is defined such that we aim at minimizing it.
- The design variable p is a function that defines the design. During the optimization this function will be modified.
- The state variable u is a function that stands for the reply of the partial differential equation for a given design p.

The partial differential equation (2.1b), called *state problem*, couples the design and state variable. The inequality (2.1c) represents the *design* and *state constraints*. The corresponding discrete problem is given by

$$\min_{\boldsymbol{p}\in\mathbb{R}^n, \boldsymbol{u}\in\mathbb{R}^m} \ \mathcal{I}_h(\boldsymbol{p},\boldsymbol{u}) \tag{2.2a}$$
$$s.t. \ \boldsymbol{K}(\boldsymbol{p})\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{p}) \tag{2.2b}$$
$$g_h(\boldsymbol{p},\boldsymbol{u}) \leq 0, \tag{2.2c}$$

where $\boldsymbol{p}$ is the vector of design variables, $\boldsymbol{u}$ the vector of state variables and $\mathcal{I}_h$, g_h are the discretized cost function and constraint function, respectively. The state equation (2.2b) is given by the Galerkin approximation of the partial differential equation in (2.1b). The way the problem is stated in (2.2) is called *simultaneous formulation* (see [5]), since the state problem is solved simultaneously with the optimization.

In formulation (2.2) the design and state variables are considered as independent quantities. However, quite often the state variable $\boldsymbol{u}$ is already uniquely determined by the state problem for given design variables. As we see later, this is also the case in our situation since the stiffness matrix $\boldsymbol{K}$ is invertible, hence we obtain

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{p}) = \boldsymbol{K}^{-1}(\boldsymbol{p})\boldsymbol{f}(\boldsymbol{p}).$$

If we plug in the function $\boldsymbol{u}(\boldsymbol{p})$ for the design variable, we end up with the *nested formulation* (cf. [5]):

$$\min_{\boldsymbol{p}\in\mathbb{R}^n} \ \mathcal{I}_h(\boldsymbol{p},\boldsymbol{u}(\boldsymbol{p})) \tag{2.3a}$$
$$s.t. \ g_h(\boldsymbol{p},\boldsymbol{u}(\boldsymbol{p})) \leq 0. \tag{2.3b}$$

Later, in Chapter 4 we discuss the numerical treatment of problems of this form.

2.3 Mathematical Model for Magnetostatics

The aim of this section is to deduce the 2D linear magnetostatic formulation. The starting point are Maxwell's equations, the next step is the magnetostatic vector potential formulation and finally we will perform a reduction to 2D. The derivation is primarily based on [18], [17] and [6].

Maxwell's equations (2.4) describe the phenomena of electromagnetism, for a thorough derivation see, e.g., [11]. Maxwell's equations in differential form are given by

$$\operatorname{curl} \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}, \tag{2.4a}$$

$$\operatorname{curl} \boldsymbol{H} = \boldsymbol{J} + \frac{\partial \boldsymbol{D}}{\partial t}, \tag{2.4b}$$

$$\operatorname{div} \boldsymbol{D} = \rho, \tag{2.4c}$$

$$\operatorname{div} \boldsymbol{B} = 0. \tag{2.4d}$$

The appearing field quantities with corresponding SI units are

the *electric field intensity*	$\boldsymbol{E}$ $[V/m]$,
the *electric flux density (displacement current density)*	$\boldsymbol{D}$ $[As/m^2]$,
the *magnetic flux density (magnetic induction)*	$\boldsymbol{B}$ $[Vs/m^2 = T]$,
the *magnetic field intensity*	$\boldsymbol{H}$ $[A/m]$,
the *electric current density*	$\boldsymbol{J}$ $[A/m^2]$,
the *charge density*	ρ $[As/m^3]$.

Note, the defined quantities depend on the spatial variable $\boldsymbol{x} = (x_1, x_2, x_3)^T$ and the time variable t. The letters set in boldface represent three-dimensional vector fields. According to Maxwell's equations (2.4) we obtain 6 equations for 15 unknowns, since by applying the curl operator information gets lost because it has a null space consisting of the gradient fields. Hence, in order to receive a closed system we have to pose 9 additional material laws. The following material laws provide relations between the electromagnetic fields:

$$\boldsymbol{B} = \mu \boldsymbol{H} + \boldsymbol{M}, \tag{2.5a}$$

$$\boldsymbol{D} = \epsilon \boldsymbol{E} + \boldsymbol{P}, \tag{2.5b}$$

$$\boldsymbol{J} = \sigma(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) + \boldsymbol{J}_i, \tag{2.5c}$$

containing

the *magnetization (remanent flux density)*	$\boldsymbol{M}$ $[Vs/m^2 = T]$,
the *polarization*	$\boldsymbol{P}$ $[As/m^2]$,
the *magnetic permeability*	μ $[Vs/Am = H/m]$,
the *electric permittivity*	ϵ $[As/Vm]$,
the *electric conductivity*	σ $[A/Vm]$,
the *velocity*	$\boldsymbol{v}$ $[m/s]$,
the *impressed current density*	$\boldsymbol{J}_i$ $[A/m^2]$.

For simplicity, we do not take the magnetization and the polarization into account and, therefore, set $\boldsymbol{M} = \boldsymbol{P} = 0$. Moreover, we assume $\boldsymbol{v} = 0$ and μ, ϵ and σ are scalar quantities that only depend on $\boldsymbol{x}$. This means we neglect the effects of hysteresis and consider only static (μ, ϵ and σ do not depend on t) isotropic linear materials. In case of a linear material the permeability μ does not depend on the magnetic field $\boldsymbol{H}$. In this context we should mention that ferromagnetic materials, like iron, are non-linear materials. However, for simplicity, we will apply this model to ferromagnetic materials in our later considerations. Therefore, we should keep in mind that by assuming linearity we induce a modelling error. Often, the electric current density is known in parts of the domain, which will be the case in our application. Then we additionally add the impressed current density in Ohm's law, as in (2.5c), and formally assume $\sigma = 0$ in this parts.

From now on we assume the appearing fields are time independent, i.e., we are in the static case where $\frac{\partial}{\partial t} = 0$. Then we receive $\boldsymbol{J} = \boldsymbol{J}_i$; therefore, the electric current density is in the following considered as known quantity. Under this assumption, we end up with the following equations, the magnetostatic formulation:

$$\operatorname{curl} \boldsymbol{H} = \boldsymbol{J}, \tag{2.6a}$$

$$\operatorname{div} \boldsymbol{B} = 0, \tag{2.6b}$$

$$\boldsymbol{B} = \mu \boldsymbol{H}. \tag{2.6c}$$

First, applying the operator div to equation (2.6a) implies a necessary condition for the electric current density

$$\operatorname{div} \boldsymbol{J} = 0. \tag{2.7}$$

Here we use that for a vector field $\boldsymbol{v}$ that is two times continuously differentiable $\operatorname{div} \operatorname{curl} \boldsymbol{v} = 0$ holds.

In the following let the computational domain $\Omega \subset \mathbb{R}^3$ be a bounded domain with a sufficiently smooth boundary $\Gamma = \partial\Omega$ composed of two disjoint parts Γ_B, Γ_H with $\overline{\Gamma}_B \cup \overline{\Gamma}_H = \Gamma$, and let $\boldsymbol{n}$ denote the outer unit normal on Γ. If we assume that the domain Ω is simply connected, equation (2.6b) guarantees the existence of a vector potential $\boldsymbol{A}$ with

$$\boldsymbol{B} = \operatorname{curl} \boldsymbol{A}.$$

This allows us to rewrite the magnetostatic system (2.6) to the so-called vector potential formulation, given by

$$\operatorname{curl}\Big(\frac{1}{\mu}\operatorname{curl}\boldsymbol{A}\Big) = \boldsymbol{J} \quad \text{in } \Omega, \tag{2.8a}$$

with possible boundary conditions

$$\boldsymbol{A}\times\boldsymbol{n} = 0 \quad \text{on } \Gamma_B, \tag{2.8b}$$

$$\frac{1}{\mu}\operatorname{curl}\boldsymbol{A}\times\boldsymbol{n} = 0 \quad \text{on } \Gamma_H. \tag{2.8c}$$

The physical interpretation of these boundary conditions is as follows: (2.8b) implies $\boldsymbol{B}\cdot\boldsymbol{n} = 0$, which is called *induction* boundary condition. The second condition (2.8c) and $\boldsymbol{H}\times\boldsymbol{n} = 0$, which is called *perfect magnetic conductor* (PMC) condition, are equivalent. Bear in mind, the solution $\boldsymbol{A}$ of (2.8a) is only unique up to adding a gradient field $\nabla\varphi$. One possibility to fix the arbitrary gradient field is to impose the additional condition $\operatorname{div}\boldsymbol{A} = 0$, called *Coulomb gauge*.

2.3.1 2D Reduction

The final step in this section is a two-dimensional reduction of the magnetostatic system stated in (2.8). Let us assume our domain $\Omega = \Omega_{2D}\times(-l,l)$ with $l \gg \operatorname{diam}(\Omega_{2D})$ is homogeneous in x_3-direction and

$$\boldsymbol{J} = \begin{pmatrix} 0 \\ 0 \\ J(x_1,x_2)\end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} H_1(x_1,x_2) \\ H_2(x_1,x_2) \\ 0\end{pmatrix}.$$

First, note the current density $\boldsymbol{J}$ as given above fulfils the necessary condition (2.7). Furthermore, we obtain by means of (2.6c)

$$\boldsymbol{B} = \begin{pmatrix} B_1(x_1,x_2) \\ B_2(x_1,x_2) \\ 0\end{pmatrix},$$

which implies

$$B_3 = [\operatorname{curl} A]_3 = \frac{\partial A_2}{\partial x_1} - \frac{\partial A_1}{\partial x_2} = 0.$$

This motivates the following ansatz for the vector potential:

$$\boldsymbol{A} = \begin{pmatrix} 0 \\ 0 \\ u(x_1,x_2)\end{pmatrix}, \tag{2.9}$$

which provides

$$\boldsymbol{B} = \operatorname{curl} \boldsymbol{A} = \begin{pmatrix} \frac{\partial u}{\partial x_2}(x_1, x_2) \\ -\frac{\partial u}{\partial x_1}(x_1, x_2) \\ 0 \end{pmatrix}. \tag{2.10}$$

Note, for the considered ansatz (2.9) the Coulomb gauge, as defined previously, is fulfilled. With this ansatz the vector potential formulation (2.8) reduces to

$$\operatorname{curl}\Big(\frac{1}{\mu} \operatorname{curl} \boldsymbol{A}\Big) = \operatorname{curl} \begin{pmatrix} \frac{1}{\mu}\frac{\partial u}{\partial x_2}(x_1, x_2) \\ -\frac{1}{\mu}\frac{\partial u}{\partial x_1}(x_1, x_2) \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\frac{\partial}{\partial x_1}\left(\frac{1}{\mu}\frac{\partial u}{\partial x_1}\right) - \frac{\partial}{\partial x_2}\left(\frac{1}{\mu}\frac{\partial u}{\partial x_2}\right) \end{pmatrix} = \boldsymbol{J},$$

and for the boundary conditions we receive

$$0 = \boldsymbol{A} \times \boldsymbol{n} = \begin{pmatrix} -u(x_1, x_2) n_2 \\ u(x_1, x_2) n_1 \\ 0 \end{pmatrix} \Leftrightarrow u(x_1, x_2) = 0$$

and

$$0 = \frac{1}{\mu} \operatorname{curl} A \times \boldsymbol{n} = \frac{1}{\mu} \begin{pmatrix} 0 \\ 0 \\ \frac{\partial u}{\partial x_2} n_2 + \frac{\partial u}{\partial x_1} n_1 \end{pmatrix},$$

where we use that $n_3 = 0$ on $\partial\Omega_{2D} \times (-l, l)$. Hence, we end up with the following 2D linear magnetostatic formulation:

$$-\operatorname{div}\Big(\frac{1}{\mu}\nabla u\Big) = J \quad \text{in } \Omega \subset \mathbb{R}^2, \tag{2.11a}$$

$$u = 0 \quad \text{on } \Gamma_B = \Gamma_D, \tag{2.11b}$$

$$\frac{1}{\mu}\frac{\partial u}{\partial \boldsymbol{n}} = 0 \quad \text{on } \Gamma_H = \Gamma_N, \tag{2.11c}$$

where $\Omega = \Omega_{2D}$, $\Gamma_B = \Gamma_B \cap \overline{\Omega}_{2D}$, $\Gamma_H = \Gamma_H \cap \overline{\Omega}_{2D}$, $\nabla = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2})^T$ denotes the two-dimensional gradient operator and $\frac{\partial u}{\partial \boldsymbol{n}}$ the normal derivative $\nabla u \cdot \boldsymbol{n}$.

2.4 Mathematical Model as Structural Shape Optimization Problem

In this section the elements of the general continuum structural optimization problem (2.1), stated in Section 2.2, are defined for our considered application, which has been introduced in Section 2.1.

In this thesis we restrict our considerations to a two-dimensional model of the cross-section of the Maltese Cross geometry. The computational domain $\Omega = (-0.2, 0.2) \times (-0.2, 0.2)$ completely contains the considered geometry of electromagnets, as displayed in Figure 2.3. The exact physical dimensions of the geometrical model can be found in [13, Ch. 7, p. 109].

2.4.1 Design Variable

In contrast to common practise in shape optimization, in our application the design variable does not describe the shape of a boundary part. Hence, the computational domain Ω does not change during the optimization. The design variable represents the forms of the pole heads or more precisely only the interior part, i.e., the boundary points on the left and right of the pole head are fixed.

In order to reduce the number of degrees of freedom of our model we make the following assumptions: First, we assume that the shapes of the four pole heads are the same; therefore, we will only optimize the shape of the south pole head and describe the shape of any other pole head by this form. Moreover, we assume the shape of the south pole head is symmetric with respect to the x_2-axis; hence, we only treat its half. Summing up, the design variable represents the shape of the half south pole head, which is emphasized in green in Figure 2.3.

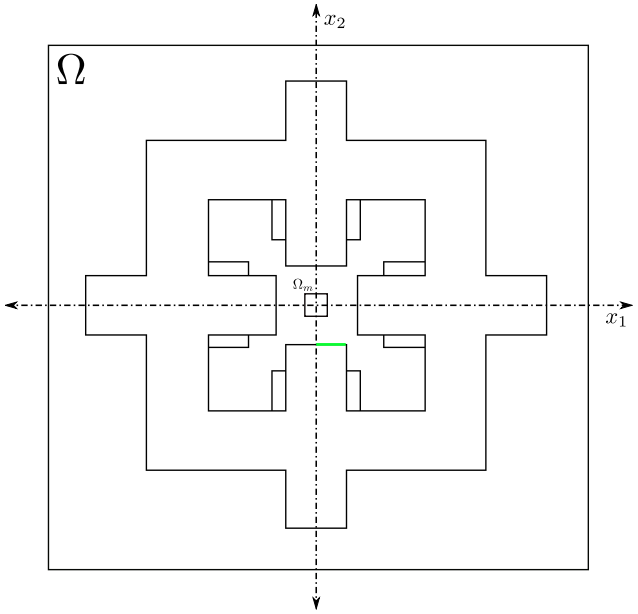


Figure 2.3: Cross section of the Maltese Cross geometry of electromagnets with computational domain Ω and design variable (emphasized in green)

2.4.2 State Problem

As mentioned above, we consider a two-dimensional setting of our problem, and the current does not change in time, i.e., we are in the static case. Therefore, the governing state equation is the 2D linear magnetostatic formulation (2.11) derived in Section 2.3.1. As already indicated, our geometry of electromagnets is completely contained in the computational domain Ω. Hence, we impose induction boundary conditions $\boldsymbol{B} \cdot \boldsymbol{n} = 0$ on the whole boundary or in terms of the vector potential $\boldsymbol{A} \times \boldsymbol{n} = 0$.

So we end up with the following state problem:

$$-\operatorname{div}\Big(\frac{1}{\mu(\boldsymbol{x})}\nabla u(\boldsymbol{x})\Big) = J(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} = (x_1, x_2)^T \in \Omega \subset \mathbb{R}^2, \tag{2.12a}$$

$$u(\boldsymbol{x}) = 0 \qquad \text{for } \boldsymbol{x} \in \Gamma, \tag{2.12b}$$

where the magnetic flux density $\boldsymbol{B}$ is given by $\boldsymbol{B}(\boldsymbol{x}) = (\frac{\partial u}{\partial x_2}(\boldsymbol{x}), -\frac{\partial u}{\partial x_1}(\boldsymbol{x}), 0)^T$.

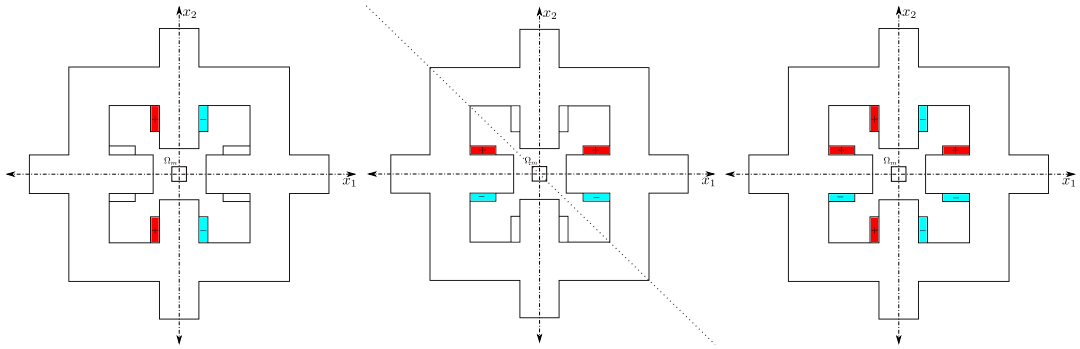
Pay attention to the fact that the shape of the pole head, the design variable, controls the material partition of the domain. The pole head form divides the whole domain into

- Ω_0, the domain composed of coils and air, with permeability $\mu_0 = 4\pi 10^{-7}$ and
- Ω_1, the ferromagnetic parts (yoke and poles), with permeability $\mu_1 = 5100\mu_0$.

According to our goal stated in Section 2.1, we have to consider different directions of the magnetic field, which are generated by changing the current excitation of the coils. Since we assume equality of all pole head shapes, it is sufficient to consider one vertical and one diagonal magnetic field. Clearly, the current density J, which reduces to a scalar in the two-dimensional case, is only non-zero in the domains occupied by the coils. For both current excitation scenarios the modulus of the current density is given by

$$|J(\boldsymbol{x})| = \frac{n_I I}{S_c},$$

where the electric current $I = 5\ [A]$, the winding number $n_I = 500$ and the cross-section area of the coils $S_c = 3 \cdot 10^{-4}\ [m^2]$. We still have to prescribe the sense of J. For the vertical magnetic field we apply the following current excitation depicted in Figure 2.4a, later denoted by *vertical current excitation*. In case of the diagonal magnetic field the used current excitation, called *diagonal current excitation*, is as depicted in Figure 2.4c.



(a) Vertical current excitation (b) Reflected vertical current excitation (c) Diagonal current excitation

Figure 2.4: Current excitations

Note, in case of the vertical current excitation the right hand side J is symmetric w.r.t. the x_1-axis and antisymmetric w.r.t. the x_2-axis. By exploiting the just established symmetry of the right hand side and the symmetry of our geometry, it is sufficient to solve the equation on one quarter of the domain Ω, e.g., on the lower right (fourth quarter). Bear in mind, we have to impose symmetry boundary conditions ($u = 0$) and antisymmetry boundary conditions ($\frac{\partial u}{\partial \boldsymbol{n}} = 0$) on the corresponding interior boundary parts, see Figure 2.5. If we denote by u the solution on the fourth quarter, symmetry considerations for our boundary value problem (2.12) yield the following:

- The solution in the first quarter $u_1(x_1, x_2) = u(x_1, -x_2)$.
- The solution in the second quarter $u_2(x_1, x_2) = -u(-x_1, -x_2)$.
- The solution in the third quarter $u_3(x_1, x_2) = -u(-x_1, x_2)$.

