Doctoral Thesis

Numerical Shape Optimization with Finite Elements

Author(s):
Paganini, Alberto

Publication Date:
2016

Permanent Link:

https://doi.org/10.3929/ethz-a-010579469 →

DISS. ETH No. 23212

NUMERICAL SHAPE OPTIMIZATION WITH FINITE ELEMENTS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

ALBERTO PAGANINI

M.Sc. ETH in Mathematics

born on 18.06.1987
citizen of St. Gallen SG, Switzerland

accepted on the recommendation of
Prof. Dr. Ralf Hiptmair, ETH Zürich, examiner
Prof. Dr. Christian Hafner, ETH Zürich, co-examiner
Prof. Dr. Boris Vexler, TU München, co-examiner

2016

Abstract

This thesis is devoted to numerical aspects of PDE-constrained shape optimization in the framework of Galerkin finite element (FE) discretization of the underlying boundary value problem (BVP). The formulation of a shape optimization problem comprises the description of a set of admissible shapes and the definition of a shape functional. An admissible shape is said to be optimal if it minimizes (or maximizes) the shape functional. Most shape functionals considered in literature are shape differentiable. In this case, optimal shapes are critical points, that is, shapes for which the shape gradient is zero.

For shape functionals constrained by BVPs, the shape gradient also depends on the solution of the underlying BVP and, usually, on the solution of an adjoint problem. Shape gradients can be approximated by replacing these functions with numerical solutions. We prove superconvergence in the approximation of shape gradients when the underlying BVP is elliptic and is discretized by means of Galerkin FEs.

Next, we develop a shape optimization algorithm tailored to preserve and exploit the approximation properties of the FE method, and that allows for arbitrarily high resolution of shapes. Optimization is carried out by constructing a deformation diffeomorphism that is based on B-splines and is updated with the help of H^1-representatives of shape gradients. We provide numerical evidence of the performance of this method both on prototypical well-posed and ill-posed shape optimization problems. For the latter case, we discuss regularization based on wavelets.

Finally, we apply the shape optimization algorithm to optimize the shape of the cross section of cylindrical microlenses. Microlenses are highly attractive for optical applications such as super resolution and photonic nanojets, but their design is demanding because resonance effects play an important role, which forces one to perform a full wave analysis. In literature, mostly spherical microlenses were studied. Starting from reasonable elliptical and semi-circular shapes, we show that strong increases of the performance of the lenses may be obtained for any physically meaningful value of the refraction index.

Riassunto

Questa tesi tratta aspetti numerici di problemi di ottimizzazione di forma con vincoli espressi da equazioni alle derivate parziali (PDE), discretizzate mediante elementi finiti (FE). Per formulare un problema di ottimizzazione di forma è necessario introdurre un insieme di forme ammissibili che funge da dominio di un funzionale di forma. Una forma ammissibile è detta ottimale se minimizza (o massimizza) il funzionale di forma. La maggior parte dei funzionali di forma trattati nella letteratura è differenziabile. In questo caso, le forme ottimali sono punti critici, ovvero, sono zeri del gradiente di forma.

Il gradiente di forma di un funzionale di forma vincolato ad una PDE dipende anche dalla soluzione di quest'ultima oltre che, in generale, dalla soluzione di un problema aggiunto. Il gradiente di forma può essere approssimato sostituendo queste soluzioni con approssimazioni numeriche delle stesse. Dimostriamo che questa approssimazione è superconvergente quando la PDE vincolante è ellittica e viene discretizzata con FE.

Successivamente, sviluppiamo un algoritmo di ottimizzazione di forma che mira a preservare e sfruttare la proprietà di approssimazione dei FE e che, allo stesso tempo, permette risoluzione arbitrariamente accurata delle forme. Questo algoritmo utilizza il rappresentante in H^1 del gradiente di forma per costruire un diffeomorfismo di deformazione basato su B-splines. Forniamo prove numeriche delle prestazioni di questo algoritmo su problemi standard di ottimizzazione di forma ben e mal posti. Per quest'ultimo caso proponiamo una regolarizzazione basata su wavelets.

Infine, l'ottimizzazione della forma della sezione di microlenti cilindriche fornisce un'interessante applicazione realistica per testare il nostro algoritmo. Infatti le microlenti vengono studiate in applicazioni quali la super risoluzione e i nanojet fotonici, ma il loro design è complicato a causa di fenomeni di risonanza non trascurabili. Nella letteratura sono state studiate principalmente microlenti sferiche. Noi consideriamo design ellittici o semi-circolari e dimostriamo che le prestazioni della lente possono essere migliorate per ogni valore "ragionevole" dell'indice di rifrazione.

Acknowledgments

I would like to thank Ralf Hiptmair and Christian Hafner for having given me the opportunity to pursue my Ph.D. studies[1]. I have received excellent guidance and support, and I am deeply grateful for all they have taught me. In addition, I would like to praise the work of all SAM professors and secretaries, who lead this research group so well, as well as the D-MATH IT Support Group, which was of crucial assistance.
I had a great time at SAM, and I am thankful to the whole group for that. I had the honor and pleasure to share my office with Laura, Pegah, and Sahar, who created a familial atmosphere at work and provided unique help during these years. I would also like to thank Cecilia, who virtually belongs to this office family, all SAM-chat authors, who made correction periods so hilarious, Andreas, for all his coding wisdom, r., for no one can be more concise, Elke and Franziska, for all the wonderful jogging sessions and superkondis, Roger, for his great approach to scientific life, Paolo and Cedric, who helped me so much when I moved my first steps in SAM.
Living in Zurich could have never been this enjoyable without my friends. In particular, I would like to mention Claudio, Elisa, Lorenzo, and Simone, with whom I started my adventure in the university dungeon. Additionally, I want to thank all players of K.Toy.A, for this is by large the best floorball team!
Finally, I owe my family a debt of gratitude for all they did for me. Without them, I could have never succeeded.

[1]This research was partly supported by ETH Grant CH1-02 11-1

Contents

1. Introduction

"La filosofia è scritta in questo grandissimo libro che continuamente ci sta aperto innanzi a gli occhi (io dico l'universo), ma non si può intendere se prima non s'impara a intender la lingua, e conoscer i caratteri, ne' quali è scritto. Egli è scritto in lingua matematica, e i caratteri son triangoli, cerchi, ed altre figure geometriche, senza i quali mezi è impossibile a intenderne umanamente parola; senza questi è un aggirarsi vanamente per un oscuro laberinto."

Galileo Galilei, *Il Saggiatore*, Chapter VI.

Galileo Galilei taught us that the universe is written in the language of mathematics and that learning this language is crucial to understanding this "grand book" [33, Excerpts from *The Assayer*, pp. 237-238]. Galilei's words have shaped the modern scientific approach. Most specifically, we believe that physical phenomena can be described by mathematical models, which can be seen as black boxes that link input and output quantities.

In this thesis, we consider mathematical models where the input, also called *control*, consists of the shape of an object, and a *shape functional* models the dependence of the output on the control. In particular, we zero in on *shape optimization*. Roughly speaking, the goal is to find the shape that achieves a specific target output. Such a shape is usually named *optimal*.

In order to perform shape optimization, it is necessary to give a structure to the *space of shapes*. This non-trivial step was solved (probably for the first time) in 1907 by J. Hadamard [36], who "used displacements along the normal to the boundary Γ of a C^∞-domain" ... "to compute the derivative of the first eigenvalue of a clamped plate" [24, Ch.9, Rmk. 3.2]. This pioneering work gave birth to the fascinating field of research known as *shape calculus*. Numerous authors have contributed to its development. Today, several books provide introductions to this topic and summarize the bulk of knowledge [3, 24, 41, 73].

In several applications, the control is the shape of a domain in which a boundary value problem (BVP) is stated, and the shape functional depends also on the solution of this BVP. In this case, we say that the shape opti-

mization problem is constrained to partial differential equations (PDEs). Often, it is possible to show the existence and uniqueness of the solution of a PDE, but it is rare that this solution can be found explicitly. Additionally, the dependence of this solution on the shape of the domain is highly nonlinear. Therefore, PDE-constrained shape optimization problems are typically difficult to solve. Usually, one has to settle for approximate optimal shapes computed with iterative optimization algorithms. And these have to be combined with numerical methods to approximate the solution of the underlying BVP. Clearly, the choice of the numerical method has a great impact on the quality of the approximate optimal shapes retrieved.

The goal of this thesis is to investigate the numerical aspects of PDE-constrained shape optimization when the solution of the PDE constraint is approximated with the finite element method (FEM).

In Chapter 2, we study the approximation of the *shape gradient*, which is the Eulerian derivative of shape functionals with respect to perturbations of the shape. This derivative is the starting point for shape optimization. A key theorem in shape calculus states that most shape gradients can be equivalently written as a boundary or a volume integral [24, Ch. 9, Thm 3.6]. Many authors have suggested that the volume-based formulation is better suited when the solution of the BVP is approximated with FEM; cf. [13], [24, Ch. 10, Rmk. 2.3], and [40, Ch. 3.3.7]. We prove and provide numerical evidence that this is indeed the case when the shape functional is constrained to an elliptic PDE.

In Chapter 3, we develop an algorithm to perform shape optimization. We describe its mathematical derivation and assess its performance in several numerical experiments. The literature certainly abounds with shape optimization algorithms. However, it is often not clear what the impact of finite element discretizations is. Our mathematical interpretation of shape optimization problems has been partly inspired by [29] and is similar to that presented in [50] and in [31]. The key idea is to employ transformation techniques to better formulate the dependence of the solution of the BVP on the control. Then, we recast the shape optimization problem as an optimal control problem whose control is a vector field that lives in an infinite-dimensional space. We show that it is possible to switch to finite-dimensional problem by employing a conforming Ritz discretization of the control. The resulting finite-dimensional problem is a nonlinear (and possibly nonconvex) optimization problem subject to PDE constraints. To tackle it, we employ a steepest descent algorithm based on shape gradients. Our results from Chapter 2 guarantee that shape

gradients can be accurately approximated by replacing the solution of the BVP with finite element approximations.

In Chapter 4, we employ the algorithm developed in Chapter 3 to perform shape optimization of microlenses. In a nutshell, a lens is an optical device the focuses incident light. The size of a microlens is comparable with the wavelength of light, and resonances inside the microlens have significant impact on the focus. As a consequence, it is difficult to find the shape of microlens that focuses at a specific location. We show that shape optimization provides a valid tool to improve the design of microlenses.

We end this thesis with Chapter 5, where we draw conclusions and present some suggestions for further directions of research. In Appendix A, we provide a complete ready-to-run MATLAB code of the algorithm presented in Chapter 3. This implementation includes a fully vectorized implementation of linear Lagrangian FEM inspired by [32].

2. Approximate shape gradients for elliptic state constraints

The key theorem of shape calculus, the Hadamard-Zolésio structure theorem, states that, under reasonable assumptions, shape gradients depend only on perturbations of the boundary $\partial\Omega$ in the normal direction. As a consequence, most shape gradients can be formulated as an integration both in the volume Ω and on the boundary $\partial\Omega$, and these formulations are equivalent.

In several applications, the shape functional $\mathcal{J}(\Omega)$ depends not only on Ω, but also on the solution u of a boundary value problem (BVP) stated on Ω. Clearly, the shape gradient of a so-called PDE-constrained shape functionals depends on u, too. Usually, exact solutions of BVP are not available, and to evaluate the shape gradient the function u has to be replaced with a numerical approximation u_h. Then, the equivalence of the volume and the boundary formulations of $d\mathcal{J}(\Omega;\mathcal{V})$ inevitably breaks down, and it is not clear which of the two is better suited for an accurate evaluation of $d\mathcal{J}(\Omega;\mathcal{V})$. In particular, we consider this issue in the context of discretizations by means of finite elements, as this is the most popular choice in shape optimization.

In this chapter, we provide theoretical and numerical evidence that the volume formulation is better suited for approximations of shape gradients. In Section 2.1 we recall the basics of shape calculus and derive the boundary and the volume formulation of a prototypical elliptic PDE-constrained shape functional. In Section 2.2 we provide a-priori convergence estimates for shape gradient approximations. In particular, in Theorem 2.2.4 we prove that superconvergence in the finite element Galerkin approximation of $d\mathcal{J}(\Omega;\mathcal{V})$ can be achieved when the volume formulation is employed. The key feature that allows to improve convergence rates is that volume formulations are continuous in energy norm, and standard duality techniques can be applied. On the other hand, boundary integrals involve traces that are not well-defined on the natural variational space, so that basic convergence rates may not be improved. In Section 2.3, we provide numerical evidence of the superiority of the volume formulation of $d\mathcal{J}(\Omega;\mathcal{V})$ over its boundary formulation. Finally, Section 2.4

extends our consideration to shape functionals constrained to elliptic interface problems with discontinuous coefficients.

The work presented in this Section is largely based on [45] and on [64], and has been partly developed in collaboration with Prof. Dr. Ralf Hiptmair and Sahar Sargheini.

2.1. Shape gradients

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be an open bounded domain with piecewise smooth boundary $\partial\Omega$, and let $\mathcal{J}(\Omega) \in \mathbb{R}$ be a real-valued quantity of interest associated to it. One is often interested in its shape sensitivity, which quantifies the impact of small perturbations of $\partial\Omega$ on the value $\mathcal{J}(\Omega)$.

For this purpose, we model perturbations of the domain Ω through maps of the form

$$T_{\mathcal{V}}(\mathbf{x}) \coloneqq \mathbf{x} + \mathcal{V}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{2.1}$$

where $\mathcal{V}$ is a vector field in $C^1(\mathbb{R}^d; \mathbb{R}^d)$. It can easily be proved that the map (2.1) is a diffeomorphism for $\|\mathcal{V}\|_{C^1} < 1$ [3, Lemma 6.13]. Therefore, it is natural to consider $\mathcal{J}(\Omega)$ as the realization of a shape functional, a real map

$$\mathcal{J} : \mathcal{U}_{\text{ad}} \to \mathbb{R}$$

defined on the family of admissible domains

$$\mathcal{U}_{\text{ad}} \coloneqq \left\{ T_{\mathcal{V}}(\Omega) \,;\, \mathcal{V} \in C^1(\mathbb{R}^d; \mathbb{R}^d) \,, \|\mathcal{V}\|_{C^1} < 1 \right\}. \tag{2.2}$$

The sensitivity of $\mathcal{J}(\Omega)$ with respect to the perturbation direction $\mathcal{V}$ can be expressed through the *Eulerian derivative* of the shape functional $\mathcal{J}$ in the direction $\mathcal{V}$, that is,

$$d\mathcal{J}(\Omega; \mathcal{V}) \coloneqq \lim_{s \searrow 0} \frac{\mathcal{J}\left(T_{s\cdot\mathcal{V}}(\Omega)\right) - \mathcal{J}(\Omega)}{s}. \tag{2.3}$$

It goes without saying that it is desirable that (2.3) exists for all possible perturbation directions $\mathcal{V}$. It is therefore natural to define a shape functional $\mathcal{J}$ to be *shape differentiable* at Ω if the mapping

$$d\mathcal{J}(\Omega; \cdot) : C^1(\mathbb{R}^d; \mathbb{R}^d) \to \mathbb{R}, \qquad \mathcal{V} \mapsto d\mathcal{J}(\Omega; \mathcal{V}). \tag{2.4}$$

defined by (2.3) is linear and bounded on $C^1(\mathbb{R}^d;\mathbb{R}^d)$. In literature, the mapping $\mathcal{V} \mapsto d\mathcal{J}(\Omega;\mathcal{V})$ is called *shape gradient* of $\mathcal{J}$ at Ω, as it is the Gâteaux derivative in $0 \in C^1(\mathbb{R}^d;\mathbb{R}^d)$ of the map

$$\mathcal{V} \mapsto \mathcal{J}\left(T_{\mathcal{V}}(\Omega)\right) ,$$

see [24, Ch. 9, Def. 2.2]. Note that Formula (2.3) is well-defined for any vector field in the Banach space $C^1(\mathbb{R}^d;\mathbb{R}^d)$, and the shape gradient is an element of its dual space.

Remark 2.1.1. *In literature, perturbations as in* (2.1) *are known as* perturbations of the identity*. From a differential geometry point of view, this approach is less general than the so called* velocity method*, which is, for instance, introduced in [24, Ch. 4]. However, both methods lead to the same formula for the shape gradient, which merely takes into account first order perturbations of the shape functional $\mathcal{J}$ [24, Ch. 9, Thm 3.2].*

An interesting property of shape gradients is expressed in the *Hadamard structure theorem* [24, Ch. 9, Thm 3.6]: If $\partial\Omega$ is smooth, $d\mathcal{J}(\Omega;\cdot)$ admits a representative $\mathfrak{g}(\Omega)$ in the space of distributions $\mathcal{D}^k(\partial\Omega)$

$$d\mathcal{J}(\Omega;\mathcal{V}) = \langle \mathfrak{g}(\Omega), \gamma_{\partial\Omega}\mathcal{V}\cdot\mathbf{n}\rangle_{\mathcal{D}^k(\partial\Omega)} , \tag{2.5}$$

where $\gamma_{\partial\Omega}\mathcal{V}\cdot\mathbf{n}$ is the normal component of $\mathcal{V}$ on the boundary $\partial\Omega$. This implies that only normal displacements of the boundary have an impact on the value of $\mathcal{J}(\Omega)$. However, we should take into account that this is no longer true if the boundary $\partial\Omega$ is only piecewise smooth.

We are particularly interested in PDE-constrained shape functionals of the form

$$\mathcal{J}(\Omega) = \int_\Omega j(u)\,\mathrm{d}\boldsymbol{x} , \tag{2.6}$$

where $j:\mathbb{R}\to\mathbb{R}$ possesses a locally Lipschitz continuous derivative j' and u is the solution of the *state problem*, a boundary value problem stated on Ω.

Clearly, the formula of the Eulerian derivative of a PDE-constrained shape functional depends on the operators involved in the state constraint. Henceforth, we restrict ourselves to scalar elliptic equations with Neumann or Dirichlet boundary conditions. In particular, we consider the test case

$$\left\{\begin{array}{rcll} -\Delta u + u &=& f & \text{in } \Omega , \\ u &=& g \text{ or } \frac{\partial u}{\partial \mathbf{n}} = g & \text{on } \partial\Omega . \end{array}\right. \tag{2.7}$$

The functions f and g are assumed to belong to $L^2(\mathbb{R}^d)$ ($H^1(\mathbb{R}^d)$ in the case of the Neumann BVP) and $H^2(\mathbb{R}^d)$, respectively, and they are identified with their restrictions onto Ω and $\partial\Omega$.

The state problem (2.7) should be interpreted in weak sense. In the case of Neumann boundary conditions, we say that $u \in H^1(\Omega)$ is the solution of (2.7) if

$$\int_\Omega \nabla u \cdot \nabla v + uv \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} gv \,\mathrm{d}S = \int_\Omega fv \,\mathrm{d}\boldsymbol{x} \quad \forall\, v \in H^1(\Omega)\,.$$

On the other hand, we say that $u \in H^1(\Omega)$ is the solution of (2.7) with Dirichlet boundary conditions if the function $\tilde{u} \in H^1_0(\Omega)$ defined as $\tilde{u} \coloneqq u - g$ satisfies

$$\int_\Omega \nabla \tilde{u} \cdot \nabla v + \tilde{u}v \,\mathrm{d}\boldsymbol{x} = \int_\Omega fv - \nabla g \cdot \nabla v - gv \,\mathrm{d}\boldsymbol{x} \quad \forall\, v \in H^1_0(\Omega)\,. \tag{2.8}$$

Explicit formulas for $d\mathcal{J}(\Omega)$ can be derived both for unconstrained and PDE-constrained shape functionals, cf. [24, Ch. 9, Sect. 4.3, and Ch. 10, Sect. 2.5]. In the case of PDE-constrained shape functionals, the formulas involve the integrals of u, the solution of (2.7), and of p, the weak solution of the adjoint problem

$$\begin{cases} -\Delta p + p &= j'(u) & \text{in } \Omega\,, \\ p &= 0 \text{ or } \frac{\partial p}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega\,. \end{cases} \tag{2.9}$$

Note that, by the Hölder Inequality and the Sobolev Embedding Theorem,

$$\int_\Omega j'(u)v \,\mathrm{d}\boldsymbol{x} \le \|j'(u)\|_{L^{6/5}(\Omega)} \|v\|_{L^6(\Omega)} < \infty \quad \forall v \in H^1(\Omega)\,.$$

This implies that we have to further assume that $j'(u) \in L^{6/5}(\Omega)$ for the adjoint problem (2.9) to be well-defined. If $u \in H^2(\Omega)$, this condition is satisfied because $H^2(\Omega) \subset L^\infty(\Omega)$ and j' is locally Lipschitz. Otherwise, it is sufficient to assume that j' satisfies the following polynomial growth condition

$$|j'(x)| \le C|x|^p \quad \forall x \in \mathbb{R}$$

with $p \le 5$. In the two dimensional case, any $p < 6$ is also allowed.

In the next Proposition, we give explicit formulas for the Eulerian derivative of the shape functional (2.6) constrained to (2.7). Note that, as mentioned in the introduction, $d\mathcal{J}(\Omega; \mathcal{V})$ can be formulated as an integral over a volume, as well as an integral on the boundary.

Proposition 2.1.2. *Let Ω be a C^2-domain. The Eulerian Derivative of the shape functional* (2.6) *constrained to* (2.7) *with Dirichlet boundary conditions $u = g$ on $\partial\Omega$ reads*

$$\begin{aligned} d\mathcal{J}(\Omega;\mathcal{V}) = \int_\Omega \Big(& \nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p - f\mathcal{V}\cdot\nabla p \\ & + \operatorname{div}\mathcal{V}(j(u) - \nabla u\cdot\nabla p - up) \\ & + (j'(u) - p)(\nabla g\cdot\mathcal{V}) - \nabla p\cdot\nabla(\nabla g\cdot\mathcal{V})\Big)\,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{2.10}$$

and can be recast as

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_{\partial\Omega} (\mathcal{V}\cdot\mathbf{n})\left(j(u) + \frac{\partial p}{\partial\mathbf{n}}\frac{\partial(u-g)}{\partial\mathbf{n}}\right) dS\,, \tag{2.11}$$

where p is the weak solution of (2.9) *with homogeneous Dirichlet boundary conditions. In the case of Neumann boundary conditions in* (2.7)*, the counterparts of Formulas* (2.10) *and* (2.11) *read*

$$\begin{aligned} d\mathcal{J}(\Omega;\mathcal{V}) = \int_\Omega \Big(& (\nabla f\cdot\mathcal{V})p + \nabla u\cdot(\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p \\ & + \operatorname{div}\mathcal{V}(fp + j(u) - \nabla u\cdot\nabla p - up)\Big)\,\mathrm{d}\boldsymbol{x} \\ & + \int_{\partial\Omega} (\nabla g\cdot\mathcal{V})p + gp\operatorname{div}_\Gamma\mathcal{V}\,dS\,, \end{aligned} \tag{2.12}$$

where $\operatorname{div}_\Gamma$ *denotes the tangential divergence on $\partial\Omega$, and*

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}\left(j(u) - \nabla u\cdot\nabla p - up + fp + \frac{\partial gp}{\partial\mathbf{n}} + \mathrm{K}gp\right) dS\,, \tag{2.13}$$

where K is the mean curvature[1] *of $\partial\Omega$ and p is the solution of* (2.9) *with homogeneous Neumann boundary conditions.*

Proof. First, we derive formula (2.10) following [43, Ch. 1.6] closely. Then, we repeat the steps of [13, Sect. 6] and show that (2.10) can be rewritten equivalently as (2.11). Finally, we run through this procedure again and derive

[1] For C^2-domains, the mean curvature can be defined as the divergence of (any unitary extension) of the normal vector field n; see [41, pp. 192]

formulas (2.12)–(2.13). We mention that the same formulas can be retrieved following [24, Ch. 10, Sect. 6], which is the standard approach in the shape optimization community.

To compute the Eulerian derivative, it is convenient to work with a variational formulation of (2.7) that involves trial and test spaces that are independent of the control $\mathcal{V}$ (and that is nevertheless equivalent to (2.8) when its solution is sufficiently smooth). Therefore, we say that $(u, \boldsymbol{\mu}) \in H^1(\mathbb{R}^d) \times \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d)$ is the weak solution of (2.7) with Dirichlet boundary conditions if

$$\begin{cases} \int_\Omega \nabla u \cdot \nabla v + uv \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} \boldsymbol{\mu} \cdot \mathbf{n} v \,\mathrm{d}S = \int_\Omega f v \,\mathrm{d}\boldsymbol{x} & \forall v \in H^1(\mathbb{R}^d)\,, \\ \int_{\partial\Omega} (u - g)\boldsymbol{\lambda} \cdot \mathbf{n} \,\mathrm{d}S = 0 & \forall \boldsymbol{\lambda} \in \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d)\,. \end{cases} \tag{2.14}$$

We consider $\boldsymbol{\mu}, \boldsymbol{\lambda} \in \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d)$ because the restrictions $\boldsymbol{\mu} \cdot \mathbf{n}|_{\partial\Omega}$ and $\boldsymbol{\lambda} \cdot \mathbf{n}|_{\partial\Omega}$ must be in $H^{-1/2}(\partial\Omega)$. The formulation (2.14) corresponds to the optimality conditions that must be fulfilled by the critical point $(u, \boldsymbol{\mu})$ of the Dirichlet energy

$$\int_\Omega \frac{1}{2}\left(|\nabla u|^2 + u^2\right) - fu \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} (u - g)\boldsymbol{\mu} \cdot \mathbf{n} \,\mathrm{d}S\,.$$

Note that the restrictions $u|_\Omega$ and $(\boldsymbol{\mu} \cdot \mathbf{n})|_{\partial\Omega}$ are uniquely defined by (2.14) and satisfy

$$\|u\|_{H^1(\Omega)} + \|\boldsymbol{\mu} \cdot \mathbf{n}\|_{H^{-1/2}(\partial\Omega)} \leq C(\|f\|_{H^{-1}(\Omega)} + \|g\|_{H^{1/2}(\partial\Omega)})\,. \tag{2.15}$$

In the presence of the PDE constraint (2.7) with Dirichlet boundary conditions, the term $\mathcal{J}(T_{s\mathcal{V}}(\Omega))$ that appears in the difference quotient (2.3) depends on the solution of

$$\begin{cases} \int_{T_{s\mathcal{V}}(\Omega)} \nabla u \cdot \nabla v + uv \,\mathrm{d}\boldsymbol{x} - \int_{\partial T_{s\mathcal{V}}(\Omega)} \boldsymbol{\mu} \cdot \mathbf{n} v \,\mathrm{d}S = \int_{T_{s\mathcal{V}}(\Omega)} f v \,\mathrm{d}\boldsymbol{x} & \forall v \in H^1(\mathbb{R}^d)\,, \\ \int_{\partial T_{s\mathcal{V}}(\Omega)} (u - g)\boldsymbol{\lambda} \cdot \mathbf{n} \,\mathrm{d}S = 0 & \forall \boldsymbol{\lambda} \in \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d)\,. \end{cases} \tag{2.16}$$

Employing transformation techniques [24, Sect. 4.1-3, Ch. 9], the constraint (2.16) can be equivalently formulated as

$$\begin{aligned} 0 = &\int_\Omega \nabla(u \circ T_{s\mathcal{V}}) \cdot \mathbf{D}T_{s\mathcal{V}}^{-1} \mathbf{D}T_{s\mathcal{V}}^{-t} \det \mathbf{D}T_{s\mathcal{V}} \nabla(v \circ T_{s\mathcal{V}}) \,\mathrm{d}\boldsymbol{x} \\ &+ \int_\Omega ((u \circ T_{s\mathcal{V}})(v \circ T_{s\mathcal{V}}) - (f \circ T_{s\mathcal{V}})(v \circ T_{s\mathcal{V}})) \det \mathbf{D}T_{s\mathcal{V}} \,\mathrm{d}\boldsymbol{x} \end{aligned}$$

$$
\begin{aligned}
&+ \int_{\partial\Omega} (u \circ T_{s\mathcal{V}} - g \circ T_{s\mathcal{V}}) \left((\boldsymbol{\lambda} \circ T_{s\mathcal{V}}) \cdot \frac{\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}}{\|\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}\|} \right) \det \mathbf{D}T_{s\mathcal{V}} \|\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}\| \mathrm{d}S \\
&- \int_{\partial\Omega} \left((\boldsymbol{\mu} \circ T_{s\mathcal{V}}) \cdot \frac{\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}}{\|\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}\|} \right) (v \circ T_{s\mathcal{V}}) \det \mathbf{D}T_{s\mathcal{V}} \|\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}\| \, \mathrm{d}S \,, \\
= &\int_{\Omega} \nabla(u \circ T_{s\mathcal{V}}) \cdot \mathbf{D}T_{s\mathcal{V}}^{-1}\mathbf{D}T_{s\mathcal{V}}^{-t} \det \mathbf{D}T_{s\mathcal{V}} \nabla(v \circ T_{s\mathcal{V}}) \, \mathrm{d}\boldsymbol{x} \\
&+ \int_{\Omega} ((u \circ T_{s\mathcal{V}})(v \circ T_{s\mathcal{V}}) - (f \circ T_{s\mathcal{V}})(v \circ T_{s\mathcal{V}})) \det \mathbf{D}T_{s\mathcal{V}} \, \mathrm{d}\boldsymbol{x} \\
&+ \int_{\partial\Omega} (u \circ T_{s\mathcal{V}} - g \circ T_{s\mathcal{V}}) \left(\det \mathbf{D}T_{s\mathcal{V}} \mathbf{D}T_{s\mathcal{V}}^{-1} (\boldsymbol{\lambda} \circ T_{s\mathcal{V}}) \right) \cdot \mathbf{n} \mathrm{d}S \\
&- \int_{\partial\Omega} \left(\det \mathbf{D}T_{s\mathcal{V}} \mathbf{D}T_{s\mathcal{V}}^{-1} (\boldsymbol{\mu} \circ T_{s\mathcal{V}}) \right) \cdot \mathbf{n} \, (v \circ T_{s\mathcal{V}}) \, \mathrm{d}S \\
&\qquad\qquad \forall (v, \boldsymbol{\lambda}) \in H^1(\mathbb{R}^d) \times \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d) \,.
\end{aligned} \tag{2.17}
$$

which, defining $u(s\mathcal{V}) \coloneqq u \circ T_{s\mathcal{V}}$ and $\boldsymbol{\mu}(s\mathcal{V}) \coloneqq \det \mathbf{D}T_{s\mathcal{V}} \mathbf{D}T_{s\mathcal{V}}^{-1} (\boldsymbol{\mu} \circ T_{s\mathcal{V}})$ (the inverse Piola transform of $\boldsymbol{\mu}$) and renaming v and $\boldsymbol{\lambda}$, is equivalent to

$$
\begin{aligned}
0 = &\int_{\Omega} \left(\nabla u(s\mathcal{V}) \cdot \mathbf{D}T_{s\mathcal{V}}^{-1}\mathbf{D}T_{s\mathcal{V}}^{-T} \nabla v + u(s\mathcal{V}) v - (f \circ T_{s\mathcal{V}}) v \right) \det \mathbf{D}T_{s\mathcal{V}} \, \mathrm{d}\boldsymbol{x} \\
&+ \int_{\partial\Omega} (u(s\mathcal{V}) - g \circ T_{s\mathcal{V}}) \boldsymbol{\lambda} \cdot \mathbf{n} - (\boldsymbol{\mu}(s\mathcal{V}) \cdot \mathbf{n}) \, v \, \mathrm{d}S \\
&\qquad\qquad \forall (v, \boldsymbol{\lambda}) \in H^1(\mathbb{R}^d) \times \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d) \,.
\end{aligned} \tag{2.18}
$$

Equation (2.18) can be used to replace the constraint (2.7). Note that the integration regions in (2.18) are independent of the control $\mathcal{V}$. This allows us to replace the trial and the test space $H^1(\mathbb{R}^d) \times \boldsymbol{H}(\mathrm{div}, \mathbb{R}^d)$ with $H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega)$. Note that we are merely interested in the restrictions of $u(s\mathcal{V})$ and $\mu(s\mathcal{V})$ to Ω and $\partial\Omega$, respectively.

It is convenient to interpret the right-hand side of (2.18) as the following operator

$$
\begin{aligned}
e : C^1(\mathbb{R}^d; \mathbb{R}^d) \times H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega) &\to \left(H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega) \right)' \,, \\
(\mathcal{V}, u, \boldsymbol{\mu}) &\mapsto e(\mathcal{V}, u, \boldsymbol{\mu}) \,,
\end{aligned}
$$

where $e(\mathcal{V}, u, \boldsymbol{\mu}) : (v, \boldsymbol{\lambda}) \mapsto e(\mathcal{V}, u, \boldsymbol{\mu}; v, \boldsymbol{\lambda})$ is given by

$$\begin{aligned} e(\mathcal{V}, u, \boldsymbol{\mu}; v, \boldsymbol{\lambda}) \coloneqq & \int_\Omega \left(\nabla u \cdot \mathbf{D}T_\mathcal{V}^{-1} \mathbf{D}T_\mathcal{V}^{-T} \nabla v + uv - (f \circ T_\mathcal{V}) v \right) \det \mathbf{D}T_\mathcal{V} \, \mathrm{d}\boldsymbol{x} \\ & - \int_{\partial\Omega} (u - g \circ T_\mathcal{V}) \boldsymbol{\lambda} \cdot \mathbf{n} + (\boldsymbol{\mu} \cdot \mathbf{n}) \, v \, \mathrm{d}S \,. \end{aligned} \tag{2.19}$$

Note that $(u, \boldsymbol{\mu})$ is the solution of (2.14) if and only if $e(0, u, \boldsymbol{\mu}) = 0$. Moreover, the operator e is continuously Fréchet differentiable in the variables u and $\boldsymbol{\mu}$ (because it is linear in these variables), and has a bounded inverse; see (2.15). Therefore, by the Implicit function theorem [43, Thm 1.41], there exists a solution operator

$$\mathcal{V} \mapsto (u(\mathcal{V}), \boldsymbol{\mu}(\mathcal{V}) \cdot \mathbf{n}) \tag{2.20}$$

that is continuously Fréchet differentiable in a neighbourhood of $0 \in C^1(\mathbb{R}^d; \mathbb{R}^d)$. In the shape optimization community, the Fréchet derivative of this solution operator is called *material derivative*; cf. [24, Ch. 10, Sect. 6].

Next, we introduce the Lagrangian

$$\begin{aligned} \mathscr{L} : C^1(\mathbb{R}^d; \mathbb{R}^d) \times H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega) \times H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega) \to \mathbb{R} \,, \\ (\mathcal{V}, u, \boldsymbol{\mu}, v, \boldsymbol{\lambda}) \mapsto \mathscr{L}(\mathcal{V}, u, \boldsymbol{\mu}, v, \boldsymbol{\lambda}) \coloneqq \mathcal{J}\left((T_\mathcal{V}(\Omega)\right) - e(\mathcal{V}, u, \boldsymbol{\mu}; v, \boldsymbol{\lambda}) \,. \end{aligned} \tag{2.21}$$

$u(\mathcal{V})$ in Ω and $\boldsymbol{\mu}(\mathcal{V}) \cdot \mathbf{n}$ on $\partial\Omega$. Note that

$$\mathscr{L}(\mathcal{V}, u(\mathcal{V}), \boldsymbol{\mu}(\mathcal{V}), v, \boldsymbol{\lambda}) = \mathcal{J}\left((T_\mathcal{V}(\Omega)\right) \quad \forall\, (v, \boldsymbol{\lambda}) \in H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega)$$

as long as $\mathcal{V}$ is in the domain of definition of the solution operator (2.20). Thus

$$\begin{aligned} d\mathcal{J}(\Omega; \mathcal{V}) &= \lim_{s \searrow 0} \frac{\mathcal{J}\left(T_{s\cdot\mathcal{V}}(\Omega)\right) - \mathcal{J}(\Omega)}{s} \\ &= \lim_{s \searrow 0} \frac{\mathscr{L}(\mathcal{V}, u(\mathcal{V}), \boldsymbol{\mu}(\mathcal{V}), v, \boldsymbol{\lambda}) - \mathscr{L}(0, u(0), \boldsymbol{\mu}(0), v, \boldsymbol{\lambda})}{s} \,, \end{aligned} \tag{2.22}$$

that is, to derive $d\mathcal{J}(\Omega; \mathcal{V})$, we can compute the derivative of the reduced functional $(\mathcal{W}, v, \lambda) \mapsto \mathscr{L}(\mathcal{W}, u(\mathcal{W}), \boldsymbol{\mu}(\mathcal{W}), v, \boldsymbol{\lambda})$ with respect to $\mathcal{W}$ in the direction $\mathcal{V} \in C^1(\mathbb{R}^d; \mathbb{R}^d)$ evaluated in $\mathcal{W} = 0$. This reads

$$
\begin{aligned}
&< \frac{d\mathscr{L}(\mathcal{W}, u(\mathcal{W}), \boldsymbol{\mu}(\mathcal{W}), v, \lambda)}{d\mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} = \\
&=< \frac{\partial \mathcal{J}\left(T_{\mathcal{W}}(\Omega)\right)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} + < \frac{\partial \mathcal{J}(\Omega)}{\partial u}, u'(0;\mathcal{V}) > \\
&\quad - < \frac{\partial e(\mathcal{W}, u(0), \boldsymbol{\mu}(0); v, \lambda)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} - < \frac{\partial e(0, u(0), \boldsymbol{\mu}(0); v, \lambda)}{\partial u}, u'(0;\mathcal{V}) > \\
&\quad - < \frac{\partial e(0, u(0), \boldsymbol{\mu}(0); v, \lambda)}{\partial \boldsymbol{\mu}}, \boldsymbol{\mu}'(0;\mathcal{V}) > ,
\end{aligned} \tag{2.23}
$$

where $(u'(0;\mathcal{W}), \boldsymbol{\mu}'(0;\mathcal{W}))$ denotes the derivative of the solution operator (2.20) in the direction $\mathcal{W} \in C^1(\mathbb{R}^d;\mathbb{R}^d)$ evaluated in $\mathcal{V} = 0$. Note that

$$
(u'(0;\mathcal{V}), \boldsymbol{\mu}'(0;\mathcal{V})) \in H^1(\Omega) \times \boldsymbol{H}(\mathrm{div}, \Omega) .
$$

The Lagrange multipliers $(v, \boldsymbol{\lambda})$ are free parameters and can be chosen so that

$$
\begin{cases}
< \frac{\partial \mathcal{J}(\Omega)}{\partial u}, \phi > - < \frac{\partial e(0, u(0), \boldsymbol{\mu}(0); v, \lambda)}{\partial u}, \phi >= 0 & \forall\, \phi \in H^1(\Omega) , \\
< \frac{\partial e(0, u(0), \boldsymbol{\mu}(0); v, \lambda)}{\partial \boldsymbol{\mu}}, \boldsymbol{\psi} >= 0 & \forall\, \boldsymbol{\psi} \in \boldsymbol{H}(\mathrm{div}, \Omega) ,
\end{cases} \tag{2.24}
$$

that is, so that

$$
\begin{cases}
\int_\Omega j'(u)\phi - \nabla\phi \cdot \nabla v - \phi v \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \phi \boldsymbol{\lambda} \cdot \mathbf{n} \,\mathrm{d}S = 0 & \forall\, \phi \in H^1(\Omega) , \\
\int_{\partial\Omega} \boldsymbol{\psi} \cdot \mathbf{n} v \,\mathrm{d}S = 0 & \forall\, \boldsymbol{\psi} \in \boldsymbol{H}(\mathrm{div}, \Omega) .
\end{cases} \tag{2.25}
$$

Note that (2.25) is the weak formulation of (2.9) with Dirichlet boundary conditions.

By (2.22), (2.23), and with $(v, \boldsymbol{\lambda})$ satisfying (2.25), it follows

$$
d\mathcal{J}(\Omega;\mathcal{V}) =< \frac{\partial \mathcal{J}\left((T_{\mathcal{W}}(\Omega)\right)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} - < \frac{\partial e(\mathcal{W}, u(0); v, \lambda)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} . \tag{2.26}
$$

The partial derivatives on the right-hand side of (2.26) can be computed employing transformation techniques. On the one hand, from

$$
\begin{aligned}
\mathcal{J}\left(T_{\mathcal{W}}(\Omega)\right) &= \int_{T_{\mathcal{W}}(\Omega)} j(u) \,\mathrm{d}\boldsymbol{x} , \\
&= \int_\Omega j(u \circ T_{\mathcal{W}}) |\det \mathbf{D}T_{\mathcal{W}}| \,\mathrm{d}\boldsymbol{x} ,
\end{aligned}
$$

$$\overset{(2.2)}{=} \int_\Omega j(u \circ T_\mathcal{W}) \det \mathbf{D}T_\mathcal{W} \,\mathrm{d}\boldsymbol{x}\,,$$

we conclude that

$$< \frac{\partial \mathcal{J}\left((T_\mathcal{W}(\Omega)\right)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} = \int_\Omega j(u) \operatorname{div} \mathcal{V} \,\mathrm{d}\boldsymbol{x}\,, \tag{2.27}$$

because, by Taylor expansion,

$$\det \mathbf{D}T_\mathcal{W} = \mathbb{1} + \operatorname{div} \mathcal{W} + \mathcal{O}(\|\mathcal{W}\|^2)\,.$$

On the other hand, by (2.19),

$$\begin{aligned} < \frac{\partial e(\mathcal{W}, u(0); v, \lambda)}{\partial \mathcal{W}}, \mathcal{V} > |_{\mathcal{W}=0} = &\int_\Omega \nabla u \cdot \left(\operatorname{div} \mathcal{V} - \mathbf{D}\mathcal{V} - \mathbf{D}\mathcal{V}^T\right) \nabla v \,\mathrm{d}\boldsymbol{x} \\ &+ \int_\Omega (uv - fv) \operatorname{div} \mathcal{V} - \nabla f \cdot \mathcal{V} v \,\mathrm{d}\boldsymbol{x} \\ &+ \int_{\partial\Omega} \nabla g \cdot \mathcal{V}\boldsymbol{\lambda} \cdot \mathbf{n} \mathrm{d}S\,, \end{aligned} \tag{2.28}$$

because,

$$\begin{aligned} \mathbf{D}T_\mathcal{W}^{-1} &= \mathbb{1} - \mathbf{D}\mathcal{W} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2)\,, \\ f \circ T_\mathcal{W} &= f + \nabla f \cdot \mathcal{W} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2)\,, \\ g \circ T_\mathcal{W} &= g + \nabla g \cdot \mathcal{W} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2)\,. \end{aligned}$$

Thus, by (2.26),(2.27) and (2.28), and renaming $p = v$,

$$\begin{aligned} d\mathcal{J}(\Omega; \mathcal{V}) = &\int_\Omega j(u) \operatorname{div} \mathcal{V} - \nabla u \cdot \left(\operatorname{div} \mathcal{V} - \mathbf{D}\mathcal{V} - \mathbf{D}\mathcal{V}^T\right) \nabla p \,\mathrm{d}\boldsymbol{x} \\ &- \int_\Omega (up - fp) \operatorname{div} \mathcal{V} - \nabla f \cdot \mathcal{V} p \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} \nabla g \cdot \mathcal{V}\boldsymbol{\lambda} \cdot \mathbf{n} \mathrm{d}S\,, \\ \overset{(2.25)}{=} &\int_\Omega \big(\nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla p + \nabla f \cdot \mathcal{V} p + (j'(u) - p) \nabla g \cdot \mathcal{V} \\ &\quad - \nabla p \cdot \nabla(\nabla g \cdot \mathcal{V}) + \operatorname{div}(\mathcal{V}) \left(j(u) - \nabla u \cdot \nabla p - u\, p + f p\right) \big) \,\mathrm{d}\boldsymbol{x}\,, \end{aligned}$$

which, after an additional integration by parts on the term $\nabla f \cdot \mathcal{V} p$, corresponds to Formula (2.10).

Next, we derive Formula (2.11). First of all, let us remark that elliptic regularity theory [15, Ch. II, Thm 7.2] implies that the weak solutions u of (2.7) and p of (2.9) are at least in $H^2(\Omega)$. Thus, we can employ the vector calculus identity [13, Sect. 6]

$$\mathcal{V}\cdot\nabla\left(\nabla u\cdot\nabla p\right)+\nabla u\cdot\left(\mathbf{D}\mathcal{V}+\mathbf{D}\mathcal{V}^T\right)\nabla p=\nabla\left(\mathcal{V}\cdot\nabla u\right)\cdot\nabla p+\nabla u\cdot\nabla\left(\mathcal{V}\cdot\nabla p\right) \tag{2.29}$$

to rewrite

$$\begin{aligned} d\mathcal{J}(\Omega;\mathcal{V})=\int_\Omega\big(&\nabla\left(\mathcal{V}\cdot\nabla u\right)\cdot\nabla p+\nabla u\cdot\nabla\left(\mathcal{V}\cdot\nabla p\right)-\mathcal{V}\cdot\nabla\left(\nabla u\cdot\nabla p\right)\\ &+\nabla f\cdot\mathcal{V}p+(j'(u)-p)\nabla g\cdot\mathcal{V}-\nabla p\cdot\nabla(\nabla g\cdot\mathcal{V})\\ &+\operatorname{div}(\mathcal{V})\left(j(u)-\nabla u\cdot\nabla p-u\,p+fp\right)\big)\,\mathrm{d}\boldsymbol{x}\,. \end{aligned}$$

Then, by the product rule,

$$\begin{aligned} d\mathcal{J}(\Omega;\mathcal{V})=\int_\Omega\big(&\nabla\left(\mathcal{V}\cdot\nabla u\right)\cdot\nabla p+\nabla u\cdot\nabla\left(\mathcal{V}\cdot\nabla p\right)-\mathcal{V}\cdot\nabla\left(\nabla u\cdot\nabla p\right)\\ &+\nabla f\cdot\mathcal{V}p+(j'(u)-p)\nabla g\cdot\mathcal{V}-\nabla p\cdot\nabla(\nabla g\cdot\mathcal{V})\\ &-\mathcal{V}\cdot\left(j'(u)\nabla u-\nabla(\nabla u\cdot\nabla p)-p\nabla u-u\nabla p+p\nabla f+f\nabla p\right)\\ &+\operatorname{div}\left(\mathcal{V}\left(j(u)-\nabla u\cdot\nabla p-u\,p+fp\right)\right)\big)\,\mathrm{d}\boldsymbol{x}\,,\\ =\int_\Omega&\nabla u\cdot\nabla\left(\mathcal{V}\cdot\nabla p\right)+u\mathcal{V}\cdot\nabla p-f\mathcal{V}\cdot\nabla p\,\mathrm{d}\boldsymbol{x}\\ &+\int_\Omega\nabla\left(\mathcal{V}\cdot\nabla u\right)\cdot\nabla p-j'(u)\mathcal{V}\cdot\nabla u+p\mathcal{V}\cdot\nabla u\,\mathrm{d}\boldsymbol{x}\\ &-\int_\Omega\nabla\left(\mathcal{V}\cdot\nabla g\right)\cdot\nabla p-j'(u)\mathcal{V}\cdot\nabla g+p\mathcal{V}\cdot\nabla g\,\mathrm{d}\boldsymbol{x}\\ &+\int_\Omega\operatorname{div}\left(\mathcal{V}\left(j(u)-\nabla u\cdot\nabla p-u\,p+fp\right)\right)\,\mathrm{d}\boldsymbol{x}\,. \end{aligned}$$

By (2.14),(2.25), and Gauss's Theorem,

$$\begin{aligned} d\mathcal{J}(\Omega;\mathcal{V})=\int_{\partial\Omega}&\frac{\partial u}{\partial\mathbf{n}}\mathcal{V}\cdot\nabla p+\frac{\partial p}{\partial\mathbf{n}}\mathcal{V}\cdot\nabla(u-g)\\ &+\left(\mathcal{V}\cdot\mathbf{n}\right)\left(j(u)-\nabla u\cdot\nabla p-u\,p+fp\right)\,\mathrm{d}S \end{aligned}$$

Since $u,p\in H^2(\Omega)\subset C^0(\overline{\Omega})$ (for $d=2,3$, [68, Thm. 2.5.4]), the equalities $u=g$ and $p=0$ hold pointwise on $\partial\Omega$, and it follows that (on $\partial\Omega$)

$$\mathcal{V}\cdot\nabla(u-g)=\mathcal{V}\cdot\mathbf{n}\,\frac{\partial(u-g)}{\partial\mathbf{n}}\,,\quad\mathcal{V}\cdot\nabla p=\mathcal{V}\cdot\mathbf{n}\frac{\partial p}{\partial\mathbf{n}}\,,\quad\nabla u\cdot\nabla p=\frac{\partial v}{\partial\mathbf{n}}\frac{\partial p}{\partial\mathbf{n}}\,,$$

and thus, we retrieve (2.11).

Next, we repeat this procedure to derive Formulas (2.12) and (2.13). We say that u is the weak solution of (2.7) with Neumann boundary conditions if

$$\int_\Omega \nabla u \cdot \nabla v + uv \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} gv \,\mathrm{d}S = \int_\Omega fv \,\mathrm{d}\boldsymbol{x} \quad \forall v \in H^1(\mathbb{R}^d)\,. \tag{2.30}$$

On the perturbed domain $T_{s\mathcal{V}}(\Omega)$, this constraint reads

$$\int_{T_{s\mathcal{V}}(\Omega)} \nabla u \cdot \nabla v + uv \,\mathrm{d}\boldsymbol{x} - \int_{\partial T_{s\mathcal{V}}(\Omega)} gv \,\mathrm{d}S = \int_{T_{s\mathcal{V}}(\Omega)} fv \,\mathrm{d}\boldsymbol{x} \quad \forall v \in H^1(\mathbb{R}^d)\,,$$

which, by transformation techniques [24, Sect. 4.1-3, Ch. 9], can be equivalently formulated as

$$\begin{aligned}\int_\Omega \Big(\nabla u(s\mathcal{V}) \cdot \mathbf{D}T_{s\mathcal{V}}^{-1}\mathbf{D}T_{s\mathcal{V}}^{-T}\nabla v + u(s\mathcal{V})v - (f \circ T_{s\mathcal{V}})v\Big) \det \mathbf{D}T_{s\mathcal{V}} \,\mathrm{d}\boldsymbol{x} \\ - \int_{\partial\Omega} (g \circ T_{s\mathcal{V}})v \det \mathbf{D}T_{s\mathcal{V}} |\mathbf{D}T_{s\mathcal{V}}^{-T}\mathbf{n}| \,\mathrm{d}S = 0 \quad \forall v \in H^1(\mathbb{R}^d)\,,\end{aligned} \tag{2.31}$$

where $u(s\mathcal{V}) \coloneqq u \circ T_{s\mathcal{V}}$. At this point, we replace the trial and test space $H^1(\mathbb{R}^d)$ with $H^1(\Omega)$ because we are merely interested in the restriction of $u(s\mathcal{V})$ to Ω. The constraint (2.31) can be interpreted as the following operator

$$e : C^1(\mathbb{R}^d;\mathbb{R}^d) \times H^1(\Omega) \to \big(H^1(\Omega)\big)'\,, \quad (\mathcal{V}, u) \mapsto e(\mathcal{V}, u)\,,$$

where $e(\mathcal{V}, u) : v \mapsto e(\mathcal{V}, u; v)$ is given by

$$\begin{aligned}e(\mathcal{V}, u; v) \coloneqq \int_\Omega \Big(\nabla u \cdot \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}\nabla v \,\mathrm{d}\boldsymbol{x} + uv - (f \circ T_{\mathcal{V}})v\Big) \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} \\ - \int_{\partial\Omega} (g \circ T_{\mathcal{V}})v \det \mathbf{D}T_{\mathcal{V}} |\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{n}| \,\mathrm{d}S\,.\end{aligned} \tag{2.32}$$

Again, we can rely on the Implicit function theorem [43, Thm 1.41] to assert that there exists a solution operator $\mathcal{V} \mapsto u(\mathcal{V})|_\Omega$ that is continuously Fréchet differentiable in a neighbourhood of $0 \in C^1(\mathbb{R}^d;\mathbb{R}^d)$. Thus, we can compute $d\mathcal{J}(\Omega;\mathcal{V})$ by differentiating the Lagrangian

$$\mathscr{L} : C^1(\mathbb{R}^d;\mathbb{R}^d) \times \big(H^1(\Omega)\big)^2 \to \mathbb{R}\,, \quad (\mathcal{V}, u, v) \mapsto \mathscr{L}(\mathcal{V}, u, v)\,,$$

$$\mathscr{L}(\mathcal{V}, u, v) \coloneqq \mathcal{J}\left((T_{\mathcal{V}}(\Omega)\right) - e(\mathcal{V}, u; v)\,.$$

We skip this differentiation because it is completely analogous to the Dirichlet case. Let us just point out that, in this case, the adjoint equation reads

$$\int_\Omega j'(u)\phi \,\mathrm{d}\boldsymbol{x} - \int_\Omega \nabla\phi \cdot \nabla v + \phi v \,\mathrm{d}\boldsymbol{x} = 0 \quad \forall\, \phi \in H^1(\Omega)\,, \tag{2.33}$$

which is the weak formulation of (2.9). To retrieve Formula (2.13), we first apply the vector calculus calculus identity (2.29), the product rule,(2.30),(2.33), and Gauss theorem. We obtain

$$\begin{aligned} d\mathcal{J}(\Omega; \mathcal{V}) = \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}\,(fp + j(u) - \nabla u\cdot\nabla p - up) \\ + (\nabla g\cdot\mathcal{V})p + gp\,\mathrm{div}_\Gamma\,\mathcal{V} + (\nabla p\cdot\mathcal{V})g\,dS\,, \end{aligned}$$

where [24, Eq. 4.12, Ch. 9]

$$\mathrm{div}_\Gamma\,\mathcal{V} \coloneqq < \frac{\partial}{\partial\mathcal{W}}\left(\det \mathbf{D}T_{\mathcal{W}} |\mathbf{D}T_{\mathcal{W}}^{-T}\mathbf{n}|\right), \mathcal{V} > |_{\mathcal{W}=0} = \mathrm{div}\,\mathcal{V} - \mathbf{n}\cdot\mathbf{D}\mathcal{V}\mathbf{n}\,.$$

Let $\mathcal{V}_\tau \coloneqq \mathcal{V} - (\mathcal{V}\cdot\mathbf{n})\mathbf{n}$. Proposition 2.57 in [73] shows that

$$\mathrm{div}_\Gamma\,\mathcal{V} = \mathrm{div}_\Gamma\,\mathcal{V}_\tau + K\mathcal{V}\cdot\mathbf{n}\,,$$

where K denotes the mean curvature. Thus,

$$\begin{aligned} \int_{\partial\Omega} \nabla(gp)\cdot\mathcal{V} + gp\,\mathrm{div}_\Gamma\,\mathcal{V}\,\mathrm{d}S &= \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}(\frac{\partial(gp)}{\partial\mathbf{n}} + Kgp)\,\mathrm{d}S \\ &\quad + \int_{\partial\Omega} \frac{\partial(gp)}{\partial\tau}\cdot\mathcal{V}_\tau + gp\,\mathrm{div}_\Gamma\,\mathcal{V}_\tau\,\mathrm{d}S\,, \\ &= \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}(\frac{\partial(gp)}{\partial\mathbf{n}} + Kgp) + \mathrm{div}_\Gamma(gp\mathcal{V}_\tau)\,\mathrm{d}S\,, \\ &= \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}(\frac{\partial(gp)}{\partial\mathbf{n}} + Kgp)\,\mathrm{d}S\,, \end{aligned} \tag{2.34}$$

where the last step follows from Stokes' Theorem. □

Remark 2.1.3. *The assumption "Ω is C^2" in Proposition 2.1.2 is not strictly necessary. Formulas* (2.10) *and* (2.12) *can be derived for any Lipschitz domain. More specifically, it is sufficient that $u, p \in H^1(\Omega)$. Thus, these formulas do not change if the PDE-contraint is replaced with a finite-dimensional counterpart that arises from Galerkin finite element discretization.*

To derive Formula (2.11) *and* (2.13) *we need that* $u, p \in H^2(\Omega)$. *This condition is satisfied for all convex domains. However, Formula* (2.13) *requires special care because it involves the mean curvature (unless* $g = 0$*). For this quantity to be well-defined, the domain* Ω *has to be at least piecewise* C^2. *However, in the presence of corners, Formula* (2.13) *has to be corrected by adding terms that arise from the application of Stokes' Theorem in* (2.34). *In 2D, these terms read*

$$\sum_i p(\mathbf{a}_i) g(\mathbf{a}_i) \mathcal{V}(\mathbf{a}_i) \cdot [[\tau(\mathbf{a}_i)]] \,, \tag{2.35}$$

where the $\mathbf{a}_i$ *denote the corner points and* $[[\tau(\mathbf{a}_i)]]$ *is the jump of the tangential unit vector field in the corner* $\mathbf{a}_i$ *[73, Ch. 3.8].*

Finally, let us remark that, in the case of homogeneous Neumann boundary conditions, Formula (2.12) *is still valid if* $\mathcal{V}$ *is in* $W^{1,\infty}(\Omega)$ *only.*

2.2. Approximation of shape gradients

In this section we investigate the approximation of the shape gradient $d\mathcal{J}$. For the sake of readability, we perform the analysis for the shape functional (2.6) constrained to (2.7) with Dirichlet boundary conditions only. The results can easily be extended to general elliptic operators in divergence form with both Dirichlet and Neumann boundary conditions.

To highlight the dependence of $d\mathcal{J}$ on the solution of the state and adjoint problem u and p, as well as to distinguish between formulas (2.10) and (2.11), we introduce the notations

$$\begin{aligned} d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Vol}} \coloneqq \int_\Omega \Big(& \nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla p - f \mathcal{V} \cdot \nabla p \\ & + \operatorname{div} \mathcal{V} (j(u) - \nabla u \cdot \nabla p - up) \\ & + (j'(u) - p)(\nabla g \cdot \mathcal{V}) - \nabla p \cdot \nabla (\nabla g \cdot \mathcal{V}) \Big) \, \mathrm{d}\boldsymbol{x} \,, \end{aligned} \tag{2.36}$$

$$d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Bdry}} \coloneqq \int_{\partial\Omega} \mathcal{V} \cdot \mathbf{n} \left(j(u) + \frac{\partial p}{\partial \mathbf{n}} \frac{\partial (u - g)}{\partial \mathbf{n}} \right) dS \,. \tag{2.37}$$

Note that, provided u and p are exact solutions of (2.7) and (2.9),

$$d\mathcal{J}(\Omega; \mathcal{V}) = d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Vol}} = d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Bdry}} \,. \tag{2.38}$$

The operator $d\mathcal{J}(\Omega;\cdot)$ can be approximated by replacing the functions u and p with Ritz-Galerkin Lagrangian finite element solutions of (2.7) and (2.9) respectively. We consider approximations based on discretization with finite elements, as this approach is very popular in shape optimization due to its flexibility for engineering applications. Approximations based on boundary element methods are also possible, cf. [26, 38, 75].

Equality (2.38) certainly breaks down when the functions u and p are approximated with finite elements [13]. Thus, a natural question is, which formula, (2.36) or (2.37), should be preferred for an approximation of $d\mathcal{J}(\Omega;\cdot)$ in the operator norm. The answer is provided by Theorems 2.2.4 and 2.2.9. Next we state a few assumptions necessary for a precise statement of the theorems.

Assumption 2.2.1. *The Dirichlet BVP for the Laplacian is H^2-regular [15, Ch. II, Def. 7.1], that is, if a function $w \in H^1_0(\Omega)$ is the (unique) weak solution of the elliptic BVP*

$$\begin{cases} -\Delta w + w &= \rho \quad \text{in } \Omega\,, \\ w &= 0 \quad \text{on } \partial\Omega\,, \end{cases}$$

for a function $\rho \in L^2(\Omega)$, then $w \in H^2(\Omega)$, and there is a constant C_r, depending only on Ω, so that

$$\|w\|_{H^2(\Omega)} \leq C_r \|\rho\|_{L^2(\Omega)}\,.$$

Remark 2.2.2. *Assumption 2.2.1 holds for convex Lipschitz domains and (possibly nonconvex) domains with C^2 boundary [15, Ch. II, Thm 7.2].*

Assumption 2.2.3. *The source function f and the boundary data g in (2.7) are restrictions of functions in $H^1(\mathbb{R}^d)$ and $H^3(\mathbb{R}^d)$ to Ω and $\partial\Omega$, respectively.*

Next, for an index set $\mathbb{H}$, we introduce a family $(V_h)_{h\in\mathbb{H}}$ of finite-dimensional subspaces of $H^1_0(\Omega)$ and define[2] $u_h \in g + V_h$, $p_h \in V_h$ as Ritz-Galerkin solutions[3] of (2.7) and (2.9), respectively, that is,

$$\int_\Omega \nabla u_h \cdot \nabla v_h + u_h v_h \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v_h \,\mathrm{d}\boldsymbol{x} \qquad \forall\, v_h \in V_h\,, \tag{2.39}$$

[2] In standard finite element implementations, the function g is replaced with an interpolant $g_h \in V_h$. To simplify the error analysis, we neglect the additional consistency error that is introduced by this substitution.

[3] Note that p_h is not a proper Ritz-Galerkin solution of (2.9), because the right-hand side is perturbed.

$$\int_\Omega \nabla p_h \cdot \nabla v_h + p_h v_h \,\mathrm{d}\boldsymbol{x} = \int_\Omega j'(u_h) v_h \,\mathrm{d}\boldsymbol{x} \qquad \forall\, v_h \in V_h\,. \tag{2.40}$$

In particular, let $(V_h)_{h\in\mathbb{H}}$ be a family of H^1-conforming piecewise linear Lagrangian finite element spaces built on a shape-regular and quasi-uniform family of simplicial meshes [15, Ch. II, Def. 5.1], and let h designate the mesh-width. We recall that the associated family of nodal interpolation operators

$$\mathcal{I}_h : H^2(\Omega) \cap H_0^1(\Omega) \to V_h$$

satisfies[4] [15, Ch. II, Thm 6.4]

$$\|w - \mathcal{I}_h w\|_{H^1(\Omega)} \le C h |w|_{H^2(\Omega)} \quad \forall\, h \in \mathbb{H}\,. \tag{2.41}$$

Theorem 2.2.4. *Let u and p be the solutions of (2.7) and (2.9), and let u_h and p_h be their Ritz-Galerkin approximations in the sense of (2.39) and (2.40) by piecewise linear Lagrangian finite elements. Furthermore, let Assumptions 2.2.1 and 2.2.3 be satisfied. Then*[5]

$$|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}| \le C(\Omega,u,p,f,g) h^2 \|\mathcal{V}\|_{W^{2,4}(\Omega)}\,, \tag{2.42}$$

where the constant $C(\Omega,u,p,f,g)$ depends on the domain Ω and its discretization, $\|u\|_{H^2(\Omega)}$, $\|p\|_{H^2(\Omega)}$, $\|f\|_{H^1(\Omega)}$, and $\|g\|_{H^3(\Omega)}$.

Proof. The proof heavily relies on duality techniques that are repeatedly used to obtain estimates for the various terms in (2.36). The impatient reader may skip the proof after (2.50) and will get main ideas nevertheless.

From the equality $d\mathcal{J}(\Omega;\mathcal{V}) = d\mathcal{J}(\Omega,u,p;\mathcal{V})^{\mathrm{Vol}}$, we immediately get by the triangle inequality

$$|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}|$$

[4] We write C for generic constants, whose value may differ between different occurrences. They may depend only on Ω, shape-regularity and quasi-uniformity of the meshes.

[5] For the sake of readability, we use the same notation for scalar and vectorial Sobolev norms.

$$
\begin{aligned}
\leq \Big(& |\int_\Omega \nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p - \nabla u_h \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p_h \,\mathrm{d}\boldsymbol{x}| \\
& + |\int_\Omega f\mathcal{V} \cdot \nabla(p - p_h)\,\mathrm{d}\boldsymbol{x}| \\
& + |\int_\Omega \operatorname{div}\mathcal{V}(j(u) - j(u_h) - \nabla u \cdot \nabla p - up + \nabla u_h \cdot \nabla p_h + u_h p_h)\,\mathrm{d}\boldsymbol{x}| \\
& + |\int_\Omega (j'(u) - j'(u_h) - p + p_h)(\nabla g \cdot \mathcal{V})\,\mathrm{d}\boldsymbol{x}| \\
& + |\int_\Omega \nabla(p - p_h) \cdot \nabla(\nabla g \cdot \mathcal{V})\,\mathrm{d}\boldsymbol{x}| \Big) .
\end{aligned}
\tag{2.43}
$$

The proof boils down to bounding each integral in the previous inequality and applying standard finite element convergence and interpolation estimates. To begin with, we split the first integral into

$$
\begin{aligned}
\int_\Omega (\nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p - \nabla u_h \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p_h \,\mathrm{d}\boldsymbol{x} & \\
= \int_\Omega \nabla(u - u_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p \,\mathrm{d}\boldsymbol{x} & \\
+ \int_\Omega \nabla u \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla(p - p_h)\,\mathrm{d}\boldsymbol{x} & \\
- \int_\Omega \nabla(u - u_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla(p - p_h)\,\mathrm{d}\boldsymbol{x} . &
\end{aligned}
\tag{2.44}
$$

To bound the first and the second integral on the right-hand side of (2.44) we make use of standard duality techniques. For the first one we introduce the function w as weak solution of the adjoint BVP [6]

$$
\left\{ \begin{array}{rll} -\Delta w + w &= -\operatorname{div}\big((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p\big) & \text{in } \Omega\,, \\ w &= 0 & \text{on } \partial\Omega\,, \end{array} \right. \tag{2.45}
$$

that is,

$$
\int_\Omega \nabla w \cdot \nabla v + wv \,\mathrm{d}\boldsymbol{x} = \int_\Omega \big((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla p\big) \cdot \nabla v \quad \forall\, v \in H_0^1(\Omega)\,. \tag{2.46}
$$

[6] Many bounds in this proof rely on duality techniques, which introduce so-called adjoint BVPs. For the sake of readability, we abuse the notation and we always denote by w the solutions of these BVPs.