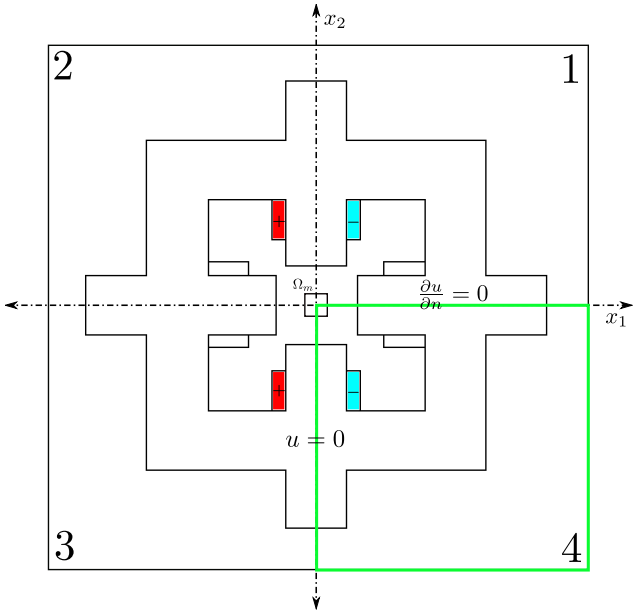


Figure 2.5: Reduced computational domain with respective symmetry and antisymmetry boundary conditions

Regarding the diagonal current excitation case, the first idea would be to consider a so-called multi-state problem and solve the corresponding state problem for both cases. However, there is a possibility to avoid solving two state problems by using a superposition argument. In order to do this, one has to realize that the diagonal current excitation can be represented as summation of the already considered vertical one (Figure 2.4a) and a second one depicted in Figure 2.4b. Observe, the second scenario is the reflection of the vertical current excitation w.r.t. to the diagonal $x_1 = -x_2$. Since we consider a linear differential equation with homogeneous boundary conditions, we

obtain by using superposition for the solution of the diagonal case

$$u_{diag} = u_{vert} + u_{vert,ref},$$

where u_{vert} is the solution of the vertical case and $u_{vert,ref}$ is the corresponding solution for the reflected right hand side. Note, $u_{vert,ref}$ can be obtained by reflecting the vertical solution along the diagonal $x_1 = -x_2$. Moreover, the same symmetry considerations as above can be applied to obtain the reflected solution on the whole domain from the solution on the fourth quarter with the difference that symmetry and antisymmetry axis are interchanged.

In summation, we only solve one state problem, namely, for the vertical excitation scenario, on the fourth quarter of the domain, as illustrated in figure Figure 2.5. By means of symmetry we obtain the solution on the whole domain; moreover, by using reflection and superposition we receive a solution for both current excitation cases.

2.4.3 Cost Function

Finally, we have to define the cost function. Recall that the first part of our goal, as stated in Section 2.1, is to minimize the inhomogeneity of the magnetic field in the respective direction in the magnetization area Ω_m. In order to achieve this, we define a function φ that measures the deviation of the magnetic flux density $\boldsymbol{B}$ from its average value in the L^2-norm. Hence, we set

$$\varphi(\boldsymbol{B}) = \frac{1}{\mathrm{meas}(\Omega_m)(B_{min}^{avg})^2} \int_{\Omega_m} |\boldsymbol{B}(\boldsymbol{x}) - B^{avg}(\boldsymbol{B})\boldsymbol{n}_m|^2 \,\mathrm{d}\boldsymbol{x}, \tag{2.13}$$

with

$$B^{avg}(\boldsymbol{B}) = \frac{1}{\mathrm{meas}(\Omega_m)} \int_{\Omega_m} \boldsymbol{B}(\boldsymbol{x}) \cdot \boldsymbol{n}_m \,\mathrm{d}\boldsymbol{x}, \tag{2.14}$$

where $\boldsymbol{B}(\boldsymbol{x}) = (\frac{\partial u}{\partial x_2}(\boldsymbol{x}), -\frac{\partial u}{\partial x_1}(\boldsymbol{x}))$ and $\boldsymbol{n}_m$ is the respective direction. The second part of our goal is to guarantee a given minimal magnitude, denoted by B_{min}^{avg}, of the magnetic field. We will impose this condition by adding a penalty term of the form

$$\theta(\boldsymbol{B}) = \Big[\max(0, B_{min}^{avg} - B^{avg}(\boldsymbol{B}))\Big]^2, \tag{2.15}$$

and end up with the following cost function:

$$\mathcal{I}(\boldsymbol{B}) = \varphi(\boldsymbol{B}) + \rho\theta(\boldsymbol{B}), \tag{2.16}$$

with a penalty parameter $\rho = 10^6$.

However, we aim at minimizing the inhomogeneity for both current excitation cases. Therefore, we have to minimize two objective functions. A problem of this form is

called *multiple criteria* optimization problem. In this context, it is important to note that the two cost functions are in general not minimized for the same design and state variable (p, u). The common approach is to look for a so-called *Pareto optimal* design, see [5] and [8].

Definition 2.1. *A design is called* Pareto optimal *if for any other design, either at least one of the objective function values becomes worse or all objective function values stay the same (cf. [8]).*

A widely used procedure to receive a Pareto optimal design is to build a scalar objective function by summing up the individual objective functions with some weighting coefficients. By applying this technique, we define

$$\mathcal{I}(\boldsymbol{B}^1, \boldsymbol{B}^2) = \frac{1}{2} \sum_{v=1}^{2} \Big[\varphi^v(\boldsymbol{B}^v) + \rho \theta^v(\boldsymbol{B}^v) \Big], \tag{2.17a}$$

with

$$\varphi^v(\boldsymbol{B}^v) = \frac{1}{\mathrm{meas}(\Omega_m)(B_{min}^{avg,v})^2} \int_{\Omega_m} |\boldsymbol{B}^v(\boldsymbol{x}) - B^{avg,v}(\boldsymbol{B}^v)\boldsymbol{n}_m^v|^2 \,\mathrm{d}\boldsymbol{x}, \tag{2.17b}$$

$$B^{avg,v}(\boldsymbol{B}^v) = \frac{1}{\mathrm{meas}(\Omega_m)} \int_{\Omega_m} \boldsymbol{B}^v(\boldsymbol{x}) \cdot \boldsymbol{n}_m^v \,\mathrm{d}\boldsymbol{x}, \tag{2.17c}$$

$$\theta^v(\boldsymbol{B}^v) = \Big[\max(0, B_{min}^{avg,v} - B^{avg,v}(\boldsymbol{B}^v)) \Big]^2. \tag{2.17d}$$

Here, $v = 1$ corresponds to the vertical current excitation and $v = 2$ to the diagonal current excitation. For completeness of the definition, the considered directions are given by

$$\boldsymbol{n}_m^v = \begin{cases} (0,1) & v = 1, \\ (1/\sqrt{2}, 1/\sqrt{2}) & v = 2, \end{cases}$$

and for the minimal average magnetic flux densities we choose

$$B_{min}^{avg,v} = \begin{cases} 0.085\ [T] & v = 1, \\ 0.12\ [T] & v = 2. \end{cases}$$

Remark 2.2. *Note, one can easily restrict the admissible shapes by means of design constraints of the form $g(p, u) \leq 0$. In this thesis, for simplicity, we do not incorporate design and state constraints in our model.*

Remark 2.3. *This final remark is related to the symmetry consideration we applied to reduce our computational domain to a quarter. According to the definition of φ (2.13) and the definition of the average magnetic flux density (2.14) we have to integrate the magnetic flux density $\boldsymbol{B}(\boldsymbol{x}) = (\frac{\partial u}{\partial x_2}(\boldsymbol{x}), -\frac{\partial u}{\partial x_1}(\boldsymbol{x}))$ over all four quarters of the*

magnetization area. Let us consider exemplarily the integral over the first quarter and transform it to an integral over the fourth quarter, where we have computed the solution u and its gradient. We know the solution in the first quarter is given by $u_1(x_1, x_2) = u(x_1, -x_2)$ and by using chain rule and transformation theorem we obtain

$$\begin{aligned}\int_{Q_1} \frac{\partial}{\partial x_2} u_1(x_1, x_2)\,\mathrm{d}\boldsymbol{x} = \int_{Q_1} \frac{\partial}{\partial x_2}\Big(u(x_1, -x_2)\Big)\,\mathrm{d}\boldsymbol{x} &= \int_{Q_1} \Big(\frac{\partial}{\partial \tilde{x}_2} u\Big)(x_1, \underbrace{-x_2}_{\tilde{x}_2})(-1)\,\mathrm{d}\boldsymbol{x} \\ &= \int_{Q_4} -\frac{\partial}{\partial \tilde{x}_2} u(\tilde{x}_1, \tilde{x}_2)\,\mathrm{d}\tilde{\boldsymbol{x}}.\end{aligned}$$

Here we use that the modulus of the Jacobian determinant $\left|\det \frac{\partial \boldsymbol{x}}{\partial \tilde{\boldsymbol{x}}}\right| = 1$, since $\boldsymbol{x}(\tilde{\boldsymbol{x}}) = (\tilde{x}_1, -\tilde{x}_2)^T$. We observe that in case of the first quarter we receive an additional minus for the derivative w.r.t. x_2. Similar considerations have to be done for the second and third quarter.

Chapter 3

Isogeometric Analysis

In this chapter we first introduce the B-spline basis and discuss B-spline geometries like curves and surfaces and their properties (Section 3.1). The next step, treated in Section 3.2, is to find a representation via a B-spline mapping of our geometry of electromagnets, which we need in order to specify the discrete design variables. In Section 3.3 we first derive a discretization scheme based on isogeometric analysis and incorporate the Dirichlet boundary conditions in a weak sense by means of Nitsche's method (Section 3.3.1). Moreover, we deduce the linear system of equations we have to solve to compute the numerical solution (Section 3.3.2). Section 3.3.3 is devoted to the question of existence and uniqueness of the solution of the discrete problem. Finally, in Section 3.4 we present some numerical results for the considered state problem, the equations of 2D magnetostatics. This chapter is mainly based on [10] and [9].

3.1 B-splines

A *knot vector* represents a sequence of coordinates in the parameter space

$$\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\},$$

where $\xi_i \in \mathbb{R}$ is called i^{th} *knot*, p is the polynomial order and n is the number of basis functions. The knots subdivide the parameter space into *knot spans* defined by the domain between two adjacent knots. In the following the knots spans are referred to as elements. Moreover, it is possible to place more than one knot at the same coordinate position, which means a certain knot value is repeated. The number of repetitions is referred to as *multiplicity*. A knot vector is called *open* if its first and last knot values have multiplicity $p + 1$.

Definition 3.1. *Given a knot vector* $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$*, the B-spline basis functions are defined by the so-called* Cox-de Boor recursion formula*:*

$$N_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1}^{p-1}(\xi), \tag{3.1}$$

where the base case ($p = 0$) is a piecewise constant function

$$N_i^0(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

B-spline basis functions have many remarkable qualities, which are summarized below.

Properties 3.2. *For the B-spline basis functions defined as above the following properties hold:*

1. *First, they form a partition of unity, i.e.,*

$$\sum_{i=1}^{n} N_i^p(\xi) = 1, \quad \forall \xi.$$

2. *They are pointwise non-negative, i.e., $N_i^p(\xi) \geq 0, \forall \xi$.*

3. *Connection continuity and multiplicity of knot values: A B-spline basis function of order p is C^{p-m_i}-continuous across the knot ξ_i, where m_i denotes the multiplicity of the knot value ξ_i.*

   - *Hence, if a knot value has multiplicity p we only receive C^0-continuity and the basis is interpolatory at the corresponding knot.*
   - *Note, using an open knot vector causes the basis to be interpolatory at the boundary knots.*

   However, in general B-spline basis functions are not interpolatory at interior knots. Note that B-spline basis functions are polynomials on each interval $[\xi_i, \xi_{i+1}]$ and, hence, belong to $C^\infty([\xi_i, \xi_{i+1}])$.

4. *Finally, B-spline basis functions have local support, namely, the support of the basis function $N_{i,p}$ is given by the interval $[\xi_i, \xi_{i+p+1}]$, which consists of $p + 1$ elements. In other words, at a point ξ at most $p+1$ basis functions are non-zero.*

Fortunately, there exist efficient algorithms for the evaluation of the basis functions and their derivatives, see, e.g., [19].

Remark 3.3. *Note, in the implementation we count every element, also these with zero measure, which appear between repeated knots. This approach guarantees that the support of the basis functions is always the same number of elements. However, during this thesis we take the more intuitive idea and only consider elements with positive measure.*

3.1.1 B-spline Curves and Surfaces

The next step is to define a B-spline curve in $\mathbb{R}^d$, which is achieved by forming a linear combination of B-spline basis functions with vector-valued coefficients in $\mathbb{R}^d$, called *control points*. Let N_i^p, $i = 1, \ldots, n$, be basis functions with corresponding control points $\boldsymbol{P}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$. Then we define a piecewise polynomial *B-spline curve* by

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} N_i^p(\xi)\boldsymbol{P}_i. \tag{3.2}$$

In Properties 3.4 we summarize the qualities we need in our further considerations, for a comprehensive discussion of B-spline curve properties we refer the reader to [10] and [19]. The majority of the properties of B-spline curves stated below are direct consequences of the qualities of the B-spline basis, cf. Properties 3.2.

Properties 3.4. *1. Geometric meaning of control points: The* control polygon *is given by the linear interpolation of the control points. Note, the control polygon is like a frame that controls the curve, which means if a control point is displaced the curve follows.*

2. *Continuity: Let us define the element boundaries of a curve in the physical space by the image of the knots under the mapping* (3.2)*. With this definition we obtain, the number of continuous derivatives across an element boundary is greater or equal to the respective number of the basis at the corresponding knot. In this context observe, if the basis is interpolatory at a knot, the B-spline curve is interpolatory at the corresponding control point.*

3. *Locality: Resulting from the local support of the B-spline basis functions, the position of single control point only has an impact on $p + 1$ elements of the curve.*

In practise we also have to represent surfaces. The idea is to consider so-called *tensor product B-splines*, which are constructed by taking the product of two univariate B-spline basis functions. For this purpose, let us consider two knot vectors $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ and $\mathcal{H} = \{\eta_1, \eta_2, \ldots, \eta_{m+q+1}\}$ with corresponding univariate B-spline basis functions $N_{i,p}$ and $M_{j,q}$. Then the tensor product B-splines are defined by

$$N_{i,j}^{p,q}(\xi, \eta) = N_i^p(\xi)M_j^q(\eta), \quad i = 1, \ldots, n,\ j = 1, \ldots, m. \tag{3.3}$$

In analogy to the construction of a B-spline curve, let $N_{i,j}^{p,q}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, be tensor product basis functions with corresponding control points $\boldsymbol{P}_{i,j} \in \mathbb{R}^d$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. Then we define a tensor product *B-spline surface* as follows:

$$\boldsymbol{S}(\xi, \eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,j}^{p,q}(\xi, \eta)\boldsymbol{P}_{i,j}. \tag{3.4}$$

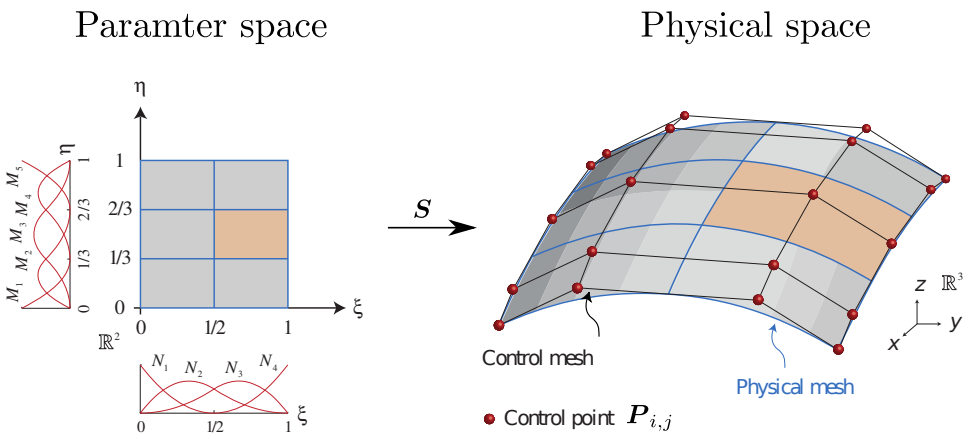


Figure 3.1: Illustration B-spline mapping. Figure is taken from [10].

Below we provide an overview of B-spline surface properties. Note, most of them follow directly from the one-dimensional results by exploiting the tensor product structure.

Properties 3.5. *1. First, tensor product B-splines are again pointwise non-negative and form a partition of unity.*

2. Note, the linear interpolation of the control points is referred to as control mesh, *and its geometrical meaning as frame that controls the geometry stays valid, see Figure 3.1.*

3. In case of a two-dimensional parameter space let us define the elements in the physical space (physical mesh) analogously by the image of the knot lines under the B-spline mapping, see Figure 3.1. Then the number of continuous partial derivatives in a parametric direction can be determined from the multiplicity in the respective one-dimensional knot vector and the polynomial order.

- *In this context observe, a control point is interpolated by the surface, more precisely by a vertex of the physical mesh, if the corresponding univariate basis functions in both parametric directions are interpolatory at the respective knots.*
- *In more dimensions using an open knot vector causes the basis to be interpolatory at the corners.*

4. Finally, tensor B-spline basis functions have local support, namely, the support of the basis function $N^{p,q}_{i,j}(\xi,\eta) = N^p_i(\xi)M^q_j(\eta)$ is given by $[\xi_i, \xi_{i+p+1}]\times[\eta_j, \eta_{j+q+1}]$.

Remark 3.6. *For the further considerations we introduce the following global numbering of the tensor product B-splines basis functions:*

$$A = n(j-1) + i,$$

where $i = 1, \ldots, n$, $j = 1, \ldots, m$ and n, m are the number of basis functions in the ξ- and η-directions, respectively .

Moreover, note that the same technique can be applied to construct B-spline volumes.

3.1.2 Knot Insertion as Refinement Strategy

By means of *knot insertion* we can add new knots without changing the geometry and parametrization of the curve (or surface). Let us consider a coarse knot vector $\overline{\Xi} = \{\overline{\xi}_1, \overline{\xi}_2, \ldots, \overline{\xi}_{n+p+1}\}$ and a refined knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+m+p+1}\}$, such that $\overline{\Xi} \subset \Xi$ holds. The new $n + m$ basis functions are constructed by applying the recursive definition (3.1) to the refined knot vector Ξ. Moreover, the new $n + m$ control points $\mathcal{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{n+m}\}^T$ are defined as linear combinations of the coarse control points $\overline{\mathcal{P}} = \{\overline{\boldsymbol{P}}_1, \overline{\boldsymbol{P}}_2, \ldots, \overline{\boldsymbol{P}}_n\}^T$:

$$\mathcal{P} = \boldsymbol{T}^p \, \overline{\mathcal{P}}, \tag{3.5}$$

where the *transfer matrix* $\boldsymbol{T}^p$ is defined recursively by

$$T_{ij}^{q+1} = \frac{\xi_{i+q} - \overline{\xi}_j}{\overline{\xi}_{j+q} - \overline{\xi}_j} T_{ij}^q + \frac{\overline{\xi}_{j+q+1} - \xi_{i+q}}{\overline{\xi}_{j+q+1} - \overline{\xi}_{j+1}} T_{ij+1}^q$$

for $q = 0, 1, 2 \ldots, p-1$, with base case ($q = 0$):

$$T_{ij}^0(\xi) = \begin{cases} 1 & \text{if } \overline{\xi}_j \leq \xi_i < \overline{\xi}_{j+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Remark 3.7. *So far we have only defined the transfer matrix for a one dimensional parameter space, however, in our application we consider a two-dimensional geometry model. Note, in case of a two-dimensional parameter space the transfer matrix can be computed by combining the transfer matrices of each basis component.*

Remark 3.8. *For our application we only need simple uniform refinement, where a certain number of new knots is inserted in every knot span (with positive measure). Therefore, we already complete our considerations of refinement strategies with the just discussed knot insertion, although there exists a huge number of other interesting techniques in this field, e.g., local refinement methods.*

3.2 B-spline Geometry Model

As already indicated, the control mesh is like a frame that controls the surface; thus, the first idea is to use control points as discrete design variables. In this context, keep in mind that our design variable represents the shape of the half south pole head,

which is not part of the domain boundary. One possibility would be to use multiple patches and describe the pole head as part of the boundary of some of the patches. However, in this thesis we choose a different approach, namely, we only use one patch and describe the pole head by a segment of the physical mesh. For this purpose we have to find a geometry representation via a B-spline mapping of the form (3.4) that allows us to distinguish between the ferromagnetic parts (yoke and poles), coils, air and the magnetization area.