We recall that for two generic functions $q_1,\ q_2 \in L^4(\Omega)$ the Cauchy-Schwarz inequality implies

$$\|q_1 q_2\|_{L^2(\Omega)} \leq \|q_1\|_{L^4(\Omega)} \|q_2\|_{L^4(\Omega)} \,. \tag{2.47}$$

By the triangle inequality, (2.47) and the Sobolev Imbedding Theorem [1, Thm 4.12], we bound the source function in (2.45) by

$$\begin{aligned} &\| \operatorname{div} \big((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla p \big) \|_{L^2(\Omega)} \\ &\qquad \leq C \big(\|\mathcal{V}\|_{W^{2,4}(\Omega)} \|p\|_{W^{1,4}(\Omega)} + \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|p\|_{H^2(\Omega)} \big) \,, \\ &\qquad \leq C \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|p\|_{H^2(\Omega)} \,. \end{aligned}$$

By Assumption 2.2.1, w is in $H^2(\Omega)$ and it satisfies

$$\|w\|_{H^2(\Omega)} \leq C \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|p\|_{H^2(\Omega)} \,. \tag{2.48}$$

By exploiting the Galerkin orthogonality of $u - u_h$ to the finite dimensional trial space $V_h \subset H_0^1(\Omega)$, we derive the bound

$$\begin{aligned} &\Big| \int_\Omega \nabla(u - u_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla p \,\mathrm{d}\boldsymbol{x} \Big| \\ &\qquad = \Big| \int_\Omega \nabla(u - u_h) \cdot \nabla w + (u - u_h) w \,\mathrm{d}\boldsymbol{x} \Big| \,, \\ &\qquad = \Big| \int_\Omega \nabla(u - u_h) \cdot \nabla(w - \mathcal{I}_h w) + (u - u_h)(w - \mathcal{I}_h w) \,\mathrm{d}\boldsymbol{x} \Big| \,, \\ &\qquad \leq \|u - u_h\|_{H^1(\Omega)} \|w - \mathcal{I}_h w\|_{H^1(\Omega)} \,. \end{aligned} \tag{2.49}$$

Then by (2.41) and the standard finite element convergence estimate [15, Ch. II, Sect. 7]

$$\|u - u_h\|_{H^1(\Omega)} \leq Ch \|u\|_{H^2(\Omega)} \,, \tag{2.50}$$

we conclude from (2.48)

$$\Big| \int_\Omega \nabla(u - u_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla p \,\mathrm{d}\boldsymbol{x} \Big| \leq Ch^2 \|u\|_{H^2(\Omega)} \|p\|_{H^2(\Omega)} \|\mathcal{V}\|_{W^{2,4}(\Omega)} \,.$$

Similarly, for the second integral on the right-hand side of (2.44) we introduce the function w as weak solution of the adjoint BVP

$$\left\{ \begin{array}{rll} -\Delta w + w &= -\operatorname{div} \big((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla u \big) & \text{in } \Omega \,, \\ w &= 0 & \text{on } \partial\Omega \,, \end{array} \right. \tag{2.51}$$

that is,

$$\int_\Omega \nabla w \cdot \nabla v + wv \,\mathrm{d}\boldsymbol{x} = \int_\Omega \left((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla u\right) \cdot \nabla v \quad \forall\, v \in H_0^1(\Omega)\,. \tag{2.52}$$

Assumption 2.2.1 and the bound

$$\| \operatorname{div}\left((\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla u\right) \|_{L^2(\Omega)} \leq C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|u\|_{H^2(\Omega)}$$

imply that $w \in H^2(\Omega)$ and that it satisfies

$$\|w\|_{H^2(\Omega)} \leq C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|u\|_{H^2(\Omega)}\,. \tag{2.53}$$

Next, we note that for every $v_h \in V_h$

$$\int_\Omega \nabla(p - p_h) \cdot \nabla v_h + (p - p_h)v_h \,\mathrm{d}\boldsymbol{x} = \int_\Omega (j'(u) - j'(u_h))v_h \,\mathrm{d}\boldsymbol{x}\,, \tag{2.54}$$

which implies

$$\begin{aligned}
&|\int_\Omega \nabla(p - p_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T)\nabla u \,\mathrm{d}\boldsymbol{x}| = |\int_\Omega \nabla(p - p_h) \cdot \nabla w + (p - p_h)w \,\mathrm{d}\boldsymbol{x}|\,,\\
\leq&|\int_\Omega \nabla(p - p_h) \cdot \nabla(w - \mathcal{I}_h w) + (p - p_h)(w - \mathcal{I}_h w) \,\mathrm{d}\boldsymbol{x}|\\
&+ |\int_\Omega \left(j'(u) - j'(u_h)\right)\mathcal{I}_h w \,\mathrm{d}\boldsymbol{x}|\,,\\
\leq&\|p - p_h\|_{H^1(\Omega)}\|w - \mathcal{I}_h w\|_{H^1(\Omega)} + \|\mathcal{I}_h w\|_{L^2(\Omega)}\|j'(u) - j'(u_h)\|_{L^2(\Omega)}\,.
\end{aligned} \tag{2.55}$$

For the concrete BVP considered the state solution u will belong to $C^0(\overline{\Omega})$. Further, $L^\infty(\Omega)$-estimates for finite element solutions [15, Ch. II, Sect. 7] ensure that $\|u - u_h\|_{L^\infty(\Omega)} \to 0$ as $h \to 0$. Hence, we can take for granted that there are h-independent bounds $\underline{u}$ and $\overline{u}$

$$-\infty < \underline{u} \leq u(\mathbf{x}), u_h(\mathbf{x}) \leq \overline{u} < \infty \quad \forall \mathbf{x} \in \Omega\,. \tag{2.56}$$

We write $I := [\underline{u}, \overline{u}]$ and point out that j' is locally Lipschitz continuous on $\mathbb{R}$ (and thus uniformly Lipschitz continuous on the compact interval I). Thus

$$\begin{aligned}
\|j'(u) - j'(u_h)\|_{L^2(\Omega)} &\leq \|j'\|_{C^{0,1}(I)}\|u - u_h\|_{L^2(\Omega)}\,,\\
&\leq Ch^2\|j'\|_{C^{0,1}(I)}\|u\|_{H^2(\Omega)}\,,
\end{aligned} \tag{2.57}$$

where the last step follows from the standard finite element convergence estimate [15, Ch. II, Sect. 7]

$$\|u - u_h\|_{L^2(\Omega)} \leq Ch^2 \|u\|_{H^2(\Omega)} \,. \tag{2.58}$$

In order to establish a bound for (2.55), we follow the arguments in the proof of Strang's first lemma [15, Ch. III, Thm. 1.1]. We note that for every $v_h \in V_h$

$$\begin{aligned} \|p_h - v_h\|^2_{H^1(\Omega)} &= \int_\Omega \nabla(p_h - p) \cdot \nabla(p_h - v_h) + (p_h - p)(p_h - v_h) \,\mathrm{d}\boldsymbol{x} \\ &\quad + \int_\Omega \nabla(p - v_h) \cdot \nabla(p_h - v_h) + (p - v_h)(p_h - v_h) \,\mathrm{d}\boldsymbol{x} \\ &\leq \left(\|j'(u_h) - j'(u)\|_{L^2(\Omega)} + \|p - v_h\|_{H^1(\Omega)} \right) \|p_h - v_h\|_{H^1(\Omega)} \,, \end{aligned} \tag{2.59}$$

where in the last step we used (2.54) and the Cauchy-Schwarz inequality. Then by the triangle inequality, (2.59) and (2.41),

$$\begin{aligned} \|p - p_h\|_{H^1(\Omega)} &\leq \|p - \mathcal{I}_h p\|_{H^1(\Omega)} + \|\mathcal{I}_h p - p_h\|_{H^1(\Omega)} \,, \\ &\leq 2\|p - \mathcal{I}_h p\|_{H^1(\Omega)} + \|j'(u_h) - j'(u)\|_{L^2(\Omega)} \,, \\ &\leq Ch\|p\|_{H^2(\Omega)} + Ch^2 \|j'\|_{C^{0,1}(I)} \|u\|_{H^2(\Omega)} \,, \end{aligned} \tag{2.60}$$

which implies

$$\begin{aligned} &\left| \int_\Omega \nabla(p - p_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla u \,\mathrm{d}\boldsymbol{x} \right| \\ &\qquad \leq Ch^2 \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|u\|_{H^2(\Omega)} (\|p\|_{H^2(\Omega)} + \|u\|_{H^2(\Omega)} \|j'\|_{C^{0,1}(I)}) \,. \end{aligned}$$

Finally, by the Cauchy-Schwarz inequality, (2.50) and (2.60), the following bound for the third integral on the right-hand side of (2.44) holds.

$$\begin{aligned} &\left| \int_\Omega \nabla(u - u_h) \cdot (\mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{V}^T) \nabla(p - p_h) \,\mathrm{d}\boldsymbol{x} \right| \\ &\qquad \leq \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|u - u_h\|_{H^1(\Omega)} \|p - p_h\|_{H^1(\Omega)} \,, \\ &\qquad \leq Ch^2 \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|u\|_{H^2(\Omega)} \|p\|_{H^2(\Omega)} \,. \end{aligned}$$

To bound the second integral on the right-hand side of (2.43) we introduce the function w as weak solution of the adjoint BVP

$$\left\{ \begin{array}{rcll} -\Delta w + w & = & -\operatorname{div}(f\mathcal{V}) & \text{in } \Omega\,, \\ w & = & 0 & \text{on } \partial\Omega\,, \end{array} \right. \tag{2.61}$$

that is,

$$\int_\Omega \nabla w \cdot \nabla v + wv \,\mathrm{d}\boldsymbol{x} = \int_\Omega f\mathcal{V} \cdot \nabla v \quad \forall\, v \in H_0^1(\Omega)\,. \tag{2.62}$$

Note that

$$\| -\operatorname{div}(f\mathcal{V}) \|_{L^2(\Omega)} \le \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|f\|_{H^1(\Omega)}\,,$$

which implies that w is in $H^2(\Omega)$ and that it satisfies

$$\|w\|_{H^2(\Omega)} \le C \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|f\|_{H^1(\Omega)}\,. \tag{2.63}$$

Then by (2.62), (2.54), (2.60), (2.41), and (2.58),

$$\begin{aligned} |\int_\Omega f\mathcal{V} \cdot \nabla(p - p_h) \,\mathrm{d}\boldsymbol{x}| &= |\int_\Omega \nabla(p - p_h) \cdot \nabla w + (p - p_h) w \,\mathrm{d}\boldsymbol{x}|\,, \\ &\le \|p - p_h\|_{H^1(\Omega)} \|w - \mathcal{I}_h w\|_{H^1(\Omega)} + \|\mathcal{I}_h w\|_{L^2(\Omega)} \|j'\|_{C^{0,1}(I)} \|u - u_h\|_{L^2(\Omega)}\,, \\ &\le Ch^2 \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|f\|_{H^1(\Omega)} \left(\|p\|_{H^2(\Omega)} + \|u\|_{H^2(\Omega)} \|j'\|_{C^{0,1}(I)} \right)\,. \end{aligned}$$

To bound the third integral on the right-hand side of (2.43), we first apply the triangle inequality

$$\begin{aligned} |\int_\Omega \operatorname{div} \mathcal{V}(j(u) - j(u_h) - \nabla u \cdot \nabla p - up + \nabla u_h \cdot \nabla p_h + u_h p_h) \,\mathrm{d}\boldsymbol{x}| \\ \le |\int_\Omega \operatorname{div} \mathcal{V}(j(u) - j(u_h)) \,\mathrm{d}\boldsymbol{x}| \\ + |\int_\Omega \operatorname{div} \mathcal{V}(\nabla u \cdot \nabla p + up - \nabla u_h \cdot \nabla p_h - u_h p_h) \,\mathrm{d}\boldsymbol{x}|\,. \end{aligned} \tag{2.64}$$

The first integral on the right-hand side of (2.64) can be bounded by

$$\begin{aligned} |\int_\Omega \operatorname{div} \mathcal{V}(j(u) - j(u_h)) \,\mathrm{d}\boldsymbol{x}| &\le C \|\mathcal{V}\|_{W^{1,\infty}} \|j'\|_{C^0(I)} \|u - u_h\|_{L^2(\Omega)}\,, \\ &\le Ch^2 \|\mathcal{V}\|_{W^{1,\infty}} \|j'\|_{C^0(I)} \|u\|_{H^2(\Omega)}\,, \end{aligned} \tag{2.65}$$

whereas the second one can conveniently be rewritten as

$$\begin{aligned}
&\Big|\int_\Omega \operatorname{div}\mathcal{V}(\nabla u\cdot\nabla p + up - \nabla u_h\cdot\nabla p_h - u_h p_h)\,\mathrm{d}\boldsymbol{x}\Big| \\
&\quad= \Big|\int_\Omega \operatorname{div}\mathcal{V}\left(\nabla(u-u_h)\cdot\nabla p + (u-u_h)p\right)\,\mathrm{d}\boldsymbol{x} \\
&\qquad+ \int_\Omega \operatorname{div}\mathcal{V}\left(\nabla u\cdot\nabla(p-p_h) + u(p-p_h)\right)\,\mathrm{d}\boldsymbol{x} \\
&\qquad- \int_\Omega \operatorname{div}\mathcal{V}\left(\nabla(u-u_h)\cdot\nabla(p-p_h) + (u-u_h)(p-p_h)\right)\,\mathrm{d}\boldsymbol{x}\Big|\,.
\end{aligned} \tag{2.66}$$

Again, the first two integrals on the right-hand side of (2.66) can be bounded with standard duality techniques. For the first one we introduce the function w as weak solution of the adjoint BVP

$$\left\{\begin{array}{rll} -\Delta w + w &= -\operatorname{div}\left(\operatorname{div}(\mathcal{V})\nabla p\right) + \operatorname{div}(\mathcal{V})p & \text{in } \Omega\,, \\ w &= 0 & \text{on } \partial\Omega\,, \end{array}\right. \tag{2.67}$$

that is,

$$\int_\Omega \nabla w\cdot\nabla v + wv\,\mathrm{d}\boldsymbol{x} = \int_\Omega \operatorname{div}(\mathcal{V})\left(\nabla p\cdot\nabla v + pv\right)\,\mathrm{d}\boldsymbol{x} \quad \forall\, v\in H_0^1(\Omega)\,. \tag{2.68}$$

Assumption 2.2.1 and the bound

$$\|-\operatorname{div}\left(\operatorname{div}(\mathcal{V})\nabla p\right) + \operatorname{div}(\mathcal{V})p\|_{L^2(\Omega)} \le C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|p\|_{H^2(\Omega)}$$

imply that w is in $H^2(\Omega)$ and that it satisfies

$$\|w\|_{H^2(\Omega)} \le C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|p\|_{H^2(\Omega)}\,. \tag{2.69}$$

Then by (2.68), Galerkin orthogonality of $u-u_h$ to V_h, the Cauchy-Schwarz inequality, (2.50), and (2.41),

$$\begin{aligned}
&\Big|\int_\Omega \operatorname{div}\mathcal{V}\left(\nabla p\cdot\nabla(u-u_h) + p(u-u_h)\right)\,\mathrm{d}\boldsymbol{x}\Big| \\
&\quad= \int_\Omega \nabla w\cdot\nabla(u-u_h) + w(u-u_h)\,\mathrm{d}\boldsymbol{x} \le \|u-u_h\|_{H^1(\Omega)}\|w-\mathcal{I}_h w\|_{H^1(\Omega)}\,, \\
&\quad\le Ch^2\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|p\|_{H^2(\Omega)}\|u\|_{H^2(\Omega)}\,.
\end{aligned}$$

For the second integral on the right-hand side of (2.66) we introduce the function w as weak solution of the adjoint BVP

$$\left\{ \begin{array}{rll} -\Delta w + w &= -\operatorname{div}\left(\operatorname{div}(\mathcal{V})\nabla u\right) + \operatorname{div}(\mathcal{V})u & \text{in } \Omega\,, \\ w &= 0 & \text{on } \partial\Omega\,, \end{array} \right. \tag{2.70}$$

that is,

$$\int_\Omega \nabla w \cdot \nabla v + wv \,\mathrm{d}\boldsymbol{x} = \int_\Omega \operatorname{div}(\mathcal{V})\big(\nabla u \cdot \nabla v + uv\big) \,\mathrm{d}\boldsymbol{x} \quad \forall\, v \in H^1_0(\Omega)\,. \tag{2.71}$$

Assumption 2.2.1 and the bound

$$\| - \operatorname{div}\left(\operatorname{div}(\mathcal{V})\nabla u\right) + \operatorname{div}(\mathcal{V})u\|_{L^2(\Omega)} \leq C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|u\|_{H^2(\Omega)}$$

imply that w is in $H^2(\Omega)$ and that it satisfies

$$\|w\|_{H^2(\Omega)} \leq C\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|u\|_{H^2(\Omega)}\,. \tag{2.72}$$

Then, by (2.71),(2.54), the Cauchy-Schwarz inequality, (2.60), (2.41), and (2.58),

$$\begin{aligned} &\Big|\int_\Omega \operatorname{div}\mathcal{V}\left(\nabla u \cdot \nabla(p-p_h) + u(p-p_h)\right) \mathrm{d}\boldsymbol{x}\Big| \\ &\quad = \Big|\int_\Omega \nabla(p-p_h)\cdot\nabla w + (p-p_h)w \,\mathrm{d}\boldsymbol{x}\Big|\,, \\ &\quad \leq \|p-p_h\|_{H^1(\Omega)}\|w - \mathcal{I}_h w\|_{H^1(\Omega)} + \|\mathcal{I}_h w\|_{L^2(\Omega)}\|j'\|_{C^{0,1}(I)}\|u-u_h\|_{L^2(\Omega)}\,, \\ &\quad \leq Ch^2\|\mathcal{V}\|_{W^{2,4}(\Omega)}\|u\|_{H^2(\Omega)}\left(\|p\|_{H^2(\Omega)} + \|u\|_{H^2(\Omega)}\|j'\|_{C^{0,1}(I)}\right)\,. \end{aligned}$$

By the Cauchy-Schwarz inequality, (2.50), and (2.60), we obtain the following bound for the third integral on the right-hand side of (2.66):

$$\begin{aligned} &\Big|\int_\Omega \operatorname{div}\mathcal{V}\left(\nabla(u-u_h)\cdot\nabla(p-p_h) + (u-u_h)(p-p_h)\right) \mathrm{d}\boldsymbol{x}\Big| \\ &\qquad\qquad \leq \|\mathcal{V}\|_{W^{1,\infty}(\Omega)}\|u-u_h\|_{H^1(\Omega)}\|p-p_h\|_{H^1(\Omega)}\,, \\ &\qquad\qquad \leq Ch^2\|\mathcal{V}\|_{W^{1,\infty}(\Omega)}\|u\|_{H^2(\Omega)}\left(\|p\|_{H^2(\Omega)} + h\|j'\|_{C^{0,1}(I)}\|u\|_{H^2(\Omega)}\right)\,. \end{aligned}$$

The fourth integral on the right-hand side of (2.43) can be bounded similarly as in (2.65), relying on $L^\infty(\Omega)$-estimates for finite element solutions. The uniform Lipschitz continuity of j' on the compact interval I yields

$$\left|\int_\Omega (j'(u) - j'(u_h) - p + p_h)(\nabla g \cdot \mathcal{V})\,\mathrm{d}\boldsymbol{x}\right| \\ \leq \|\mathcal{V}\|_{L^\infty(\Omega)}\|g\|_{H^1(\Omega)}\left(\|j'\|_{C^{0,1}(I)}\|u - u_h\|_{L^2(\Omega)} + \|p - p_h\|_{L^2(\Omega)}\right).$$

Next, we derive an estimate of $\|p - p_h\|_{L^2(\Omega)}$. Let w be the weak solution of the adjoint BVP

$$\begin{cases} -\Delta w + w &= p - p_h \quad \text{in } \Omega\,, \\ w &= 0 \quad \text{on } \partial\Omega\,. \end{cases}$$

Assumption 2.2.1 implies

$$\|w\|_{H^2(\Omega)} \leq C\|p - p_h\|_{L^2(\Omega)}\,. \tag{2.73}$$

Then, by (2.54), (2.60),(2.41), (2.73), and (2.57), we have

$$\begin{aligned} \|p - p_h\|^2_{L^2(\Omega)} &= \int_\Omega (p - p_h)^2\,\mathrm{d}\boldsymbol{x} = \int_\Omega \nabla w \cdot \nabla(p - p_h) + w(p - p_h)\,\mathrm{d}\boldsymbol{x}\,, \\ &= \int_\Omega \nabla(w - \mathcal{I}_h w) \cdot \nabla(p - p_h) + (w - \mathcal{I}_h w)(p - p_h)\,\mathrm{d}\boldsymbol{x} \\ &\quad + \int_\Omega (j'(u) - j'(u_h))\mathcal{I}_h w\,\mathrm{d}\boldsymbol{x}\,, \\ &\leq \|p - p_h\|_{H^1(\Omega)}\|w - \mathcal{I}_h w\|_{H^1(\Omega)} + \|\mathcal{I}_h w\|_{L^2(\Omega)}\|j'(u) - j'(u_h)\|_{L^2(\Omega)} \\ &\leq \|p - p_h\|_{L^2(\Omega)}\left(Ch^2\left(\|p\|_{H^2(\Omega)} + \|j'\|_{C^{0,1}(I)}\|u\|_{H^2(\Omega)}\right)\right), \end{aligned}$$

and thus,

$$\|p - p_h\|_{L^2(\Omega)} \leq Ch^2\left(\|p\|_{H^2(\Omega)} + \|j'\|_{C^{0,1}(I)}\|u\|_{H^2(\Omega)}\right) \tag{2.74}$$

With the estimates (2.58) and (2.74), we conclude

$$\left|\int_\Omega (j'(u) - j'(u_h) - p + p_h)(\nabla g \cdot \mathcal{V})\,\mathrm{d}\boldsymbol{x}\right| \\ \leq Ch^2\|\mathcal{V}\|_{L^\infty(\Omega)}\|g\|_{H^1(\Omega)}\left(\|p\|_{H^2(\Omega)} + \|j'\|_{C^{0,1}(I)}\|u\|_{H^2(\Omega)}\right).$$

Finally, the fifth integral on the right-hand side of (2.43) can be bounded with standard duality techniques by introducing the function w as weak solution of the adjoint BVP

$$\begin{cases} -\Delta w + w &= -\Delta\left(\nabla g \cdot \mathcal{V}\right) \quad \text{in } \Omega\,, \\ w &= 0 \quad \text{on } \partial\Omega\,, \end{cases} \tag{2.75}$$

that is,

$$\int_\Omega \nabla w \cdot \nabla v + wv \,\mathrm{d}\boldsymbol{x} = \int_\Omega \nabla \left(\nabla g \cdot \mathcal{V}\right) \cdot \nabla v \,\mathrm{d}\boldsymbol{x} \quad \forall\, v \in H_0^1(\Omega)\,. \tag{2.76}$$

Assumption 2.2.1 and the bound

$$\begin{aligned}
&\|\Delta \left(\nabla g \cdot \mathcal{V}\right)\|_{L^2(\Omega)} \\
&\quad \le C\Big(\|\mathcal{V}\|_{L^\infty(\Omega)} \|g\|_{H^3(\Omega)} + \|\mathcal{V}\|_{W^{1,\infty}(\Omega)} \|g\|_{H^2(\Omega)} + \|\mathcal{V}\|_{H^2(\Omega)} \|g\|_{W^{1,\infty}(\Omega)} \Big) \\
&\quad \le C \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|g\|_{H^3(\Omega)}
\end{aligned}$$

imply that w is in $H^2(\Omega)$ and that it satisfies

$$\|w\|_{H^2(\Omega)} \le C \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|g\|_{H^3(\Omega)}\,. \tag{2.77}$$

Then by (2.76), (2.54), (2.60), (2.41), and (2.58),

$$\begin{aligned}
&\Big| \int_\Omega \nabla(\nabla g \cdot \mathcal{V}) \cdot \nabla(p - p_h) \,\mathrm{d}\boldsymbol{x} \Big| \\
&\quad = \int_\Omega \nabla w \cdot \nabla(p - p_h) + w(p - p_h) \,\mathrm{d}\boldsymbol{x} \\
&\quad \le \|p - p_h\|_{H^1(\Omega)} \|w - \mathcal{I}_h w\|_{H^1(\Omega)} + \|\mathcal{I}_h w\|_{L^2(\Omega)} \|j\|_{C^{1,1}(I)} \|u - u_h\|_{L^2(\Omega)}\,, \\
&\quad \le C h^2 \|\mathcal{V}\|_{W^{2,4}(\Omega)} \|g\|_{H^3(\Omega)} \left(\|p\|_{H^2(\Omega)} + \|u\|_{H^2(\Omega)} \|j\|_{C^{1,1}(I)} \right)\,.
\end{aligned}$$

□

Remark 2.2.5. *Theorem 2.2.4 can be extended to the energy functional*

$$\mathcal{J} = \int_\Omega \nabla u \cdot \nabla u \,\mathrm{d}\boldsymbol{x}\,.$$

In this case, the right-hand side of the adjoint equation (2.9) *becomes the linear continuous map*

$$H^1(\Omega) \ni \phi \mapsto \int_\Omega \nabla u \cdot \nabla \phi \,\mathrm{d}\boldsymbol{x}$$

The proof of the quadratic rate of convergence for piecewise linear Lagrangian finite elements follows the line of the proof of Theorem 2.2.4 with few modifications.

1) The right-hand side in (2.54) *becomes*

$$\int_\Omega \nabla(u-u_h)\cdot\nabla v_h \,\mathrm{d}\boldsymbol{x}\,.$$

Galerkin orthogonality of $u-u_h$ *to the finite-dimensional trial space* $V_h \subset H_0^1(\Omega)$ *yields*

$$\int_\Omega \nabla(u-u_h)\cdot\nabla v_h \,\mathrm{d}\boldsymbol{x} = -\int_\Omega (u-u_h)v_h \,\mathrm{d}\boldsymbol{x}\,,$$

and the term

$$\int_\Omega \big(j'(u)-j'(u_h)\big)\,\mathcal{I}_h w \,\mathrm{d}\boldsymbol{x}$$

that appears in several instances in the proof can be replaced (and estimated) by

$$\int_\Omega \nabla(u-u_h)\cdot\nabla\mathcal{I}_h w \,\mathrm{d}\boldsymbol{x} \le \|\mathcal{I}_h w\|_{L^2(\Omega)}\|u-u_h\|_{L^2(\Omega)} \le Ch^2\|u\|^2_{H^1(\Omega)}\,.$$

2) To estimate the term

$$\int_\Omega \operatorname{div}\mathcal{V}(j(u)-j(u_h))\,\mathrm{d}\boldsymbol{x}$$

in (2.64)*, we first reformulate*

$$\int_\Omega \operatorname{div}\mathcal{V}(\nabla u\cdot\nabla u-\nabla u_h\cdot\nabla u_h)\,\mathrm{d}\boldsymbol{x} = \int_\Omega 2\operatorname{div}\mathcal{V}\nabla(u-u_h)\cdot\nabla u\,\mathrm{d}\boldsymbol{x} - \int_\Omega \operatorname{div}\mathcal{V}(\nabla(u-u_h))^2\,\mathrm{d}\boldsymbol{x} \quad (2.78)$$

The first integral in (2.78) *can be estimated with standard duality techniques (repeating the procedure used in* (2.45)-(2.49) *above) because* $u\in H^2(\Omega)$ $\mathcal{I}_h w\in H^1(\Omega)$ *, whilst for the second term we employ the Cauchy-Schwarz inequality and* (2.50)

3) The term

$$\int_\Omega (j'(u)-j'(u_h))(\nabla g\cdot\mathcal{V})\,\mathrm{d}\boldsymbol{x}$$

that appears in the fourth integral on the right-hand side of (2.43) *becomes*

$$\int_\Omega (\nabla u-\nabla u_h))\cdot\nabla(\nabla g\cdot\mathcal{V})\,\mathrm{d}\boldsymbol{x}$$

and can be estimated as the fifth integral on the right-hand side of (2.43)*.*

Remark 2.2.6. *The shape gradient formula* (2.10) *clearly represents a linear continuous operator on* $C^1(\mathbb{R}^d)$*. Nevertheless, to exploit finite element superconvergence as in Theorem* (2.2.4)*, we have to*[7] *restrict ourselves to vector fields in* $W^{2,4}(\mathbb{R}^d)$*. If this condition is violated, only first order convergence of* $d\mathcal{J}(\Omega, u_h, p_h; \mathcal{V})^{\mathrm{Vol}}$ *to* $d\mathcal{J}(\Omega; \mathcal{V})$ *as* $h \to 0$ *can be shown, because the bounds* (2.48),(2.53),(2.69), (2.72)*, and* (2.77) *are no longer available. Note also that Theorem* (2.2.4) *neglects errors introduced by numerical quadrature. Standard approach to estimate quadrature errors relies on the Bramble-Hilbert Lemma [20, Thm 4.1.3], which requires sufficient regularity of the integrands. For instance, we focus on the term*

$$\int_\Omega \operatorname{div}(\mathcal{V}) u_h p_h \,\mathrm{d}\boldsymbol{x} \tag{2.79}$$

from (2.36)*, where we replaced the functions* u *and* p *with their finite element counterparts. The impact of numerical quadrature on* (2.79) *can be studied following closely [20, Thm 4.1.5]. To achieve quadratic convergence with respect to the meshwidht* h*, the quadrature rule employed should be exact for linear functions and the term* $\operatorname{div}(\mathcal{V})$ *should be in* $W^{2,\infty}(\Omega)$*. Thus, numerical quadrature imposes even stricter requirements on the regularity of the vector field* $\mathcal{V}$*.*

Remark 2.2.7. *The quadratic rate of convergence in Theorem 2.2.4 depends on the regularity of the functions* u *and* p*. If the assumption on the* H^2*-regularity of* (2.7) *is not fulfilled, the provable rate of convergence deteriorates to* $O(h^\alpha)$ *with fractional* $\alpha < 2$*, but the formula* (2.36) *remains meaningful, as long as a weak solutions in* $H^1(\Omega)$ *exist. On the other hand, if the functions* u *and* p *enjoy higher smoothness, the convergence may be improved by increasing the polynomial degree of the finite element space.*

Remark 2.2.8. *Theorem 2.2.4 holds true for Dirichlet boundary conditions only. However, a similar result can be achieved for Neumann boundary conditions. The proof follows the same lines as for the Dirichlet case and relies on* H^2*-regularity of the state problem and regularity assumptions on the source function* f *and the boundary data* g*. In particular, convergence for the*

[7] Since $H^1(\Omega) \hookrightarrow L^6(\Omega)$, one could replace the Cauchy-Schwarz inequality in (2.47) with the Hölder Inequality. The estimate (2.42) would still hold true, but with $\|\mathcal{V}\|_{W^{2,4}(\Omega)}$ replaced by $\|\mathcal{V}\|^2_{W^{2,3}(\Omega)}$. This is not desirable because (2.42) would not estimate the approximation error in an operator norm.

boundary term in (2.12) *can be concluded either via duality techniques or by continuity of the Dirichlet trace operator with respect to* $H^1(\Omega)$.

For Formula (2.37), the following holds:

Theorem 2.2.9. *Let* u_h *and* p_h *be Ritz-Galerkin linear Lagrangian finite element approximations of the solutions* u *and* p *of* (2.7) *and* (2.9). *In addition to the hypothesis of Theorem 2.2.4, let us assume that*

$$\|u\|_{W^{2,q}(\Omega)} \leq C\, q\, \|f\|_{L^\infty(\Omega)} \quad \text{for all } q \in \mathbb{N}\,. \tag{2.80}$$

Then

$$|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}| \leq C|\log h|h\|\mathcal{V}\cdot\mathbf{n}\|_{L^\infty(\partial\Omega)}\,,$$

where h *stands for the meshwidth, and* $C>0$ *does not depend on* h.

Proof. By the equality $d\mathcal{J}(\Omega;\mathcal{V}) = d\mathcal{J}(\Omega,u,p;\mathcal{V})^{\mathrm{Bdry}}$, we immediately deduce from (2.37)

$$\begin{aligned}
&|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}| \\
&\quad\leq \|\mathcal{V}\cdot\mathbf{n}\|_{L^\infty(\Omega)} \int_{\partial\Omega} \left| j(u) - j(u_h) + \frac{\partial p}{\partial \mathbf{n}}\frac{\partial(u-g)}{\partial\mathbf{n}} - \frac{\partial p_h}{\partial\mathbf{n}}\frac{\partial(u_h-g)}{\partial\mathbf{n}} \right| dS\,.
\end{aligned} \tag{2.81}$$

By linearity, and similarly as in (2.44), we find

$$\begin{aligned}
&\frac{\partial p}{\partial\mathbf{n}}\frac{\partial(u-g)}{\partial\mathbf{n}} - \frac{\partial p_h}{\partial\mathbf{n}}\frac{\partial(u_h-g)}{\partial\mathbf{n}} \\
&= \frac{\partial p}{\partial\mathbf{n}}\frac{\partial u}{\partial\mathbf{n}} - \frac{\partial p_h}{\partial\mathbf{n}}\frac{\partial u_h}{\partial\mathbf{n}} + \frac{\partial p_h}{\partial\mathbf{n}}\frac{\partial g}{\partial\mathbf{n}} - \frac{\partial p}{\partial\mathbf{n}}\frac{\partial g}{\partial\mathbf{n}} \\
&= \frac{\partial p}{\partial\mathbf{n}}\frac{\partial(u-u_h)}{\partial\mathbf{n}} + \frac{\partial(p-p_h)}{\partial\mathbf{n}}\frac{\partial u_h}{\partial\mathbf{n}} + \frac{\partial(p_h-p)}{\partial\mathbf{n}}\frac{\partial g}{\partial\mathbf{n}}\,.
\end{aligned}$$

Therefore, the estimate of the theorem follows straightforwardly from (2.81), the triangle inequality, and the following finite element error estimates in $W^{1,\infty}(\Omega)$:

$$\|u-u_h\|_{W^{1,\infty}(\Omega)} \leq C|\log h|h\|f\|_{L^\infty(\Omega)}\,, \tag{2.82}$$

$$\|p-p_h\|_{W^{1,\infty}(\Omega)} \leq C|\log h|h\|u\|_{L^\infty(\Omega)}\,. \tag{2.83}$$

The estimate (2.82) follows from stability of the Ritz projection with respect to $W^{1,\infty}(\Omega)$ [16, Thm 8.1.11], the interpolation estimate

$$\|u - \mathcal{I}_h u\|_{W^{1,\infty}(\Omega)} \leq C h^{1-d/q} \|u\|_{W^{2,q}(\Omega)} ,$$

and the continuity assumption (2.80) (with $q = |\log h|$). To show

$$\|p_h\|_{W^{1,\infty}(\Omega)} \leq C \|p\|_{W^{1,\infty}(\Omega)}, \tag{2.84}$$

which in turn implies the second estimate in (2.83), it is necessary to repeat the proof of [16, Thm 8.1.11] tracking the consistency term

$$\int_\Omega \nabla(p - p_h) \cdot \nabla g_h^z + (p - p_h) g_h^z \,\mathrm{d}\boldsymbol{x} , \tag{2.85}$$

where the function $g_h^z \in V_h$ is given in [16, Sect 8.2] and satisfies

$$\|g_h^z\|_{H^1(\Omega)} = \mathcal{O}(h^{-d/2}) .$$

More specifically, we have to show that the consistency term (2.85) is bounded independently of h. However, this follows from (2.9) and (2.40), the Cauchy-Schwarz inequality and (2.57),

$$\left| \int_\Omega \nabla(p - p_h) \cdot \nabla g_h^z + (p - p_h) g_h^z \,\mathrm{d}\boldsymbol{x} \right| = \left| \int_\Omega (j'(u) - j'(u_h)) g_h^z \,\mathrm{d}\boldsymbol{x} \right| = \mathcal{O}(h^{2-d/2}) .$$

□

Remark 2.2.10. *For $d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Bdry}}$ to be well-defined, the functions u and p must be smoother than merely belonging to $H^1(\Omega)$.*

2.3. Numerical experiments

The function space $C^1(\mathbb{R}^d; \mathbb{R}^d)$ is infinite-dimensional and nonreflexive. This makes it challenging to investigate numerically convergence rates with respect to the $C^1(\mathbb{R}^d; \mathbb{R}^d)$ dual norm. Therefore, we adopt two simplifications to provide numerical evidence of the theoretical results from section 2.2

Firstly, we consider only an operator norm over a finite dimensional space of vector fields of the form

$$\mathcal{V}(x, y) = \sum_{1 \leq m_1, m_2, n_1, n_2 \leq M} \lambda_{m_1, n_1} \begin{pmatrix} v(x, y, m_1, n_1) \\ 0 \end{pmatrix} + \lambda_{m_2, n_2} \begin{pmatrix} 0 \\ v(x, y, m_2, n_2) \end{pmatrix} \tag{2.86}$$

with $v(x,y,m,n) = \sin(m\,x)\sin(n\,y)$, $M = 5$, and $\lambda_{m_i,n_i} \in \mathbb{R}$ (we tacitly restrict ourselves to 2 dimensional problems).

Secondly, we replace the $C^1(\mathbb{R}^d;\mathbb{R}^d)$-norm with the $H^1(D)$-norm, which is more tractable computationally. The domain D is a square centered in zero. The length of its edges is a multiple of 2π and is chosen sufficiently large so that all domains Ω are subsets of D. Note that

$$\left\{v(x,y,m,n)\mathbf{e}_j\,; m,n \in \mathbb{N}^+, j = 1,2\right\} \tag{2.87}$$

is a basis of $C_0^1(\overline{D};\mathbb{R}^2)$

The convergence studies are performed monitoring the approximate dual norms

$$\mathrm{err}^{\mathrm{Vol}} := \max_{\|\mathcal{V}\|_{H^1(D)}=1} \left(d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}\right)$$

and

$$\mathrm{err}^{\mathrm{Bdry}} := \max_{\|\mathcal{V}\|_{H^1(D)}=1} \left(d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}\right)$$

on different meshes generated through uniform refinement.

To compute the values $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$, we introduce a basis $\{\mathcal{V}_i\}_{i=1}^{50}$, as in (2.87), and define the column vectors

$$\mathbf{z}^{\mathrm{Vol}} := \left(d\mathcal{J}(\Omega;\mathcal{V}_i) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V}_i)^{\mathrm{Vol}}\right)_{i=1}^{m},$$
$$\mathbf{z}^{\mathrm{Bdry}} := \left(d\mathcal{J}(\Omega;\mathcal{V}_i) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V}_i)^{\mathrm{Bdry}}\right)_{i=1}^{m}.$$

Let $\mathbf{M}$ be the Gramian matrix of $\{\mathcal{V}_i\}_{i=1}^{50}$ with respect to the $H^1(D)$ inner product. Then, up to a mesh-independent scaling factor, $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ can be computed by

$$(\mathbf{z}^{\mathrm{Vol}})^T\mathbf{M}^{-1}\mathbf{z}^{\mathrm{Vol}} \quad \text{and} \quad (\mathbf{z}^{\mathrm{Bdry}})^T\mathbf{M}^{-1}\mathbf{z}^{\mathrm{Bdry}},$$

respectively.

To compute an approximation of the reference value $d\mathcal{J}(\Omega;\mathcal{V})$, we evaluate both $d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}$ and $d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}$ on a mesh with an extra level of refinement. To avoid biased results we display convergence history both with self- and cross-comparison.

In the implementation, we opt for linear Lagrangian finite elements on quasi-uniform triangular meshes. The experiments are performed in MATLAB and

are partly based on the library LehrFEM developed at ETHZ. Mesh generation and uniform refinement are performed with the functions `initmesh` and `refinemesh` of the MATLAB PDE Toolbox [58]. The boundary of computational domains is approximated by a polygon, which is generally believed not to affect the convergence of linear finite elements [16, Sect. 10.2]. For domains with curved boundaries, the refined mesh is always adjusted to fit the boundary. Integrals in the domain are computed by a 3-point quadrature rule of order 3 in each triangle and line integrals with a 10-point Gauss quadrature on each segment.

As a first test case, we choose the quadratic shape functional

$$\mathcal{J}(\Omega) = \int_\Omega u^2 \,\mathrm{d}\boldsymbol{x},$$

constrained to the the scalar boundary value problem

$$\left\{ \begin{array}{rcll} -\Delta u + u &=& f & \text{in } \Omega\,, \\ u &=& g & \text{on } \partial\Omega\,. \end{array} \right. \tag{2.88}$$

We consider three different geometries for the domain Ω (see Figure 2.1):

1. A disc of radius $\sqrt{6/5}$ centered in (0.01,0.02).
2. A triangle with corners located at
$$(-\sqrt{6/5}, -\sqrt{6/5})\,, (\sqrt{6/5}, -\sqrt{6/5})\,, (-\sqrt{6/5}, \sqrt{6/5})\,.$$
3. A circular sector of radius $\sqrt{6/5}$ centered in $(\pi/4 + 0.01, \pi/4 + 0.02)$ of angle $0.9 \cdot 2\pi$.

The source function and the Dirichlet boundary conditions in (2.88) are

$$f(x, y) = \cos(x)\,, \quad g(x, y) = 1/3\,,$$

respectively. This choice is arbitrary. Numerical experiments for different choices of both source function and Dirichlet boundary conditions are reported in [45].

In Figure 2.2 (first row) we plot the convergence history when the domain Ω is a disc. As expected, the values of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ decrease quadratically and linearly with respect to the meshwidth h for, respectively. In the cross-comparison plot, the convergence line of $\mathrm{err}^{\mathrm{Vol}}$ saturates due to an insufficient

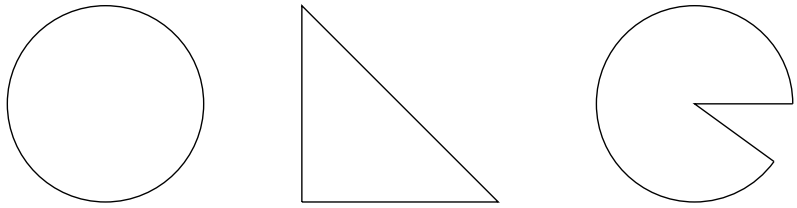

Figure 2.1.: The domain Ω is chosen to be either a disc or a triangle or a sector.

accuracy of the reference values computed with Formula (2.37). The same behavior is observed when the domain Ω is a triangle; see Figure 2.2 (second row).

To assess the impact of the restriction to a finite-dimensional space of vector fields, we repeat the convergence experiments by varying the parameter M in (2.86). In Figure 2.3, we display the convergence history for $M = 3, 4, 5$ when Ω is a disc. We observe that the discretization level does not affect the convergence behavior. This test has been repeated for all numerical experiments of this section and in no case we observed that the discretization parameter M affects the convergence rates. Thus, the arbitrary choice of fixing the discretization parameter M to 5 does not seem to compromise our observations.

In Figure 2.4 we plot the convergence history when the domain Ω is a sector. This domain does not guarantee H^2-regularity of the state problem (2.7) because it has a reentrant corner. We observe that the convergence rates of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ decrease to fractional values. This is a consequence of the lower regularity of the functions u and p. Additionally, the convergence rate of $\mathrm{err}^{\mathrm{Bdry}}$ differs between self- and cross-comparison. This may be due to a poor accuracy of the reference solution employed in the self-comparison plot. However, note that Formulas (2.36) and (2.37) may not be equivalent due to the lack of regularity of the functions u and p; cf. Remark 2.1.3.

Next, we replace the PDE constraint (2.88) with the homogeneous Neumann boundary value problem

$$\begin{cases} -\Delta u + u &= f \quad \text{in } \Omega\,, \\ \frac{\partial u}{\partial \mathbf{n}} &= 0 \quad \text{on } \partial\Omega\,. \end{cases} \tag{2.89}$$

and repeat the convergence experiments for the same quadratic shape functional. Again, the source function reads $f(x, y) = \cos(x)$. Numerical experiments for different choices of both source function and Neumann boundary

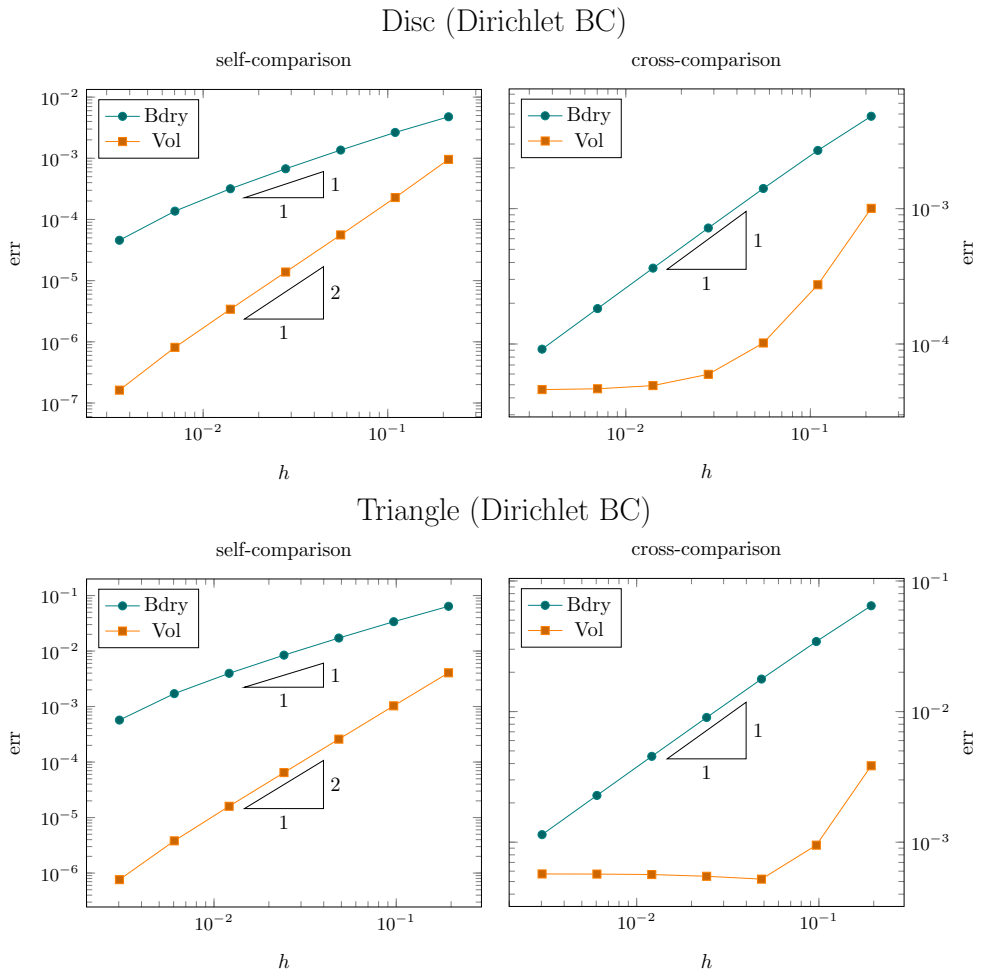

Figure 2.2.: Convergence history of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ when Ω is a disc (*first row*) and a triangle (*second row*). We observe the convergence rates predicted by Theorems 2.2.4 and 2.2.9.

conditions are reported in [45].

In Figure 2.5 (first row) we plot the convergence history when the domain Ω is a disc. Surprisingly, we observe that Formula (2.13) performs as well as Formula (2.12), showing quadratic convergence in the meshwidth h, too. This

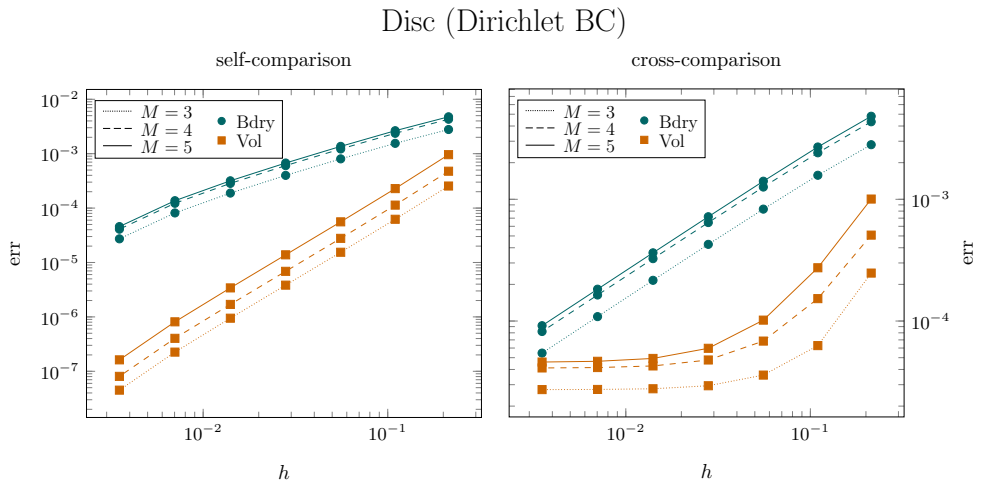

Figure 2.3.: Convergence history of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ when Ω is a disc for different values of the discretization parameter M. We observe that the discretization level does not affect the convergence behavior.

unpredicted behavior is also observed when the domain Ω is a triangle; see Figure 2.5 (second row). In this case, we observe that Formula (2.13) is even superior to Formula (2.12) on coarse meshes. This unpredicted superconvergence has already been observed in [45]. Note that there is no need to correct Formula (2.13) by adding the terms (2.35) because we consider homogeneous Neumann boundary conditions.

In Figure 2.6 we plot the convergence history when the domain Ω is a sector. This domain does not guarantee H^2-regularity of the state problem (2.7) because it has a reentrant corner. In the self-comparison plot, we observe that the convergence rates decrease to fractional values. This is a consequence of the lower regularity of the functions u and p. Similarly to the Dirichlet case, the convergence rate of $\mathrm{err}^{\mathrm{Bdry}}$ is smaller in the cross-comparison plot. This may be caused by insufficient accuracy of the reference solution employed in the self-comparison plot. However, note that, for this domain Ω Formulas (2.36) and (2.37) may not be equivalent; cf. Remark 2.1.3.

A closer look at Formula (2.13) reveals a cancellation of the normal deriva-

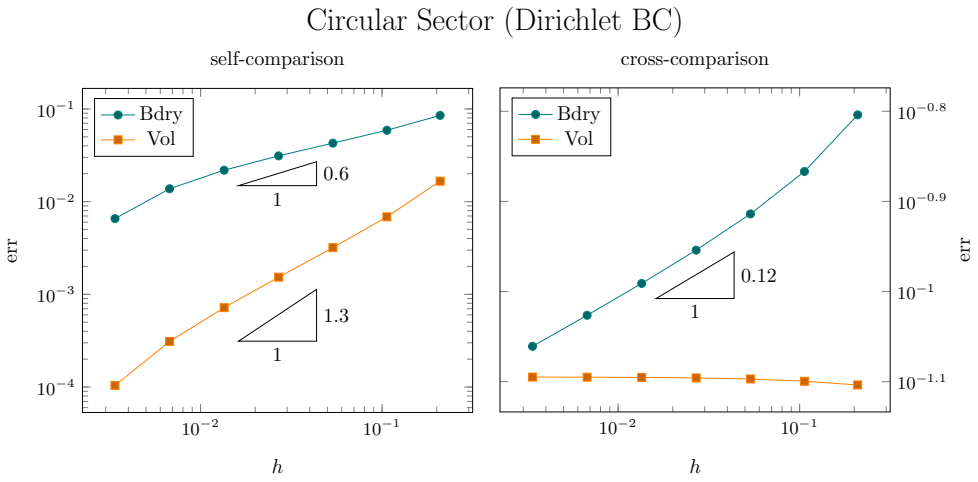

Figure 2.4.: Convergence history of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ when Ω is a sector. This domain does not guarantee H^2-regularity of the state problem (2.7), and it is questionable whether Formulas (2.36) and (2.37) are still equivalent.

tives of u and p, so that the formula is equivalent to

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}\left(j(u) - \nabla_\Gamma u \nabla_\Gamma p - up + fp + \mathrm{K}gp + p\frac{\partial g}{\partial \mathbf{n}}\right) dS\,, \quad (2.90)$$

where ∇_Γ stands for the tangential derivative. To elucidate the behavior of different contributions, we split Formula (2.90) according to

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}\left(j(u) - up + fp + \mathrm{K}gp + p\frac{\partial g}{\partial \mathbf{n}}\right) dS \quad (2.91a)$$

$$- \int_{\partial\Omega} \mathcal{V}\cdot\mathbf{n}\,(\nabla_\Gamma u \nabla_\Gamma p)\, dS\,. \quad (2.91b)$$

An approximation of the first integral (2.91a) by finite elements converges quadratically in h. This can be shown as in the proof of Theorem 2.2.4, since the Dirichlet trace operator is bounded on $H^1(\Omega)$. On the other hand, the good approximation of the tangential derivative of u and p in (2.91b) still defies a theoretical explanation. We limit ourself to provide additional numerical evidence of this unpredicted behavior. In Table 2.1we report the

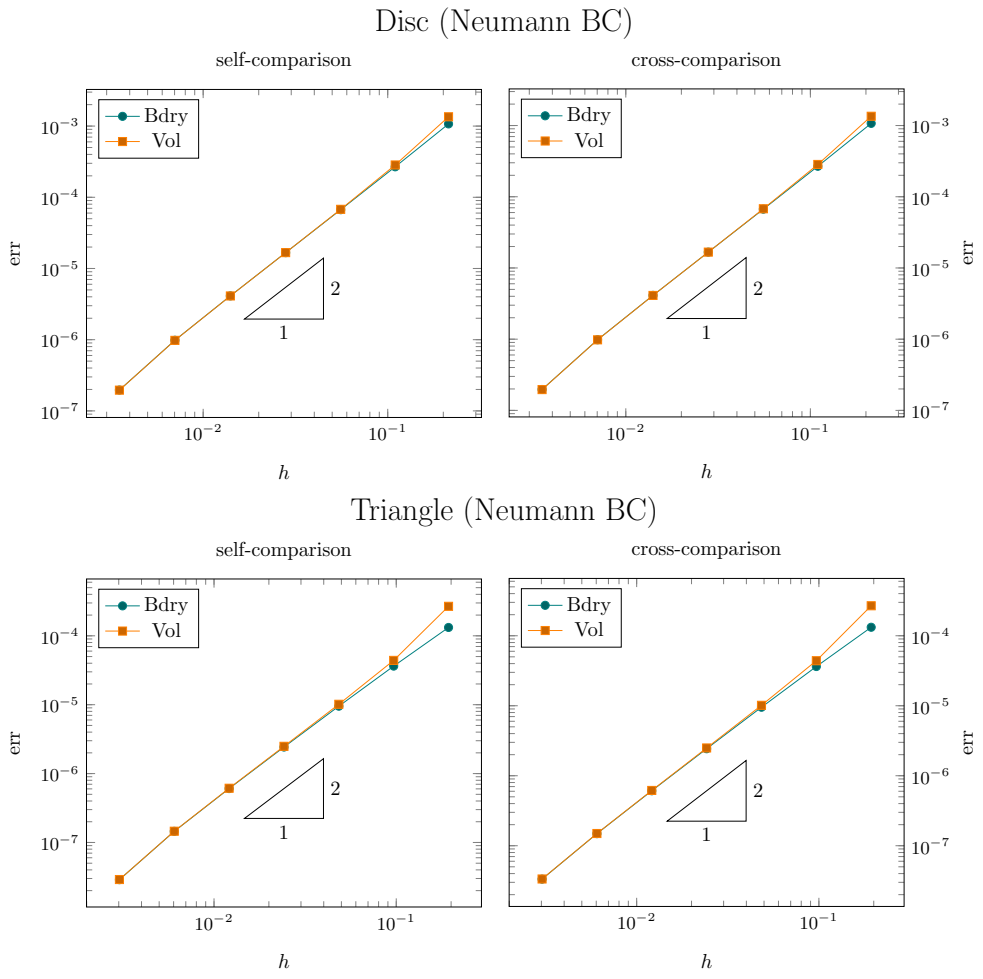

Figure 2.5.: Convergence history of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ when Ω is a disc (*first row*) and a triangle (*second row*). Surprisingly, Formula (2.13) performs as well as Formula (2.12).

approximate convergence rate of the error with respect to the $H^1(\Omega)$- and in the $L^2(\Omega)$-norm, as well as the convergence rate of the boundary integrals

$$\int_{\partial\Omega} \nabla_\Gamma u \nabla_\Gamma u \, dS\,, \quad \int_{\partial\Omega} \nabla u \cdot \nabla u \, dS\,, \quad \int_{\partial\Omega} \nabla_\Gamma u \nabla_\Gamma p \, dS\,, \text{ and } \int_{\partial\Omega} \nabla u \cdot \nabla p \, dS\,,$$

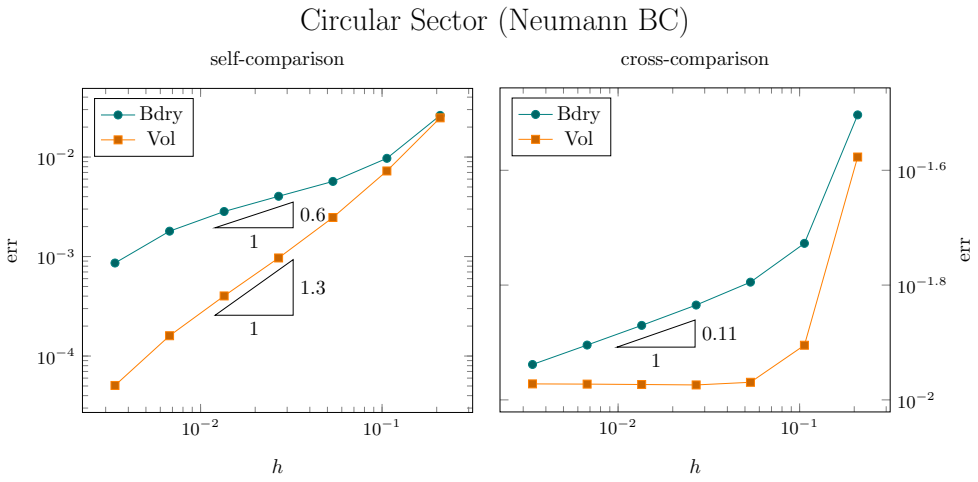

Figure 2.6.: Convergence history of $\mathrm{err}^{\mathrm{Vol}}$ and $\mathrm{err}^{\mathrm{Bdry}}$ when Ω is a sector. This domain does not guarantee H^2-regularity of the state problem (2.7), and it is questionable whether Formulas (2.36) and (2.37) are still equivalent.

when the domain Ω is circular sector with internal angle of size $0.3 \cdot 2\pi$, $0.5 \cdot 2\pi$, $0.7 \cdot 2\pi$, and $0.9 \cdot 2\pi$; see Figure 2.7 We observe that the (unexpected) superconvergence of the boundary integrals breaks down when the domain has a reentrant corner. On the other hand, the approximates convergence rates with respect to the $H^1(\Omega)$- and in the $L^2(\Omega)$-norm behave has predicted by standard finite element estimates [15, Ch. II, Sect. 7].

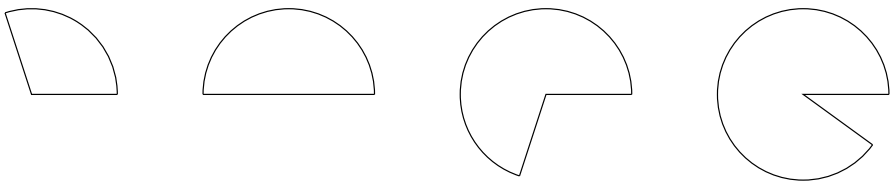

Figure 2.7.: We vary the internal angle of a circular sector Ω to investigate numerically the superconvergence of the boundary integrals.

internal angle	$H^1(\Omega)(u)$	$L^2(\Omega)(u)$	$\int_{\partial\Omega} \nabla_\Gamma u \nabla_\Gamma u \, dS$	$\int_{\partial\Omega} \nabla u \cdot \nabla u \, dS$
$0.3 \cdot 2\pi$	1.02	2.06	2	1.93
$0.5 \cdot 2\pi$	1.04	2.08	2.08	1.95
$0.7 \cdot 2\pi$	0.81	1.67	0.78	0.75
$0.9 \cdot 2\pi$	0.71	1.33	0.55	0.44

internal angle	$H^1(\Omega)(p)$	$L^2(\Omega)(p)$	$\int_{\partial\Omega} \nabla_\Gamma u \nabla_\Gamma p \, dS$	$\int_{\partial\Omega} \nabla u \cdot \nabla p \, dS$
$0.3 \cdot 2\pi$	1.11	2.06	2.02	1.86
$0.5 \cdot 2\pi$	1.06	2.09	2.08	2.12
$0.7 \cdot 2\pi$	0.81	1.69	0.82	0.84
$0.9 \cdot 2\pi$	0.66	1.31	0.56	0.53

Table 2.1.: Dependence of approximate convergence rates on the internal angle when the domain Ω is circular sector. The superconvergence of the boundary integrals breaks down when the domain has a reentrant corner.

2.4. Extension to transmission problems

Here we consider

$$\mathcal{J}(\Omega) = \int_D j(u) \, \mathrm{d}\boldsymbol{x} \,, \tag{2.92}$$

where $D \subset \Omega$, $j : \mathbb{R} \to \mathbb{R}$ is a Lipschitz differentiable function and u is the solution of the scalar transmission problem

$$\begin{cases} -\operatorname{div}(\sigma(\mathbf{x})\nabla u) &= f \quad \text{in } \Omega = \Omega_1 \cup \Omega_2 \,, \\ [\![u]\!] &= 0 \quad \text{on } \Gamma \,, \\ [\![\sigma \frac{\partial u}{\partial \mathbf{n}}]\!] &= 0 \quad \text{on } \Gamma \,, \\ u &= 0 \quad \text{on } \partial\Omega \,, \end{cases} \tag{2.93}$$

with real piecewise constant coefficient

$$\sigma(\mathbf{x}) \coloneqq \sigma_1 \chi_{\Omega_1}(\mathbf{x}) + \sigma_2 \chi_{\Omega_2}(\mathbf{x}) \,.$$

The jump symbol $[\![\cdot]\!]$ denotes discontinuity across the interface Γ. Note that for the Neumann jump the vector $\mathbf{n}$ points outward, see Figure 2.8.

Let us assume that the piecewise constant coefficient σ follows the perturbations introduced by the testing vector field $\mathcal{V}$ (that is, $\dot{\sigma} = 0$) and that the vector field $\mathcal{V}$ vanishes on $\partial\Omega$ (because we are mostly interested in the

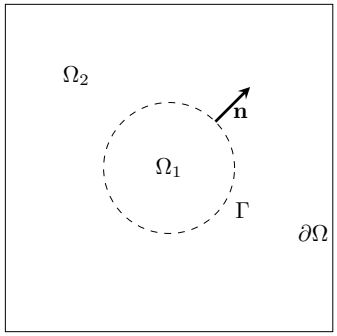

Figure 2.8.: Computational domain Ω of (2.93).

contribution of the interface). Then, the shape gradient of (2.92) takes the forms

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_\Omega \Big(\sigma \nabla u \cdot (D\mathcal{V} + D\mathcal{V}^T)\nabla p + p\nabla f \cdot \mathcal{V} \\ + \operatorname{div}(\mathcal{V})\left(j(u) - \sigma\nabla u \cdot \nabla p + fp\right) \Big) \mathrm{d}\boldsymbol{x} \tag{2.94}$$

and

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_\Gamma (\mathcal{V}\cdot\mathbf{n}) \left[\!\!\left[2\sigma \frac{\partial p}{\partial \mathbf{n}} \frac{\partial u}{\partial \mathbf{n}} - \sigma \nabla u \cdot \nabla p \right]\!\!\right] \mathrm{d}S\,, \tag{2.95}$$

where p is the solution of the adjoint problem

$$\begin{cases} -\operatorname{div}(\sigma(\mathbf{x})\nabla p) &= j'(u)\chi_D & \text{in } \Omega\,, \\ [\![p]\!] &= 0 & \text{on } \Gamma\,, \\ [\![\sigma \frac{\partial p}{\partial \mathbf{n}}]\!] &= 0 & \text{on } \Gamma\,, \\ p &= 0 & \text{on } \partial\Omega\,. \end{cases} \tag{2.96}$$

Formula (2.94) can be derived following the same procedure as in the proof of Proposition 2.1.2. In this case, the boundary $\partial\Omega$ is fixed, and it is possible to impose $u \in H^1_{\partial\Omega}(\Omega)$ from the beginning without considering the saddle point formulation (2.14). This notably simplifies the computations, and Formula (2.94) can be obtained by differentiating the Lagrangian

$$\mathscr{L} : C^1(\overline{\Omega};\mathbb{R}^d) \times H^1_{\partial\Omega}(\Omega) \times H^1_{\partial\Omega}(\Omega) \to \mathbb{R}$$

$$(\mathcal{V}, u, p) \mapsto \int_\Omega (j(u)\chi_D + (f \circ T_\mathcal{V})p) \det \mathbf{D}T_\mathcal{V} - \sigma \nabla u \cdot \mathbf{D}T_\mathcal{V}^{-1} \mathbf{D}T_\mathcal{V}^{-T} \nabla p \, \mathrm{d}\boldsymbol{x}$$

with respect to the control variable $\mathcal{V}$ (note that both the subdomain D and the parameter σ do not depend on $\mathcal{V}$). Since this is a straightforward computation, we only show how to recast (2.94) as the boundary integral (2.95). For a complete derivation of (2.94) we refer to [54]. Henceforth, we refer to (2.94) and (2.95) with the notation $d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Vol}}$ and $d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Bdry}}$, respectively.

Proposition 2.4.1. *Let u and p be exact solutions of* (2.93) *and* (2.96)*, respectively. Then, the following equality holds*

$$d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Vol}} = d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Bdry}} . \tag{2.97}$$

Proof. Subdomain-wise integration by parts on Formula (2.94) yields

$$\begin{aligned} d\mathcal{J}(\Omega; \mathcal{V}) = \int_\Omega \Big(& \sigma \nabla u \cdot (D\mathcal{V} + D\mathcal{V}^T) \nabla p \\ & - \mathcal{V} \cdot \big(j'(u) \nabla u - \sigma \nabla (\nabla u \cdot \nabla p) + f \nabla p \big) \Big) \mathrm{d}\boldsymbol{x} \\ & + \int_\Gamma [\![\mathcal{V} \cdot \mathbf{n} \, (j(u) - \sigma \nabla u \cdot \nabla p + fp)]\!] \mathrm{d}S . \end{aligned} \tag{2.98}$$

With the vector calculus identity [13, Eq. (44)]

$$\nabla u \cdot (D\mathcal{V} + D\mathcal{V}^T) \nabla p + \mathcal{V} \cdot \nabla (\nabla u \cdot \nabla p) = \nabla p \cdot \nabla (\mathcal{V} \cdot \nabla u) + \nabla u \cdot \nabla (\mathcal{V} \cdot \nabla p) , \tag{2.99}$$

Formula (2.98) can be rewritten as

$$\begin{aligned} d\mathcal{J}(\Omega; \mathcal{V}) = \int_\Omega \Big(& \sigma \nabla p \cdot \nabla (\mathcal{V} \cdot \nabla u) + \sigma \nabla u \cdot \nabla (\mathcal{V} \cdot \nabla p) \\ & - j'(u) \mathcal{V} \cdot \nabla u - f \mathcal{V} \cdot \nabla p \Big) \mathrm{d}\boldsymbol{x} \\ & + \int_\Gamma [\![\mathcal{V} \cdot \mathbf{n} \, (j(u) - \sigma \nabla u \cdot \nabla p + fp)]\!] \mathrm{d}S . \end{aligned} \tag{2.100}$$

Then, integration by parts yields

$$d\mathcal{J}(\Omega; \mathcal{V}) = \int_\Gamma \left[\!\!\left[\sigma \frac{\partial p}{\partial \mathbf{n}} \mathcal{V} \cdot \nabla u \right]\!\!\right] - \int_\Omega \mathrm{div}(\sigma \nabla p)(\mathcal{V} \cdot \nabla u) + j'(u)(\mathcal{V} \cdot \nabla u) \mathrm{d}\boldsymbol{x}$$

$$+\int_\Gamma \left[\!\!\left[\sigma\frac{\partial u}{\partial \mathbf{n}}\mathcal{V}\cdot\nabla p\right]\!\!\right] - \int_\Omega \operatorname{div}(\sigma\nabla u)(\mathcal{V}\cdot\nabla p) + f(\mathcal{V}\cdot\nabla p)\mathrm{d}\boldsymbol{x}$$
$$+\int_\Gamma [\![\mathcal{V}\cdot\mathbf{n}\,(j(u)-\sigma\nabla u\cdot\nabla p + fp)]\!]\mathrm{d}S\,. \tag{2.101}$$

The two domain integrals in (2.101) vanish because of (2.93) and (2.96). Moreover, since $[\![u]\!]=0$ on Γ,

$$\left[\!\!\left[\sigma\frac{\partial p}{\partial \mathbf{n}}\mathcal{V}\cdot\nabla u\right]\!\!\right] = \mathcal{V}\cdot\mathbf{n}\left[\!\!\left[\sigma\frac{\partial p}{\partial \mathbf{n}}\frac{\partial u}{\partial \mathbf{n}}\right]\!\!\right] \quad \text{and} \quad [\![\mathcal{V}\cdot\mathbf{n}j(u)]\!]=0\,,$$

and since $[\![p]\!]=0$, $[\![\mathcal{V}\cdot\mathbf{n}fp]\!]=0$, so that we retrieve

$$d\mathcal{J}(\Omega;\mathcal{V}) = \int_\Gamma \mathcal{V}\cdot\mathbf{n}\left[\!\!\left[2\sigma\frac{\partial p}{\partial \mathbf{n}}\frac{\partial u}{\partial \mathbf{n}} - \sigma\nabla u\cdot\nabla p\right]\!\!\right]\mathrm{d}S\,. \tag{2.95}$$

□

Remark 2.4.2. *For $d\mathcal{J}(\Omega,u,p;\mathcal{V})^{\mathrm{Vol}}$ to be well-defined, it is sufficient to assume that $u,\,p\in H^1(\Omega)$. On the other hand, higher regularity of u and p is required for $d\mathcal{J}(\Omega,u,p;\mathcal{V})^{\mathrm{Bdry}}$ to be well-defined because the latter is not continuous on $H^1(\Omega)$.*

Usually, exact solutions of IPs are not available, and one has to rely on numerical approximations $u_h,\,p_h\in W^{1,\infty}(\Omega)$. Equality (2.97) breaks down when u and p are replaced with their approximate counterparts [13], and both formulas (2.94) and (2.95) become approximations

$$d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}} \approx d\mathcal{J}(\Omega;\mathcal{V}) \approx d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}} \tag{2.102}$$

of the exact value $d\mathcal{J}(\Omega;\mathcal{V})$. Again, the natural question is then which among $d\mathcal{J}(\Omega,u_h,p_h;\cdot)^{\mathrm{Vol}}$ and $d\mathcal{J}(\Omega,u_h,p_h;\cdot)^{\mathrm{Bdry}}$ is closer to $d\mathcal{J}(\Omega;\cdot)$.