The idea is to map a knot line of the parameter space to a specific mesh line in the physical space. In order to achieve this we make the basis interpolatory at particular knots. The prescribed lines for our considered geometry are indicated in blue in Figure 3.2. Note, we need ten lines (nine elements) in each direction.

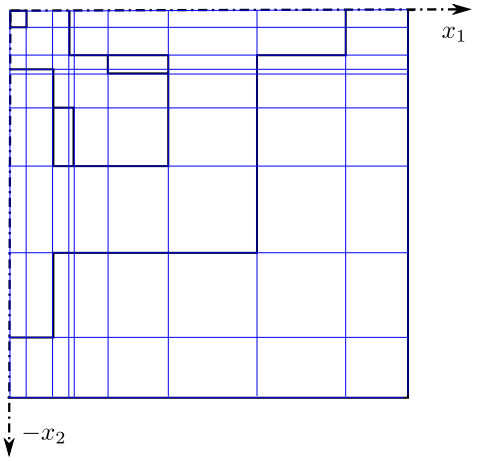


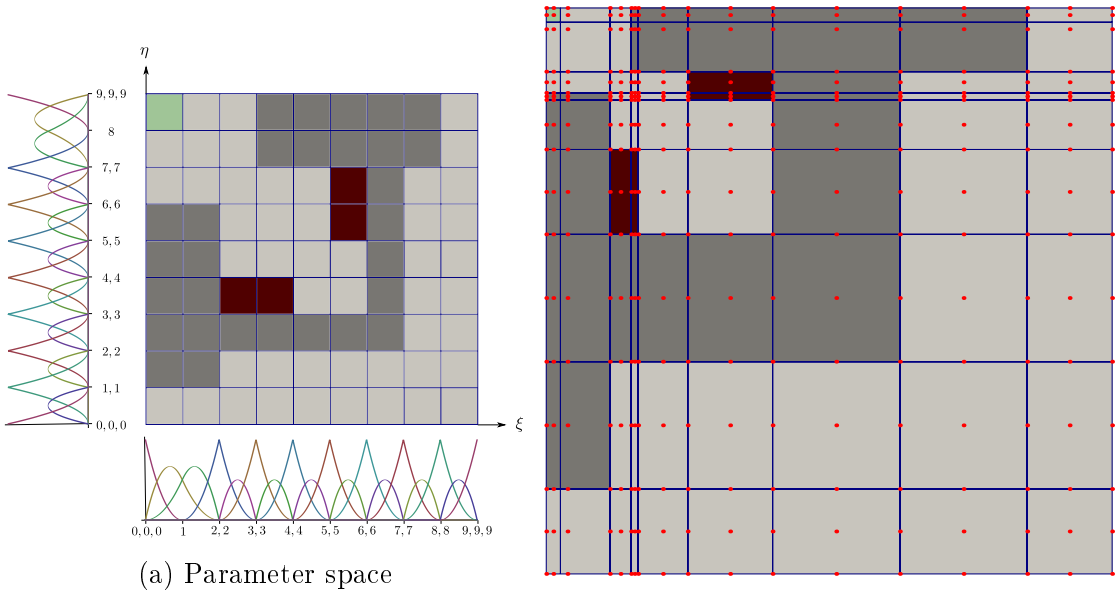
Figure 3.2: Prescribed lines

As a first step, we set the polynomial order for both parametric directions $p = q = 2$. The reason for this choice is that as result of our optimization we want to receive a smooth pole head shape. For this purpose, we demand C^1-continuity across the element boundaries of the pole head, so we at least need quadratic B-splines, according to Properties 3.5.

Moreover, we have to decide what knot vectors we use. For the definition of the knot vectors we have to take the required number of elements and continuity across element boundaries into account. As already stated we need nine elements and the basis should be interpolatory, which can be achieved by increasing the multiplicity of the knot values to 2. However, as indicated above, we require smooth pole head shapes. Hence, we choose for both directions knot vectors consisting of nine elements with C^1-continuity between the elements that belong to the pole head and C^0-continuity elsewhere:

$$\Xi = \{0, 0, 0, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 9\},$$
$$\mathcal{H} = \{0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 9, 9, 9\}.$$

The resulting tensor B-spline basis is given by product of the two one-dimensional B-spline bases depicted in Figure 3.3a.



(a) Parameter space

(b) Physical space

Figure 3.3: Illustration B-spline mapping for the considered geometry of electromagnets. Concerning the colouring, the ferromagnetic parts are grey, the coils red, the magnetization area green and the air light grey. The control points are denoted by the red points, and the blue lines (in the physical space) represent the physical mesh.

Finally, we are able to determine the positions of the control points. We start with the control points which are interpolated by the physical mesh, i.e., those corresponding to interpolatory basis functions in both parametric directions, cf. Properties 3.5. Obviously, we set them such that the vertices of the physical mesh are as for the prescribed lines in Figure 3.2. However, since we are using quadratic basis functions we have an additional control point (corresponding to an interpolatory basis function in one parametric direction and a non-interpolatory in the other) between two interpolated control points. In order to obtain straight lines between the vertices we have to place them on the paraxial lines between the interpolated points. Moreover, positioning the just discussed points and the further points (corresponding to non-interpolatory basis functions in both parametric directions) centred, as depicted in Figure 3.3b, results in a linear parametrization of the elements. Note, the mesh lines that define the magnetization area require special treatment, since the 1D bases are not interpolatory at the corresponding knots. Here we exploit that the knots are equally spaced in the parameter space, which provides that at $\xi = 1$ and $\eta = 8$ both non-zero basis functions have value 0.5, see Figure 3.3a. Hence, the discussed knot lines are placed in the middle of the two lines of control points to the left and to the right.

3.2.1 Definition of the Discrete Design Variables

The final task of this section is the definition of the discrete design variables. According to the coarse geometry description depicted in Figure 3.3b, the shape of the south pole head is described by three control points, where we assume the right boundary control point of the pole head to be fixed. In order to obtain a richer set of representable shapes, we insert additional knot values, with single multiplicity, between the values $1, 2$ (in ξ-direction) and $7, 8$ (in η-direction) in Figure 3.3a. In the practical implementation we insert three or six further knots in each parametric direction and position the corresponding additional lines of control points equally spaced. So we end up with six or nine free control points as design variables; in Figure 3.4 the case with six design variables is illustrated. For simplicity, we only consider their x_2-component as design variables and define

$$\boldsymbol{p} = \{[\boldsymbol{P}_A]_2 : A \in \mathcal{D}\}, \tag{3.6}$$

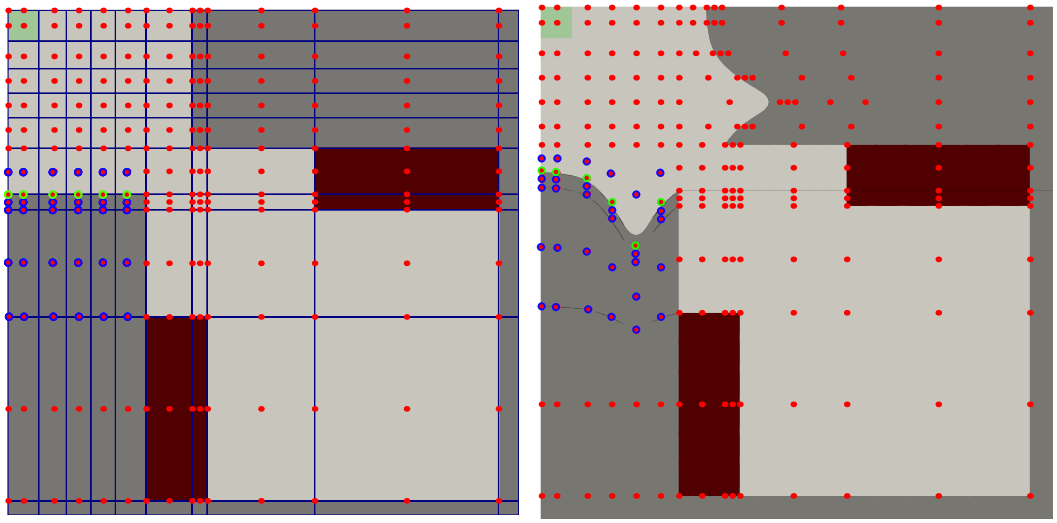
where $\mathcal{D}$ denotes the set of global numbers of control points that define the shape of the half south pole head, so for our choice, $\mathcal{D}$ consists of six or nine elements. The assumption that only the x_2-components are free to move is not a big limitation, since we are still able to represent a wide range of pole head shapes, see Figure 3.4b.

Remark 3.9. *If the control points of the pole head are displaced, additionally, a certain number of control points above and below should be adjusted in order to reduce the risk of overlapping of the elements. In the concrete implementation one row of control points above and four rows of control points below, emphasized in blue in Figure 3.4, are displaced. The first two rows below are translated in the same way as the control points of the pole head, since the x_2-dimension of the elements is quite small there. In case of the further points above and below, the displacement is decreased linearly.*

3.3 Galerkin Method

3.3.1 Variational Formulation (Nitsche's Method)

The aim of this subsection is to derive a discrete variational formulation for our state problem, the 2D linear magnetostatic formulation (2.12). For this purpose, let us consider as model problem a slight generalization of problem (2.12), including both inhomogeneous Dirichlet and Neumann boundary conditions. Note, for the reduction to a quarter of the domain, illustrated in Section 2.4, homogeneous Dirichlet and Neumann boundary conditions have to be taken into account. Throughout this chapter, let $\Omega \subset \mathbb{R}^2$ be a bounded domain with a sufficiently smooth boundary $\Gamma = \partial\Omega$ composed of two disjoint parts Γ_D, Γ_N with $\overline{\Gamma}_D \cup \overline{\Gamma}_N = \Gamma$. Moreover, we assume $\mathrm{meas}(\Gamma_D) > 0$ in order to exclude the case of pure Neumann boundary conditions, where the solution



(a) Rectangular pole head shape (b) Modified pole head shape

Figure 3.4: Control mesh and physical mesh. The control points representing design variables are coloured green and the adjusted control points blue.

can only be unique up to an additive constant. The model problem reads

$$-\operatorname{div}\Big(\frac{1}{\mu}\nabla u\Big) = J \quad \text{in } \Omega, \tag{3.7a}$$

$$u = g_D \quad \text{on } \Gamma_D, \tag{3.7b}$$

$$\frac{1}{\mu}\frac{\partial u}{\partial \boldsymbol{n}} = g_N \quad \text{on } \Gamma_N, \tag{3.7c}$$

where $\frac{\partial u}{\partial \boldsymbol{n}}$ denotes the normal derivative $\nabla u \cdot \boldsymbol{n}$ and $\boldsymbol{n}$ is the unit outer normal on Γ. In the classical setting we look for a solution $u \in C^2(\Omega) \cap C^1(\Omega \cup \Gamma_N) \cap C(\Omega \cup \Gamma_D)$ under classical assumptions on the data, i.e., $J \in C(\Omega)$, $g_D \in C(\Gamma_D)$, $g_N \in C(\Gamma_N)$. Moreover, the coefficient function μ, which represents in our practical application the permeability, is assumed to be bounded from above and below by positive constants μ_0, μ_1, i.e.,

$$\mu_0 \le \mu(\boldsymbol{x}) \le \mu_1, \quad \forall \boldsymbol{x} \in \Omega. \tag{3.8}$$

Let us assume the classical solution u of the model problem (3.7) and the data fulfil additional integrability conditions, e.g., $u \in C^2(\overline{\Omega})$, $J \in C(\overline{\Omega})$, $g_D \in C(\overline{\Gamma}_D)$, $g_N \in C(\overline{\Gamma}_N)$, and let the test function $v \in H^1(\Omega)$. Now, we multiply the differential equation (3.7a) with the test function, integrate over the domain Ω and perform integration by parts. Then we end up with the following variational equation:

$$\int_\Omega \frac{1}{\mu}\nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_D} \frac{1}{\mu}\frac{\partial u}{\partial \boldsymbol{n}} v \,\mathrm{d}s = \int_\Omega J v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} \underbrace{\frac{1}{\mu}\frac{\partial u}{\partial \boldsymbol{n}}}_{=g_N} v \,\mathrm{d}s, \quad \forall v \in H^1(\Omega), \tag{3.9}$$

where we already split the boundary integral into a Dirichlet part and a Neumann part. Moreover, we bring the Neumann part to the right-hand side and plug in the Neumann boundary condition g_N. Note, function values on the boundary of functions in $H^1(\Omega)$ have to be interpreted in the sense of the Trace Theorem (Theorem 3.10) stated below.

Theorem 3.10 (Trace Theorem)**.** *Let $\Omega \subset \mathbb{R}^n$ be a bounded domain with a Lipschitz-continuous boundary. Then there exists a unique bounded linear operator*

$$\gamma_0 : H^1(\Omega) \to L^2(\Gamma) \tag{3.10}$$

with

$$\|\gamma_0 v\|_{L^2(\Gamma)} \leq C\|v\|_{H^1(\Omega)}, \quad \forall v \in H^1(\Omega), \tag{3.11}$$

such that $\gamma_0 v = v|_\Gamma$ holds, for all $v \in C^1(\overline{\Omega})$. Moreover, $\gamma_0(H^1(\Omega)) = H^{1/2}(\Gamma)$, i.e., every function in $H^{1/2}(\Gamma)$ is trace of an $H^1(\Omega)$ function.

Proof. A proof can be found in, e.g., [1]. □

One possibility would be to incorporate the Dirichlet boundary condition into the solution space and use test functions that vanish on the Dirichlet boundary. However, in this theses we have chosen a different approach called Nitsche's Method. Before we incorporate the Dirichlet boundary conditions in a weak sense by means of Nitsche's method, we need the following definition of the discrete function space V_h.

Definition 3.11. *By taking all linear combinations of basis functions we obtain the discrete space*

$$V_h = \{v_h : v_h(\boldsymbol{x}) = \sum_{A=1}^{n_{bf}} N_A(\boldsymbol{x}) v_A\}, \tag{3.12}$$

where n_{bf} denotes the total number of basis functions. The basis functions $N_A(\boldsymbol{x})$ are defined by means of the so-called mapping principle, i.e.,

$$N_A(\boldsymbol{x}) = \hat{N}_A(\boldsymbol{\xi}(\boldsymbol{x})), \quad \boldsymbol{x} \in \Omega,$$

where $\hat{N}_A$ is the basis function on the parameter space. The geometrical mapping $\boldsymbol{x} : \hat{\Omega} \to \Omega$ which maps the parameter space $\hat{\Omega}$ to the physical space Ω is defined by

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{A=1}^{n_{bf}} \hat{N}_A(\boldsymbol{\xi}) \boldsymbol{P}_A, \quad \boldsymbol{\xi} \in \hat{\Omega},$$

and $\boldsymbol{\xi} = \boldsymbol{x}^{-1} : \Omega \to \hat{\Omega}$.

In other words, we only have to give a definition of the basis functions on the parameter space, and the basis functions in the physical space are defined via the geometrical mapping $\boldsymbol{x}$.

Remark 3.12. *In the previous definition the basis functions $\hat{N}_A$ may represent any kind of basis function (univariate, bivariate, trivariate, ...), where the dimension of the parameter space $\hat{\Omega}$ has to be chosen accordingly. For our practical application we consider $\hat{\Omega} \subset \mathbb{R}^2$ and use bivariate B-splines, as defined in* (3.3)*. Then the geometrical mapping has the form of a tensor product B-spline surface, see* (3.4)*, with control points in $\mathbb{R}^2$, since $\Omega \subset \mathbb{R}^2$.*

Remark 3.13. *Note, we use the same bases for the definition of the geometrical mapping and the solution space. This approach called* isoparametric concept *is widely used in classical finite element analysis.*

For the further analysis it is important to observe that functions in V_h are polynomials on every element and so C^∞ on the elements, and across element boundaries they are at least continuous. Hence, V_h is a subspace of $H^1(\Omega)$, which means we are in the conforming case.

Notation. The physical mesh is given by the image of the knot lines under the geometrical mapping, cf. Properties 3.5. In the following we will denote its elements by Ω_e and their preimage in the parameter space by $\hat{\Omega}_e$, where $e = 1, \dots, n_{el}$, with n_{el} is the total number of elements. Moreover, the physical mesh also induces a partition of the boundary Γ, and by E we denote an edge on the boundary.

The next step is to derive a discrete variational formulation, based an the discrete space V_h, that allows us to incorporate the Dirichlet boundary condition in a weak sense, by means of Nitsche's method (cf. [12]). In order to do this, let $u \in C^2(\overline{\Omega})$ be again the classical solution of the model problem (3.7), and in contrast to above we test with a discrete test function $v_h \in V_h$. As before, by means of integration by parts we obtain

$$\int_\Omega \frac{1}{\mu} \nabla u \cdot \nabla v_h \,\mathrm{d}\boldsymbol{x} - \sum_{E \in \Gamma_D} \int_E \frac{1}{\mu} \frac{\partial u}{\partial n} v_h \,\mathrm{d}s = \int_\Omega J v_h \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} g_N v_h \,\mathrm{d}s, \tag{3.13a}$$

where we have split the Dirichlet boundary integral into the summation over the edges that belong to the Dirichlet boundary. We still have to incorporate the Dirichlet boundary condition. To do so, we impose the following additional equations:

$$-\sum_{E \in \Gamma_D} \int_E \frac{1}{\mu} u \frac{\partial v_h}{\partial \boldsymbol{n}} \,\mathrm{d}s = -\sum_{E \in \Gamma_D} \int_E \frac{1}{\mu} g_D \frac{\partial v_h}{\partial \boldsymbol{n}} \,\mathrm{d}s, \tag{3.13b}$$

$$\sum_{E \in \Gamma_D} \frac{\sigma}{h_E} \int_E u v_h \,\mathrm{d}s = \sum_{E \in \Gamma_D} \frac{\sigma}{h_E} \int_E g_D v_h \,\mathrm{d}s, \tag{3.13c}$$

where the stabilization parameter $\sigma > 0$ and h_E denotes the length of the edge E. First, the requested equations hold for the exact solution, since it fulfils the Dirichlet boundary condition $u = g_D$ on Γ_D. Recall that on every element a function in V_h belongs to C^∞. Therefore, if we split the Dirichlet integral into the summation over

the edges, the trace of the normal derivative $\frac{\partial v_h}{\partial \boldsymbol{n}} = \nabla u \cdot \boldsymbol{n}$ is on every edge well defined, although a function in V_h is on the whole domain only in H^1. The Trace Theorem (Theorem 3.10) guarantees for a function $v \in H^1(\Omega)$ the existence of the trace on the boundary of the functions itself but not for its derivative, which belongs only to $L^2(\Omega)$.