Let assume that both the interface Γ and the source function f are sufficiently smooth, and let u_h and p_h be Ritz-Galerkin solutions computed with piecewise linear Lagrangian finite elements on a family of quasi-uniform triangular meshes with nodal basis functions. Following the same lines as for the proofs of Theorems 2.2.4 and 2.2.9, it can be shown that[8]

[8] We denote by C a generic constant, which may depend on Ω, its discretization, the source function f, and the coefficient σ. Its value may differ between different occurrences.

$$|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}| \le Ch^2\|\mathcal{V}\|_{W^{2,4}(\mathbb{R}^d;\mathbb{R}^d)} \tag{2.103}$$

and that

$$|d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}| \le Ch\|\mathcal{V}\|_{L^\infty(\mathbb{R}^d;\mathbb{R}^d)}. \tag{2.104}$$

We next give numerical evidence of the convergence rates in (2.103) and (2.104). As in Section 2.3, we have to content ourself with considering convergence with respect to a numerically tractable operator norm (the H^1-norm) over a finite-dimensional space of vector fields.

Since we are mainly interested in contributions of the interface, we select vector fields that vanish on $\partial\Omega$. We set $\Omega =]-2,2[^2$ (a square centered in the origin and with side equal 4), and we restrict ourself to the finite-dimensional space of vector fields of the form[9]

$$\mathcal{V}(x,y) = \sum_{\substack{m_1+n_1\le 5\\ m_2+n_2\le 5\\ m_1,m_2,n_1,n_2\ge 1}} \lambda_{m_1,n_1}\begin{pmatrix} v(x,y,m_1,n_1)\\ 0\end{pmatrix} + \lambda_{m_2,n_2}\begin{pmatrix} 0\\ v(x,y,m_2,n_2)\end{pmatrix}$$

with $v(x,y,m,n) = \sin(m\,x\,\pi/2)\sin(n\,y\,\pi/2)$ and $\lambda_{m_i,n_i}\in\mathbb{R}$.

To investigate the convergence, we monitor the approximate dual norms

$$\mathrm{err}^{\mathrm{Vol}} := \max_{\|\mathcal{V}\|_{H^1(\Omega)}=1}\Big(d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}\Big) \tag{2.105}$$

and

$$\mathrm{err}^{\mathrm{Bdry}} := \max_{\|\mathcal{V}\|_{H^1(\Omega)}=1}\Big(d\mathcal{J}(\Omega;\mathcal{V}) - d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}\Big) \tag{2.106}$$

on different meshes generated through uniform refinement. The reference value $d\mathcal{J}(\Omega;\mathcal{V})$ is approximated by evaluating both $d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Vol}}$ and $d\mathcal{J}(\Omega,u_h,p_h;\mathcal{V})^{\mathrm{Bdry}}$ on a mesh with an extra level of refinement. To avoid biased results we display convergence history both with self- and cross-comparison.

As in Section 2.3, we consider finite element discretizations based on linear Lagrangian finite elements on quasi-uniform triangular meshes with nodal

[9] Repeating the experiments for $m_i + n_i \le 3$ produces results in agreement with the observations made for $m_i + n_i \le 5$. Therefore, the arbitrary choice of restricting the sum of the indices to 5 does not seem to compromise our observations.

basis functions. The experiments are performed in MATLAB and are based on the library LehrFEM developed at ETHZ. Mesh generation and uniform refinement are performed with the functions `initmesh` and `refinemesh` of the MATLAB PDE Toolbox [58]. Integrals in the domain are computed by 7 point quadrature rule in each triangle, while line integrals are computed by 6 point Gauss quadrature on each segment. In experiment 1, the interface is approximated by a polygon. Nevertheless, the convergence of linear finite elements is not affected by this discretization [56]. Note that new meshes are always adjusted to fit the curved interface.

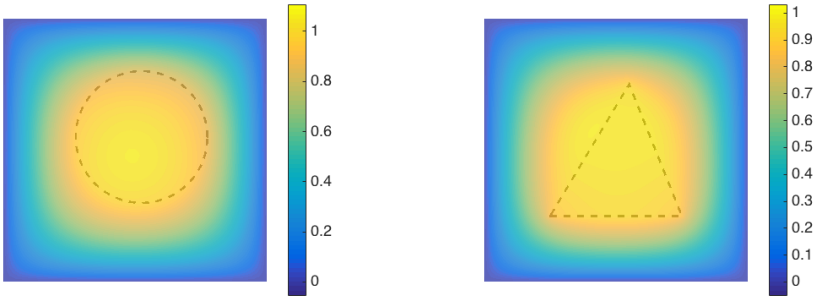

Figure 2.9.: Plot of the solution u of the state problem in the computational domain Ω for the **first** (left) and the **second** (right) **numerical experiment**. The interface is drawn with a dashed line.

In the **first numerical experiment** the interface Γ is a circle centered in $(0.1, 0.2)$ and with radius equal to 1, see Figure 2.9 (left). The problem data are

$$f(\mathbf{x}) = 1 \quad \text{and} \quad \sigma(\mathbf{x}) = 2\chi_{\Omega_1}(\mathbf{x}) + 1\chi_{\Omega_2}(\mathbf{x})\,. \tag{2.107}$$

The numerical results are displayed in Figure 2.10 (first row). We clearly see that the volume-based formulation converges faster and is more accurate than its boundary-based counterpart. The convergence rates agree with what has been predicted by (2.103) and (2.104). In the cross-comparison plot $d\mathcal{J}(\Omega, u_h, p_h; \mathcal{V})^{\mathrm{Vol}}$ saturates due to insufficient accuracy of the reference solution computed with $d\mathcal{J}(\Omega, u_h, p_h; \mathcal{V})^{\mathrm{Bdry}}$, whereas the boundary-based formulation converges with the same rate as for the self-comparison.

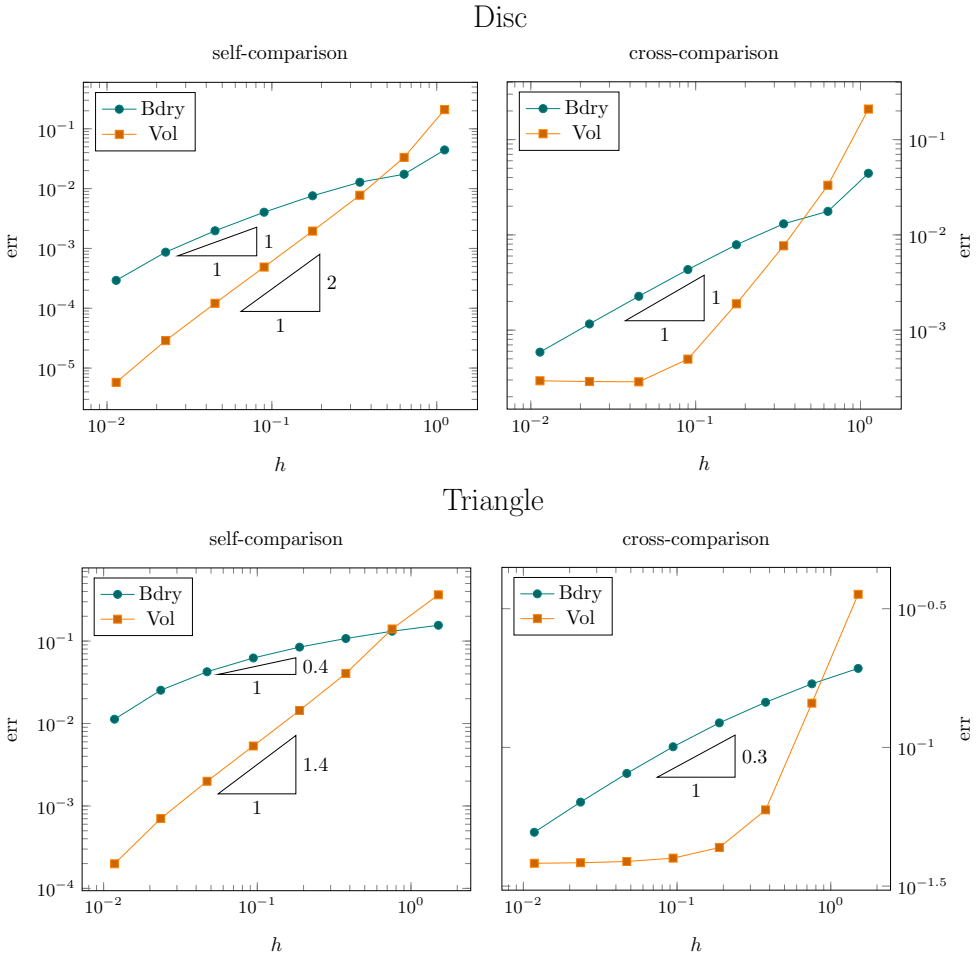

Figure 2.10.: Convergence history for the **first** (first row) and the **second** (second row) **numerical experiment**. In the left column the reference value $d\mathcal{J}(\Omega;\mathcal{V})$ is computed with an extra level of refinement. The second column displays cross-comparisons.

In the **second numerical experiment** the interface Γ is a triangle with corners located at $(-1,-1)$, $(1,-1)$ and $(0.2,1)$, see Figure 2.9 (right). Interface corners are known to affect the regularity of the solution of interface

problems [14]. Therefore, the estimates (2.103) and (2.104) can not be proved in this case, and we expect to observe lower convergence rates. To better highlight the impact of the corners we increase the contrast of the diffusion coefficient by setting

$$\sigma(\mathbf{x}) = 10\chi_{\Omega_1}(\mathbf{x}) + 1\chi_{\Omega_2}(\mathbf{x})\,.$$

The source function is the same as in (2.107). From the results displayed in Figure 2.10 (second row) we observe that the volume-based formulation converges faster and is more accurate then its boundary-based counterpart. Again, in the cross-comparison the convergence history of the volume-based formulation saturates due to an insufficient accuracy of the reference solution computed with $d\mathcal{J}(\Omega, u_h, p_h; \mathcal{V})^{\mathrm{Vol}}$. We suspect that this inaccuracy gives rise to the difference in the convergence rates of the boundary based formulation between self- and cross-comparisons.

In the **third numerical experiment** we investigate the impact of the choice of the diffusion coefficient σ on the results obtained in the **first** and in the **second numerical experiment**. For $\sigma_1 = 0.1, 0.5, 0.8, 1.25, 2, 10$ and $\sigma_2 = 1$ fixed, we monitor the approximate relative error constructed by dividing the approximate dual norms (2.105) and (2.106) by

$$\max_{\mathcal{V}} \frac{|d\mathcal{J}(\Omega; \mathcal{V})|}{\|\mathcal{V}\|_{H^1(\Omega)}}\,.$$

The reference solution is computed evaluating $d\mathcal{J}(\Omega, u, p; \mathcal{V})^{\mathrm{Vol}}$ on a mesh with an extra level of refinement. In Figure 2.11 (left), we see that the choice of the diffusion coefficient σ has no influence on the convergence rates in case of a circular interface. On the other hand, for nonsmooth interfaces, the effect of the singularity in the functions u and p is visible only for high contrasts σ_1/σ_2.

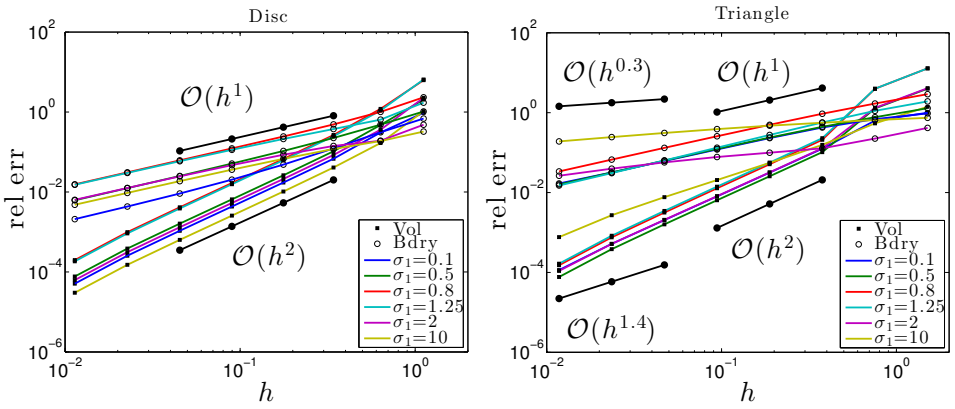

Figure 2.11.: Convergence history for the **third numerical experiment**. The choice of the diffusion coefficient has no influence on the convergence rates in case of a circular interface (left). On the other hand, for a triangular interface (right), the effect of the singularity in the functions u and p is visible only for high contrasts σ_1/σ_2.

3. Shape optimization by pursuing diffeomorphisms

Shape gradients are a key ingredient to perform shape optimization. When the shape functional is constrained by boundary value problem (BVP), the shape gradient depends on the solution of this BVP. Often, this solution can only be computed approximatively employing numerical methods. The challenge is to devise an algorithm that allows to perform shape updates in a way that does not destroy the approximation properties of the numerical method used to approximate the solution of the BVP.

Literature abounds with numerical schemes to solve PDE-constrained shape optimization problems; see, for instance, the monographs [2, 3, 39, 52, 66]. An accurate method for PDE-constrained shape optimization problems has been developed relying on boundary element discretization of the underlying BVP [28,29]. This approach has been further investigated in [10]. However, the bulk of literature considers discretization by means of the finite element method (FEM) [3, 8, 9, 11, 31, 50, 53, 61, 66, 74].

The most naïve approach discretizes an initial shape, the initial guess, with a mesh and then optimize the coordinates of the mesh nodes [3, 66, 74]. This means that in 2D the shape is initially discretized as a polygon, and that mesh updates occur by modifying the vertices of this polygon. Unfortunately, this is a very delicate task because the mesh might get distorted or self-intersect as the optimization routine proceeds and one has to employ technical remedies to avoid these issues [4, 5].

The level set approach is a popular alternative [6, 7]. It is based on the boundary formulation of shape gradients and on level set functions [71]. The boundary of the domain is implicitly represented as the zero-level of a function, and optimization is carried out by updating this function. To include information on the shape in the PDE constraint, the latter is first formulated on a hold-all domain (that contains all admissible shapes). Then, the PDE is modified pursuing the so-called "ersatz material" approach: the boundary of the domain to be optimized is interpreted as "the interface between two

constituents occupying" the hold-all domain [7]. In [7], the authors state that "This is a well-known approach in topology optimization which can be rigorously justified in some cases [2]" . However, updating the level set function is a delicate process because, on the one hand, this function should have sufficiently steep slope at the zero-level to identify the boundary of the domain, and, on the other hand, a level set function that is too steep "implies a bad approximation of the normal n or of the curvature H" [7], which appear in the boundary formula of the shape gradient. In practice, the level set function has to be periodically reinitialized by solving an evolution problem [7]. Recently, a level set method based on the volume formula of the shape gradient has been presented in [55]. For this novel approach, it has been observed experimentally that the level set function "stays close to a distance function during the iterations" and that reinitialization may not be necessary.

Another alternative are free-form deformation methods [9, 53], which recast the shape optimization problem as an optimal control problem. Shapes are parametrized by applying a transformation to a reference shape that corresponds to an initial guess. This transformation is constructed with (piecewise) polynomials defined on a lattice of control points, and optimization is carried out on their coordinates. This approach allows to preserve the approximation properties of FEM. However, the infinite-dimensional shape optimization problem is replaced with a model with a fixed small number of control parameters, and the dependence of the quality of the discrete solution on the number of control parameters is not clear.

We present an algorithm developed to preserve and exploit the approximation properties of FEM, and that allows for arbitrarily high resolution of shapes. Similar to the free-form deformation approach, we recast the shape optimization problem as an optimal control problem. Shapes are parametrized by letting a diffeomorphism act on a reference shape Ω_0 that corresponds to an initial guess . Pursuing a Ritz approach, we discretize the diffeomorphism with conforming basis functions based on B-splines. We show that, under reasonable assumptions, the sequence of optimal discrete solutions converges to the global minimum as the dimension of the trial space tends to infinity. We also investigate the impact of FEM approximations in the context of elliptic PDE-constrained shape optimization and formulate a descent method that enjoys superconvergence in the approximation of the Fréchet derivative. We test the performance of the proposed method both on a well-posed model problem stemming from the class of exterior Bernoulli free boundary prob-

lems and on a prototypical ill-posed inverse problem. Our method has been partly inspired by [29] and can be interpreted as a generalization of the naïve aproach that updates the coordinates of the nodes of a mesh. In contrast to the similar works [31, 50], we directly work with volume transformations (instead of extending a perturbation of the boundary to the volume), we do not deal with a parametrization of the domain boundary, and we do not assume that the domain is star-shaped. The advantage of our approach is that we can easily construct C^2-transformations. The drawback is that we do not have unique representation of shapes and this makes it difficult to prove a-priori convergence estimates. Some of the results presented here have already been published in [44].

3.1. Shape optimization in parametric form

Let $D \subset \mathbb{R}^d, d = 2, 3$, be a bounded Lipschitz domain (hold-all domain), and let Ω_0 be a compact subset of D with Lipschitz boundary. We fix $\varepsilon \in (0, 1)$ and define the set of admissible shapes as

$$\mathcal{U}_{\text{ad}}(\Omega_0) \coloneqq \{T_{\mathcal{V}}(\Omega_0);\, T_{\mathcal{V}} \coloneqq \mathcal{I} + \mathcal{V}, \mathcal{V} \in B_{1-\varepsilon}\}\,, \tag{3.1}$$

where

$$B_{1-\varepsilon} \coloneqq \left\{\mathcal{V} \in C_c^1(\overline{D}; \mathbb{R}^d);\, |\mathcal{V}|_{C^1} \leq 1 - \varepsilon\right\}$$

and

$$|\mathcal{V}|_{C^1} \coloneqq \sup_{\mathbf{x}\in\mathbb{R}^d} \sup_{\mathbf{y}\in\mathbb{R}^d, \mathbf{y}\neq 0} \frac{\|\mathbf{D}\mathcal{V}(\mathbf{x})\mathbf{y}\|_{\mathbb{R}^d}}{\|\mathbf{y}\|_{\mathbb{R}^d}}\,.$$

Proposition 3.1.1. *The transformation $T_{\mathcal{V}}$ defined in* (3.1) *is a diffeomorphism from D onto itself.*

Proof. We follow closely the proof of Lemma 6.13 in [3]. We tacitly assume that the vector field $\mathcal{V}$ is extended by zero outside D.

By the Fundamental Theorem of Calculus, $\mathcal{V}$ is a contraction on $\mathbb{R}^d$, because

$$\begin{aligned} |\mathcal{V}(\mathbf{x}) - \mathcal{V}(\mathbf{y})| &= \left| \int_0^1 \mathbf{D}\mathcal{V}(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{x} - \mathbf{y}) dt \right|, \\ &\leq |\mathcal{V}|_{C^1} \|\mathbf{x} - \mathbf{y}\|_{\mathbb{R}^d} \leq (1 - \varepsilon)\|\mathbf{x} - \mathbf{y}\|_{\mathbb{R}^d} \qquad \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d\,. \end{aligned}$$

For a generic $\mathbf{b} \in \mathbb{R}^d$, we define the map $\mathbf{K}_\mathbf{b}(\mathbf{x}) \coloneqq \mathbf{b} - \mathcal{V}(\mathbf{x})$, which is a contraction on $\mathbb{R}^d$, too. By the Banach Fixpoint Theorem, there exists exactly one $\mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{K}_\mathbf{b}(\mathbf{y}) = \mathbf{y}$, which implies that

$$\mathbf{b} = \mathbf{y} + \mathcal{V}(\mathbf{y}) = T_\mathcal{V}(\mathbf{y})\,,$$

and thus, that $T_\mathcal{V}$ is a bijection.

The continuous differentiability of the inverse $T_\mathcal{V}$ follows from the convergence of the Neumann series

$$(\mathbb{1} + \mathcal{V})^{-1} = \sum_{k\geq 0}(-\mathcal{V})^k$$

and the Inverse Function Theorem [30, Sect. C.6.] Finally, $T_\mathcal{V}(D) = D$ is a consequence of $T_\mathcal{V}$ being the identity on $\mathbb{R}^d \setminus D$. □

Let $\mathcal{J}$ be a real-valued functional defined on $\mathcal{U}_{\text{ad}}(\Omega_0)$, and let $\tilde{\mathcal{J}}$ be defined by

$$\tilde{\mathcal{J}} : B_{1-\varepsilon} \to \mathbb{R}\,, \quad \mathcal{V} \mapsto \mathcal{J}\left(T_\mathcal{V}(\Omega_0)\right)\,.$$

The shape optimization problem

$$\inf_{\Omega\in\mathcal{U}_{\text{ad}}(\Omega_0)} \mathcal{J}(\Omega)$$

can be recast as

$$\inf_{\mathcal{V}\in B_{1-\varepsilon}} \tilde{\mathcal{J}}(\mathcal{V})\,. \tag{3.2}$$

In general, the optimization problem (3.2) does not admit a minimizer in $C^1_c(\overline{D};\mathbb{R}^d)$. In particular, $C^1_c(\overline{D};\mathbb{R}^d)$ is not reflexive, and the Banach-Alaoglu Theorem cannot be employed to assert that $B_{1-\varepsilon}$ is weakly compact. This reflects the ill-posedness of most shape optimization problems: optimal shapes may not lie in $\mathcal{U}_{\text{ad}}(\Omega_0)$. However, the next theorem shows that shape optimization still provides a valid tool to improve the performance (measured through $\mathcal{J}$) of an initial design Ω_0.

Theorem 3.1.2. *Let $\tilde{\mathcal{J}}$ be continuous with respect to the $C^1(\overline{D};\mathbb{R}^d)$ -norm and restrict the shape optimization problem* (3.2) *to*

$$\inf_{\mathcal{V}\in B_{1-\varepsilon}\cap B^W_r} \tilde{\mathcal{J}}(\mathcal{V})\,, \tag{3.3}$$

where

$$B_r^W := \{\mathcal{V} \in W^{2,d+1}(D;\mathbb{R}^d);\ \|\mathcal{V}\|_{W^{2,d+1}(D;\mathbb{R}^d)} \leq r\}$$

for a fixed real number $r > 0$ that can be arbitrarily large. Then, there exists a vector field $\mathcal{V}^* \in B_{1-\varepsilon}$ so that

$$\tilde{\mathcal{J}}(\mathcal{V}^*) = \inf_{\mathcal{V} \in B_{1-\varepsilon} \cap B_r^W} \tilde{\mathcal{J}}(\mathcal{V})\,. \tag{3.4}$$

Proof. We follow closely [3, Thm 5.12]. The main ingredient is the compact embedding

$$W^{2,d+1}(D;\mathbb{R}^d) \overset{c}{\hookrightarrow} C^1(\overline{D};\mathbb{R}^d)\,.$$

This follows from Morrey's inequality [1, Thm 4.12], which states that

$$W^{2,d+1}(D;\mathbb{R}^d) \hookrightarrow C^{1,\lambda}(\overline{D};\mathbb{R}^d)\,, \quad \text{for } \lambda = 1 - d/4\,,$$

and the compact embedding $C^{1,\lambda}(\overline{D};\mathbb{R}^d) \overset{c}{\hookrightarrow} C^1(\overline{D};\mathbb{R}^d)$ [1, Thm 1.34].

A minimizing sequence $\{\mathcal{V}_i\}_{i\in\mathbb{N}}$ of (3.3) is bounded because $\mathcal{V}_i \in B_r^W$ for every index i. Thus, by compactness, we can extract a subsequence that converges to a limit function $\mathcal{V}^*$ in the $C^1(\overline{D};\mathbb{R}^d)$-norm. Finally, the continuity assumption on $\tilde{\mathcal{J}}$ implies (3.4). Note that $\mathcal{V}^* \in B_{1-\varepsilon}$ because $\mathcal{V}_i \in B_{1-\varepsilon}$ for every index i. □

Remark 3.1.3. *The continuity assumption on $\tilde{\mathcal{J}}$ in Theorem 3.1.2 is fulfilled by most of the shape functionals considered in literature. For instance, this is the case for the volume and the surface area shape functionals.*

Remark 3.1.4. *There is little hope for uniqueness in this framework. Let $\mathcal{V}^*$ be an optimal solution. If there is a vector field $\tilde{\mathcal{V}} \neq 0$ so that $(\mathcal{I}+\tilde{\mathcal{V}})(\partial\Omega_0) = \partial\Omega_0$ (from the set point of view), then the composition $\mathcal{V}^* \circ (\mathcal{I} + \tilde{\mathcal{V}})$ is an optimal solution, too.*

Approximate solutions can be obtained easily with a Ritz approach.

Theorem 3.1.5. *Let $\{V_N\}_{N\in\mathbb{N}}$ be a nested sequence of $W^{2,d+1}(D;\mathbb{R}^d)$-conforming trial spaces that satisfies*

$$\overline{\cup_{N\in\mathbb{N}} V_N}^{W^{2,d+1}(D;\mathbb{R}^d)} = W^{2,d+1}(D;\mathbb{R}^d)\,.$$

Let $\{\mathcal{V}_N^*\}_{N\in\mathbb{N}}$ be the sequence of discrete solutions defined by

$$\mathcal{V}_N^* \in \operatorname*{argmin}_{\mathcal{V}_N\in V_N\cap B_{1-\varepsilon}\cap B_r^W} \tilde{\mathcal{J}}(\mathcal{V}_N)\,. \tag{3.5}$$

Then, under the assumptions of Theorem 3.1.2, $\{\mathcal{V}_N^*\}_{N\in\mathbb{N}}$ is a minimizing sequence of $\tilde{\mathcal{J}}$ in $B_{1-\varepsilon}\cap B_r^W$.

Proof. We follow closely the proof of the classic result on the convergence of Ritz methods given in [34, Sect. 40.1].

Let $\mu\in\mathbb{R}$ be the infimum of (3.3). Note that $\mu > -\infty$. Let $a>0$, and let $\mathcal{V}\in B_{1-\varepsilon}\cap B_r^W$ satisfy

$$\tilde{\mathcal{J}}(\mathcal{V}) < \mu + a\,.$$

By continuity of $\tilde{\mathcal{J}}$ with respect to $C^1(\overline{D};\mathbb{R}^d)$, $\mathcal{V}$ can be rescaled so that

$$|\mathcal{V}|_{C^1(\overline{D};\mathbb{R}^d)} < 1-\varepsilon\,,\quad \|\mathcal{V}\|_{W^{2,d+1}(D;\mathbb{R}^d)} < r \quad\text{and}\quad \tilde{\mathcal{J}}(\mathcal{V}) < \mu + 2a\,.$$

Let $b>0$, and let $N=N(b)\in\mathbb{N}$ be sufficiently large. Then, there exists a $\mathcal{V}_N\in V_N\cap B_{1-\varepsilon}\cap B_r^W$ that satisfies

$$\|\mathcal{V}-\mathcal{V}_N\|_{W^{2,d+1}(D;\mathbb{R}^d)} < b\,,$$

and thus, by Morrey's inequality [1, Thm 4.12],

$$\|\mathcal{V}-\mathcal{V}_N\|_{C^1(\overline{D};\mathbb{R}^d)} < b$$

for N large enough. Furthermore, for $b=b(a)$ small enough, it holds

$$\tilde{\mathcal{J}}(\mathcal{V}_N) < \mu + 3a\,.$$

Let $\mathcal{V}_N^*$ be defined as in (3.5). It holds

$$\mu \le \tilde{\mathcal{J}}(\mathcal{V}_N^*) \le \tilde{\mathcal{J}}(\mathcal{V}_N) \le \mu + 3a\,.$$

Since a is arbitrary, it follows

$$\lim_{N\to\infty} \tilde{\mathcal{J}}(\mathcal{V}_N^*) = \mu\,.$$

□

Remark 3.1.6. *Refined convergence theories can be found in [29, 31, 50]. These articles rely on a parametrization of the boundary, and consider as admissible shapes those that can be reached via a normal perturbation of the boundary $\partial\Omega_0$. In this case, the parametrization of shapes is unique, and a priori convergence rates can be proved.*

3.2. PDE-constrained shape optimization

In PDE-constrained shape optimization, the goal is to find the domain Ω that minimizes the functional $\mathcal{J}(\Omega, u)$ subject to a PDE constraint $\mathcal{A}u = f$ in Ω. Here, $\mathcal{A} : X(\Omega) \to X(\Omega)^*$ denotes a second order $X(\Omega)$-elliptic operator between the Hilbert space $X(\Omega)$ and its dual $X(\Omega)^*$, which are function spaces on the domain Ω. Similarly as in (3.2), the shape optimization problem can be recast in a parametric form relying on the characterization of admissible domains (3.1), that is,

$$\inf_{\mathcal{V}\in B_{1-\varepsilon}} \tilde{\mathcal{J}}(\mathcal{V}, u), \quad \text{subject to} \quad \tilde{\mathcal{A}}_{\mathcal{V}} u = \tilde{f}_{\mathcal{V}} \text{ in } \Omega_0\,. \tag{3.6}$$

Both the elliptic operator $\tilde{\mathcal{A}}_{\mathcal{V}} : X(\Omega_0) \to X(\Omega_0)^*$ and the linear functional $\tilde{f}_{\mathcal{V}} \in X(\Omega_0)^*$ depend on the vector field $\mathcal{V}$ and are created in a way so that $u \in X(\Omega_0)$ is the solution to $\tilde{\mathcal{A}}_{\mathcal{V}} u = \tilde{f}_{\mathcal{V}}$ in Ω_0 if and only if $\hat{u} \coloneqq u \circ T_{\mathcal{V}}^{-1} \in X(\Omega)$ is the solution to $\mathcal{A}\hat{u} = f$ in $\Omega = T_{\mathcal{V}}(\Omega_0)$.

The idea of transforming both the shape functional and the PDE constraint on a reference domain is not new to shape optimization. It has already been used, for instance, in [11,31,61,66], and is, de facto, the standard approach for shape optimization based on free-form deformations; see [9,53] and references therein.

Example 3.2.1. *The parametric form of the shape optimization problem*

$$\inf_{\Omega\in\mathcal{U}_{\mathrm{ad}}(\Omega_0)} \mathcal{J}(\Omega, \hat{u}), \quad \textit{subject to} \quad \begin{cases} -\Delta\hat{u} &= f \quad \textit{in } \Omega\,, \\ \hat{u} &= 0 \quad \textit{on } \partial\Omega\,, \end{cases} \tag{3.7}$$

with $\mathcal{J}(\Omega, \hat{u}) \coloneqq \int_\Omega j(\hat{u})\mathrm{d}\boldsymbol{x}$, $j \in C^1(\mathbb{R})$, *and* $f \in H^{-1}(\Omega)$, *reads*

$$\inf_{\mathcal{V}\in B_{1-\varepsilon}} \tilde{\mathcal{J}}(\mathcal{V}, u)\,, \quad \textit{subject to} \tag{3.8}$$
$$\begin{cases} -\operatorname{div}(\mathbf{M}_{\mathcal{V}}\,\mathbf{grad}\, u) &= (\det \mathbf{D}T_{\mathcal{V}}) T_{\mathcal{V}}^*(f) \quad \textit{in } \Omega_0\,, \\ u &= 0 \quad \textit{on } \partial\Omega_0\,, \end{cases}$$

where the pullback $T_{\mathcal{V}}^*$ *is defined as the composition* $T_{\mathcal{V}}^*(f) \coloneqq f \circ T_{\mathcal{V}}$,

$$\tilde{\mathcal{J}}(\mathcal{V}, u) \coloneqq \int_{\Omega_0} j(u)(\det \mathbf{D}T_{\mathcal{V}})\mathrm{d}\boldsymbol{x}\,, \quad \textit{and} \quad \mathbf{M}_{\mathcal{V}} \coloneqq (\det \mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}\,.$$

Assuming continuity of the map $\mathcal{V} \mapsto \tilde{\mathcal{J}}(\mathcal{V}, u)$ on $C^1(\overline{D}; \mathbb{R}^d)$, an approximate solution of (3.6) can be obtained as in Theorem 3.1.5 by computing

$$\mathcal{V}_N^* \in \operatorname*{argmin}_{\mathcal{V}_N \in V_N \cap B_{1-\varepsilon}} \tilde{\mathcal{J}}(\mathcal{V}_N, u), \quad \text{subject to} \quad \tilde{\mathcal{A}}_{\mathcal{V}_N} u = \tilde{f}_{\mathcal{V}_N} \text{ in } \Omega_0 \tag{3.9}$$

for N large enough. Note that, if $\mathcal{J}$ is shape differentiable, the approximate optimal solution $\mathcal{V}_N^*$ must satisfy the variational inequality [43, Thm 1.48]

$$d\tilde{\mathcal{J}}(\mathcal{V}_N^*, u; \mathcal{W}_N - \mathcal{V}_N^*) \geq 0 \quad \text{for all} \quad \mathcal{W}_N \in V_N \cap B_{1-\varepsilon}\,, \tag{3.10}$$

where $d\tilde{\mathcal{J}}$ denotes the Fréchet derivative of $\tilde{\mathcal{J}}$.

Remark 3.2.2. *In Example 3.2.1, a minimizing sequence $\{\mathcal{V}_N^*\}_{N\in\mathbb{N}}$ that satisfies* (3.9) *contains a subsequence $\{\mathcal{V}_{N_i}^*\}_{i\in\mathbb{N}}$ that converges strongly in $C^1(\overline{D}; \mathbb{R}^d)$ to a $\hat{\mathcal{V}} \in B_{1-\varepsilon}$. Therefore, the ellipticity constants of $\{\tilde{\mathcal{A}}_{\mathcal{V}_{N_i}}\}_{i\in\mathbb{N}}$ are bounded from below by a constant $c > 0$. This implies that*

$$\|u_{\hat{\mathcal{V}}} - u_{N_i}\|_{H^1(\Omega_0)} \to 0 \quad \text{as } i \to \infty\,,$$

where u_{N_i} is the solution to $\tilde{A}_{\mathcal{V}_{N_i}} u = \tilde{f}_{\mathcal{V}_{N_i}}$ and $u_{\hat{\mathcal{V}}}$ is the solution to $\tilde{A}_{\hat{\mathcal{V}}} u = \tilde{f}_{\hat{\mathcal{V}}}$, see [3, Lemma 5.3]. With this result it is easy to show $C^1(\overline{D}; \mathbb{R}^d)$-continuity of the constraint functional (3.8).

3.3. Finite-dimensional trial space of vector fields

Theorem 3.1.2 conveys that an initial design Ω_0 can be improved by solving the shape optimization problem (3.3). This optimization problem is stated on an infinite-dimensional space. Theorem 3.1.5 ensures that we can compute an accurate approximate solution (3.3) by solving its finite dimensional counterpart (3.5). For this counterpart to be well-defined, the nested sequence $\{V_N\}_{N\in\mathbb{N}}$ of finite-dimensional trial spaces of vector fields has to be at least $W^{1,\infty}(D; \mathbb{R}^d)$-conforming. However, the finite-dimensional shape optimizaton problem (3.5) is inherently highly nonlinear, and we have to rely on iterative methods to solve it. Therefore, in light of Theorem 2.2.4, it is computationally convenient to strengthen the conformity assumption on $\{V_N\}_{N\in\mathbb{N}}$ and work with (at least) $W^{2,4}(D; \mathbb{R}^d)$-conforming vector fields. Moreover, it is desirable that the solution of state problem in parametric form lies in $H^2(\Omega_0)$. By elliptic lifting

theorems [35, Ch. 8.4], this can be guaranteed by choosing a convex (or sufficiently smooth) initial guess Ω_0 and working with a $W^{2,\infty}(D;\mathbb{R}^d)$-conforming discretization (so that the diffusion matrix $\mathbf{M}_\mathcal{V}$ is in $W^{1,\infty}(D;\mathbb{R}^{d,d})$).

We opt for spaces spanned by multivariate B-splines of degree ≥ 2. We briefly recall the definition of B-splines and their main properties from Chapters 3 and 4 of the book [46].

Definition 3.3.1. *The uniform univariate B-spline b^n of degree n is defined by the recursion*

$$b^n(x) \coloneqq \int_{x-1}^{x} b^{n-1}\,,$$

starting from the characteristic function b^0 on the unit interval $[0,1)$.

We next list few basic properties of B-splines. The B-spline b^n is

- positive on $(0, n+1)$ and vanishes outside this interval.
- $(n-1)$-times continuously differentiable with discontinuities of the nth derivative at the break points $0,\dots,n+1$.
- a polynomial of degree n on each interval $[k,k+1], k=0,\dots,n$.

In Figure 3.1 we plot b^0, b^1, b^2. Their expression as piecewise polynomials is given in (3.11).

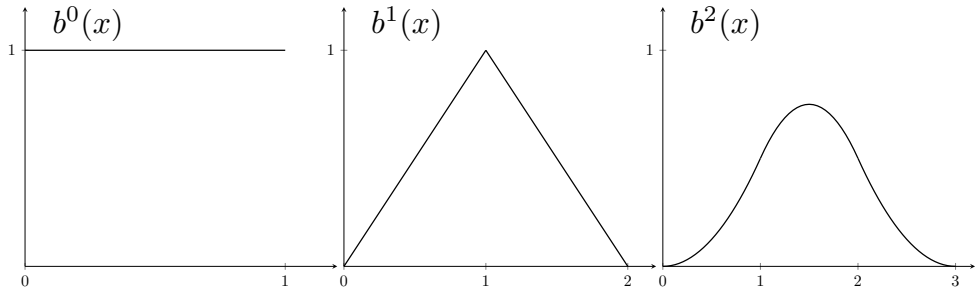

Figure 3.1.: Uniform univariate B-splines of degree $n = 0, 1, 2$.

$$b^0(x) = \begin{cases} 1 & \text{for } 0 \le x < 1\,, \\ 0 & \text{otherwise}\,, \end{cases} \qquad b^1(x) = \begin{cases} x & \text{for } 0 \le x < 1\,, \\ 2-x & \text{for } 1 \le x < 2\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

$$b^2(x) = \begin{cases} \frac{x^2}{2} & \text{for } 0 \le x < 1\,, \\ \frac{(-2x^2+6x-3)}{2} & \text{for } 1 \le x < 2\,, \\ \frac{(x-3)^2}{2} & \text{for } 2 \le x < 3\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.11}$$

Definition 3.3.2. *Let $h\mathbb{Z} \coloneqq \{hk; k \in \mathbb{Z}\}$ denote the bi-infinite grid with grid width[1] h, $h > 0$. The functions*

$$b^n_{k,h}(x) \coloneqq b^n(x/h - k)\,, \quad k \in \mathbb{Z}\,,$$

are the B-splines on the grid $h\mathbb{Z}$. Their linear combinations $\sum_{k\in\mathbb{Z}} c_k b^n_{k,h}$ are called cardinal splines of degree $\le n$ with grid width h.

Marden's Identity implies that cardinal splines can represent polynomials.

Theorem 3.3.3 (Marden's Identity)**.** *For $x, t \in \mathbb{R}$,*

$$(x-t)^n = \sum_{k\in\mathbb{Z}} h^n \left(\prod_{j=1}^{n} (k + j - t/h) \right) b^n_{k,h}(x)\,.$$

From this identity it follows that B-spline translates are also linear independent.

Theorem 3.3.4. *For any grid interval $Q = [\ell, \ell+1]h$, the B-splines $b^n_{k,h}, k = \ell - n, \dots, \ell$, which are nonzero on Q, are linearly independent.*

The simplest generalization of univariate B-splines to the multivariate case relies on tensor products. We denote by

$$B^n(x_1, \dots, x_d) \coloneqq \prod_{j=1}^{d} b^n(x_d)\,, \quad (x_1, \dots, x_d) \in \mathbb{R}^d\,, \tag{3.12}$$

[1] We employ the unusual term "grid width" to denote the width of the grid used to generate B-splines because we reserve the term "mesh width" for the width of the mesh employed for finite element approximations of state and adjoint variables.

the d-variate B-spline with degree n in each variable. Similarly to the univariate case, the space of multivariate cardinal B-splines on the tensor grid $(h\mathbb{Z})^d$ can be defined by rescaling and shifting the *mother B-spline* (3.12). In Figure 3.2 we plot the 2-variate B-splines B^0, B^1, B^2. For a conforming discretization

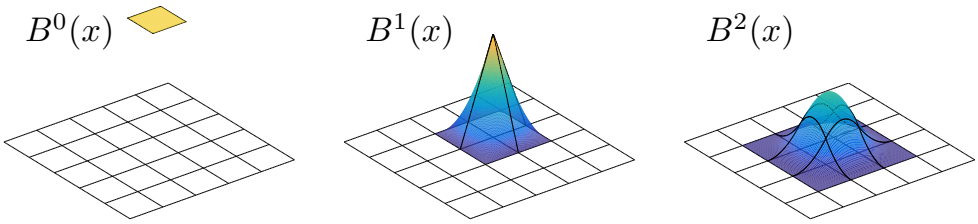

Figure 3.2.: Multivariate B-splines of degree $n = 0, 1, 2$ for $d = 2$.

of $\mathcal{U}_{\mathrm{ad}}(\Omega_0)$ (3.1) we need to construct a spline space on the bounded hold-all domain D. This can be done by

1. constructing a multivariate cardinal spline space of degree n on the tensor grid $(h\mathbb{Z})^d, h > 0$,

2. filtering all B-splines translates whose support is not completely contained in $\overline{D}$,

3. completing the space by adding additional "boundary" basis functions so that all polynomials of degree $\leq n$ with support in $\overline{D}$ can be represented.

There are several possible ways to perform step 3 in the procedure above. For instance, one may employ WEB-splines [46, Ch. 4.4] or the Schoenberg-spline basis on the interval with multiple knots at the endpoints [67]. However, we recall that the shape gradient is a distribution living on $\partial\Omega_0$ [24, Ch. 9, Thm 3.6]. Since it is usually possible to choose the hold-all domain D so that the set distance between ∂D and Ω_0 is sufficiently large, in most cases the technical step 3 can be omitted without introducing a relevant discretization error. Note also that often the hold-all domain D can be chosen to be of simple shape, e.g., a tensor product domain.

The finite-dimensional space of vector fields V_N is constructed employing multivariate cardinal B-splines of degree 2 constructed on D. More precisely,

vector fields belonging to V_N can be written as

$$\mathcal{V}_N(\mathbf{x}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{d} c_i^j \mathbf{e}_j \right) B_i^2(\mathbf{x}), \quad c_i^j \in \mathbb{R}, \tag{3.13}$$

where B_i^2 denotes the i-th multivariate B-spline of degree 2, and $\mathbf{e}_j$, $j = 1, \dots, d$, are basis vectors of $\mathbb{R}^d$.

The trial space V_N fulfills the assumptions of Theorem 3.1.5. Moreover, B-splines have compact support and are polynomial in each grid cell. These two properties are crucial for an efficient implementation of the algorithm presented in Section 3.4. Finally, using B-splines defined on a regular grid greatly simplifies the implementation.

3.4. Algorithm

As already mention in Section 3.3, the finite-dimensional shape optimizaton problem (3.5) is inherently highly nonlinear, and we have to rely on iterative methods to solve it. An approximation of the discrete optimal solution $\mathcal{V}_N^*$ can be retrieved with descent methods, which rely on the Fréchet derivative $d\tilde{\mathcal{J}}$ of $\tilde{\mathcal{J}}$ and are guaranteed to converge to (local) minima by the compactness of the finite-dimensional space $V_N \cap B_{1-\varepsilon}$. Formulas for the Fréchet derivative of $\tilde{\mathcal{J}}$ can easily be derived with the Lagrangian approach used in the proof of Proposition 2.1.2. The only difference is that in Proposition 2.1.2 we derived the Fréchet derivative in the origin, whereas here we need it for a generic vector field $\mathcal{V}$.

Lemma 3.4.1. *The Fréchet derivative of the operators defined on* $C^1(\overline{D}; \mathbb{R}^d)$

$$\mathcal{X} \mapsto \det \mathbf{D}T_{\mathcal{X}} \quad \textit{and} \quad \mathcal{X} \mapsto \mathbf{M}_{\mathcal{X}} \coloneqq (\det \mathbf{D}T_{\mathcal{X}}) \mathbf{D}T_{\mathcal{X}}^{-1} \mathbf{D}T_{\mathcal{X}}^{-T}$$

in the direction $\mathcal{W}$ *evaluated in* $\mathcal{V} \in B_{1-\varepsilon}$ *are given by*

$$\partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \coloneqq < \frac{\partial(\det \mathbf{D}T_{\mathcal{X}})}{\partial \mathcal{X}}, \mathcal{W} > |_{\mathcal{X}=\mathcal{V}} \quad = \det(\mathbf{D}T_{\mathcal{V}}) \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}\mathcal{W}), \tag{3.14}$$

and

$$\partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}} \coloneqq < \frac{\partial \mathbf{M}_{\mathcal{X}}}{\partial \mathcal{X}}, \mathcal{W} > |_{\mathcal{X}=\mathcal{V}}$$

$$= \det(\mathbf{D}T_{\mathcal{V}}) \Big(\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} - \mathbf{D}T_{\mathcal{V}}^{-1}(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}T_{\mathcal{V}}^{-T} \Big) , \tag{3.15}$$

respectively.

Proof. First of all, note that

$$\begin{aligned} \mathbf{D}T_{\mathcal{V}+\mathcal{W}} = \mathbb{1} + \mathbf{D}\mathcal{V} + \mathbf{D}\mathcal{W} &= \mathbf{D}T_{\mathcal{V}} + \mathbf{D}T_{\mathcal{V}}\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} , \\ &= \mathbf{D}T_{\mathcal{V}} \left(\mathbb{1} + \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} \right) . \end{aligned} \tag{3.16}$$

Thus, for $\|\mathcal{W}\|_{C^1}$ sufficiently small,

$$\begin{aligned} \det(\mathbf{D}T_{\mathcal{V}+\mathcal{W}}) &= \det(\mathbf{D}T_{\mathcal{V}} \left(\mathbb{1} + \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} \right)) , \\ &= \det(\mathbf{D}T_{\mathcal{V}}) \det \left(\mathbb{1} + \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} \right)) , \\ &= \det(\mathbf{D}T_{\mathcal{V}}) \left(1 + \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W}) + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2) \right) , \end{aligned} \tag{3.17}$$

which implies (3.14). The third equality in (3.17) follows from Taylor expansion of the determinant; see [57, Part 3, Sect. 8.3]. On the other hand, from (3.16) we see that

$$\begin{aligned} \mathbf{D}T_{\mathcal{V}+\mathcal{W}}^{-1} &= \left(\mathbb{1} + \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} \right)^{-1} \mathbf{D}T_{\mathcal{V}}^{-1} , \\ &= \left(\mathbb{1} - \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2) \right) \mathbf{D}T_{\mathcal{V}}^{-1} , \\ &= \mathbf{D}T_{\mathcal{V}}^{-1} - \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2) , \end{aligned}$$

and thus

$$\begin{aligned} \mathbf{D}T_{\mathcal{V}+\mathcal{W}}^{-1}\mathbf{D}T_{\mathcal{V}+\mathcal{W}}^{-T} = \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} &- \mathbf{D}T_{\mathcal{V}}^{-1}(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T\mathbf{D}T_{\mathcal{V}}^{-T}) \\ &- (\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}T_{\mathcal{V}}^{-T} + \mathcal{O}(\|\mathcal{W}\|_{C^1}^2) , \end{aligned}$$

which, multiplied with (3.17), gives (3.15). □

Remark 3.4.2. *The Fréchet derivative of $\tilde{\mathcal{J}}(\cdot, u)$ at $\mathcal{V}$ evaluated in the direction $\mathcal{W}$ is equal to the Eulerian derivative of $\mathcal{J}(T_{\mathcal{V}}(\Omega), u)$ in the direction $\mathcal{W} \circ (T_{\mathcal{V}}^{-1})$, because $T_{\mathcal{V}+\mathcal{W}} = T_{\mathcal{W}\circ T_{\mathcal{V}}^{-1}} \circ T_{\mathcal{V}}$.*

The formulas in the following example can be derived easily following the approach in the proof of Proposition 2.1.2 and employing Lemma 3.4.1.

Example 3.4.3. *The Fréchet derivative of* $\tilde{\mathcal{J}}$ *from* (3.8) *reads*

$$d\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W}) = \int_{\Omega_0} (j(u) - fp)\partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \\ - (\mathbf{grad}\, f \cdot \mathcal{W})p \det \mathbf{D}T_{\mathcal{V}} + \mathbf{grad}\, p \cdot (\partial_{\mathcal{W}}\mathbf{M}_{\mathcal{V}})\, \mathbf{grad}\, u \,\mathrm{d}\boldsymbol{x}\,,$$

where

$$\partial_{\mathcal{W}}\mathbf{M}_{\mathcal{V}} \coloneqq \det(\mathbf{D}T_{\mathcal{V}}) \Big(\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} \\ -\mathbf{D}T_{\mathcal{V}}^{-1}(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}T_{\mathcal{V}}^{-T} \Big)\,,$$

$$\partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \coloneqq \det(\mathbf{D}T_{\mathcal{V}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\,,$$

and where $p \in H_0^1(\Omega_0)$ *is the solution to the adjoint problem*

$$\begin{cases} -\operatorname{div}(\mathbf{M}_{\mathcal{V}}\, \mathbf{grad}\, p) &= -j'(u)(\det \mathbf{D}T_{\mathcal{V}}) \quad \text{in } \Omega_0\,, \\ p &= 0 \quad \text{on } \partial\Omega_0\,. \end{cases}$$

As Example 3.4.3 clearly illustrates, the Fréchet derivative of PDE constrained functionals depends on the solution u of the state problem and, possibly, on the solution p of the adjoint problem. As explicit analytic solutions of these boundary value problems are usually not available, one can replace them with approximate solutions, at the cost of introducing a perturbation error when solving the first order optimality condition (3.10). In particular, this perturbation error affects the quality of the descent directions.

Henceforth, we consider approximations by means of the finite element method, and we discuss what is the impact of replacing the solution u by its finite element counterpart u_h.

When stated as a volume integral, the map $u \mapsto d\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W})$ is usually continuous with respect to the energy norm of u. In Section 2.2, we showed that one can expect to observe superconvergence in the approximation of the operator $d\tilde{\mathcal{J}}(\mathcal{V},u;\cdot)$ for $\mathcal{V}=0$, which corresponds to the Eulerian derivative of $\mathcal{J}(T_{\mathcal{V}}(\Omega),u)$. Moreover, Remark 3.4.2 implies that this holds true for all $\mathcal{V} \in B_{1-\varepsilon}$. The same holds for evaluating the shape functional $\tilde{\mathcal{J}}(\mathcal{V},u)$.

To compute an approximation of the optimal solution $\mathcal{V}_N^*$, we adopt a "simulation-based optimization policy": the routines to compute the solution of the state problem and the Fréchet derivative are "embedded into an optimization loop" [42, Sect. 1.1]. In Algorithm 1 we present a pseudo-code for

a descent method based on the Armijo rule [43, Sect. 2.2.1.1]. The optimization algorithm is kept simple for the sake of readability of the pseudo-code. However, we mention that more advanced optimization algorithms have been succesfully applied to shape optimization problems [28,70,75]. The next paragraphs give a detailed description of the steps in Algorithm 1.

Algorithm 1 Projected gradient method with Armijo rule

1: Select initial design Ω_0, optimization step length δ, and parameters ε,$\gamma \in (0,1)$
2: Initialize $\mathcal{V}_N = 0$, $\mathcal{V}_N^{\text{temp}} = 0$,
3: Precompute all $\mathbf{B}_i$'s on quadrature nodes
4: **for** $ii = 1, \ldots, MAX^{\text{ITER}}$ **do**
5: Assemble finite element stiffness matrix $\tilde{\mathcal{A}}_{\mathcal{V}_N^{\text{temp}}}$ and load vector $\tilde{f}_{\mathcal{V}_N^{\text{temp}}}$
6: Compute finite element solution u_h of $\tilde{\mathcal{A}}_{\mathcal{V}_N^{\text{temp}}} u = \tilde{f}_{\mathcal{V}}$ in Ω_0
7: Compute $\tilde{\mathcal{J}}^{\text{new}} \coloneqq \tilde{\mathcal{J}}(\mathcal{V}_N^{\text{temp}}, u_h)$
8: **if** $ii > 1$ **and** $\tilde{\mathcal{J}}^{\text{new}} - \tilde{\mathcal{J}}^{\text{old}} > \gamma\delta d\tilde{\mathcal{J}}(\mathcal{V}_N, u_h; \mathcal{V}_N^{\text{new}})$ **then**
9: Update $\delta \leftarrow \delta/2$
10: **else**
11: Update $\delta \leftarrow 2\delta$
12: Update $\mathcal{V}_N \leftarrow \mathcal{V}_N^{\text{temp}}$, $\tilde{\mathcal{J}}^{\text{old}} \leftarrow \tilde{\mathcal{J}}^{\text{new}}$
13: Compute $\mathcal{V}_N^{\text{new}} = \text{argmin}_{\mathcal{W}_N \in V_N, \|\mathcal{W}_N\|_{H^1(D)}=1} d\tilde{\mathcal{J}}(\mathcal{V}_N, u_h; \mathcal{W}_N)$
14: **end if**
15: Set $\mathcal{V}_N^{\text{temp}} \coloneqq \mathcal{V}_N + \delta\mathcal{V}_N^{\text{new}}$
16: **while** $|\mathcal{V}_N^{\text{temp}}|_{C^1} > 1 - \varepsilon$ **do**
17: Update $\delta \leftarrow \delta/2$ and set $\mathcal{V}_N^{\text{temp}} \coloneqq \mathcal{V}_N + \delta\mathcal{V}_N^{\text{new}}$
18: **end while**
19: **end for**

Lines 1-3 are an initialization of the optimization routines. Step 3 is performed to improve computational efficiency; see Section 3.5.

In Line 6 we compute the finite element solution u_h of the state problem, which is required to evaluate the objective functional in Line 7 and the Fréchet derivative in Line 13.

In line 8 we test whether the Armijo rule [43, Sect. 2.2.1.1] is satisfied. If this is not the case, we reduce the optimization step δ. Otherwise, we increase it.

In Line 13 we compute the descent direction (if the Armijo condition is fulfilled). This is a delicate task and deserves a more thorough explanation. Let us recall that, for a Banach space S, a steepest descent direction d of a Fréchet differentiable operator $F : S \to \mathbb{R}$ at $s \in S$ is a solution of [43, Page 103]

$$\inf_{\|d\|_S=1} < dF(s), d >_{S^*,S} \ .$$

Thus, the steepest descent direction strongly depends on the Banach space S. Often, there is a certain freedom in the choice of the space S [48], and one has to decide which metric should be best taken into account. This choice has to be taken with great care because it may affect the performance of the steepest decent algorithm [62]. Note that this is also the case for finite-dimensional optimization problem (and is a motivation for the conjugate gradient method).

In our framework, the formulation (3.2) suggests to chose $S = C^1(\overline{D}; \mathbb{R}^d)$. However, the space $C^1(\overline{D}; \mathbb{R}^d)$ is not reflexive, and the steepest descent direction might not exists. The Banach space $W^{2,4}(D; \mathbb{R}^d)$ is introduced in Theorem 3.1.2 to show that an initial design Ω_0 may be improved via shape optimization, and in Theorem 2.2.4 to ensure superconvergence in the approximation of the shape gradient. Although relevant, these aspects are not related to the intrinsic properties of the shape optimization problem (3.2). These properties are usually encoded in the second order Fréchet derivative.

Employing knowledge on the shape Hessian is also not straightforward. To derive the second order Fréchet derivative $d^2\tilde{\mathcal{J}}$, the perturbation vector field $\mathcal{W}$ should be in $C^2(\overline{D}; \mathbb{R}^d)$ [23]. However, $d^2\tilde{\mathcal{J}}$ cannot be expected to be coercive in the $C^2(\overline{D}; \mathbb{R}^d)$-norm. Indeed, for any vector field $\mathcal{W}$ tangential to Ω_0 as well as for vector fields with a compact support that does not intersect $\partial\Omega_0$, it holds

$$d\tilde{\mathcal{J}}(\mathcal{V}, u; \mathcal{W}) = 0 \qquad \text{and} \qquad d^2\tilde{\mathcal{J}}(\mathcal{V}, u; \mathcal{W}, \mathcal{W}) = 0 \, .$$

Nevertheless, in several situations, the shape Hessian is a positive bilinear form when evaluated on vector fields with nonzero normal component on $\partial\Omega_0$. For instance, this is the case for the shape functional defined in (3.7), see [26]. Moreover, the shape Hessian can be expected to be a continuous bilinear form with respect to the $H^s(\partial\Omega_0)$-norm of the normal component of the vector fields. The regularity s of the "energy space" $H^s(\partial\Omega_0)$ depends on the problem under consideration; cf. [28]. In the seminal works [22, 28] it has been shown that shape optimization problems admit strict local minima, also called stable

minimizers, if the shape Hessian is also coercive with respect to the $H^s(\partial\Omega_0)$-norm of the normal component of the vector fields, that is,

$$d^2\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W},\mathcal{W}) \geq C\|\mathcal{W}\cdot\mathbf{n}\|^2_{H^s(\partial\Omega_0)}\,, \tag{3.18}$$

where $\mathbf{n}$ is the normal vector field on $\partial\Omega_0$ and $C>0$ is a constant independent of $\mathcal{W}$. Thus, coercivity in the $H^s(\partial\Omega_0)$-norm can be used as a criterion to distinguish between well- and ill-posed shape optimization problems [28]. Therefore, from a theoretical point of view, it is natural to consider the $H^s(\partial\Omega_0)$ representative of the Fréchet derivative, which is unique up to extensions into the domain D of its values on $\partial\Omega_0$ in the normal direction $\mathbf{n}$. Note that for ill-posed shape optimization problems this choice provides a regularization in the spirit of regularized sequential quadratic programming [17].

Therefore, we consider descent directions given as $H^1_0(D;\mathbb{R}^d)$-representatives of the Fréchet derivative, that is, solutions to

$$\min_{\|\mathcal{W}\|_{H^1(D;\mathbb{R}^d)}=1} d\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W})\,. \tag{3.19}$$

Up to a scaling factor, this amounts to solving the linear variational problem

$$(\mathcal{V}_N^{\text{new}},\mathcal{W}_N)_{H^1(D;\mathbb{R}^d)} = d\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W}_N) \quad \forall\,\mathcal{W}_N\in V_N\,, \tag{3.20}$$

which is equivalent to solving a discrete Laplacian with homogeneous Dirichlet boundary conditions on ∂D and a spline based Galerkin discretization.

By the continuity of the normal component trace

$$H^1_0(D;\mathbb{R}^d)\to H^{1/2}(\partial\Omega_0)\,,\quad \mathcal{W}\mapsto\mathcal{W}\cdot\mathbf{n}|_{\partial\Omega_0}\,,$$

the $H^1_0(D;\mathbb{R}^d)$-representative of the Fréchet derivative is the unique $H^1_0(D;\mathbb{R}^d)$-extension of the normal values of the $H^{1/2}(\partial\Omega_0)$-representative. Note that this is the proper "energy space" for the shape optimization problems considered in Sections 3.6 and 3.7; cf. [26, 28]. Finally, let us remark that constructing descent direction by solving (3.20) has also been used in [54] and agrees with the approach suggested in [70], which is motivated by deeper investigations on the metric of the "space of shapes" [69].

In Line 16 of Algorithm 1 we check that the updated vector field is admissible. If this is not the case, the optimization stepsize is reduced. By and large, it is a challenging task to compute the supremum norm numerically, and one

should content oneself with the evaluation of this quantity on a finite number of points (for instance, on the quadrature nodes employed in the FE code). It is also worth mentioning that the main goal of this step is to guarantee that the transformation $T_{\mathcal{V}}$ is a diffeomorphism. For this goal, the restrictions on the norm of $\mathcal{V}$ imposed in (3.1) are sufficient, but not necessary. Alternatively, one may verify that the values of $\det \mathbf{D}T_{\mathcal{V}}$ are bigger than a threshold value (as suggested in [9]). Transformations $T_{\mathcal{V}}$ that satisfy this condition are locally diffeomorphisms (but not globally).

Remark 3.4.4. *It might nevertheless happen that the (continuous) optimal solution $\mathcal{V}^*$ lies on the boundary of $B^1_{1-\varepsilon}$, and that, however, the value of $\tilde{\mathcal{J}}(\mathcal{V}^*)$ is not yet satisfactory for convergence purposes. For instance, this might be the case when the initial guess Ω_0 is poorly chosen. In this situation a remedy is to select the retrieved shape as initial guess, and to restart the algorithm. This approach constructs a so-called " discrete gradient flow" [25].*

Practically, this can be done by either creating a new mesh of $T_{\mathcal{V}^*}(\Omega_0)$ or by replacing the transformation $T_{\mathcal{V}}$ with the composition $T_{\mathcal{V}} \circ T_{\mathcal{V}^*}$ (exploiting the fact that the composition of diffeomorphisms is again a diffeomorphism). This latter approach can be made computationally affordable by simply re-evaluating all $\mathbf{B}_i$'s on the mapped quadrature nodes; see the next paragraph on the implementation of Algorithm 1. Note also that Theorems 3.1.2 and 3.1.5 still hold as long as a finite number of compositions is considered. However, the ellipticity constant of the operator $\tilde{\mathcal{A}}_{\mathcal{V}}$ in (3.6) *might tend to zero as more and more compositions are considered, and this would adversely affect the constants in the convergence estimates of FEM.*

3.5. Implementation in Matlab

We give details for an efficient implementation of Algorithm 1 in MATLAB for a two dimensional problem. A complete MATLAB code is available in Appendix A. The state problem is solved by piecewise linear Lagrangian finite elements on triangles. The code can easily be extended to higher order polynomials by updating the routines accordingly. Our implementation follows the lines of [32, Sect. 3].

Firstly, we note that for each iteration the computational cost of Algorithm 1 is mainly due to:

- assembling the linear system in Line 5,

- solving the linear system in Line 6,
- evaluating the shape functional in Line 7,
- computing the descent direction in Line 13 (see (3.20)),
- testing the feasibility of the new descent direction in Line 16.

Except for solving the linear system, these steps require the evaluation of the Jacobian $\mathbf{D}\mathcal{V}$, and thus of all partial derivatives of the B-splines, in each quadrature point on the finite element mesh. Since function calls are generally expensive, we pre-evaluate all these partial derivatives in all quadrature points in Line 3, and assemble two sparse matrices `VdX` and `VdY` of size (`nQP` · `nElements`) × `nBsplines`, where `nQP` is the number of quadrature points for a triangle, `nElements` is the number of triangles, and `nBsplines` is the number of B-splines B_i (which corresponds to N in (3.13)). In these two matrices we store the values of the partial derivatives $\partial_x B_i$ and $\partial_y B_i$, respectively. Then, the entries of the Jacobian $\mathbf{D}\mathcal{V}$ can be obtained by multiplying the matrices `VdX` and `VdY` with the vectors `cX` and `cY`, whose entries are the coefficients c_i^1 and c_i^2 of the expansion of $\mathcal{V}$ given in (3.13), respectively. This yields an effective speed-up of the computations at the cost of memory. Note that the number of nonzero entries of `VdX` and `VdY` is significantly less than `nQP` · `nElements` · `nBsplines` because B-splines have compact support.

With the matrices `VdX` and `VdY` at our disposal, it is very simple to test on the quadrature points the feasibility of the new descent direction in Line 16. We conclude this section by discussing the assembly of the stiffness matrix in Line 5, the computation of the solution of the related linear system, the evaluation of the the shape functional in Line 7, and the computation of the descent direction in Line 13.

The weak formulation of the state constraint of (3.8) suggests that

$$(u, v) \mapsto \int_{\Omega_0} \nabla u \cdot \mathbf{M} \nabla v \, \mathrm{d}\boldsymbol{x}$$

can represent the bilinear form of a generic second order elliptic operator. By and large, $\mathbf{M}$ is a nonconstant positive defined diffusion matrix. However, as the gradient of piecewise linear functions is piecewise constant, the matrix function $\mathbf{M}$ can be replaced by its mean values in each triangle. Then, a fully vectorized matrix assembly of the stiffness matrix can easily be implemented

based on the details given in [32, Sect. 3.4], where the authors describe the assembly of the stiffness matrix arising from the bilinear form

$$(u, v) \mapsto \int_{\Omega_0} \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} \,.$$

The following step is to solve the linear system to compute the finite element solution u_h. In [8] it has been shown that multigrid strategies can be successfully applied in the context of shape optimization with moving meshes. We believe that similar ideas can be employed in our strategy. However, not to introduce additional error terms, we rely on the MATLAB function `mldivide`, which implements a direct solver.

Let $u^{\mathrm{ref}} \in H^1(\mathbb{R}^d)$. We consider the quadratic functional

$$\mathcal{J} \coloneqq \int_{\Omega} \nabla(u - u^{\mathrm{ref}}) \cdot \nabla(u - u^{\mathrm{ref}}) + (u - u^{\mathrm{ref}})^2 \mathrm{d}\boldsymbol{x} \,,$$

as a representative for general shape functionals. The corresponding functional in parametric form reads

$$\tilde{\mathcal{J}} \coloneqq \int_{\Omega_0} \nabla(u - \tilde{u}^{\mathrm{ref}}) \cdot \mathbf{M}_{\mathcal{V}} \nabla(u - \tilde{u}^{\mathrm{ref}}) + (u - \tilde{u}^{\mathrm{ref}})^2 \det(\mathbf{D}T_{\mathcal{V}}) \mathrm{d}\boldsymbol{x} \,,$$

where $\tilde{u}^{\mathrm{ref}} \coloneqq u^{\mathrm{ref}} \circ T_{\mathcal{V}}$. The contribution of the first integrand can be evaluated efficiently employing the stiffness matrix assembled in Line 6. The second integrand is a scalar function, and does not represent any computational challenge, because the term $\det(\mathbf{D}T_{\mathcal{V}})$ can be computed efficiently with the matrices `VdX` and `VdY`.