By summing up the equations (3.13a), (3.13b) and (3.13c) we obtain that the classical solution $u \in C^2(\overline{\Omega})$ solves the equation

$$a_h(u, v_h) = \langle F_h, v_h \rangle, \quad \forall v_h \in V_h, \tag{3.14}$$

with

$$a_h(u, v_h) = \int_\Omega \frac{1}{\mu} \nabla u \cdot \nabla v_h \, \mathrm{d}\boldsymbol{x} + \sum_{E \in \Gamma_D} \left[- \int_E \frac{1}{\mu} \frac{\partial u}{\partial n} v_h \, \mathrm{d}s - \int_E \frac{1}{\mu} u \frac{\partial v_h}{\partial n} \, \mathrm{d}s + \frac{\sigma}{h_E} \int_E u v_h \, \mathrm{d}s \right], \tag{3.15}$$

$$\langle F_h, v_h \rangle = \int_\Omega J v_h \, \mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} g_N v_h \, \mathrm{d}s + \sum_{E \in \Gamma_D} \left[- \int_E \frac{1}{\mu} g_D \frac{\partial v_h}{\partial n} \, \mathrm{d}s + \frac{\sigma}{h_E} \int_E g_D v_h \, \mathrm{d}s \right]. \tag{3.16}$$

Note that the bilinear form $a_h(\cdot, \cdot)$ is symmetric due to the minus sign in equation (3.13b). This motivates to define the discrete problem as follows:

Definition 3.14 (Discrete Problem)**.** *Find $u_h \in V_h$, such that*

$$a_h(u_h, v_h) = \langle F_h, v_h \rangle, \quad \forall v_h \in V_h, \tag{3.17}$$

where $a_h(\cdot, \cdot)$ and $\langle F_h, . \rangle$ are defined as above.

The assumptions on the data of (3.17) can be weakened, e.g., $J \in L^2(\Omega)$, $g_D \in H^{1/2}(\Gamma_D)$, $g_N \in L^2(\Gamma_N)$. Moreover, also the bounds of coefficient function μ (3.8) only have to hold almost everywhere in Ω.

Note, in the Interior Penalty Discontinuous Galerkin method the Dirichlet boundary conditions are incorporated in the same way.

3.3.2 Linear System of Equations

Due to linearity of $a_h(\cdot, \cdot)$ and $\langle F_h, . \rangle$, it is sufficient to test in the discrete variational formulation (3.17) only with a basis of the space V_h. Therefore, we obtain that the discrete problem (3.17)

$$\begin{aligned} \Leftrightarrow \quad & \text{Find } u_h \in V_h \text{ s.t.} \\ & a_h(u_h, N_A) = \langle F_h, N_A \rangle, \quad \forall N_A, \, A = 1, \ldots, n_{bf}. \end{aligned}$$

The next step is to plug in for u_h the representation

$$u_h(\boldsymbol{x}) = \sum_{B=1}^{n_{bf}} N_B(\boldsymbol{x}) u_B, \tag{3.18}$$

which yields that the discrete problem (3.17)

$$\begin{aligned} \Leftrightarrow \quad & \text{Find } \boldsymbol{u} = (u_1, \dots, u_{n_{bf}})^T \in \mathbb{R}^{n_{bf}} \text{ s.t.} \\ & \sum_{B=1}^{n_{bf}} a_h(N_B, N_A)\, u_B = \langle F_h, N_A \rangle, \quad \forall N_A, A = 1, \dots, n_{bf}, \\ \Leftrightarrow \quad & \text{Find } \boldsymbol{u} \in \mathbb{R}^{n_{bf}} \text{ s.t.} \\ & \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \end{aligned} \tag{3.19}$$

with

$$\begin{aligned} [\boldsymbol{K}]_{AB} &= a_h(N_B, N_A) = a_h(N_A, N_B), \\ [\boldsymbol{f}]_A &= \langle F_h, N_A \rangle, \\ [\boldsymbol{u}]_A &= u_A, \end{aligned}$$

for all $A, B = 1, \dots, n_{bf}$. Here, we use the symmetry of the bilinear form. In order to compute the solution u_h, we have to solve the linear system (3.19). In this context it is important to observe that the stiffness matrix $\boldsymbol{K}$ is sparse, due to the local support of the basis functions, cf. Properties 3.2 and Properties 3.5. More precisely, the number of basis functions with support on an element, referred to as n_{bf_el}, is given by $(p+1)(q+1)$ in the case of bivariate B-spines. Note, the term number of local basis functions is also common. Therefore, the assembling of the linear system is done element wise, as in classical finite element analysis. The procedure is to loop through the elements:

- Build the element stiffness matrix $\boldsymbol{K}^e \in \mathbb{R}^{n_{bf\_el} \times n_{bf\_el}}$.
- Build the element load vector $\boldsymbol{f}^e \in \mathbb{R}^{n_{bf\_el}}$.
- Add the entries of the element stiffness matrix and load vector to the corresponding entries of the global stiffness matrix and load vector. In order to do this, we need a connection between the local ordering of the basis functions on an element and the global basis function numbers, which is provided by the so-called connectivity array.

Remark 3.15. *The task of the connectivity array is to link for each element $e = 1, \dots, n_{el}$, every local basis function number $a = 1, \dots, n_{bf\_el}$, to the corresponding global basis function number $A \in \{1, \dots, n_{bf}\}$. Which means the connectivity array is a mapping of the form $CA(a, e) = A$. In order to define a connectivity array we need a local numbering of the basis functions on every element and a global numbering of the basis functions (cf. Remark 3.6) and of the elements. The numbering technique used for the implementation and the algorithm for building the connectivity array is taken from [10, Appendix A]*

Notation. In the following we denote by $A \in \{1, \dots, n_{bf}\}$ the global number of a basis function, and $a \in \{1, \dots, n_{bf\_el}\}$ denotes the local number of the basis function on a given element. When further indices are needed, the letters B and b respectively will be used.

What is still missing is the computation of the element stiffness matrix $\boldsymbol{K}^e$ and element load vector $\boldsymbol{f}^e$. Let us start with the first part of the bilinear form (3.15), which means we consider

$$[\boldsymbol{K}^e]_{ab} = \int_{\Omega_e} \frac{1}{\mu(\boldsymbol{x})} \nabla N_a(\boldsymbol{x}) \cdot \nabla N_b(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

over an element Ω_e in the physical space, where N_a, N_b are local basis functions. The approach is to first transform the integral back to the parameter space and then to the so-called parent element $[-1,1] \times [-1,1]$, as illustrated in Figure 3.5. On the

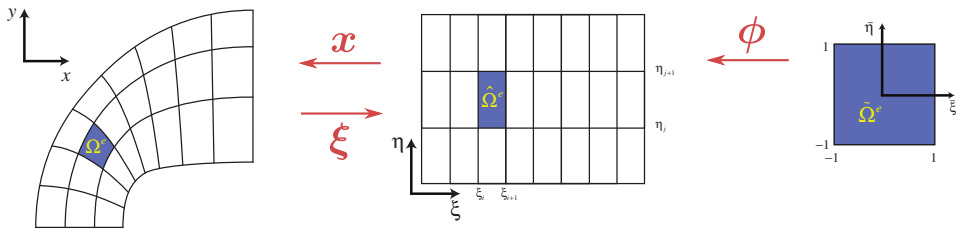


Figure 3.5: Diagram of the mappings between physical space, parameter space and parent element. Figure is taken from [10].

parent element we apply a quadrature rule, e.g., Gaussian quadrature, to compute the integral. Remember, Ω_e denotes the element in the physical space, $\hat{\Omega}_e$ denotes the corresponding element in the parameter space and now let us refer to the parent element as $\tilde{\Omega}$. Analogous, we denote by $\boldsymbol{x}$ coordinates in the physical space, by $\boldsymbol{\xi}$ coordinates in the parameter space and by $\tilde{\boldsymbol{\xi}}$ coordinates in the parent element. The mapping $\boldsymbol{x} : \hat{\Omega}_e \to \Omega_e$ which maps the element from the parameter space to the physical space and its inverse $\boldsymbol{\xi} : \Omega_e \to \hat{\Omega}_e$ are defined as in Definition 3.11. Furthermore, we defined an affine mapping $\boldsymbol{\phi} : \tilde{\Omega} \to \hat{\Omega}_e$ which maps the parent element to the element in the parameter space.

Notation. For convenience, we introduce the following notation for the Jacobian determinant of a mapping. Let us exemplarily consider the Jacobian of the mapping $\boldsymbol{x} : \hat{\Omega}_e \to \Omega_e$ and define

$$\left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \right| = \left| \det \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \right|. \tag{3.20}$$

Note, if the mapping is defined such that the determinant of its Jacobian is always positive, which is common practise, we can omit the absolute value of the determinant.

In so doing, we obtain

$$[\boldsymbol{K}^e]_{ab} = \int_{\hat{\Omega}_e} \frac{1}{\mu(\boldsymbol{x}(\boldsymbol{\xi}))} \nabla_x N_a(\boldsymbol{x}(\boldsymbol{\xi})) \cdot \nabla_x N_b(\boldsymbol{x}(\boldsymbol{\xi})) \left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \right| \mathrm{d}\boldsymbol{\xi}.$$

Moreover, we have to transform the derivatives with respect to physical coordinates $\boldsymbol{x}$ to derivatives with respect to parametric coordinates $\boldsymbol{\xi}$. Note, there exist efficient

algorithms to compute the derivatives on the parameter space. To do so, we represent the basis function on the physical space by means of the basis function on the parameter space and use chain rule. Then we receive

$$\frac{\partial}{\partial \boldsymbol{x}} N(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}} \Big(\hat{N}(\boldsymbol{\xi}(\boldsymbol{x})) \Big) = \Big(\frac{\partial}{\partial \boldsymbol{\xi}} \hat{N} \Big)(\boldsymbol{\xi}(\boldsymbol{x})) \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \Big(\frac{\partial}{\partial \boldsymbol{\xi}} \hat{N} \Big)(\boldsymbol{\xi}(\boldsymbol{x})) \Big(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}(\boldsymbol{x})) \Big)^{-1},$$

where we use that

$$\boldsymbol{x}(\boldsymbol{\xi}(\boldsymbol{x})) = \boldsymbol{x} \Rightarrow \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}(\boldsymbol{x})) \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = 1.$$

Summing up, we obtain for the gradient

$$\nabla_x N(\boldsymbol{x}) = \nabla_x \Big(\hat{N}(\boldsymbol{\xi}(\boldsymbol{x})) \Big) = \Big(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}(\boldsymbol{x})) \Big)^{-T} \Big(\nabla_\xi \hat{N} \Big)(\boldsymbol{\xi}(\boldsymbol{x})),$$

and the gradient at the considered point $\boldsymbol{x}(\boldsymbol{\xi})$ reads

$$\nabla_x N(\boldsymbol{x}(\boldsymbol{\xi})) = \Big(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \Big)^{-T} \nabla_\xi \hat{N}(\boldsymbol{\xi}).$$

If we plug in the transformed gradient and transform the integral back to the parent element, we end up with

$$\int_{\tilde{\Omega}} \frac{1}{\mu(\boldsymbol{x}(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})))} \Big(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})) \Big)^{-T} \nabla_\xi \hat{N}_a(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})) \cdot \Big(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})) \Big)^{-T} \nabla_\xi \hat{N}_b(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})) \Big| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\phi}(\tilde{\boldsymbol{\xi}})) \frac{\partial \boldsymbol{\phi}}{\partial \tilde{\boldsymbol{\xi}}} \Big| \, \mathrm{d}\tilde{\boldsymbol{\xi}}, \tag{3.21}$$

where we multiply with the Jacobian determinant of the composition of $\boldsymbol{x}$ and $\boldsymbol{\phi}$. Note, the Jacobian of the affine mapping $\boldsymbol{\phi}$ is independent of $\tilde{\boldsymbol{\xi}}$.

The contribution consisting of an integral over the domain Ω to the linear functional (3.16) can be computed in the same way. The remaining parts are boundary integrals; so let us consider exemplarily the Neumann boundary integral appearing in the linear functional (3.16). Since the assembling is done boundary edge wise, we have to compute the following integral:

$$\int_E \frac{1}{\mu(\boldsymbol{x})} g_N(\boldsymbol{x}) N_a(\boldsymbol{x}) \, \mathrm{d}s.$$

In order to compute the line integral we need a parametrization of the boundary edge. A possible parametrization corresponding to the lower edge of the parent element is given by

$$\begin{aligned} \gamma : [-1, 1] &\to \mathbb{R}^2 \\ t &\mapsto \boldsymbol{x}(\boldsymbol{\phi}(t, -1)). \end{aligned}$$

If we plug in the parametrization, we end up with

$$\int_{-1}^{1} \frac{1}{\mu(\gamma(t))} g_N(\gamma(t)) \underbrace{N_a(\boldsymbol{x}(\boldsymbol{\phi}(t,-1)))}_{=\hat{N}_a(\boldsymbol{\phi}(t,-1))} \|\gamma'(t)\| \,\mathrm{d}t, \tag{3.22}$$

where the derivative of the parametrization is given by $\gamma'(t) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\phi}(t,-1))\frac{\partial \boldsymbol{\phi}}{\partial \tilde{\xi}}$. As next step we use again a quadrature rule to compute the line integral. If there is a derivative involved, we apply the same considerations as above (chain rule) to transform it to a derivative with respect to the parameter space coordinates $\boldsymbol{\xi}$.

Now, the question of existence and uniqueness of the solution of the discrete problem (3.17) arises.

3.3.3 Numerical Analysis of the Bilinear Form: Existence and Uniqueness

In this section we will show ellipticity and boundedness of the bilinear form $a_h(\cdot,\cdot)$ and conclude by means of the Theorem of Lax and Milgram existence and uniqueness of the solution for a bounded linear functional as right hand side. The majority of this section is based on [15] and [12].

As already indicated, we apply the Theorem of Lax and Milgram, which is stated below.

Theorem 3.16 (Lax-Milgram)**.** *Let V be a Hilbert space and $a : V \times V \to \mathbb{R}$ a bilinear form that is elliptic (coercive), i.e.,*

$$a(v,v) \geq C_1^a \|v\|_V^2, \quad \forall v \in V,$$

with a constant $C_1^a > 0$ and bounded, i.e.,

$$a(v,w) \leq C_2^a \|v\|_V \|w\|_V, \quad \forall v, w \in V,$$

with a constant $C_2^a > 0$. Further, let $F \in V^*$. Then the variational problem: Find $u \in V$ s.t.

$$a(u,v) = \langle F, v \rangle, \quad \forall v \in V$$

has a unique solution $u \in V$, which fulfils

$$\frac{1}{C_2^a}\|F\|_{V^*} \leq \|u\|_V \leq \frac{1}{C_1^a}\|F\|_{V^*}.$$

Proof. For the proof we refer to a book about the finite element method, e.g., [3]. □

In the analysis of the bilinear form $a_h(\cdot,\cdot)$ we will use the following discrete mesh-dependent norm:

$$\|v_h\|_h^2 = \int_\Omega \frac{1}{\mu}|\nabla v_h|^2 \,\mathrm{d}\boldsymbol{x} + \sum_{E \in \Gamma_D} \frac{\sigma}{h_E} \int_E v_h^2 \,\mathrm{d}s = \left\| \left(\frac{1}{\mu}\right)^{\frac{1}{2}} \nabla v_h \right\|_{L^2(\Omega)}^2 + \sum_{E \in \Gamma_D} \frac{\sigma}{h_E} \|v_h\|_{L^2(E)}^2. \tag{3.23}$$

The next Lemma provides an inverse inequality, which we need in order to prove ellipticity and boundedness of our bilinear form $a_h(\cdot,\cdot)$.

Lemma 3.17 (cf. [12]). *For $v_h \in V_h$ the estimate*

$$\sum_{E\in\Gamma_D} h_E \left\| \frac{\partial v_h}{\partial \boldsymbol{n}} \right\|^2_{L^2(E)} \leq C_I \|\nabla v_h\|^2_{L^2(\Omega)}$$

holds, with a mesh independent constant $C_I > 0$.

Proof. In order to show the estimate one uses a so-called scaling argument. □

Remark 3.18. *For the availability of the just stated inverse inequality we need assumptions on the mapping $\boldsymbol{F}_e = \boldsymbol{x} \circ \boldsymbol{\phi} : \tilde{\Omega} \to \Omega_e$ from the parent element to the element in the physical space, see Figure 3.5. More precisely, we need upper bounds for the norm of the Jacobian matrix and the Jacobian determinant, given by*

$$\left\| \frac{\partial \boldsymbol{F}_e}{\partial \tilde{\boldsymbol{\xi}}} \right\| \leq C_1 h_e, \qquad \left\| \frac{\partial \boldsymbol{F}_e^{-1}}{\partial \boldsymbol{x}} \right\| \leq C_2 h_e^{-1},$$
$$\left| \frac{\partial \boldsymbol{F}_e}{\partial \tilde{\boldsymbol{\xi}}} \right| \leq C_3 h_e^2, \qquad \left| \frac{\partial \boldsymbol{F}_e^{-1}}{\partial \boldsymbol{x}} \right| \leq C_4 h_e^{-2},$$

where h_e denotes the diameter of the element in the physical space Ω_e and all involved constants are independent of h_e (cf. [7]).

Now, we are able to prove ellipticity of the bilinear form.