The formulas of shape gradients of PDE constraint shape functionals strongly depend both on the shape functional itself and on the PDE constraint. Therefore, it is not possible to give a detailed description of its efficient implementation. We simply remark that, to compute the descent direction in Line 13, the shape gradient has to be evaluated on all basis vector fields $B_i \mathbf{e}_j$. By and large, the evaluation of the shape gradient on a fixed direction corresponds to an integration in volume, when the solution of the state problem is approximated with FEM [45]. Thus, 2·`nBsplines` integrals have to be computed. Employing MATLAB's pointwise arithmetics [32], these integrations can be performed simultaneously with a fully vectorized implementation. However, we stress that this step requires a large amount of memory. We strongly recommend to exploit sparsity to reduce the active memory requirements. We

finally recall that, in MATLAB, function input variables are not copied as long as they are not modified within the body of the function. We refer to the MATLAB documentation for further details.

3.6. Numerical experiments: the well-posed case

We investigate the performance of the algorithm proposed in Section 3.4 on a prototypical well-posed shape optimization problem that stems from the class of Bernoulli free boundary problems.

We consider the Dirichlet boundary value problem

$$-\Delta u = 0 \ \text{ in } \Omega\,, \quad u = u_{\text{in}} \quad \text{on } \partial\Omega^{\text{in}}\,, \quad u = u_{\text{out}} \ \text{ on } \partial\Omega^{\text{out}}\,, \tag{3.21}$$

stated on the annular domain $\Omega \subset \mathbb{R}^2$, see Figure 3.3. The goal is to find the shape of $\partial\Omega^{\text{in}}$ (or of $\partial\Omega^{\text{out}}$) so that the Neumann trace $\frac{\partial u}{\partial \mathbf{n}}$ is equal to a prescribed function g on $\partial\Omega^{\text{in}}$ (or on $\partial\Omega^{\text{out}}$). This additional requirement is stated on just one of the two boundaries, and only this one is considered as "free" to move. For the sake of simplicity, we assume that the Dirichlet data u_{in} and u_{out}, as well as the Neumann data g are constants.

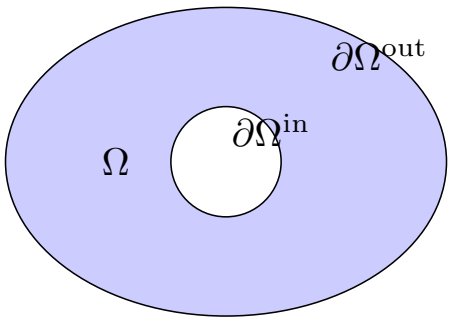

Figure 3.3.: Annular domain for Bernoulli free boundary problems.

This Bernoulli free boundary problem can be reformulated as the following shape optimization problem [27]

$$\inf_{\Omega\in\mathcal{U}_{\text{ad}}(\Omega_0)} \int_\Omega (\nabla u)^2 + g^2 \mathrm{d}\boldsymbol{x}\,, \quad \text{subject to} \begin{cases} -\Delta u &= 0 & \text{in } \Omega\,, \\ u &= u_{\text{in}} & \text{on } \partial\Omega^{\text{in}}\,, \\ u &= u_{\text{out}} & \text{on } \partial\Omega^{\text{out}}\,. \end{cases} \tag{3.22}$$

To construct a test case for numerical simulations, we set the optimal shape of Ω to be a ring with radii r_{in} and r_{out}. The solution of the Dirichlet problem (3.21) reads (in polar coordinates)

$$u(r) = c_1 \log(r) + c_2\,, \quad r \in [r_{\text{in}}, r_{\text{out}}]\,,$$

where the coefficients c_1, and c_2 are determined by the Dirichlet boundary conditions. It immediately follows that admissible Neumann data g must satisfy $g = -c_1/r_{\text{in}}$ (or $g = c_1/r_{\text{out}}$), and that the minimal value of the shape functional in (3.22) reads

$$\mathcal{J}^{\min} = 2\pi \left((u_{\text{out}} - u_{\text{in}})c_1 + g^2(r_{\text{out}}^2 - r_{\text{in}}^2)/2\right)\,.$$

Note that $\mathcal{J}^{\min}$ is positive. Well-posedness of this shape optimization problem is due to the $H^{1/2}(\partial\Omega_0)$-coercivity (see Equation (3.18)) of the Shape Hessian in the optimal shape, which can be showed performing a shape analysis in polar coordinates [27].

The parametric form of (3.22) reads

$$\inf_{\mathcal{V}\in B_{1-\varepsilon}} \int_{\Omega_0} \nabla u \cdot \boldsymbol{M}_{\mathcal{V}} \nabla u + g^2(\det \mathbf{D}T_{\mathcal{V}})\mathrm{d}\boldsymbol{x}\,, \tag{3.23}$$

$$\text{subject to} \quad \begin{cases} -\operatorname{div} \mathbf{M}_{\mathcal{V}}\, \mathbf{grad}\, u &= 0 \qquad \text{in } \Omega_0\,, \\ u &= u_{\text{in}} \quad \text{on } \partial\Omega_0^{\text{in}}\,, \\ u &= u_{\text{out}} \quad \text{on } \partial\Omega_0^{\text{out}}\,, \end{cases}$$

where $\mathbf{M}_{\mathcal{V}} \coloneqq (\det \mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}$. The Fréchet derivative of the shape functional in (3.23) reads

$$d\tilde{\mathcal{J}}(\mathcal{V}, u; \mathcal{W}) = \int_{\Omega_0} \nabla u \cdot (\partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}}) \nabla u + g^2(\partial_{\mathcal{W}} \det \mathbf{D}T_{\mathcal{V}})\,\mathrm{d}\boldsymbol{x}\,. \tag{3.24}$$

For our first investigations, we impose the additional constraint $\frac{\partial u}{\partial \mathbf{n}} = g$ on $\partial\Omega^{\text{in}}$. We choose a hold-all domain D that does not intersect the external boundary $\partial\Omega^{\text{out}}$, so that the set of admissible domains comprises domains obtained by perturbing only the internal boundary $\partial\Omega^{\text{in}}$.

Henceforth, $\partial\Omega_0^{\text{out}}$ is a circle of radius 1.2 centered in the origin, $u_{\text{in}} = 0$, and $u_{\text{out}} = -1$. We set $g = (-0.55\log(0.55/1.2))^{-1}$, so that the internal boundary of the optimal solution is a circle of radius 0.55 centered in the origin. The internal boundary $\partial\Omega_0^{\text{in}}$ of the initial domain is a circle of radius 0.5 centered

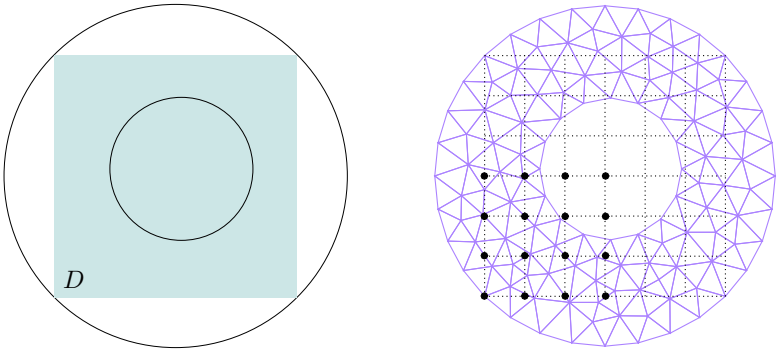

Figure 3.4.: *Left:* Initial guess Ω_0 and hold-all domain D. *Right:* A triangular grid is generated on Ω_0 to compute the finite element solution u_h. The hold-all domain D is covered with a regular grid used to generate multivariate quadratic B-splines. Dots indicate the lower left corner of the support of the B-splines.

in $[0.04, 0.05]$, see Figure 3.4. The hold-all Domain is a square centered in the origin with edges of length $1.2 \cdot \sqrt{2}$.

We consider finite element solutions computed with linear Lagrangian finite elements on quasi-uniform triangular meshes. Integrals in the domain are computed by a 7-point quadrature rule of order 6 in each triangle. The boundary of the computational domain is approximated by a polygon, which will not affect the convergence of linear finite elements [16, Sect. 10.2]. The optimization step δ is initially set to $\delta = 0.3$ and the parameter ε to $\varepsilon = 0.05$ (instead of $|\mathcal{V}_N^{\text{temp}}|_{C^1} > 1-\varepsilon$ we test that $\det \mathbf{D}\mathcal{V}_N^{\text{temp}} > \varepsilon$). Finally, we replace the Armijo rule condition

$$\tilde{\mathcal{J}}^{\text{new}} - \tilde{\mathcal{J}}^{\text{old}} \leq \gamma \delta d\tilde{\mathcal{J}}(\mathcal{V}_N, u_h; \mathcal{V}_N^{\text{new}})$$

with

$$|\tilde{\mathcal{J}}^{\text{new}} - \mathcal{J}^{\min}| - |\tilde{\mathcal{J}}^{\text{old}} - \mathcal{J}^{\min}| \leq \gamma \delta d\tilde{\mathcal{J}}(\mathcal{V}_N, u_h; \mathcal{V}_N^{\text{new}}),$$

so that the algorithm does not get stuck if, due to numerical error, $\tilde{\mathcal{J}}^{\text{new}}$ becomes smaller than the minimal value $\mathcal{J}^{\min}$. The parameter γ is set to $\gamma = 0.1$.

We perform 10 optimization steps employing 36 uniform multivariate quadratic B-splines. The finite element mesh has 1728 nodes and and 3264 triangles. In

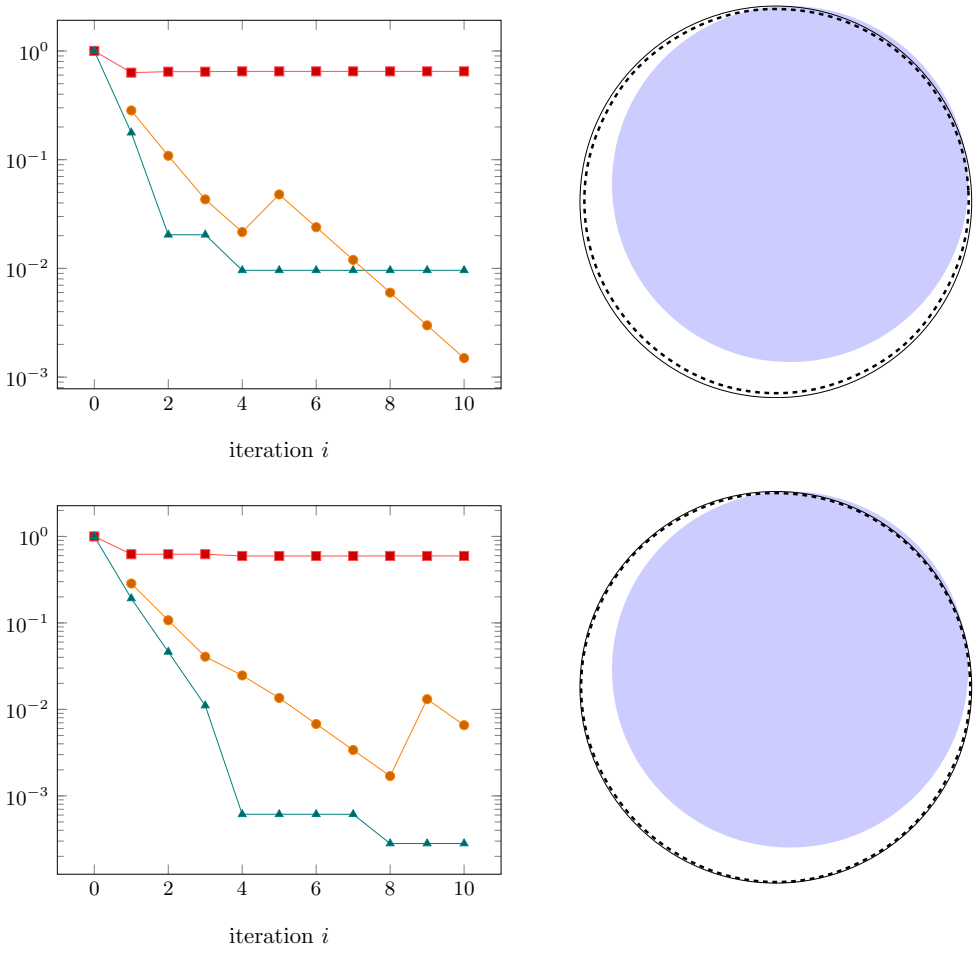

Figure 3.5.: Shape optimization employing 36 uniform multivariate quadratic B-splines and a finite element mesh with 1728 nodes and 3264 triangles (*first row*) and 6720 nodes and 13056 triangles (*second row*). *Left:* convergence history (▲), optimization steps (●), and $\min \det \mathbf{D}\mathcal{V}$ (■). *Right:* Initial shape (filled area), target shape (continuous line), and shape recovered after 10 optimization steps (dashed line).

Figure 3.5 (first row),we display (on the left) the convergence history

$$|\tilde{\mathcal{J}}^{(k)} - \mathcal{J}_{\min}|/|\tilde{\mathcal{J}}^{(0)} - \mathcal{J}_{\min}|\,, \tag{3.25}$$

the optimization step length $\delta^{(k)}$, the minimum of $\det \mathbf{D}\mathcal{V}_N^{\text{temp}}$, and (on the right) the initial interior boundary $\partial\Omega_0^{\text{in}}$, the exact solution, the retrieved approximate optimal solution. We observe that the algorithm succeeds in recovering a decent approximation of the exact solution. After 5 steps the optimization routine stagnates because the finite element error in the approximation of the state variable u affects the computation of the descent direction (as well as the evaluation of $\tilde{\mathcal{J}}$, and thus the reliability in the evaluation the Armijo rule condition). The results improve when the experiment is repeated on a finite element mesh obtained after one uniform refinement, see Figure 3.5 (second row).

In the next experiment we investigate the impact of the finite element and of the B-spline discretization on the quality of the recovered approximate optimal shapes. The design of the experiment reads as follows. We consider 10 B-spline discretizations constructed employing from 3^2 to 12^2 uniform multivariate quadratic B-splines. For each B-spline discretization we start from the coarsest mesh and perform 20 optimization steps. The recovered shape is considered as optimal for the current finite element mesh. Then, we perform a uniform refinement of the finite element mesh and compute 20 optimization steps taking as initial guess the optimal shape of the previous finite element discretization. The procedure is repeated on four additional meshes obtained through uniform refinement. Finally, for each optimal shape we evaluate the misfit functional employing a finite element mesh that is obtained after an additional uniform refinement. In Figure 3.6 we plot the relative difference of the misfit functional computed on the approximate optimal shapes with respect to $\mathcal{J}^{\min}$ versus the width of the B-spline grids and the width of the finite element meshes. We observe quadratic convergence with respect to the finite element discretization when enough B-splines are employed. Conversely, we observe that the error introduced by the B-spline discretization is negligible compared to the finite element discretization error as soon as 25 B-splines are employed.

We repeat this experiment by employing second order Lagrangian finite elements with parabolic boundary approximation of the domain boundaries. The results are displayed in Figure 3.7. In this case it is less immediate to extract a convergence rate with respect to finite element discretization because for fine finite element meshes we observe a saturation effect. This is partly due to the small number of optimization steps. In Figure 3.8 (left) we plot the relative difference of the misfit functional with respect to its minimal value $\mathcal{J}^{\min}$

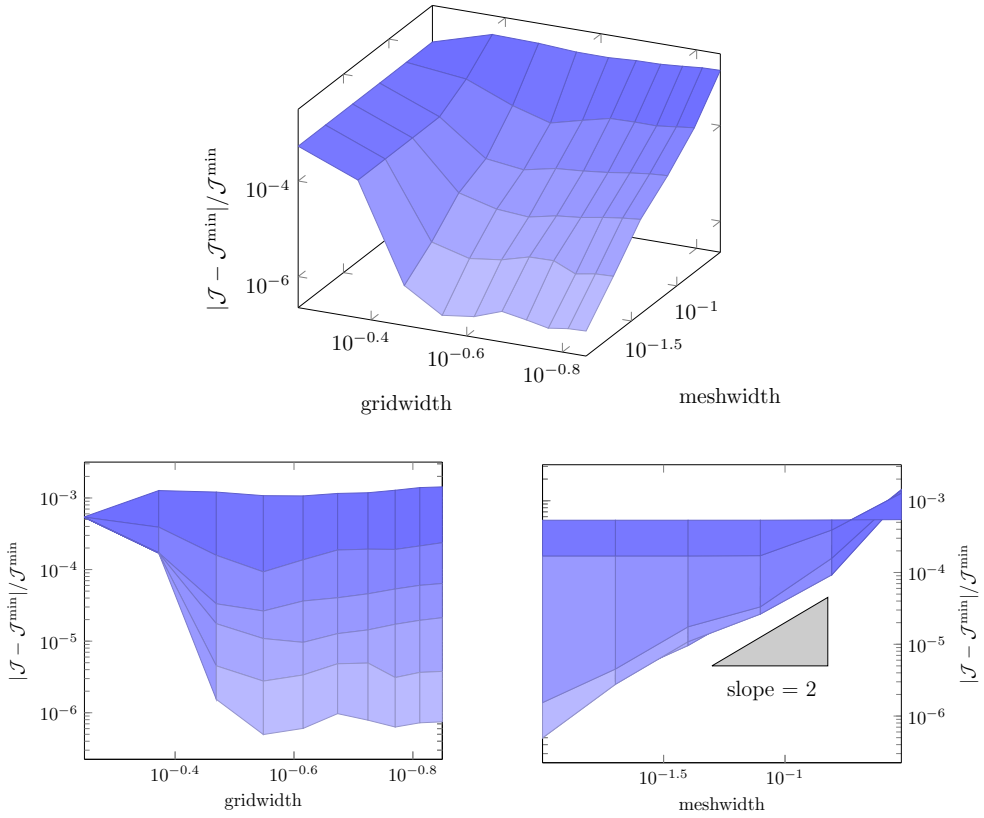

Figure 3.6.: Impact of B-spline and (linear) finite element discretization on the quality of the retrieved approximate solutions. *Left:* the error introduced by the B-spline discretization is negligible as soon as 25 B-splines are employed. *Right:* We observe quadratic convergence with respect to the finite element discretization.

versus the width of the finite element meshes when the approximate optimal shapes are obtained employing 144 B-splines and performing 20, 100, and 200 optimization steps. Firstly, we observe that the number of optimization steps affects the accuracy that is achieved for the fifth and the sixth finite element meshes. Secondly, two different rates of convergence are noticeable. On the first three meshes we observe algebraic convergence of rate 4.2, whereas on the last 4 meshes the rate decreases to 1.8. A possible explanation of this be-

havior is that the entries of matrix $\mathbf{M}_{\mathcal{V}}$ are in $W^{1,\infty}(D;\mathbb{R}^d)$, when the vector field $\mathcal{V}$ is discretized with quadratic B-splines. It follows that the solution u of (3.23) may lie in $H^2(\Omega_0)$ only [35, Ch. 8.4]. In this case, we could not expect to improve the convergence rate by increasing the polynomial order of the finite element discretization. Indeed, this bending in the convergence line is not observed when the vector field $\mathcal{V}$ is discretized with cubic B-splines, see Figure 3.8 (right). In this picture, the algebraic convergence rate on the first 3 meshes reads 3.93. Finally, we observe a more relevant impact of the B-spline discretization compared to the previous experiment. However, the convergence behavior is not conspicuous and we refrain from computing an algebraic convergence rate.

Finally, we design a numerical experiment to stress the importance of the choice of a "good" initial shape Ω_0. Again, we consider the interior Bernoulli free boundary problem (3.23). We choose $\partial\Omega^{\text{out}}$ as well as the hold-all domain D to be squares with edges of length 2 centered in the origin and $\partial\Omega^{\text{in}}$ to be a circle with radius of length 0.5 centered in the origin. The internal boundary of the optimal solution is a circle of radius 0.8 centered in the origin. In Figure 3.9, we display (on the left) the convergence history, the optimization step length $\delta^{(k)}$ and the minimum of $\det \mathbf{D}\mathcal{V}$, and (on the right) the initial interior boundary $\partial\Omega_0^{\text{in}}$, the exterior boundary $\partial\Omega^{\text{out}}$, the exact solution, the retrieved approximate optimal solution, and the quadrature nodes where $\det \mathbf{D}\mathcal{V}$ is smaller than 0.1 (after the fourth iteration). We observe that the algorithm struggles to generate a descent direction that, at the same time, has sufficiently big radial values on $\partial\Omega^{\text{in}}$, vanishes on ∂D, and for which $\det \mathbf{D}\mathcal{V}$ is not too small.

Next, we consider a test case where the additional constraint $\frac{\partial u}{\partial \mathbf{n}} = g$ is imposed on the exterior boundary $\partial\Omega^{\text{out}}$. This case presents an additional issue because the internal boundary $\partial\Omega^{\text{in}}$ cannot be modified. Therefore, the hold-all domain cannot be simply connected. A very simple way to construct a conforming finite-dimensional space of vector fields consists in covering the initial guess Ω_0 with a regular grid which is used to construct B-splines and then discarding all B-splines whose support intersects the internal boundary $\partial\Omega^{\text{in}}$. The resulting hold-all domain is an annular domain with staircase boundary, see Figure 3.10. For the next numerical experiments, we employ uniform multivariate cubic B-splines and linear Lagrangian finite elements.

As initial guess for this experiment, we select $\partial\Omega_0^{\text{out}}$ to be an ellipse with major semi-axis of length 1.5 and minor semi-axis of length 1.3. The internal

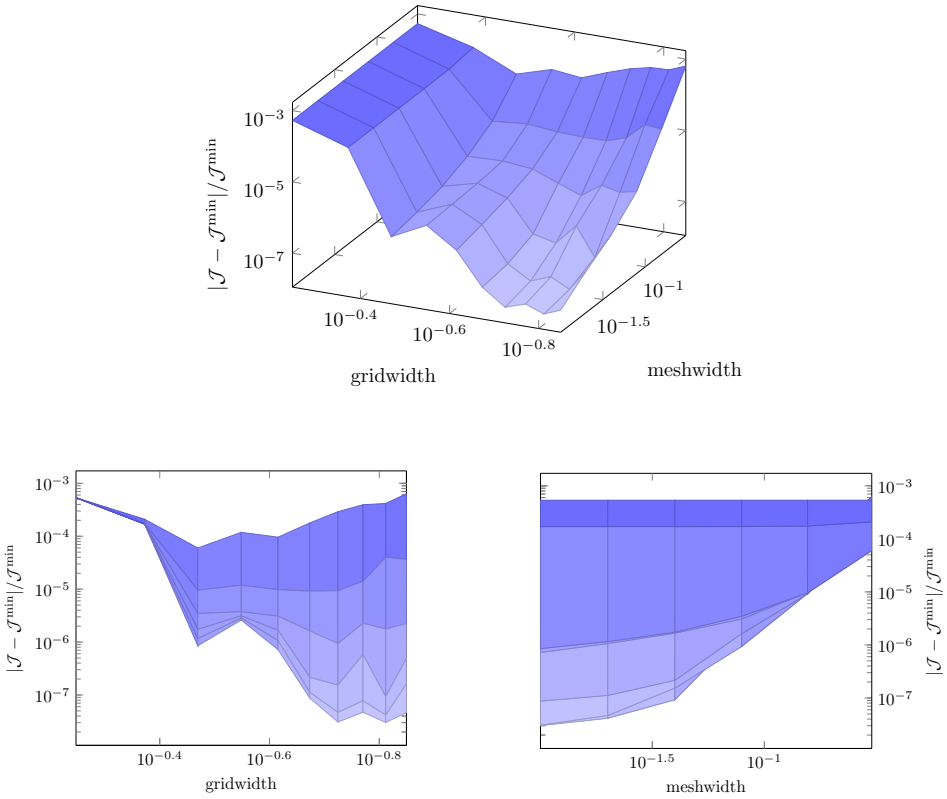

Figure 3.7.: Impact of B-spline and (second order) finite element discretization on the quality of the retrieved approximate solutions.

boundary $\partial\Omega_0^{\text{in}}$ is a circle of radius 0.5 centered in the origin. The domain Ω_0 is covered with a regular grid of width 0.255 over which the trial space V_N is constructed. The finite element solution u_h is computed on the mesh displayed in Figure 3.10 (left). Despite the coarseness of the mesh and the moderate resolution of the B-spline grid, after twelve optimization steps we already recover a satisfactory approximation of the target boundary; see Figure 3.11 (top left). As already observed above, we can improve the quality of the recovered solution by computing the finite element solution u_h on a finer mesh. The results obtained after 3 uniform refinements of the mesh are displayed in

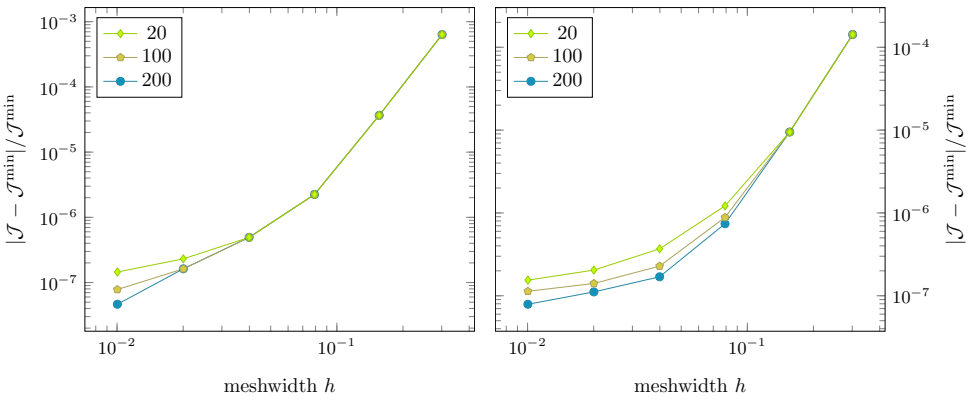

Figure 3.8.: Impact of number of optimization steps on convergence with respect to finite element discretization. *Left:* When the vector field $\mathcal{V}$ is discretized with quadratic B-splines, the number of optimization steps affect the accuracy reached on the fifth and the sixth finite element mesh. From the third mesh on, the rate of convergence decreases due to a lack of regularity of the state variable u. *Right:* When the vector field $\mathcal{V}$ is discretized with cubic B-splines, saturation in the convergence lines is due to an insufficient amount of optimization steps.

Figure 3.11 (bottom left).

In this case it is interesting to investigate the performance of the algorithm when the initial domain possesses corners, whereas the optimal solution does not. Therefore, we repeat the experiment and select as initial design a square with edges of length 2.06 (see Figure 3.10, right). Again, after twelve steps we recover a satisfactory approximation of the target boundary; see Figure 3.11 (top right). Although corners cannot be smoothed with a diffeomorphism, the quality of the approximate solution improves by computing the finite element solution u_h on a finer mesh; see Figure 3.11 (bottom right). Note that it would be more difficult to interpret an equivalent test for an Bernoulli interior free boundary problem because the corners of the initial guess would introduce a singularity in the state variable u [51].

Finally, we briefly discuss how to balance the B-spline resolution and the finite element discretization errors. Unfortunately, the nonuniqueness in the representation of shapes via diffeomorphisms makes it difficult to derive con-

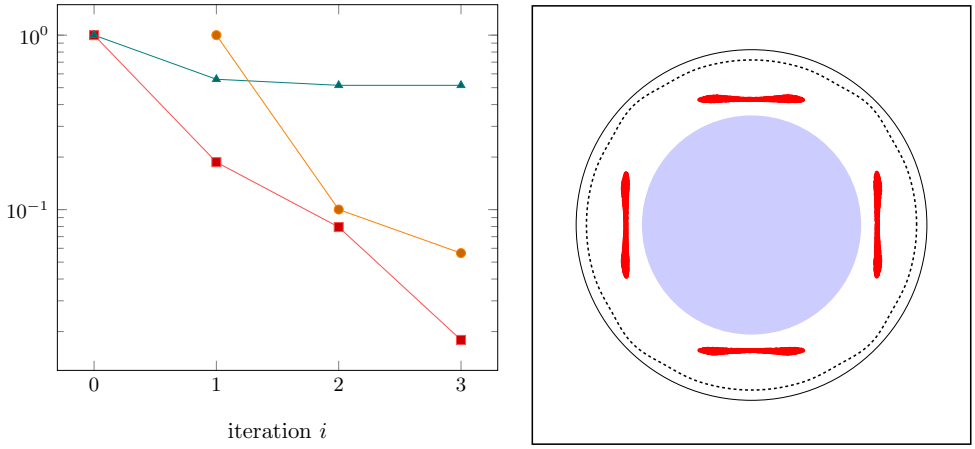

Figure 3.9.: *Left:* convergence history (▲), optimization steps (●), and $\min \det \mathbf{D}\mathcal{V}$ (■). *Right:* Initial shape (filled area), target shape (continuous line), shape recovered after 3 optimization steps (dashed line), and quadrature points where $\det \mathbf{D}\mathcal{V}$ is smaller than 0.1 (red dots).

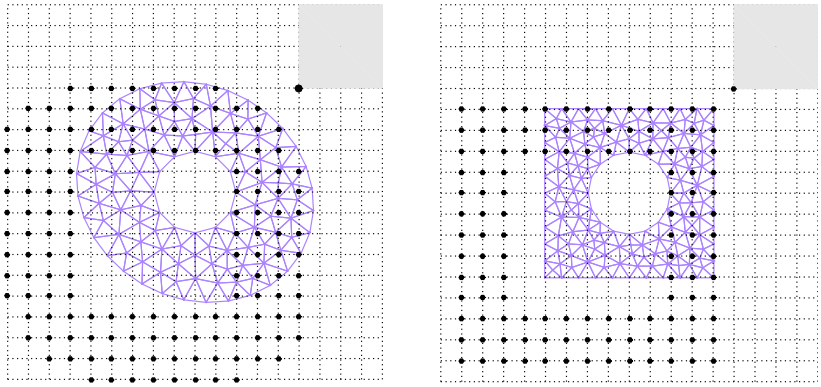

Figure 3.10.: The initial guess Ω_0 is covered with a regular grid used to generate cubic B-splines. Dots indicate the lower left corner of the support of the active B-splines. The square in the top right corner indicates the support of a cubic B-spline. A triangular grid is generated on Ω_0 to compute the finite element solution u_h.

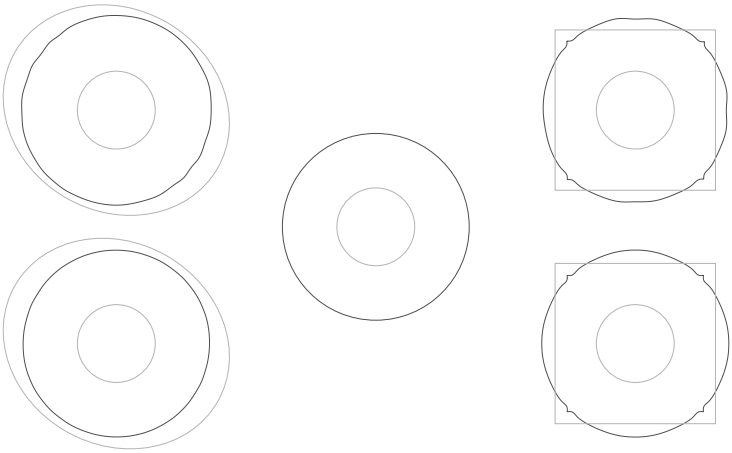

Figure 3.11.: Approximate optimal boundary retrieved after twelve iterations of Algorithm 1. Light gray lines indicate the boundary of the initial guess Ω_0. Results in the first row are obtained using the meshes displayed in Figure 3.10. Despite the coarseness of the mesh and the low resolution of the B-spline grid, we recover a decent approximation (dark gray line) of the optimum (exterior boundary of the annulus in the middle). The results can be improved by computing on finer meshes (second row).

vergence rates for the fully discretized problem. Therefore, we have to content ourselves with a naive discussion. From Figures 3.6 and 3.7 we observe that the finite element discretization has a very high impact on the quality of the retrieved solution. This is also observed in Figure 3.11. Therefore, in real applications, one should approximate the state variable (and the adjoint variable) on the finest mesh possible. As for the B-spline resolution, we observe that good results can be obtained even employing relatively few B-splines. Thus, to save computation time, it is possible to first perform shape optimization with a relatively coarse resolution. Then, to investigate whether the algorithm has fully converged, one could embed the so far computed discrete vector field on a nested space spanned by basis functions generated on a grid with half the meshwidth [47, Sect. 7.6], and restart the shape optimization routine. This approach has been pursued in [10]. We show the viability of this approach with a numerical experiment. As model problem, we consider the

Bernoulli exterior free boundary problem from above with an ellipse as initial guess, see Figure 3.10 (left). To approximate the state variable u we employ a finite element mesh with 25696 nodes and 50688 triangles. In Figure 3.12 we show the evolution of (3.25) for V_N constructed on a regular grid of width 0.51 (▲) and 0.255 (■). The former trial space comprises 54 active[2] B-splines, whilst the latter has 152 active B-splines. We see that the resolution of V_N affects the quality of the retrieved approximate optimal solution. Then, we pursue the adaptive strategy by switching (via spline interpolation) to a finer space of vector fields after 10 iterations. From this point on, only coefficients of new basis functions that intersect the boundary are updated. The evolution of (3.25) for this strategy is displayed in Figure 3.12 (●). We see that we are able to improve the quality of the approximate optimal solution by switching to a finer space after 10 iterations.

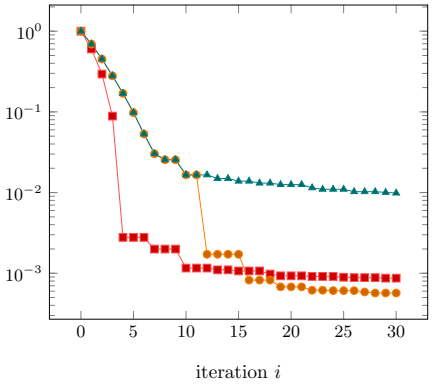

Figure 3.12.: Evolution of the scaled absolute error (3.25) for a coarse (▲) and a finer (■) trial space V_N. Switching to a finer space after 10 iterations, it is possible start with a coarse trial space and still retrieve an approximate solution with good quality (●).