Lemma 3.19. *Let $\sigma \geq \frac{4C_I\mu_1}{\mu_0^2}$, where C_I is the constant of the inverse inequality of Lemma 3.17, and let μ_0, μ_1 be the lower and upper bound of the permeability μ (3.8). Then there holds the ellipticity estimate*

$$a_h(v_h, v_h) \geq \frac{1}{2}\|v_h\|_h^2, \quad \forall v_h \in V_h.$$

Proof. For $v_h \in V_h$ we receive for the bilinear form

$$\begin{aligned} a_h(v_h, v_h) &= \int_\Omega \frac{1}{\mu} |\nabla v_h|^2 \,\mathrm{d}\boldsymbol{x} + \sum_{E\in\Gamma_D} \left[-2 \int_E \frac{1}{\mu} \frac{\partial v_h}{\partial \boldsymbol{n}} v_h \,\mathrm{d}s + \frac{\sigma}{h_E} \int_E v_h^2 \,\mathrm{d}s \right] \\ &= \|v_h\|_h^2 - 2 \sum_{E\in\Gamma_D} \int_E \frac{1}{\mu} \frac{\partial v_h}{\partial \boldsymbol{n}} v_h \,\mathrm{d}s, \end{aligned} \tag{3.24}$$

where we combine the first and last integral and obtain $\|v_h\|_h^2$. Now, our goal is to estimate the part with the negative sign from above by $\|v_h\|_h^2$ with constant $\frac{1}{2}$. To do so, we first apply Cauchy-Schwarz inequality and receive

$$\sum_{E\in\Gamma_D} \int_E \frac{1}{\mu} \frac{\partial v_h}{\partial \boldsymbol{n}} v_h \,\mathrm{d}s \leq \sum_{E\in\Gamma_D} \left\| \frac{1}{\mu} \frac{\partial v_h}{\partial \boldsymbol{n}} \right\|_{L^2(E)} \|v_h\|_{L^2(E)}.$$

Before we apply Cauchy-Schwarz inequality in the Euclidean space we expand with an artificial one of the form $\frac{h_E}{\sigma}\frac{\sigma}{h_e}$:

$$= \sum_{E\in\Gamma_D} \Big(\frac{h_E}{\sigma}\Big)^{\frac{1}{2}} \Big\| \frac{1}{\mu}\frac{\partial v_h}{\partial \boldsymbol{n}}\Big\|_{L^2(E)} \Big(\frac{\sigma}{h_E}\Big)^{\frac{1}{2}} \|v_h\|_{L^2(E)}.$$

Now, we apply Cauchy-Schwarz inequality in the Euclidean space and end up with

$$\leq \left[\sum_{E\in\Gamma_D} \frac{h_E}{\mu_0^2\sigma}\Big\|\frac{\partial v_h}{\partial \boldsymbol{n}}\Big\|^2_{L^2(E)}\right]^{\frac{1}{2}} \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)}\right]^{\frac{1}{2}},$$

where we additionally estimate μ by its lower bound μ_0. By means of the inverse inequality of Lemma 3.17 we can estimate the first sum from above and obtain

$$\leq \Big(\frac{C_I}{\mu_0^2\sigma}\Big)^{\frac{1}{2}} \|\nabla v_h\|_{L^2(\Omega)} \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)}\right]^{\frac{1}{2}}.$$

In the next step we apply Young's inequality, $xy \leq \frac{1}{2\epsilon}x^2 + \frac{\epsilon}{2}y^2$, with $x = \Big(\frac{C_I}{\mu_0^2\sigma}\Big)^{\frac{1}{2}}\|\nabla v_h\|_{L^2(\Omega)}$ and $y = \Big[\sum_{E\in\Gamma_D}\frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)}\Big]^{\frac{1}{2}}$ and get for some $\epsilon > 0$

$$\leq \frac{1}{2\epsilon}\frac{C_I}{\mu_0^2\sigma}\|\nabla v_h\|^2_{L^2(\Omega)} + \frac{\epsilon}{2}\sum_{E\in\Gamma_D}\frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)}.$$

Finally, we can estimate the term with the negative sign in (3.24) from above and receive as lower bound

$$a_h(v_h, v_h) \geq \|v_h\|_h^2 - \frac{1}{\epsilon}\frac{C_I\mu_1}{\mu_0^2\sigma}\Big\|\Big(\frac{1}{\mu}\Big)^{\frac{1}{2}}\nabla v_h\Big\|^2_{L^2(\Omega)} - \epsilon\sum_{E\in\Gamma_D}\frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)},$$

where we additionally use the estimate

$$\|\nabla v_h\|^2_{L^2(\Omega)} \leq \Big\|\Big(\frac{\mu_1}{\mu}\Big)^{\frac{1}{2}}\nabla v_h\Big\|^2_{L^2(\Omega)}.$$

For the choice $\epsilon = \frac{1}{2}$ and with the assumption $\sigma \geq \frac{4C_I\mu_1}{\mu_0^2}$, as stated in the Lemma, we end up with the following lower bound:

$$a_h(v_h, v_h) \geq \|v_h\|_h^2 - \underbrace{\frac{2C_I\mu_1}{\mu_0^2}\frac{\mu_0^2}{4C_I\mu_1}}_{=\frac{1}{2}}\Big\|\Big(\frac{1}{\mu}\Big)^{\frac{1}{2}}\nabla v_h\Big\|^2_{L^2(\Omega)} - \frac{1}{2}\sum_{E\in\Gamma_D}\frac{\sigma}{h_E}\|v_h\|^2_{L^2(E)}$$

$$= \frac{1}{2}\|v_h\|_h^2.$$

□

Lemma 3.20. *The bilinear form $a_h(\cdot,\cdot)$ is bounded, i.e.,*

$$a_h(u_h, v_h) \leq C_2^a \|u_h\|_h \|v_h\|_h, \quad \forall u_h, v_h \in V_h.$$

Proof. Let u_h, v_h be in V_h; then we obtain by means of Cauchy-Scharz inequality the estimate

$$\begin{aligned}
&a_h(u_h, v_h) \\
&= \int_\Omega \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla u_h \cdot \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla v_h \,\mathrm{d}\boldsymbol{x} \\
&\quad - \sum_{E\in\Gamma_D} \int_E \frac{1}{\mu} \frac{\partial u_h}{\partial \boldsymbol{n}} v_h \,\mathrm{d}s - \sum_{E\in\Gamma_D} \int_E \frac{1}{\mu} u_h \frac{\partial v_h}{\partial \boldsymbol{n}} \,\mathrm{d}s + \sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \int_E u_h v_h \,\mathrm{d}s \\
&\leq \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla u_h \Big\|_{L^2(\Omega)} \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla v_h \Big\|_{L^2(\Omega)} + \sum_{E\in\Gamma_D} \Big(\frac{h_E}{\sigma}\Big)^{\frac{1}{2}} \Big\| \frac{1}{\mu} \frac{\partial u_h}{\partial \boldsymbol{n}} \Big\|_{L^2(E)} \Big(\frac{\sigma}{h_E}\Big)^{\frac{1}{2}} \|v_h\|_{L^2(E)} \\
&\quad + \sum_{E\in\Gamma_D} \Big(\frac{\sigma}{h_E}\Big)^{\frac{1}{2}} \|u_h\|_{L^2(E)} \Big(\frac{h_E}{\sigma}\Big)^{\frac{1}{2}} \Big\| \frac{1}{\mu} \frac{\partial v_h}{\partial \boldsymbol{n}} \Big\|_{L^2(E)} + \sum_{E\in\Gamma_D} \Big(\frac{\sigma}{h_E}\Big)^{\frac{1}{2}} \|u_h\|_{L^2(\Omega)} \Big(\frac{\sigma}{h_E}\Big)^{\frac{1}{2}} \|v_h\|_{L^2(\Omega)},
\end{aligned}$$

where we expand the summands that contain a normal derivative with an artificial one of the form $\frac{h_E}{\sigma}\frac{\sigma}{h_E}$. In the next step we apply Cauchy-Schwarz inequality in the Euclidean space and receive

$$\begin{aligned}
&\leq \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla u_h \Big\|_{L^2(\Omega)} \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla v_h \Big\|_{L^2(\Omega)} + \left[\sum_{E\in\Gamma_D} \frac{h_E}{\mu_0^2 \sigma} \Big\| \frac{\partial u_h}{\partial \boldsymbol{n}} \Big\|^2_{L^2(E)}\right]^{\frac{1}{2}} \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|v_h\|^2_{L^2(E)}\right]^{\frac{1}{2}} \\
&\quad + \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|u_h\|^2_{L^2(E)}\right]^{\frac{1}{2}} \left[\sum_{E\in\Gamma_D} \frac{h_E}{\mu_0^2 \sigma} \Big\| \frac{\partial v_h}{\partial \boldsymbol{n}} \Big\|^2_{L^2(E)}\right]^{\frac{1}{2}} \\
&\quad + \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|u_h\|^2_{L^2(E)}\right]^{\frac{1}{2}} \left[\sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|v_h\|^2_{L^2(E)}\right]^{\frac{1}{2}},
\end{aligned}$$

where we additionally estimate μ by its lower bound μ_0. By means of the inverse inequality of Lemma 3.17 we can estimate the L^2 norm on the edge of the normal derivative with the L^2 norm on Ω of the gradient. If we do so and use Cauchy-Schwarz inequality for the four summands, we end up with the following upper bound for $a_h(u_h, v_h)$:

$$\begin{aligned}
a_h(u_h, v_h) \leq &\left[\Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla u_h \Big\|^2_{L^2(\Omega)} + \frac{C_I \mu_1}{\mu_0^2 \sigma} \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla u_h \Big\|^2_{L^2(\Omega)} + 2 \sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|u_h\|^2_{L^2(E)} \right]^{\frac{1}{2}} \\
&\cdot \left[\Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla v_h \Big\|^2_{L^2(\Omega)} + \frac{C_I \mu_1}{\mu_0^2 \sigma} \Big\| \Big(\frac{1}{\mu}\Big)^{\frac{1}{2}} \nabla v_h \Big\|^2_{L^2(\Omega)} + 2 \sum_{E\in\Gamma_D} \frac{\sigma}{h_E} \|v_h\|^2_{L^2(E)} \right]^{\frac{1}{2}} \\
&\leq \max\{1 + \frac{C_I \mu_1}{\mu_0^2 \sigma}, 2\} \|u_h\|_h \|v_h\|_h.
\end{aligned}$$

Note, for the first estimate we additionally use that there holds

$$\|\nabla v_h\|^2_{L^2(\Omega)} \leq \left\| \left(\frac{\mu_1}{\mu}\right)^{\frac{1}{2}} \nabla v_h \right\|^2_{L^2(\Omega)}.$$

□

Moreover, one can show by means of Cauchy-Schwarz inequality that the linear functional (3.16) is also bounded.

Theorem 3.21. *Let σ be as in Lemma 3.19, i.e., $\sigma \geq \frac{4C_I\mu_1}{\mu_0^2}$. Then the discrete variational problem* (3.17) *is uniquely solvable in V_h and the solution depends continuously on the data.*

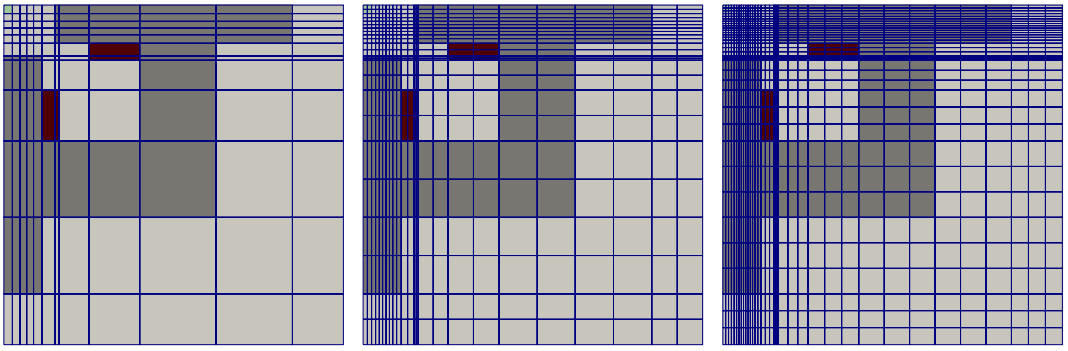
Proof. According to Lemma 3.19 the bilinear form is elliptic, and Lemma 3.20 provides the boundedness of the bilinear form. Furthermore, the linear functional is also bounded. Hence, the Theorem of Lax and Milgram (Theorem 3.16) provides the stated result. □

3.4 Numerical Experiments: Application to the State Problem

Finally, we use the derived discretization scheme to compute a solution of our state problem, the 2D linear magnetostatic problem defined in Section 2.4.2. In this section we present the obtained results.

In Section 3.2 we have defined a coarse mesh that represents the geometry. However, in order to achieve good accuracy of the numerical solution, we have to compute the solution of the state problem on a refined mesh. For this purpose, we take advantage of the knot insertion technique described in Section 3.1.2. Recall that knot insertion preserves the geometry and the parametrization. Keep in mind, the second property is important, since as a result of no change in the parametrization we still have a partition of our domain (into ferromagnetic parts, coils, air and magnetization area) after refinement.

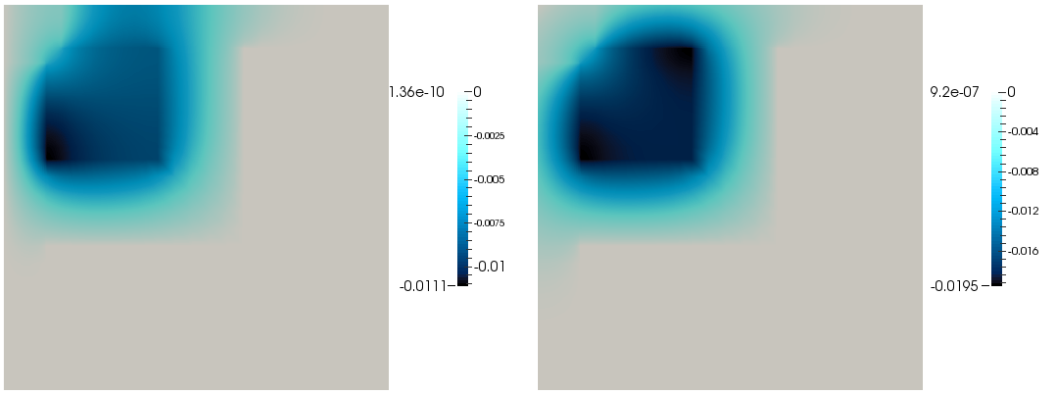
In Figure 3.6a we see the coarse mesh that describes the geometry (in case of six design variables), given by the B-spline mapping defined in Section 3.2. It consists of 12 elements (with positive measure) in each parametric direction. This leads to a total number of basis functions of 441, which is equal to the number of degrees of freedom (later referred to as DOF) of our discrete solution. Figure 3.6b shows the refined mesh, where one new knot is inserted in each element in each parameter direction. A further refined mesh, where two knots are inserted in each element is depicted in Figure 3.6c. In case of one new inserted knot we end up with 1089 DOF and in case of two new inserted knots with 2025 DOF. For the computation of the solution of the state problem we used a refined mesh, where four new knots are inserted in every element, which leads to 4761 DOF for the solution. The results are depicted in Figure 3.7. On the



(a) Coarse mesh that describes the geometry (b) Uniform refined mesh (one knot inserted) (c) Uniform refined mesh (two knots inserted)

Figure 3.6: Meshes with different refinement level

left (Figure 3.7a) we see the solution for the vertical excitation case, where only one coil is switched on, and Figure 3.7b on the right shows the solution for the diagonal excitation scenario, where both coils are switched on.



(a) Vertical current excitation (b) Diagonal current excitation

Figure 3.7: Solution of the state problem on the lower right quarter of the computational domain

Remark 3.22. *As already indicated in Section 2.4.2, we do not solve a second state problem corresponding to the diagonal current excitation scenario. By using reflection and superposition, the solution for the vertical current excitation is given by*

$$u_{diag} = u_{vert} + u_{vert,ref},$$

where u_{vert} is the solution of the vertical case and $u_{vert,ref}$ is the reflection of the vertical solution w.r.t. the diagonal $x_1 = -x_2$. Therefore, the discrete solution for the vertical

case can be computed by

$$u_{diag,h}(\boldsymbol{x}) = u_{vert,h}(\boldsymbol{x}) + u_{vert,ref,h}(\boldsymbol{x}) = \sum_{A=1}^{n_{bf}} N_A(\boldsymbol{x}) u_A + \sum_{A=1}^{n_{bf}} N_A(\boldsymbol{x}) u_{A,ref}.$$

Here, u_A is the coefficient in the basis representation of the vertical solution and $u_{A,ref}$ is the basis coefficient that corresponds to the reflected basis function.

Chapter 4

Numerical Methods for Optimization

In this chapter we focus on the numerical method used for the optimization. At the beginning, we complete the definition of the considered discrete optimization problem by discetizing the cost function. In Section 4.2 we give a brief overview of the BFGS method, the most popular quasi-Newton algorithm, which we use for the optimization. Section 4.3 is dedicated to the derivation of the gradient of the cost function (sensitivity analysis). In this context we consider three groups of methods: approximate, discrete and continuous approaches. In the remaining chapter we will discuss the first two in more detail, since these fit in the concept of "first discretize, then optimize", which we apply in this thesis.

4.1 Discretization Cost Function

Before we can start with the discussion of numerical methods for solving the discrete optimization problem in nested formulation (2.3), we have to discretize the cost function defined in (2.17). Let us define the discretized cost function $\mathcal{I}_h$ by

$$\mathcal{I}_h(\boldsymbol{B}_h^1, \boldsymbol{B}_h^2) = \frac{1}{2} \sum_{v=1}^{2} \Big[\varphi_h^v(\boldsymbol{B}_h^v) + \rho \theta_h^v(\boldsymbol{B}_h^v) \Big], \tag{4.1}$$

where the integrals over the magnetization area Ω_m are split into the summation over the elements:

$$\varphi_h^v(\boldsymbol{B}_h^v) = \frac{1}{\operatorname{meas}(\Omega_m)(B_{min}^{avg,v})^2} \sum_{\Omega_e \in \Omega_m} \int_{\Omega_e} |\boldsymbol{B}_h^v(\boldsymbol{x}) - B_h^{avg,v}(\boldsymbol{B}_h^v) \boldsymbol{n}_m^v|^2 \, \mathrm{d}\boldsymbol{x}, \tag{4.2}$$

$$B_h^{avg,v}(\boldsymbol{B}_h^v) = \frac{1}{\operatorname{meas}(\Omega_m)} \sum_{\Omega_e \in \Omega_m} \int_{\Omega_e} \boldsymbol{B}_h^v(\boldsymbol{x}) \cdot \boldsymbol{n}_m^v \, \mathrm{d}\boldsymbol{x}, \tag{4.3}$$

$$\theta_h^v(\boldsymbol{B}_h^v) = \Big[\max(0, B_{min}^{avg,v} - B_h^{avg,v}(\boldsymbol{B}_h^v)) \Big]^2. \tag{4.4}$$

Keep in mind, the involved discrete magnetic flux density $\boldsymbol{B}_h$ can be easily computed from the discrete solution u_h. Let $\boldsymbol{x} \in \Omega_e$; then the discrete solution u_h reads

$$u_h(\boldsymbol{x}) = \sum_{a=1}^{n_{bf\_el}} N_a(\boldsymbol{x}) u_a,$$

where the basis coefficients u_a are defined by the discrete state problem (3.19). Note, we only have to sum over the basis functions that have support on the given element. According to Section 2.4.2, the magnetic flux density is defined by

$$\boldsymbol{B}_h(\boldsymbol{x}) = \Big(\frac{\partial u_h}{\partial x_2}(\boldsymbol{x}), -\frac{\partial u_h}{\partial x_1}(\boldsymbol{x})\Big), \tag{4.5}$$

where the involved derivatives are given by

$$\frac{\partial u_h}{\partial x_i}(\boldsymbol{x}) = \sum_{a=1}^{n_{bf\_el}} \frac{\partial N_a}{\partial x_i}(\boldsymbol{x}) u_a. \tag{4.6}$$

Remark 4.1. *In order to compute the integrals appearing in* (4.2) *and* (4.3) *we apply the same technique as in Section 3.3.2, i.e., we first transform the integral back to the parameter space and then to the parent element. On the parent element we apply Gaussian quadrature to actually compute the integral. Note, we again have to transform the derivatives of the basis functions to derivatives with respect to parametric coordinates* $\boldsymbol{\xi}$*. This can be done by means of chain rule, as shown in Section 3.3.2.*

Remark 4.2. *Remember, we only solve one state problem for the vertical current excitation case. The discrete solution for the diagonal current excitation scenario can be computed as illustrated in Remark 3.22. Then the discrete magnetic flux density can be computed analogous as above.*

Keep in mind, in order to evaluate the cost function we have to compute an integral, involving the magnetic flux density $\boldsymbol{B}_h$*, over all four quarters of the magnetization area. As indicated in Remark 2.3, we should take the respective sign of the derivative of* u_h*, arising from reflection, into account when we use a quadrature rule to approximate the integral.*

4.2 Optimization Algorithm: BFGS Method

First of all, we recall that the discrete nested optimization problem (2.3) without constraints reads

$$\min_{\boldsymbol{p} \in \mathbb{R}^n} \ \mathcal{I}_h(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})). \tag{4.7}$$

Note, no constraints have to be taken into account, since we do not impose design and state constrains in our model, as indicated in Section 2.4. Moreover, our cost function does not depend explicitly on the design variables. The reason is that the

magnetization area is fixed and does not depend on the shape of the pole head. So we end up with a problem of the following form:

$$\min_{\boldsymbol{p}\in\mathbb{R}^n} \ \hat{\mathcal{I}}_h(\boldsymbol{p}) = \mathcal{I}_h(\boldsymbol{u}(\boldsymbol{p})). \tag{4.8}$$

For the choice of the optimization algorithm it is important to observe that the cost function depends nonlinearly on the control points of the pole head, i.e., on the design variables $\boldsymbol{p}$. This results from the implicit dependence of the state variable $\boldsymbol{u}(\boldsymbol{p})$ on the design variables through the state equation $\boldsymbol{K}(\boldsymbol{p})\boldsymbol{u}(\boldsymbol{p}) = \boldsymbol{f}(\boldsymbol{p})$. Therefore, we are in the case of unconstrained nonlinear optimization. For comparability reasons we use an external tool for the optimization, namely, the MATLAB routine `fminunc` with the quasi-Newton algorithm or more precisely the BFGS method.

In the following we give a brief introduction to the BFGS method. This introduction is mainly based on [16]. As model problem let us consider the general unconstrained nonlinear minimization problem

$$\min_{x\in\mathbb{R}^n} f(x). \tag{4.9}$$

Throughout this section we use the notation $f_k = f(x_k)$. First of all, quasi-Newton algorithms need only information of the gradient of the objective function in every iteration. By means of the changes in the gradient they construct a quadratic model of the objective function (at the iterate x_k) of the form:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p,$$

where B_k is a symmetric positive definite matrix that will be updated in every iteration. The search direction p_k is given by the minimizer of the just defined convex quadratic problem, which can be solved explicitly, i.e.,

$$p_k = -B_k^{-1}\nabla f_k.$$

Keep in mind, in contrast to the Newton method, we use instead of the exact Hessian an approximation B_k. Instead of computing a new approximation in every iteration, one uses an updating technique based on curvature information of the previous iteration. An efficient and widely used one is the BFGS formula. Before we state the formula let us introduce the following notations: $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$. Now, the BFGS update of the Hessian B_k reads

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{4.10}$$

and the corresponding update of the inverse Hessian approximation H_k is given by

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \tag{4.11}$$

with $\rho_k = \frac{1}{y_k^T s_k}$.

The next step is to state the BFGS algorithm.

Algorithm 1 (BFGS Method [16])**.**
Given: starting point x_0, convergence tolerance $\epsilon > 0$, initial inverse Hessian approximation H_0
$k = 0$
while convergence criterion not fulfilled **do**
Compute search direction $p_k = -H_k \nabla f_k$.
Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k is computed by means of line search.
Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$.
Compute H_{k+1} by means of (4.11).
k=k+1
end while

Concerning the initial approximation H_0, the simplest choice is a multiple of the identity matrix.

Remark 4.3. *The used step length α resulting from line search has to satisfy the Wolfe conditions:*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k,$$
$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k,$$

with constants c_2, c_2, $0 < c_1 < c_2 < 1$.

In order to apply the BFGS algorithm to the unconstrained optimization problem (4.8), we need information about the gradient of the objective function $\mathcal{I}_h$ with respect to the design variables $\boldsymbol{p}$. The computation of this derivatives, or sensitivities, therefore, called sensitivity analysis, is the task of the next section.

4.3 Sensitivity Analysis

The three main approaches how to compute the design sensitivity are the following:

1. Approximation approach (finite difference method)
2. Discrete approach
3. Continuum approach

The basis for the majority of this section are [4] and [5]. In the continuum approach the derivative is taken before discretization, i.e., of the still continuum problem. This approach is related to the concept "*first optimize, then discretize*". Note, this group of methods provides often higher accuracy, since the derivatives are taken before we approximate the continuum problem by a finite dimensional discrete problem. However, in this thesis we apply, as already indicated in the beginning, the approach "*first discretize, then optimize*". Therefore, we restrict our considerations to the remaining two other approaches, which we discuss in the following subsections.

4.3.1 Approximation Approach (Finite Difference Method)

In case of the approximation approach the derivatives are computed approximately by means of finite differences, for example, forward difference quotients or central difference quotients are used. Let us denote by $p_j, j = 1, \ldots, n_{design}$, a single design variable, where n_{design} is the total number of design variables. Then the forward finite difference approximation at a design $\boldsymbol{p}$ is given by

$$\frac{\mathrm{d}\hat{\mathcal{I}}_h}{\mathrm{d}p_j} \approx \frac{\hat{\mathcal{I}}_h(\boldsymbol{p} + h\boldsymbol{e}_j) - \hat{\mathcal{I}}_h(\boldsymbol{p})}{h}, \tag{4.12}$$

where $\boldsymbol{e}_j$ is the unit vector that has a one in the j-th row. It can be shown that the truncation error is of order $\mathcal{O}(h)$. The more accurate central difference approximation (truncation error is $\mathcal{O}(h^2)$) at $\boldsymbol{p}$ reads

$$\frac{\mathrm{d}\hat{\mathcal{I}}_h}{\mathrm{d}p_j} \approx \frac{\hat{\mathcal{I}}_h(\boldsymbol{p} + h\boldsymbol{e}_j) - \hat{\mathcal{I}}_h(\boldsymbol{p} - h\boldsymbol{e}_j)}{2h}. \tag{4.13}$$

The big advantage of finite difference methods is that they are very easy to implement. However, they have considerable drawbacks in terms of computation cost and accuracy. The difficult task for finite difference methods is the determination of the right perturbation size h. Clearly, too large perturbations h leads to bad approximation, but for too small parameters h numerical noise becomes dominant and the results are again not accurate.

A differentiation technique by means of finite differences is directly implemented in the used MATLAB routine `fminunc`. It is applied if the option `GradObj` is disabled. In section Chapter 5 we compare the results of the just described method with the results obtained by the so-called discrete method, which we will describe in the next section.

4.3.2 Discrete Method

Let us denote by p a single design variable. Then we obtain by means of chain rule

$$\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}p}\big(\boldsymbol{u}(\boldsymbol{p})\big) = \frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}\big(\boldsymbol{u}(\boldsymbol{p})\big)\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p}(\boldsymbol{p}), \tag{4.14}$$

where the derivatives are interpreted as Fréchet derivatives, i.e., $\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}$ is a row vector and $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p}$ is a column vector. In the following we omit the function arguments for better readability. In order to obtain the derivative $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p}$, we differentiate the discrete state equation $\boldsymbol{K}(\boldsymbol{p})\boldsymbol{u}(\boldsymbol{p}) = \boldsymbol{f}(\boldsymbol{p})$. In so doing, we end up with

$$\frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}\boldsymbol{u} + \boldsymbol{K}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p}, \tag{4.15}$$

which can be rewritten as the following linear system of equations:

$$\boldsymbol{K}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} - \frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}\boldsymbol{u}. \tag{4.16}$$

For the further proceeding there exist two different approaches: the direct method and the adjoint method.

Direct Method

In the direct method we solve the linear system (4.16) for $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p}$ and insert the result into (4.14). In other words, for every design variable we have to solve the linear system of equation (4.16), which is numerically quite expensive. Therefore, we use a more efficient approach, called adjoint method, which we describe in the sequel.

Adjoint Method

The idea in the adjoint method is to plug in (4.14) the from (4.16) obtained representation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}p} = \boldsymbol{K}^{-1}\boldsymbol{r},$$

where the residual $\boldsymbol{r}$ is given by $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} - \frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}\boldsymbol{u}$. Then we obtain

$$\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}p} = \frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}\boldsymbol{K}^{-1}\boldsymbol{r}. \tag{4.17}$$

In the previous expression we define the part in front of the residual, which does not include a design derivative, by $\boldsymbol{\lambda}^T$, i.e.,

$$\boldsymbol{\lambda}^T = \frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}\boldsymbol{K}^{-1} \quad \Leftrightarrow \quad \boldsymbol{K}\boldsymbol{\lambda} = \Big(\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}\Big)^T, \tag{4.18}$$

where we use that the stiffness matrix $\boldsymbol{K}$ is symmetric. In the adjoint method the first step is to solve the just stated linear system for $\boldsymbol{\lambda}$. Then the derivative with respect to a single design variable p is obtained by multiplying $\boldsymbol{\lambda}$ with the respective residual:

$$\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}p} = \boldsymbol{\lambda}^T\Big(\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} - \frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}\boldsymbol{u}\Big). \tag{4.19}$$

Summing up, we only have to solve one linear system (4.18) and for every design variable we only have to compute the residual and multiply with $\boldsymbol{\lambda}$.

Remark 4.4. *Note, in the adjoint method we have to solve the linear system* (4.18) *for the objective function and if present for every constraint function. In case of the direct method we have seen that we have to solve the linear system* (4.16) *for every design variable. Hence, we can conclude that when we have fewer constraints than design variables, as in our situation, the adjoint method is beneficial. However, in the other case the direct method is advantageous.*

Finally, in order to determine the design sensitivity of our cost function $\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}p}$, via the direct or adjoint method, we have to compute the design derivative of the stiffness matrix $\frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}(\boldsymbol{p})$ and of the load vector $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p}(\boldsymbol{p})$. Moreover, we need the state dependence of the cost function $\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{u}}\big(\boldsymbol{u}(\boldsymbol{p})\big)$. If this derivatives are obtained analytically, as illustrated in the next two subsections, we speak of an analytical method. However, if finite differences are used to determine this derivatives the method is called semianalytical.

4.3.3 Design Derivative of the Stiffness Matrix and Load Vector

This subsection deals with the computation of the design derivative of stiffness matrix and load vector. The presented approach is based on the techniques for the derivation of analytical sensitivities for isogeometric discretizations presented in [20] and [21]. First, we recall that the entries of the stiffness matrix $\boldsymbol{K}$ are defined by

$$\begin{aligned}[\boldsymbol{K}]_{AB} &= a_h(N_A, N_B) \\ &= \int_\Omega \frac{1}{\mu} \nabla N_A \cdot \nabla N_B \,\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{E \in \Gamma_D} \left[- \int_E \frac{1}{\mu} \frac{\partial N_A}{\partial n} N_B \,\mathrm{d}s - \int_E \frac{1}{\mu} N_A \frac{\partial N_B}{\partial n} \,\mathrm{d}s + \frac{\sigma}{h_E} \int_E N_A N_B \,\mathrm{d}s \right].\end{aligned} \tag{4.20}$$

In the following we use the same approach as for the assembling of the finite element linear system (see Section 3.3.2), namely, we compute the derivatives element wise. Let us start with the first part, i.e., $\int_\Omega \frac{1}{\mu} \nabla N_A \cdot \nabla N_B \,\mathrm{d}\boldsymbol{x}$. As indicated, the assembling is carried out element wise, so let us consider the integral

$$[\boldsymbol{K}^e]_{ab} = \int_{\Omega_e} \frac{1}{\mu} \nabla N_a \cdot \nabla N_b \,\mathrm{d}\boldsymbol{x}$$

over an element Ω_e in the physical space, where N_a, N_b are local basis functions. First of all, we transform the integral back to the parameter space. Let us denote in the following the Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}$ by $\boldsymbol{J}$. Then we end up with

$$[\boldsymbol{K}^e]_{ab} = \int_{\hat{\Omega}_e} \frac{1}{\mu} \nabla_x N_a \cdot \nabla_x N_b |\boldsymbol{J}| \,\mathrm{d}\boldsymbol{\xi},$$

where the gradient w.r.t. $\boldsymbol{x}$ can be transformed by chain rule to a gradient w.r.t. parametric coordinates $\boldsymbol{\xi}$ (as in Section 3.3.2):

$$\nabla_x N = \boldsymbol{J}^{-T} \nabla_\xi \hat{N}.$$

For the further considerations it is important to note that the element in the parameter space $\hat{\Omega}_e$ does not depend on the control points and, thus, does not depend on the

design variables. By means of product rule we obtain for the derivative with respect to a single design variable p

$$\frac{\mathrm{d}[\boldsymbol{K}^e]_{ab}}{\mathrm{d}p} = \int_{\hat{\Omega}_e} \frac{1}{\mu}\Big(\frac{\mathrm{d}}{\mathrm{d}p}\nabla_x N_a \cdot \nabla_x N_b |\boldsymbol{J}| + \nabla_x N_a \cdot \frac{\mathrm{d}}{\mathrm{d}p}\nabla_x N_b |\boldsymbol{J}| + \nabla_x N_a \cdot \nabla_x N_b \frac{\mathrm{d}}{\mathrm{d}p}|\boldsymbol{J}|\Big)\,\mathrm{d}\boldsymbol{\xi}, \tag{4.21}$$

where we use that the permeability μ is in our application constant on every element. The next step is to take a closer look at the involved derivatives. In the first and second summand in (4.21) a derivative of $\nabla_x N$ appears, which reduces to

$$\frac{\mathrm{d}}{\mathrm{d}p}\nabla_x N = \Big(\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}^{-T}\Big)\nabla_\xi \hat{N}, \tag{4.22}$$

since the basis function on the parameter space $\hat{N}$ is independent of the control points and, hence, independent of the design variable p. Moreover, the derivative of the inverse of the Jacobian can be computed as follows, cf. [20]:

$$\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}^{-1} = -\boldsymbol{J}^{-1}\Big(\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}\Big)\,\boldsymbol{J}^{-1}.$$

In the third summand in (4.21) a derivative of the Jacobian determinant appears, which is according to Jacobi's formula (see [21]) given by

$$\frac{\mathrm{d}}{\mathrm{d}p}|\boldsymbol{J}| = |\boldsymbol{J}|\,\mathrm{tr}\Big(\boldsymbol{J}^{-1}\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}\Big).$$

Here, we assume the Jacobian determinant to be positive and omit the absolute value; otherwise, we can only compute the derivative of the Jacobian determinant piecewise. Summing up, in order to evaluate the integral (4.21), we only need to compute the derivative of the Jacobian $\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}$. To do so, let p be the x_2-component of some control point $\boldsymbol{P}_{\bar{A}}$, i.e., $p = [\boldsymbol{P}_{\bar{A}}]_2$. Then we have

$$\frac{\mathrm{d}}{\mathrm{d}p}[\boldsymbol{J}]_{ij} = \frac{\mathrm{d}}{\mathrm{d}p}\left[\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\right]_{ij} = \frac{\mathrm{d}}{\mathrm{d}p}\left(\sum_{a=1}^{n_{bf\_el}} \frac{\partial \hat{N}_a}{\partial \xi_j}[\boldsymbol{P}_a]_i\right) = \begin{cases} \frac{\partial \hat{N}_a}{\partial \xi_j} & \exists a \leftrightarrow A = \bar{A}, i = 2, \\ 0 & \text{otherwise,} \end{cases} \tag{4.23}$$

where A is the global number of the local basis function with index a. Note, if the derivative of the Jacobian $\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}$ is zero, then all three summands in (4.21) vanish. This means we only receive a non-zero contribution $\frac{\mathrm{d}}{\mathrm{d}p}[\boldsymbol{K}^e]_{ab}$ from these elements where the basis function that corresponds to the control point $\boldsymbol{P}_{\bar{A}}$ has support.

Moreover, we have to differentiate the boundary integrals involved in (4.20). To do so, let us consider the following integral:

$$\big[\boldsymbol{K}^E\big]_{ab} = \int_E \frac{1}{\mu}\nabla N_a(\boldsymbol{x}) \cdot \boldsymbol{n}\, N_b(\boldsymbol{x})\,\mathrm{d}s.$$

As discussed in Section 3.3.2, we need a parametrization of the boundary edge, so let us exemplarily consider an edge with parametrization

$$\begin{aligned} \boldsymbol{\gamma} : [-1,1] &\to \mathbb{R}^2 \\ t &\mapsto \boldsymbol{x}(\boldsymbol{\phi}(t,-1)). \end{aligned} \tag{4.24}$$

Hence, we have to differentiate

$$\left[\boldsymbol{K}^E\right]_{ab} = \int_{-1}^{1} \frac{1}{\mu} \nabla_x N_a(\boldsymbol{x}(\boldsymbol{\phi}(t,-1))) \cdot \boldsymbol{n} \underbrace{N_b(\boldsymbol{x}(\boldsymbol{\phi}(t,-1)))}_{=\hat{N}_b(\boldsymbol{\phi}(t,-1))} \|\boldsymbol{\gamma}'(t)\| \,\mathrm{d}t,$$

where $\gamma'(t) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\phi}(t,-1)) \frac{\partial \boldsymbol{\phi}}{\partial \tilde{\xi}} = \boldsymbol{J}(\boldsymbol{\phi}(t,-1)) \frac{\partial \boldsymbol{\phi}}{\partial \tilde{\xi}}$, with respect to a single design variable p. By means of product rule we receive

$$\frac{\mathrm{d}\left[\boldsymbol{K}^E\right]_{ab}}{\mathrm{d}p} = \int_{-1}^{1} \frac{1}{\mu} \Big(\frac{\mathrm{d}}{\mathrm{d}p} \nabla_x N_a \cdot \boldsymbol{n} \, \hat{N}_b \|\boldsymbol{\gamma}'(t)\| + \nabla_x N_a \cdot \boldsymbol{n} \, \hat{N}_b \frac{\mathrm{d}}{\mathrm{d}p} \|\boldsymbol{\gamma}'(t)\| \Big) \,\mathrm{d}t. \tag{4.25}$$

Here, we use that the basis function on the parameter space $\hat{N}$ is independent of the control points, i.e., independent of the design variable. Moreover, note that the normal vector is for our definition of the design variables, as x_2-component of the control points of the south pole head, independent of them.

The derivative of the gradient w.r.t. $\boldsymbol{x}$, which appears in the first summand, can be computed as in (4.22). Additionally, we have to consider $\frac{\mathrm{d}}{\mathrm{d}p}\|\boldsymbol{\gamma}'\|$, which is given by

$$\frac{\mathrm{d}}{\mathrm{d}p}\|\boldsymbol{\gamma}'\| = \frac{\mathrm{d}}{\mathrm{d}p}\Big({\gamma_1'}^2 + {\gamma_2'}^2\Big)^{\frac{1}{2}} = \Big({\gamma_1'}^2 + {\gamma_2'}^2\Big)^{-\frac{1}{2}} \Big(\gamma_1' \frac{\mathrm{d}\gamma_1'}{\mathrm{d}p} + \gamma_2' \frac{\mathrm{d}\gamma_2'}{\mathrm{d}p}\Big), \tag{4.26}$$

where

$$\frac{\mathrm{d}\boldsymbol{\gamma}'}{\mathrm{d}p} = \frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}p} \frac{\partial \boldsymbol{\phi}}{\partial \tilde{\xi}}.$$

Here, we use that the mapping $\boldsymbol{\phi}$ from the parent element to the element in the parameter space is independent of the design variable p. Note, the derivative of the Jacobian $\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}$ has already been computed in (4.23). Since the derivative of the Jacobian $\frac{\mathrm{d}}{\mathrm{d}p}\boldsymbol{J}$ is involved in both summands, we only receive a non-zero contribution $\frac{\mathrm{d}}{\mathrm{d}p}[\boldsymbol{K}^E]_{ab}$ from these elements where the corresponding basis function has support.

The second type of boundary integral involved in (4.20) is of the form

$$\left[\boldsymbol{K}^E\right]_{ab} = \frac{\sigma}{h_E} \int_E N_a(\boldsymbol{x}) N_b(\boldsymbol{x}) \,\mathrm{d}s.$$

By means of the parametrization $\boldsymbol{\gamma}$ (4.24), the boundary integral can be written as

$$\left[\boldsymbol{K}^E\right]_{ab} = \frac{\sigma}{h_E} \int_{-1}^{1} \underbrace{N_a(\boldsymbol{x}(\boldsymbol{\phi}(t,-1)))}_{=\hat{N}_a(\boldsymbol{\phi}(t,-1))} \hat{N}_b(\boldsymbol{\phi}(t,-1)) \|\boldsymbol{\gamma}'(t)\| \,\mathrm{d}t.$$

Then we obtain for the derivative with respect to a single design variable p by applying product rule

$$\frac{\mathrm{d}\big[\boldsymbol{K}^E\big]_{ab}}{\mathrm{d}p} = \sigma\Big(\frac{\mathrm{d}}{\mathrm{d}p}\frac{1}{h_E}\Big)\int_{-1}^{1}\hat{N}_a\hat{N}_b\|\boldsymbol{\gamma}'(t)\|\,\mathrm{d}t + \frac{\sigma}{h_E}\int_{-1}^{1}\hat{N}_a\hat{N}_b\frac{\mathrm{d}}{\mathrm{d}p}\|\boldsymbol{\gamma}'(t)\|\,\mathrm{d}t, \tag{4.27}$$

since the basis functions $\hat{N}$ on the parameter space are independent of the design variable. An often applied simplification is to treat the length of the edge h_E as a constant. This means in the previous expression only the second summand has to be considered. However, experiments have shown a measurable difference in the derivative if the first summand has not been taken into account.

Keep in mind, the derivative arising in the second integral can be computed as in (4.26). Additionally, we have to differentiate $\frac{1}{h_E}$, where h_E represent the length of the edge. Since the boundary edges are in case of our geometry straight lines, their length is given by

$$h_E = \|\boldsymbol{e}\| = \|\boldsymbol{x}(\boldsymbol{\xi}_1) - \boldsymbol{x}(\boldsymbol{\xi}_2)\|,$$

with $\boldsymbol{\xi}_1 = \boldsymbol{\phi}(-1,-1)$ and $\boldsymbol{\xi}_2 = \boldsymbol{\phi}(1,-1)$. Then we receive for the derivative

$$\frac{\mathrm{d}}{\mathrm{d}p}\frac{1}{h_E} = \frac{\mathrm{d}}{\mathrm{d}p}\big(e_1^2+e_2^2\big)^{-\frac{1}{2}} = -\big(e_1^2+e_2^2\big)^{-\frac{3}{2}}\Big(e_1\frac{\mathrm{d}e_1}{\mathrm{d}p} + e_2\frac{\mathrm{d}e_2}{\mathrm{d}p}\Big).$$

In order to compute the derivative of $\boldsymbol{e}$, we have to differentiate the mapping $\boldsymbol{x}(\boldsymbol{\xi})$ with respect to p. To do so, let p be the x_2-component of some control point, i.e., $p = [\boldsymbol{P}_{\bar{A}}]_2$. Then, analogous as for the Jacobian in (4.23), we have

$$\frac{\mathrm{d}}{\mathrm{d}p}\big[\boldsymbol{x}(\boldsymbol{\xi})\big]_i = \frac{\mathrm{d}}{\mathrm{d}p}\left(\sum_{a=1}^{n_{bf\_el}}\hat{N}_a(\boldsymbol{\xi})\,[\boldsymbol{P}_a]_i\right) = \begin{cases}\hat{N}_a(\boldsymbol{\xi}) & \exists a \leftrightarrow A = \bar{A}, i = 2,\\ 0 & \text{otherwise.}\end{cases} \tag{4.28}$$

As above, we only receive a non-zero contribution $\frac{\mathrm{d}}{\mathrm{d}p}[\boldsymbol{K}^E]_{ab}$ from these elements where the basis function corresponding to the control point $\boldsymbol{P}_{\bar{A}}$ has support.

Remark 4.5. *Summing up, we only receive a non-zero contribution for the design derivative of the stiffness matrix $\frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}p}$ w.r.t. a single design variable from those elements where the corresponding basis function has support. Therefore, in the implementation we only loop through those elements.*

The second task of this subsection is the computation of the design sensitivity of the load vector. To do so, let us recall the definition of the entries of the load vector:

$$[\boldsymbol{f}]_A = \langle F_h, N_A\rangle = \int_\Omega JN_A\,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} g_N N_A\,\mathrm{d}s + \sum_{E\in\Gamma_D}\left[-\int_E \frac{1}{\mu}g_D\frac{\partial N_A}{\partial n}\,\mathrm{d}s + \frac{\sigma}{h_E}\int_E g_D N_A\,\mathrm{d}s\right]. \tag{4.29}$$

First of all, note that in our application we only consider homogeneous boundary conditions (Dirichlet and Neumann), since they represent symmetry and antisymmetry,

as indicated in Section 2.4.2. Hence, the boundary integrals in (4.29) vanish. In terms of the remaining first integral, keep in mind that the form of the coils, where the direct electric current is located, is independent of the shape of the pole heads. This implies that the impressed current is independent of the design variable. Summing up, we end up with

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} = 0. \tag{4.30}$$

4.3.4 State Derivative of the Cost Function

In order to apply the direct or adjoint method, we have to differentiate the cost function $\mathcal{I}_h(\boldsymbol{u}(\boldsymbol{p}))$ with respect to the vector of state variables $\boldsymbol{u}$ for a given design $\boldsymbol{p}$. First, let us recall the definition of the discrete cost function (4.1):

$$\mathcal{I}_h(\boldsymbol{B}_h^1, \boldsymbol{B}_h^2) = \frac{1}{2}\sum_{v=1}^{2}\Big[\varphi_h^v(\boldsymbol{B}_h^v) + \rho\theta_h^v(\boldsymbol{B}_h^v)\Big],$$

with

$$\varphi_h^v(\boldsymbol{B}_h^v) = \frac{1}{\mathrm{meas}(\Omega_m)(B_{min}^{avg,v})^2}\sum_{\Omega_e\in\Omega_m}\int_{\Omega_e}|\boldsymbol{B}_h^v(\boldsymbol{x}) - B_h^{avg,v}(\boldsymbol{B}_h^v)\boldsymbol{n}_m^v|^2\,\mathrm{d}\boldsymbol{x},$$

$$\theta_h^v(\boldsymbol{B}_h^v) = \Big[\max(0, B_{min}^{avg,v} - B_h^{avg,v}(\boldsymbol{B}_h^v))\Big]^2.$$

In the cost function we sum up the contribution of each of the two current excitation scenarios and, therefore, we can consider their derivative separately. In the subsequent derivation we focus on the vertical current excitation, whose solution is directly given by the state problem. In case of the diagonal excitation case the approach is analogous, except that the solution is built by superposition from the vertical solution and the reflected vertical solution, which both depend on the state variables $\boldsymbol{u}$. Therefore, in the computation of the state sensitivity contributions from both parts have to be taken into account.

If we restrict ourself to the vertical excitation scenario $v = 1$, we still have a sum of two terms, which are differentiated separately. Let us start with the differentiation of φ_h^v, where it is sufficient to consider one of the integrals that we sum up. To do so, let u be a single state variable; then we obtain by means of product rule

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}u}\int_{\Omega_e}|\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m|^2\,\mathrm{d}\boldsymbol{x}\\ &= \frac{\mathrm{d}}{\mathrm{d}u}\int_{\Omega_e}\big(\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\cdot\big(\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

$$= \int_{\Omega_e} 2\frac{\mathrm{d}}{\mathrm{d}u}\big(\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\cdot\big(\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\,\mathrm{d}\boldsymbol{x}$$
$$= \int_{\Omega_e} 2\big(\frac{\mathrm{d}}{\mathrm{d}u}\boldsymbol{B}_h(\boldsymbol{x}) - \frac{\mathrm{d}}{\mathrm{d}u}B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\cdot\big(\boldsymbol{B}_h(\boldsymbol{x}) - B_h^{avg}(\boldsymbol{B}_h)\boldsymbol{n}_m\big)\,\mathrm{d}\boldsymbol{x}.$$

In the next step we take a closer look at the involved derivatives. As we know, the state variables $\boldsymbol{u} = (u_1, \ldots, u_{n_{bf}})^T$ are the coefficients in the basis representation of the discrete solution u_h, see (3.18). In the following let us consider the state variable corresponding to a particular basis function $N_{\bar{A}}$, i.e., $u = u_{\bar{A}}$. By using the definition of the discrete magnetic flux density (4.5), we obtain for its derivative

$$\frac{\mathrm{d}}{\mathrm{d}u_{\bar{A}}}\boldsymbol{B}_h(\boldsymbol{x}) = \frac{\mathrm{d}}{\mathrm{d}u_{\bar{A}}}\Big(\frac{\partial u_h}{\partial x_2}(\boldsymbol{x}), -\frac{\partial u_h}{\partial x_1}(\boldsymbol{x})\Big)^T, \tag{4.31}$$

with

$$\frac{\mathrm{d}}{\mathrm{d}u_{\bar{A}}}\left(\frac{\partial u_h}{\partial x_i}(\boldsymbol{x})\right) = \frac{\mathrm{d}}{\mathrm{d}u_{\bar{A}}}\left(\sum_{a=1}^{n_{bf\_el}}\frac{\partial N_a}{\partial x_i}(\boldsymbol{x})u_a\right) = \begin{cases}\frac{\partial N_a}{\partial x_i}(\boldsymbol{x}) & \exists a \leftrightarrow A = \bar{A},\\ 0 & \text{otherwise.}\end{cases}$$

Moreover, the derivative of the average value B_h^{avg} of the magnetic flux density, defined as in (4.3), appears. The respective derivative is easy to compute:

$$\frac{\mathrm{d}}{\mathrm{d}u}B_h^{avg}(\boldsymbol{B}_h) = \frac{1}{\mathrm{meas}(\Omega_m)}\sum_{\Omega_e\in\Omega_m}\int_{\Omega_e}\frac{\mathrm{d}}{\mathrm{d}u}\boldsymbol{B}_h(\boldsymbol{x}).\boldsymbol{n}_m\,\mathrm{d}\boldsymbol{x}, \tag{4.32}$$

where the involved derivative of the magnetic flux density is given by (4.31). The penalty term in (4.1) consisting of a maximum is due to the square still differentiable, and the derivative is given by

$$\begin{cases}2(B_{min}^{avg} - B_h^{avg}(\boldsymbol{B}_h))(-\frac{\mathrm{d}}{\mathrm{d}u}B_h^{avg}(\boldsymbol{B}_h)) & 0 \le B_{min}^{avg} - B_h^{avg}(\boldsymbol{B}_h),\\ 0 & \text{otherwise.}\end{cases} \tag{4.33}$$

In the just stated expression we can compute the arising derivative of the average magnetic flux density as above.

Remark 4.6. *Note, the cost function consists of integrals, involving the magnetic flux density $\boldsymbol{B}_h$, over all four quarters of the magnetization area. Analogous as for the computation of the cost function, we have to take the respective sign of the derivative of u_h, arising from reflection, also for the derivative of the cost function into account.*

By means of the following remark we conclude the sensitivity analysis.

Remark 4.7. *As indicated in Section 3.4, we solve the state problem on a refined mesh in order to receive a high accuracy of the solution. However, we have defined the design variables as x_2-component of control points of the coarse mesh that describes the*

geometry, see Section 3.2.1. Therefore, in order to apply the just developed techniques for sensitivity analysis, we need a connection between control points of the coarse and the refined mesh, after knot insertion. The transfer matrix $\boldsymbol{T}$ *defined in Section 3.1.2 provides the required relation*

$$\mathcal{P} = \boldsymbol{T}\,\overline{\mathcal{P}},$$

where $\overline{\mathcal{P}}$ *are the control points of the coarse mesh and* $\mathcal{P}$ *are the control points of the refined mesh. Let us denote by* $\boldsymbol{p}$ *the* x_2*-components of all control points corresponding to the fine mesh and by* $\overline{\boldsymbol{p}}$ *our actual design variables, i.e., the* x_2*-components of particular control points of the coarse mesh. Then we obtain by means of chain rule*

$$\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\overline{\boldsymbol{p}}} = \frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{p}}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\overline{\boldsymbol{p}}} = \frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}\boldsymbol{p}}\tilde{\boldsymbol{T}}, \tag{4.34}$$

where the first term can by computed by means of the derived sensitivity analysis, and $\tilde{\boldsymbol{T}}$ *consists of the columns of* $\boldsymbol{T}$ *corresponding to our design variables.*

Concerning the validity of $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} = 0$ *for a fine control point, one should keep the following in mind. We only need the derivative* $\frac{\mathrm{d}\mathcal{I}_h}{\mathrm{d}p}$ *for those control points of the fine mesh which depend on our actual design variables, since otherwise the corresponding entry of the transfer matrix is zero. For those control points the assumption that the form of the coils is independent of the considered design variable (*x_2*-component of the control point) which provides* $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}p} = 0$ *is also fulfilled. This results from the fact that knot insertion preserves the parametrization of the geometry, i.e., also after refinement moving a fine control point that depends on a coarse control point of the pole head has no influence on the shape of the coils.*

In this context it is important to note that we did not use for the derivation of the design sensitivity of the stiffness matrix any special properties of the design variables, beside that they are x_2*-components of some given control points. Similar considerations as above, for the load vector, yield that the normal vector is still independent of the fine design variables, which we use in* (4.25).

In the previous subsections we derived an analytical method to calculate the gradient of the cost function. The MATLAB routine `fminunc` used for the optimization can by supplied with this analytically computed gradient. In the next section we present optimization results received by means of the MATLAB routine `fminunc` with the analytically computed gradient of the cost function derived in Section 4.3.2. Moreover, we compare the results obtained by using the analytically computed gradient of the cost function with results for the finite difference method, introduced in Section 4.3.1.

Chapter 5

Numerical Results

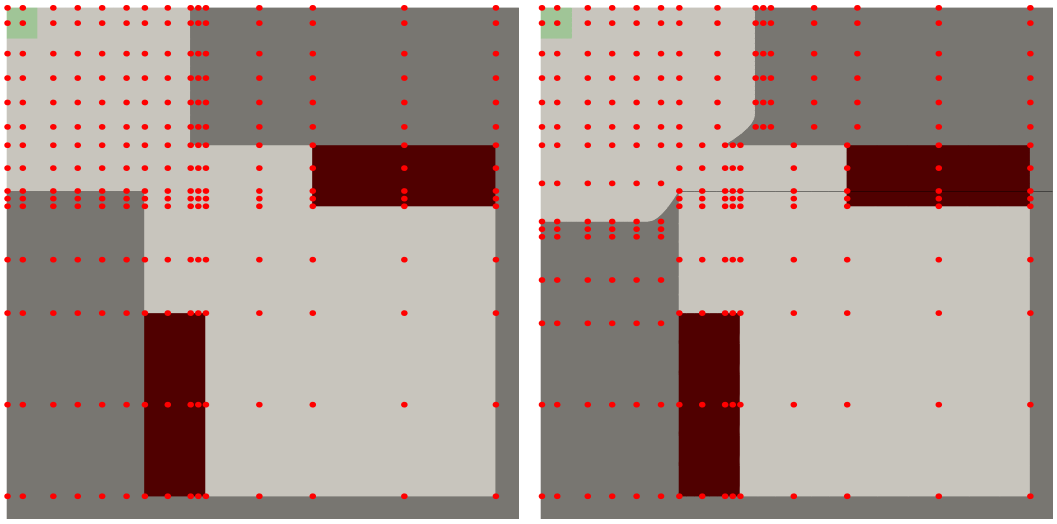
In this final chapter we present and discuss the obtained numerical results. We apply the optimization method described in Chapter 4 to our considered physical application, the shape optimization problem of electromagnets defined in Section 2.4.

Before we show the achieved results, let us give an overview of the considered settings.

- For the optimization we employ the MATLAB routine `fminunc` with the BFGS method, which has been discussed in Section 4.2. There exist two modes:
  - If `GradObj` is enabled, a user-defined gradient of the cost function is applied. In this case we supply the algorithm with the by means of the discrete method analytically computed gradient, see Section 4.3.2.
  - If `GradObj` is disabled, the MATLAB routine uses finite differences to approximate the gradient, see Section 4.3.1.
- As initial shape we consider the intuitive choice of a straight line, depicted in Figure 5.1a, and a lowered line, shown in Figure 5.1b.
- We consider the following two approaches for the choice of the design variables:
  - The design variables are defined by the x_2-component of the control points that describe the shape of the half south pole head in the coarse geometry representation (cf. Section 3.2). In our implementation we restrict ourself to two cases:
    * Six coarse design variables
    * Nine coarse design variables

    This first approach is the common one; additionally, we consider the following alternative choice.
  - The design variables are defined by the x_2-component of the control points that describe the shape of the half south pole head in the fine mesh, which we obtain from the coarse mesh (geometry description) by knot insertion. In other words we consider as many design variables as possible.

- The computations were performed on a Linux PC with the processor Intel Core i5-4200U (2 × 1,6 GHz) and 8 GB main storage.
- In the implementation we use for the geometry representation the G+SMO (Geometry + Simulation Modules) library, which is developed at the JKU in Linz. It provides necessary functionality, e.g., efficient evaluation of the B-spline basis functions and their derivatives in the parameter space and uniform refinement by means of knot insertion.



(a) Straight line (b) Lowered line

Figure 5.1: Considered initial pole head shapes for the case of six coarse design variables

Let us consider as first test scenario the following settings:

Settings 1

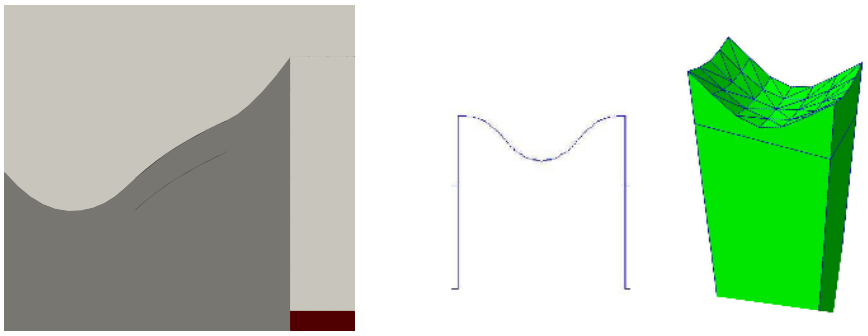
- We supply the MATLAB function with the analytically computed gradient.
- We set the lowered line as initial shape.
- We use six coarse design variables.

The first test confirms the decrease of the cost function during the optimization. For solving the state problem we use a refined mesh with 4761 DOF, which is obtained by inserting four new knots in each element. First of all, the optimization stops after three quasi-Newton iterations with the output that it cannot decrease the objective function along the current search direction. The value of the cost function in each step of the iteration is shown in Table 5.1. Keep in mind, we end up with a cost function value of $1.5947 \cdot 10^{-4}$. This represents an absolute decrease of the cost function of $= 0.7881 \cdot 10^{-4}$, which are $\approx 33\%$ of the initial value.

iteration	cost function value
0	$2.3828 \cdot 10^{-4}$
1	$1.6099 \cdot 10^{-4}$
2	$1.5947 \cdot 10^{-4}$
3	$1.5947 \cdot 10^{-4}$

Table 5.1: Decrease of the cost function.

The obtained optimized design is depicted in Figure 5.2a. Note, in case of this optimized design the lower bounds of the average magnetic flux density (see Section 2.4.3) are reached for both (vertical and diagonal) current excitation cases. From a practical point of view this seems quite realistic for an optimal design. For comparison, in Figure 5.2b the optimized 2D and 3D designs for a similar number of design variables computed by D. Lukáš in [13] are depicted. Note that especially the form of the obtained 3D design bears analogy to our received optimized pole head shape. In this context, one should keep in mind that we have not been able to exactly reproduce the problem settings considered in [13], since there have been some minor ambiguities. Therefore, we cannot consider the received cost function values for meaningful comparison.



(a) Optimized design for six design variables

(b) Optimized 2D and 3D designs computed by D. Lukáš in [13]

Figure 5.2: Obtained optimized pole head shapes

Figure 5.3 and Figure 5.4 show the magnetic field in case of the optimized design for the vertical and the diagonal current excitation respectively. In the right pictures in Figure 5.3 and Figure 5.4 we zoom in on the magnetization area and obtain a graphical confirmation that the magnetic field seems quite constant in the respective direction.

The next test treats the question, how the optimized design depends on the fineness of the discretization of the state problem. In the first row of Table 5.2 we see the obtained optimized design if we compute the solution of the state problem on the coarse mesh (cf. Section 3.2) that represents geometry with only 441 DOF. The received cost function value is $1.7577 \cdot 10^{-4}$. In the next step we solve the state problem of the optimization on a refined mesh, where two new knots are inserted in every element, which corresponds to 2025 DOF. Then we receive a considerable change

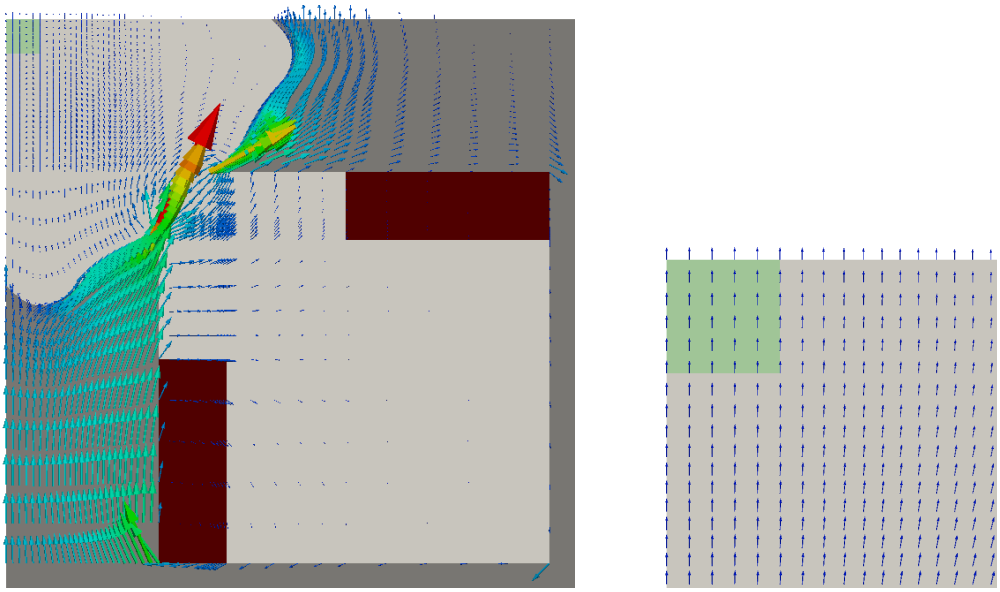


Figure 5.3: Magnetic field for the vertical current excitation in case of the optimized design

of the optimized design, and the change also becomes apparent in a reduction of the cost function value to $1.5979 \cdot 10^{-4}$. If we solve the state problem on a further refined mesh, where four or eight new knots are inserted, we no longer observe an appreciable change in the optimized design or the value of the cost function. Summing up, we can conclude a convergence behaviour of the optimized designs.

Table 5.3 provides further information about the optimization. Note that in all four cases the optimization takes only three quasi-Newton iterations. Moreover, also the total number of evaluations of cost function and gradient, stated in the fourth column, stays approximately constant. In the fifth column the number of conjugate gradient method (CG) iterations per evaluation of cost function and gradient is shown. In this context recall that we have to solve a linear system for the state problem and for the adjoint method. Here, we consider the number of iterations for both systems. Note, as solver we apply the conjugate gradient method with a simple Jacobi preconditioner. Figure 5.5 displays the number of CG iterations for different discretizations of the state problem; more precisely, the number of inserted knots runs from 0 (coarsest) to 8 (finest). In blue we see the needed CG iterations and in red the theoretical result as a function of the number of DOF, given by

$$\mathcal{O}\Big(\frac{1}{h}\Big) = \mathcal{O}\big((\#\mathrm{DOF})^{\frac{1}{2}}\big), \tag{5.1}$$

since the mesh size h is of order $\mathcal{O}\big((\#\mathrm{DOF})^{-\frac{1}{2}}\big)$ in the 2D case. For the validity of this theoretical result the considered mesh has to be quasi uniform. Hence, we observe that the increase of the needed CG iterations is as expected. The last row of Table 5.3 shows the total computation time. We see that also for the finest discretization with 13689 DOF the optimization only took less then a minute.

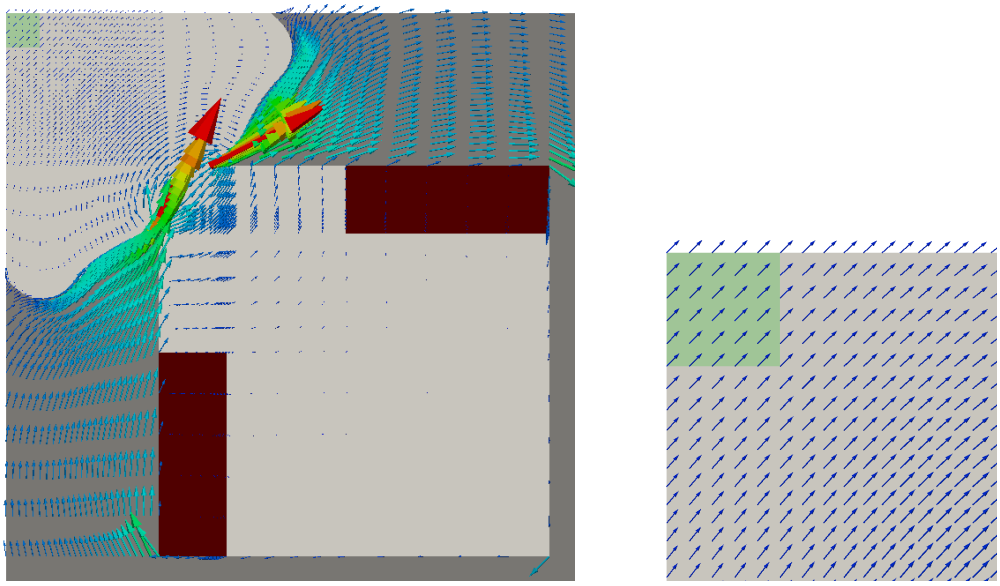


Figure 5.4: Magnetic field for the diagonal current excitation in case of the optimized design

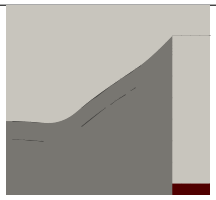
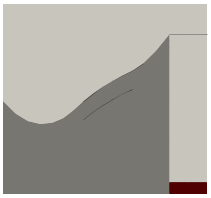
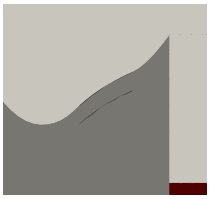
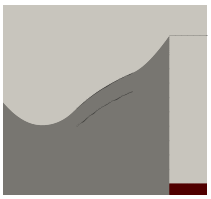
optimized design	number of inserted knots	DOF	cost function value
	0	441	$1.7577 \cdot 10^{-4}$
	2	2025	$1.5979 \cdot 10^{-4}$
	4	4761	$1.5947 \cdot 10^{-4}$
	8	13689	$1.5937 \cdot 10^{-4}$

Table 5.2: Optimized design for different fine discretizations of the state problem

number of inserted knots	DOF	optimization iter.	number of eval. cost funct., gradient	CG iter.	total time
0	441	3	27	251	2.38s
2	2025	3	29	299	4.54s
4	4761	3	31	446	12.59s
8	13689	3	28	760	56.59s

Table 5.3: Further information about the optimization. The number of CG iterations in the fifth column is the averaged number of iterations per one evaluation of cost function and gradient.

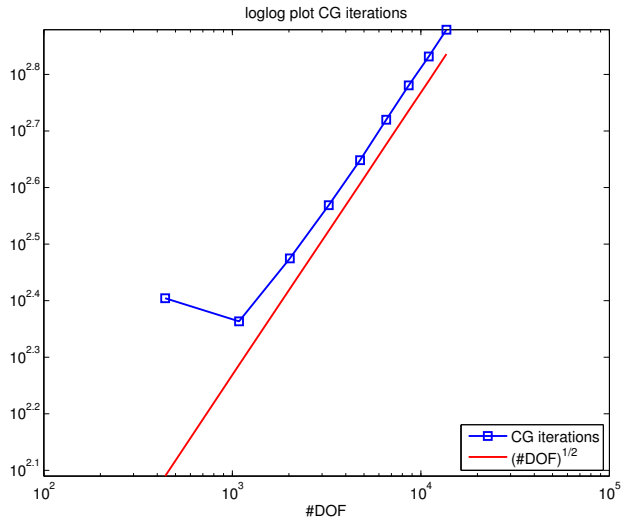


Figure 5.5: CG iterations for different fine discretizations of the state problem. The number of inserted knots runs from 0 (coarsest) to 8 (finest).

Remark 5.1. *In the current model we do not incorporate constraints to maintain physical meaningful domains, e.g., avoid overlapping of the elements in the mesh (we only reduce the risk by adjusting adjacent control points, see Remark 3.9) and guarantee that the pole head stays in the computational domain. For ideas how to avoid overlapping of the elements we refer the reader to [20] and [14]. The testing has shown that even without additional constraints we obtain meaningful results.*

Now, the question arises, if we also receive similar results if we do not provide the analytically computed gradient. Note that using finite differences would simplify the sensitivity analysis significantly. Let us consider the following settings:

Settings 2 (finite differences)

- We let the algorithm approximate the gradient by means of finite differences.
- We set the lowered line as initial shape (as in settings 1).

- We use six coarse design variables (as in settings 1).

The experiments show that often no improvement of the cost function can be achieved, i.e., the optimization stops with a cost function value that is approximately equal to the initial one. In cases where a change occurs, we end up with wiggly unrealistic shapes. Note that we receive similar behaviour for the straight line as initial shape. So we can conclude that in case of the used gradient-based quasi-Newton algorithm (BFGS method) the analytical computation of the gradient is necessary in order to receive meaningful results.

Moreover, we also tried to use the intuitive choice of a straight line as initial shape. To do so, we apply the following settings:

Settings 3 (straight line as initial shape)

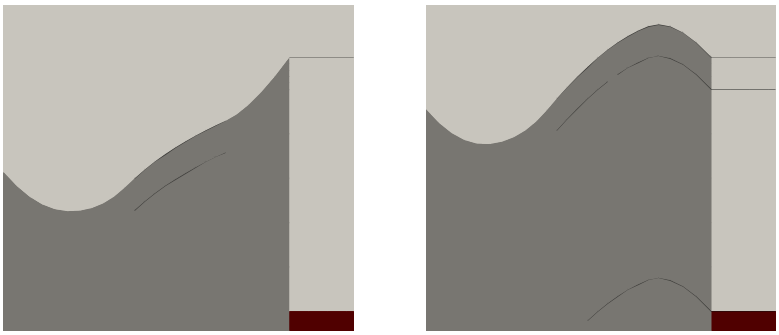
- We supply the MATLAB function with the analytically computed gradient (as in settings 1).
- We set the straight line as initial shape.
- We use six coarse design variables (as in settings 1).

As before, we compute the solution of the state problem on a refined mesh with 4761 DOF, where four new knots are inserted in every element. With this settings the optimization already stops after two iterations, again, with the output that the cost function value cannot be decreased in the current search direction. As stated in Table 5.4, we end up with a cost function value of $3.8499 \cdot 10^{-4}$, which is considerably larger than the received cost function value for the lowered line as initial shape, given by $1.5947 \cdot 10^{-4}$.

iteration	cost function value
0	$5.3791 \cdot 10^{-4}$
1	$3.8499 \cdot 10^{-4}$
2	$3.8499 \cdot 10^{-4}$

Table 5.4: Decrease of the cost function

Moreover, also the form of the received optimized design is observably different, as one can see by comparing the optimized designs for the lowered line (Figure 5.6a) and for the straight line (Figure 5.6b). This findings indicate that the considered cost function has many local minima, and in case of the straight line we end up with a different one as for the lowered line. In this context, we should mention that in case of the optimized design for the straight line (Figure 5.6b) the lower bounds for the average magnetic flux density (see Section 2.4.3) are not reached, in contrast to the optimized design for the lowered line. This observation gives rise to consider the design achieved for the lowered line is a reasonable result. Summing up, we have to argue



(a) Lowered line as initial shape

(b) Straight line as initial shape

Figure 5.6: Obtained optimized pole head shapes

from this results a quite strong dependence of the optimized design on the initial pole head shape.

In order to enable higher flexibility of the designs, we considered as a next step more design variables and carried out experiments with the following settings:

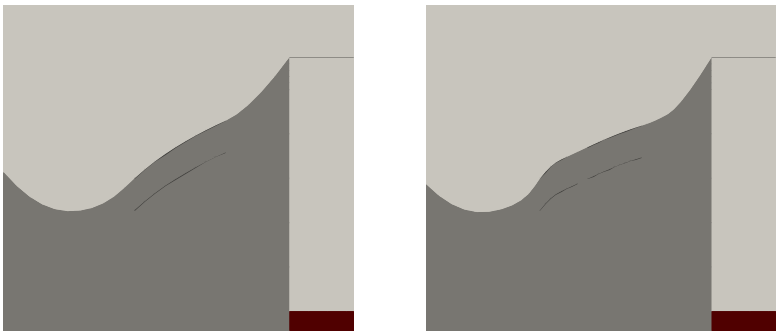
Settings 4 (nine coarse design variables)

- We supply the MATLAB function with the analytically computed gradient (as in settings 1).
- We set the lowered line as initial shape (settings 4a).
- We set the straight line as initial shape (settings 4b).
- We use nine coarse design variables.

Generally speaking, the results are comparable to the results for six design variables. In the following we take a closer look at the test case, where we compute the solution of the state problem on a refined mesh with 7056 DOF, which corresponds to four new inserted knots in every element.

Let us first consider the lowered line as initial shape (settings 4a). By comparing the received optimized design for six design variables (Figure 5.7a) with the optimized design in case of nine design variables (Figure 5.7b), we observe that the basic form of the pole head is the same. The similarity is reflected in the received cost function value, which is $1.5968 \cdot 10^{-4}$ in the case of nine design variables and, as stated before, $1.5947 \cdot 10^{-4}$ for six design variables. Note, as in the case of six design variables, the lower bounds for the average magnetic flux density are reached for the optimized design.

If we use the straight line (settings 4b) as initial shape of the pole head, we observe the same behaviour as in the case of six design variables. The basic form of the



(a) Six design variables (b) Nine design variables

Figure 5.7: Obtained optimized pole head shapes

received design changes in a similar way as before and also the received cost function value increases again considerably.

From the just described test case with nine control points one could conclude that the number of design variables has no significant influence on the basic form of the obtained design. However, this is not completely valid as the next test scenario shows.

Settings 5 (fine design variables)

- We supply the MATLAB function with the analytically computed gradient (as in settings 1).
- We set the lowered line as initial shape (as in settings 1).
- We use all fine control points of the pole head as design variables.

Let us consider the following situation: We compute the solution of the state problem on a refined mesh with 13689 DOF, which corresponds to eight new inserted knots in every element. If we consider the x_2-component of all fine control points of the pole head as design variables, we end up with 46 design variables. Figure 5.8 shows the received optimal design, with a cost function value of $1.5500 \cdot 10^{-4}$. If we compare the

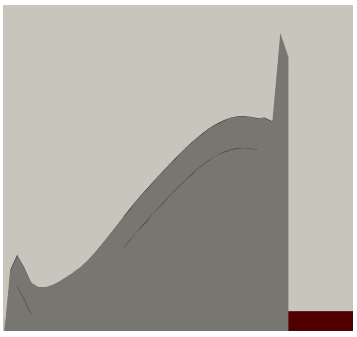


Figure 5.8: Optimized pole head shape for 46 design variables

cost function value with the one we obtain for the same discretization (eight inserted knots) with only six coarse design variables, given by $1.5937 \cdot 10^{-4}$, we observe a slight improvement. However, on the other hand the obtained design seems wiggly and unrealistic. One explanation for the improved cost function value is that we reach a different local minimum of the cost function. Another one is that we already observe the indication of ill-posedness, since it is known that shape optimization problems with a huge number of design variables are ill-posed.

Tests show that even with all fine control points as design variables we still observe a quite strong dependence of the optimized design on the initial shape of the pole head.

Chapter 6

Conclusion and Possible Further Work

In this thesis we considered shape optimization based on isogeometric analysis. We followed the concept "first discretize, then optimize" and used for the optimization a gradient-based quasi-Newton method. We performed an analytical sensitivity analysis of the discrete problem for a B-spline discretization. All the techniques were applied to a physical application consisting of a shape optimization problem of electromagnets.

In Chapter 5 we presented and discussed the obtained results for different settings. It became apparent that it is not sufficient to simply use finite differences to approximate the gradient needed for the BFGS method. Moreover, we observed a quite strong dependence of the received optimized design on the initial shape of the pole head. Even for high number of design variables this dependence occurred.

This work can be continued in the following directions:

- *Design constraints:*
  In order to maintain physical meaningful domains, we have to prevent overlapping of the elements in the mesh and guarantee that the pole head stays inside the computational domain. In this thesis, for simplicity, we did not incorporate any design constraints in our model. For more reliable results proper constraints should be included.

- *Weights of control points as design variables:*
  In Section 3.2 we described the geometry by means of a B-Spline mapping. If we use instead of B-splines NURBS that are defined by control points and control weights, then we could use the weights as additional design variables. One the one hand this enriches our design space of representable shapes, but on the other hand the derivation of analytical design sensitivities becomes more difficult.

- *Different optimization methods:*
  It would be interesting to see how gradient-free methods work for the considered problem. We saw in Chapter 5 that simply using finite differences to approximate the gradient needed for the BFGS method does not work. If we want to avoid an analytical sensitivity analysis, e.g., if we also use control weights as design variables, gradient-free methods could be a reasonable alternative.

- *Fine control points as design variables:*
  In Chapter 5 we observed a strong dependence of the optimized design on the initial shape. A possible way to overcome this problem is to compute in a first step the optimized design for a small number of design variables. Then step-by-step the number of design variables for the optimization is increased, where the obtained optimized design of the previous step is used as initial shape. Note that the state problem is always solved on a sufficiently fine mesh. It would be interesting to take a closer look at this approach. When considering really high numbers of design variables it is probably beneficial to apply in a post-processing step smoothing filters to obtain physical meaningful designs.

Bibliography

[1] D. Braess. *Finite Elemente: Theorie, Schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer, 2013.

[2] V. Braibant and C. Fleury. Shape optimal design using B-splines. *Computer Methods in Applied Mechanics and Engineering*, 44(3):247 – 267, 1984.

[3] S.C. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*. Texts in Applied Mathematics. Springer, 2008.

[4] K.K. Choi and N.H. Kim. *Structural Sensitivity Analysis and Optimization 1: Linear Systems*. Mechanical Engineering Series. Springer, 2006.

[5] P.W. Christensen and A. Klarbring. *An Introduction to Structural Optimization*. Solid Mechanics and its Applications. Springer, 2009.

[6] P. Gangl. Topology Optimization in Electrical Engineering. Master's thesis, Johannes Kepler University Linz, 2004. `http://www.numa.uni-linz.ac.at/Teaching/Diplom/Finished/gangl-dipl.pdf`.

[7] V. Girault and P.A. Raviart. *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*. Springer, 1986.

[8] R.T. Haftka and Z. Gürdal. *Elements of Structural Optimization*. Springer, 1992.

[9] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[10] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009.

[11] J. D. Jackson. *Classical Electrodynamics*. Wiley, 1999.

[12] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. *Mathematics of Computation*, 78(267):1353–1374, 2009.

[13] D. Lukáš. *Optimal Shape Design in Magnetostatics*. PhD thesis, Technical University of Ostrava, 2003.

[14] A. P. Nagy, M. M. Abdalla, and Z. Gürdal. Isogeometric sizing and shape optimisation of beam structures. *Computer Methods in Applied Mechanics and Engineering*, 199(17–20):1216 – 1230, 2010.

[15] M. Neumüller. Discontinuous Galerkin Methods in Space and Time. Lecture Notes, 2014.

[16] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[17] C. Pechstein. Multigird-Newton-Methods For Nonlinear Magnetostatic Problems. Master's thesis, Johannes Kepler University Linz, 2004. `http://www.numa.uni-linz.ac.at/Teaching/Diplom/Finished/pechstein_dipl.pdf`.

[18] C. Pechstein. Numerische Methoden der Elektrotechnik. Lecture Notes, 2012.

[19] L. Piegl and W. Tiller. *The NURBS Book*. Monographs in Visual Communication. Springer, 1997.

[20] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197:2976–2988, 2008.

[21] Q. Xiaoping. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199(29–32):2059–2071, 2010.

Eidesstattliche Erklärung

Ich, Katharina Rafetseder, erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.
Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, September 2014

_____

Katharina Rafetseder