[2]We just consider B-splines whose support intersects $\partial\Omega^{\text{out}}$.

3.7. Numerical experiments: the ill-posed case

We test our algorithm on a prototypical ill-posed inverse problem. Let B be a fixed subdomain of a domain Ω and let $u_t \in L^2(B)$ be a given target function. The goal is to find the optimal domain that contains B, so that the shape functional

$$\mathcal{J}(\Omega) \coloneqq \int_B (u - u_t)^2 \mathrm{d}\boldsymbol{x}\,, \quad \text{subject to} \quad \begin{cases} -\Delta u &= 0 \quad \text{in } \Omega\,, \\ u &= g \quad \text{on } \partial\Omega\,, \end{cases} \tag{3.26}$$

attains its minimum, where g is a prescribed Dirichlet boundary condition.

As explained in [19], elliptic regularity theory implies that the solution u of the state problem is in $H^2(\Omega)$ as soon as Ω is convex or of class C^2. Therefore, the range of the operator $\mathcal{V} \mapsto u|_B$ is at most a dense subset of $L^2(\Omega)$ [15, Thm. 7.2]. Thus, the shape optimization problem (3.26) is ill-posed. An alternative explanation of the ill-posedness of (3.26) from a shape optimization point of view can be found in [26]. Here, the functional $\mathcal{J}$ is replaced with a second order expansion, and it is shown that the shape hessian is a compact operator, which cannot be positive definite.

In our test case, Ω_0 is a rectangle with corners located at (-4,-2),(4,-2),(4,2),(-4,2), with a hole of the shape of a circle of radius 0.5 centered in (-2,0), see Figure 3.13. The goal is to find the shape of this internal boundary. The region of interest B is the square with corners (0,-2),(4,-2),(4,2),(0,2). The hold-all domain D is the square with corners (-3,-1),(-1,-1),(-1,1),(-3,1), so that both the external boundary and the region of interest B cannot be modified during the optimization routine. The target function u_t is of the form

$$u_t(\mathbf{x}) = c_1 \log(\sqrt{(\mathbf{x} - \mathbf{c}_t)^2}) + c_2\,,$$

where $c_1, c_2 \in \mathbb{R}$ and $\mathbf{c}_t \in \mathbb{R}^2$. In particular, we choose these parameters so that $u_t = 0$ on a circle of radius $r_t \in \mathbb{R}$ centered in $\mathbf{c}_t$ and $u_t = 1$ on a circle of radius 1 centered in $\mathbf{c}_t$. We prescribe $g = 1$ on the internal boundary and $g = u_t$ on the external boundary, so that the internal boundary of the optimal solution of (3.26) is a circle of radius r_t centered in $\mathbf{c}_t$.

The shape derivative of the shape optimization problem (3.26) recast in parametric form reads

$$d\tilde{\mathcal{J}}(\mathcal{V}, u; \mathcal{W}) = \int_{\Omega_0} \mathbf{grad}\, p \cdot \partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}}\, \mathbf{grad}\, u \,\mathrm{d}\boldsymbol{x}\,,$$

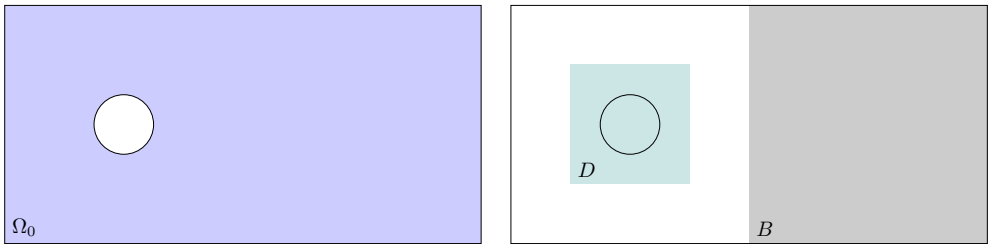

Figure 3.13.: *Left:* Initial guess Ω_0 for the ill-posed test case (3.26). *Right:* hold-all domain D and region of interest B.

where u and p are the weak solutions of

$$\begin{aligned} -\operatorname{div}(\mathbf{M}_{\mathcal{V}} \mathbf{grad}\, u) &= 0 && \text{in } \Omega_0\,, && u = g \quad \text{on } \partial\Omega_0\,, \\ -\operatorname{div}(\mathbf{M}_{\mathcal{V}} \mathbf{grad}\, p) &= -\chi_B 2(u - u_t) && \text{in } \Omega_0\,, && u = 0 \quad \text{on } \partial\Omega_0\,, \end{aligned}$$

respectively.

In the numerical experiments, we approximate u and p with linear Lagrangian finite elements on a triangular mesh with 82560 nodes and 163840 triangles. The internal boundary is approximated by a polygon.

An interesting consideration that is related to the ill-posedness of the test case under consideration is the low-sensitivity of the misfit functional $\mathcal{J}$ with respect to perturbations of the boundary. In Table 3.1 we list the value of $\mathcal{J}$ when the coefficients of u_t are chosen so that the optimal solution is a circle of radius $r_t = 0.6$ centered in $\mathbf{c}_t = 0.1(\cos\phi, \sin\phi)^T$, where $\phi = 0, 2\pi/6, \ldots, 10\pi/6$. We observe that the value of $\mathcal{J}$ is very small although the initial guess is quite far from the optimal solution (for instance, the radius of the optimal solution is 20% longer than the radius of the initial guess). Note that the L^2-norm of u in the domain B is orders of magnitude larger than $\mathcal{J}$. To assess the finite element error we evalute $\mathcal{J}$ when the initial guess coincide with the optimal solution (that is, when $r_t = 0.5$ and $\mathbf{c}_t = (0,0)^T$). The computed value reads (approximatively) 2e-9.

Next, we investigate the performance of our shape optimization algorithm on the ill-posed test case (3.26). We consider a B-spline discretization that comprises 196 uniform multivariate quadratic B-splines. We choose the target functional u_t so that the optimal solution is either a circle of radius $r_t = 0.5$ centered in $\mathbf{c}_t = (-1.9, 0)^T$ or a circle of radius $r_t = 0.5$ centered in $\mathbf{c}_t =$

$\phi/(2\pi)$	0	1/6	2/6	3/6	4/6	5/6
$\mathcal{J}$	9e-4	4e-3	2e-2	3e-2	2e-2	4e-3

Table 3.1.: Value of misfit functional $\mathcal{J}$ when the parameters of u_t are choosen so that $r_t = 0.6$ and $\mathbf{c}_t = 0.1(\cos\phi, \sin\phi)^T$. Note that $\|u\|_{L^2(B)} \approx 15$ for every ϕ, and that $\mathcal{J} \approx$ 2e-9 when $r_t = 0.5$ and $\mathbf{c}_t = (0,0)^T$.

$(-2.1, 0)^T$ or a circle of radius $r_t = 0.55$ centered in $\mathbf{c}_t = (-1.9, 0.05)^T$. For each case, we perform 10 optimization steps (the optimization step length is chosen so that the Armijo rule condition is satisfied). The results are reported in Figure 3.14. We observe that the right side of the recovered shape is close to the target optimal shape, whilst the left side remains basically unchanged. These results are expected, because we employ a steepest descent optimization algorithm, and the part of the boundary that is closer to the region of interest B is clearly more sensitive. On the other hand, the opposite part of the boundary has a very little impact on the value of the misfit functional. This is evident from the convergence histories plotted in Figure 3.14 (left). This behaviour of the misfit functional is also visible in Table 3.1. The value of $\mathcal{J}$ is the smallest when the right part of the boundary of the optimal solution and of the initial guess are close (that is, when $\phi = 0$).

Next, we investigate the descent direction employed in the first optimisation step for the case $r_t = 0.5$, $\mathbf{c}_t = (-1.9, 0)^T$. In Figure (3.15) (first row), we plot its x-component (left) and its y-component (right). These pictures highlight the sensitivity of different parts of the internal boundary. In the second row, we display the same descent direction when the B-spline discretization comprises 3844 uniform multivariate quadratic B-splines. There is no substantial difference with respect to the descent direction computed with the coarser B-spline resolution. We solely notice that the vector field is more concentrated on the boundary when the discretization is finer. This is an obvious consequence of the Hadamard structure theorem; see Equation (2.5).

In Figure 3.16, we plot the x-component (left) and the y-component (right) of the descent direction evaluated on $\partial\Omega_0^{\text{in}}$ for several B-spline discretizations. Again, we observe that there is no substantial difference in the values of the descent direction. We only see some tiny wiggles on the angle $\pm\pi/4$ when the descent direction is computed with fine B-spline discretizations. Most likely, these wiggles are due to the ill-posedness of the optimization problem

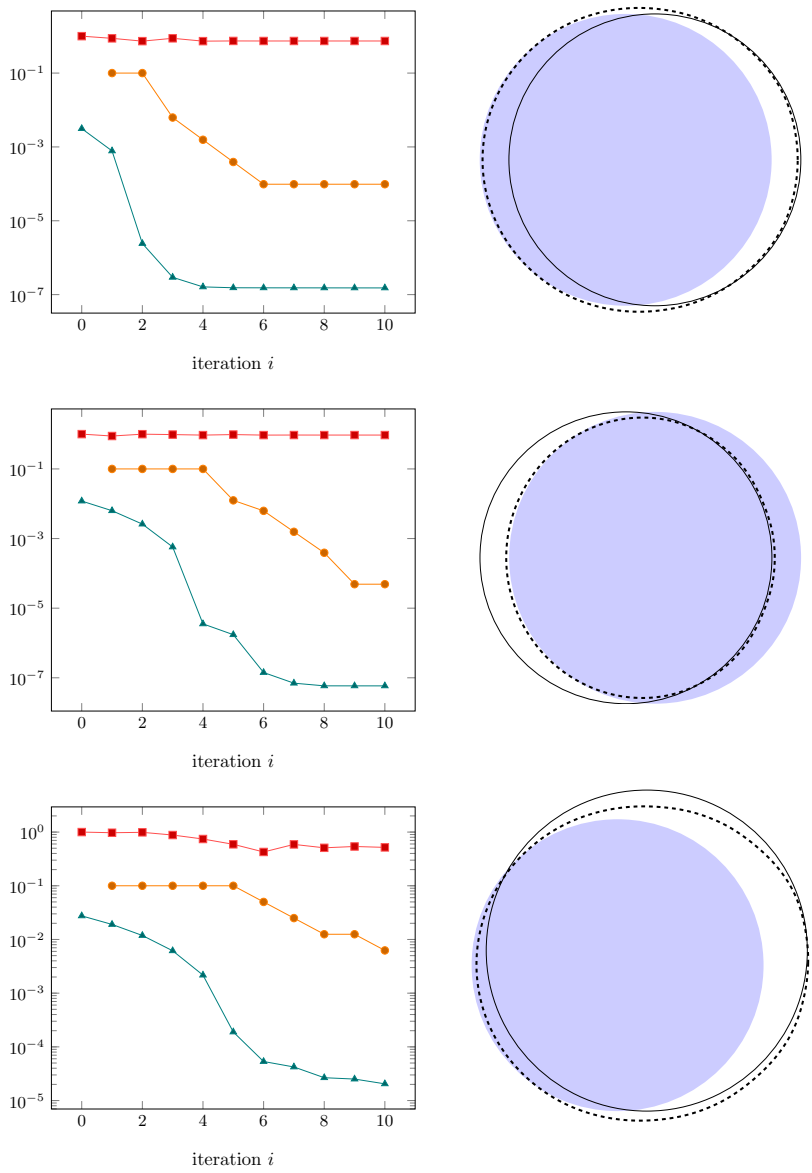

Figure 3.14.: Shape optimization experiment for the ill-posed test case (3.26) when $r_t = 0.5$ and $\mathbf{c}_t = (-1.9, 0)^T$ (*first row*), $r_t = 0.5$ and $\mathbf{c}_t = (-2.1, 0)^T$ (*second row*), $r_t = 0.55$ and $\mathbf{c}_t = (-1.9, 0.05)^T$ (*third row*). *Left:* convergence history (—▲—), optimization steps (—●—), and $\min \det \mathbf{D}\mathcal{V}$ (—■—). *Right:* Initial shape (filled area), target shape (continuous line), and shape recovered after 10 optimization steps (dashed line).

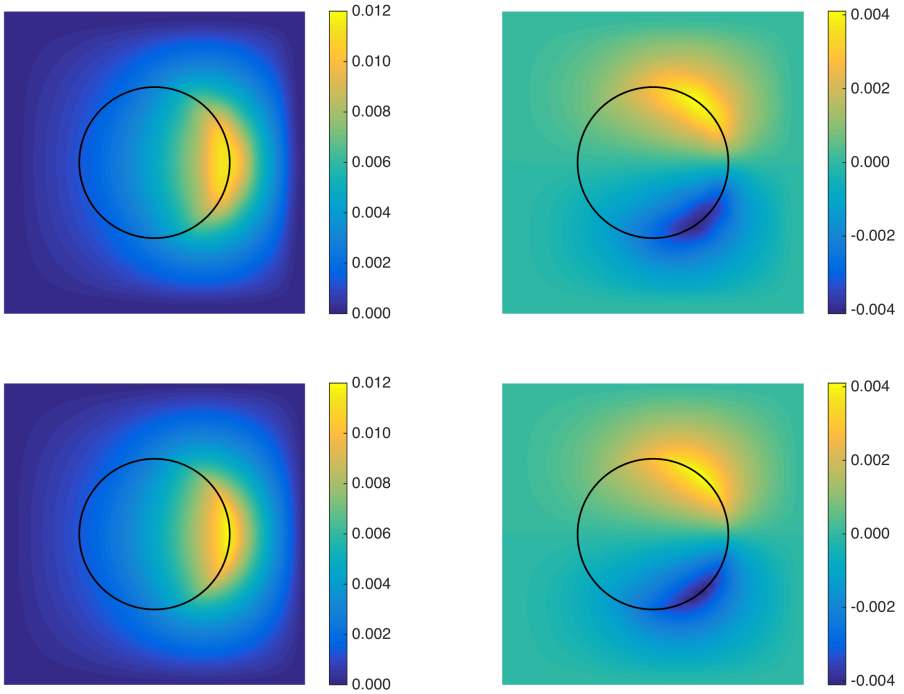

Figure 3.15.: x-component (left) and y-component (right) of the descent direction employed in the first optimisation step for the case $r_t = 0.5$, $\mathbf{c}_t = (-1.9, 0)^T$ for a B-spline discretization that comprises 196 (*first row*) and 3844 (*second row*) uniform multivariate quadratic B-splines.

and to the noise introduced by the finite element approximation error. Their absence in the descent directions computed with fewer B-splines may be due to a regularization by discretization.

The presence of these tiny oscillations in the descent directions clearly affects the optimization algorithm and the resulting recovered shape. It goes without saying that one would like to avoid unnecessary wiggles in optimized shapes. However, it is hard to tell whether a chosen B-spline discretization generates oscillatory descent directions. A possible way to regularize these descent directions is to employ the fast wavelet transform. In a nutshell, the idea is to perform a change of basis in the vector field representation (3.13)

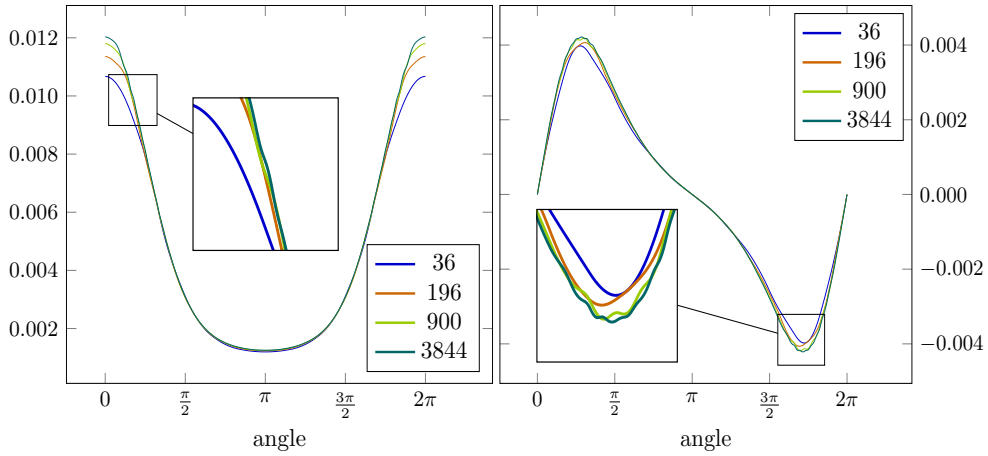

Figure 3.16.: x-component (left) and y-component (right) of the descent direction employed in the first optimisation step for the case $r_t = 0.5$, $\mathbf{c}_t = (-1.9, 0)^T$ evaluated on $\partial\Omega_0^{\text{in}}$ for several B-spline discretizations.

so that the H^1-norm of the vector field is equivalent to the ℓ^2-norm of the coefficients of its new representation, that is,

$$\begin{aligned}\|\mathcal{V}_N\|_{H^1(D;\mathbb{R}^d)} &= \|\sum_{i=1}^{N}\left(\sum_{j=1}^{d} c_i^j \mathbf{e}_j\right) B_i^2(\mathbf{x})\|_{H^1(D;\mathbb{R}^d)}\,,\\ &= \|\sum_{i=1}^{N}\left(\sum_{j=1}^{d} \hat{c}_i^j \mathbf{e}_j\right) \hat{B}_i^2(\mathbf{x})\|_{H^1(D;\mathbb{R}^d)} \sim \|\{\hat{c}_i^j\}\|_{\ell^2(\mathbb{R}^{dN})}\,. \qquad (3.27)\end{aligned}$$

With this representation at hand, small oscillations in the descent directions can be cut by setting to zero the coefficients $\hat{c}_i^j$ that are relatively small. Furthermore, it is possible to employ a discretization of vector fields based on B-spline wavelets from the very beginning. For this discretization, there is a diagonal operator that is an "asymptotically optimal preconditioner" for the stiffness matrix that arises in the computation of descent directions (3.20) [76, Thm 6.1]. Alternatively, the standard H^1 inner product in (3.20) could be replaced by the inner product induced by wavelet basis functions, so that no linear system has to be solved. This strategy is particularly useful to save computational power in 3-dimensional problems.

Performing fast wavelet transform for functions defined on a bounded domain is a rather technical procedure. A neat introduction would require the tedious repetition of a well-established theory. Since we are merely interested in its application, we skip the details and refer an interested reader to the monograph [76]. We only mention that we employ the B-spline wavelets introduced in [67] because for these the constants hidden in the norm equivalence (3.27) are smaller compared to the B-spline wavelets used in [76, Ch. 8].

In Figure 3.17, we plot the x-component (left) and the y-component (right) of the descent direction evaluated on $\partial\Omega_0^{\text{in}}$ for a B-spline discretization that comprises 900 uniform multivariate quadratic B-splines before and after performing a regularization based on fast wavelet transform. In particular, per each component we set to zero all coefficients $\hat{c}_i^j$ whose absolute value is smaller than $\max_i |\hat{c}_i^j|/100$. This regularization sets to zero approximatively 87% of the coefficients $\hat{c}_i^j$, which results in a reduction of wiggles in the descent direction.

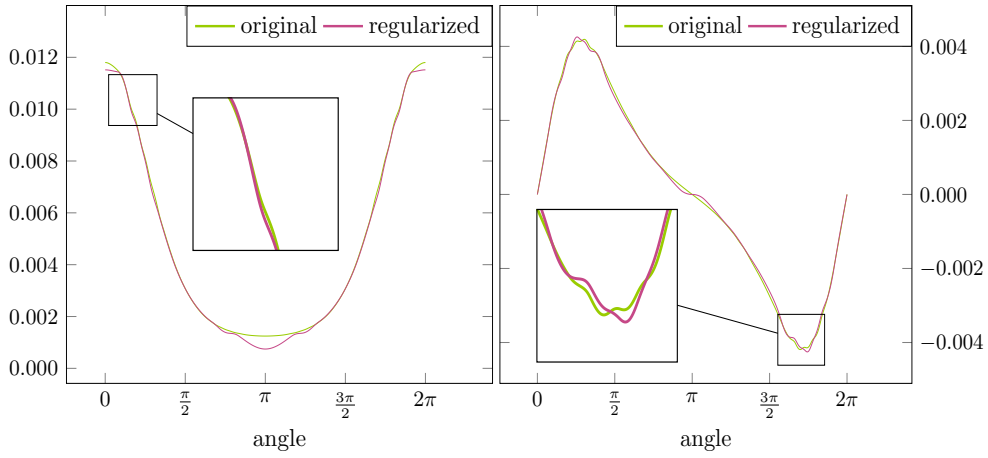

Figure 3.17.: Impact of regularization based on fast wavelet transform on x-component (left) and y-component (right) of the descent direction employed in the first optimisation step for the case $r_t = 0.5$, $\mathbf{c}_t = (-1.9, 0)^T$ evaluated on $\partial\Omega_0^{\text{in}}$ for a B-spline discretization that comprises 900 uniform multivariate quadratic B-splines.

4. Shape optimization of microlenses

In this chapter, we show that shape optimization can be a valuable tool for engineering problems that arise in the field of electromagnetics. In particular, we consider the task of optimizing an optical microlens: a dieletric structure of size comparable to the wavelength λ of the incident field. This chapter is based on [65].

It is well known that the classical diffraction limit of optical lenses restricts the resolution of optical microscopes to approximately half of the wavelength. One way to increase the resolution is to work within media (e.g., oil) with an index of refraction higher than the one of air, i.e., with a wavelength smaller than the one of air. Since lenses always have an index of refraction higher than the one of air, sufficiently thick lenses will have a focus point inside that is smaller than half the wavelength in air. By tuning the index of refraction of a microlens of spherical shape (3D) or of circular-cylinder shape (2D), one may move the focus on or close to the surface of the lens. By doing this, one may overcome the diffraction limit in some sense. Ultramicroscopy based on this concept has been proposed in [18].

When the focus of a microlens is on or near its surface, one may also observe a very much elongated shape of the focus area, which is called nanojet. Nanojets have been shown both by simulations and experiments [49].

The length of a nanojet may be strongly increased, for example, by considering layered spheres [72]. It should be mentioned that these ultra-long nanojets have a waist of more than half wavelength, but they have various promising applications as mentioned in [72].

In literature, the lenses employed to generate such nanojets have simple shapes. Obviously, this simplicity is the main advantage of studying microlenses of spherical or of circular-cylindrical shape: only two parameters (radius and index of refraction of the lens) have to be optimized for an incident plane wave of a certain wavelength. Furthermore, Mie scattering theory may be used for obtaining solutions with high accuracy. However, the restriction to simple shapes is arbitrary, and more complex models with more tuning parameters may be needed for engineering the shape of the nanojet.

Furthermore, it is important to note that appropriate low-loss dielectrics for manufacturing microlenses are only available with certain values of the index of refraction, i.e., this index should not be considered as a parameter that may be optimized.

The standard way to tune the optical features of lenses is shape optimization. An obvious geometry with still very few optimization parameters is a 2D lens of cylindrical shape with elliptical cross section [37] or a 3D lens with the shape of an ellipsoid. In [37], it was also shown that lenses with another topology (e.g., toroidal) may cause nanojets that are located considerably away from the surface.

By increasing the number of parameters describing the microlens, one can increase the search space for finding nanojets with desired shape and one may hope for better solutions than what has been found so far. This leads to very demanding optimization problems because the computation time of conventional numerical optimizers increases drastically with the number of parameters. In this paper, we demonstrate that this problem may be overcome by using the shape optimization algorithm developed in Chapter 3. This algorithm is based on shape gradients, and allows us to start with a lens of arbitrary shape and index of refraction and to deform it until a local optimum is reached. We can handle an unlimited number of shape parameters and obtain various promising structures without high computational cost. However, it goes without saying that best results are achieved with a reasonable initial guess.

In order to demonstrate the procedure, we consider a rather simple 2D test problem, starting from elliptic or semicircular lenses. We optimize their shape in such a way that the field inside a specified rectangular box is maximized. This procedure can easily be applied to lenses of axisymmetric shape and for various optimization goals, for example, strong gradients of the intensity near the nanojet, which is interesting, e.g., for optical trapping. Since we utilize a standard finite element method (FEM), a full 3D particle shape could also be considered.

4.1. Modeling

Let D_{L} be the cross section in the x-y-plane of a simply connected, homogeneous, nonmagnetic, cylindrical lens that is infinitely long in the z-direction

and has a refractive index n_{L}. The lens is surrounded by nonmagnetic material with refractive index $n_{\mathrm{A}} = 1$. It is illuminated by a plane wave

$$\mathbf{E}^{\mathrm{in}}(\mathbf{x}) \coloneqq E_z^{\mathrm{in}}(\mathbf{x})\mathbf{e}_z\,, \quad E_z^{\mathrm{in}}(\mathbf{x}) = C_E \exp(i\mathbf{k}\cdot\mathbf{x})\,, \tag{4.1}$$

where $\mathbf{e}_z$ denotes the unit vector in the z-direction and C_E is a scalar constant. The wave vector $\mathbf{k}$ of the incident field $\mathbf{E}^{\mathrm{in}}$ lies in the x-y-plane; see Figure 4.1.

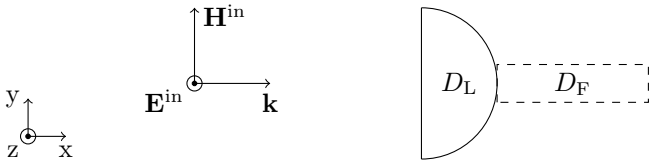

Figure 4.1.: A dielectric lens D_{L} is illuminated by a plane wave $\mathbf{E}^{\mathrm{in}}$. The goal is to find the shape of D_{L} so that the focused light in D_{F} is maximized.

We model the behavior of the electromagnetic fields with the linear time harmonic Maxwell's equations without electric current and electric charge densities [60, Eq. 1.9], which read

$$\operatorname{div}\mathbf{E} = 0\,, \quad \operatorname{div}\mathbf{B} = 0\,, \quad \mathbf{curl}\,\mathbf{E} = i\omega\mathbf{B}\,, \quad \mathbf{curl}\,\mathbf{B} = -i\omega\mu\epsilon\mathbf{E}\,, \tag{4.2}$$

where $\mathbf{B}$ is the magnetic field, $\mathbf{E}$ is the electric field, ω is the frequency, ϵ is the permittivity, and μ is the permeability. We assume that the materials are homogeneous and isotropic. Thus, the permittivity and the permeability are piecewise constant and scalar.

Maxwell's equation can be equivalently written in the decoupled form

$$\begin{cases} \operatorname{div}\mathbf{E} = 0\,, \\ \mathbf{curl}\,\mathbf{curl}\,\mathbf{E} = \mu\epsilon\omega^2\mathbf{E}\,, \end{cases} \qquad \begin{cases} \operatorname{div}\mathbf{B} = 0\,, \\ \mathbf{curl}\,\mathbf{curl}\,\mathbf{B} = \mu\epsilon\omega^2\mathbf{B}\,. \end{cases} \tag{4.3}$$

We consider illumination by a polarized plane wave $\mathbf{E}^{\mathrm{in}}$ (4.1) and restrict our attention to the TM mode, that is,

$$\mathbf{E} = (0, 0, E_z)^T \quad \text{and} \quad \mathbf{B} = (B_x, B_y, 0)^T\,. \tag{4.4}$$

We assume that the cylinder with cross section D_{L} is infinitely long in the z-direction. Additionally, we assume that the incident wave $\mathbf{E}^{\mathrm{in}}$ is translation invariant in the z-direction. Then, the fields $\mathbf{E}$ and $\mathbf{B}$ satisfy

$$\frac{\partial \mathbf{E}}{\partial z} = 0 \quad \text{and} \quad \frac{\partial \mathbf{B}}{\partial z} = 0\,. \tag{4.5}$$

By (4.4) and (4.5), the decoupled Maxwell's equations (4.3) become

$$-\Delta E_z - k^2 E_z = 0\,, \quad -\Delta B_x - k^2 B_x = 0\,, \quad -\Delta B_y - k^2 B_y = 0\,, \tag{4.6}$$

where $k := \omega\sqrt{\mu\epsilon}$. Since we have assumed translation invariance in the z-direction, we can restrict the equations (4.6) to the x-y-plane $\mathbb{R}^2$. Note that the wavenumber k depends on the material properties at $\mathbf{x} \in \mathbb{R}^2$ and satisfies

$$k(\mathbf{x}) := \begin{cases} n_{\mathrm{L}}\,|\mathbf{k}| & \text{in } D_{\mathrm{L}}\,, \\ |\mathbf{k}| & \text{in } \mathbb{R}^2 \setminus D_{\mathrm{L}}\,. \end{cases} \tag{4.7}$$

Since equations (4.6) are decoupled, we can restrict our attention to

$$-\Delta E_z - k^2 E_z = 0\,. \tag{4.8}$$

The components B_x and B_y can be retrieved by computing $\mathbf{curl}\,\mathbf{E}$; see (4.2).

By the general interface conditions in electromagnetism [60, Sect. 1.2.2], the tangential component of the electric field and the normal component of the magnetic field on the boundary of D_{L} are continuous. This implies that the E_z and its normal derivative are continuous across the boundary of D_{L}.

Additionally, the scattered field $E_z^{\mathrm{s}}(\mathbf{x}) := E_z(\mathbf{x}) - E_z^{\mathrm{in}}(\mathbf{x})$ must satisfy the Sommerfeld radiation condition [21, Eq. 1.3]

$$\lim_{|\mathbf{x}|\to\infty} \sqrt{|\mathbf{x}|}\left(\nabla E_z^{\mathrm{s}}(\mathbf{x}) \cdot \frac{\mathbf{x}}{|\mathbf{x}|} - i\,|\mathbf{k}|\,E_z^{\mathrm{s}}(\mathbf{x})\right) = 0\,, \tag{4.9}$$

which is the 2D-equivalent of the Silver Müller radiation condition [21, Thm 6.8]

We want to find the shape of the lens D_{L} so that the focused light in the focus or nanojet area D_{F} is maximized. To model this target property, we introduce the shape functional (energy content in D_{F})

$$\mathcal{J} : D_{\mathrm{L}} \mapsto \mathcal{J}(D_{\mathrm{L}}) := \int_{D_{\mathrm{F}}} |E_z|^2 \mathrm{d}\boldsymbol{x}\,. \tag{4.10}$$

Then, we say that the shape of the lens D_{L} is optimal if it solves the following PDE-constrained shape optimization problem

$$\operatorname*{argmax}_{D_{\mathrm{L}} \in \mathcal{U}_{\mathrm{ad}}(D_{\mathrm{L},0})} \mathcal{J}(D_{\mathrm{L}}) \quad \text{subject to (4.8) and (4.9)} \tag{4.11}$$

To construct the set of admissible shapes $\mathcal{U}_{\mathrm{ad}}(D_{\mathrm{L},0})$, we collect all domains that can be obtained by letting a diffeomorphism (a bijective continuously differentiable map with continuosly differentiable inverse) T act on an reasonable initial guess $D_{\mathrm{L},0}$ for which the PDE-contraint in (4.11) is well-defined. In particular, we consider maps of the form $T_{\mathcal{V}}(\mathbf{x}) \coloneqq \mathbf{x} + \mathcal{V}(\mathbf{x})$, where $\mathcal{V}$ is a vector field on $\mathbb{R}^2$ with compact support contained in a hold-all domain D that encloses the initial guess $D_{\mathrm{L},0}$ and does not intersect the region of interest D_{F}. It is well known that $T_{\mathcal{V}}$ is a diffeomorphism if $\|\mathcal{V}\|_{C^1(\mathbb{R}^2;\mathbb{R}^2)} < 1$ [3, Lemma 6.13]. Note that all admissible shapes share the same topology.

Since $T_{\mathcal{V}}$ is a diffeomorphism, we can employ transformation techniques (change of variables) to replace the dependence of (4.11) on D_{L} with a dependence on $\mathcal{V}$. We obtain the equivalent optimal control problem

$$\operatorname*{argmax}_{\mathcal{V}} \tilde{\mathcal{J}}(\mathcal{V}) \coloneqq \mathcal{J}(T_{\mathcal{V}}(D_{\mathrm{L},0})) \quad \text{subject to} \tag{4.12a}$$

$$-\operatorname{div}(\mathbf{M}_{\mathcal{V}} \operatorname{\mathbf{grad}} E_z) - k^2 E_z \det \mathbf{D}T_{\mathcal{V}} = 0 \quad \text{in } \mathbb{R}^2 \text{ and (4.9)}\,, \tag{4.12b}$$

where the matrix $\mathbf{M}_{\mathcal{V}}$ is given by $\mathbf{M}_{\mathcal{V}} = (\det \mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}$, and $\mathbf{D}T_{\mathcal{V}} = \mathbf{I} + \mathbf{D}\mathcal{V}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{D}\mathcal{V}$ is the Jacobian of $\mathcal{V}$. Reformulating the optimization problem (4.11) in the form (4.12) simplifies its mathematical analysis because the vector field $\mathcal{V}$ can be interpreted as an element of a Banach space (a complete normed vector space). Moreover, the state constrain (4.12b) is formulated on a fixed domain. This greatly simplifies the use of the finite element method to compute approximations of the electric field E_z.

4.2. Method

The optimization problem (4.12) is infinite-dimensional, because the control $\mathcal{V}$ belongs to the Banach space of continuously differentiable vector fields, that is, to a space of functions. By Theorem 3.1.5, we can circumvent the infinite dimensionality issue of the problem by restricting the control $\mathcal{V}$ to a

finite dimensional subspace of vector fields. The optimal solution of the corresponding finite-dimensional optimization problem is a good approximation of the original infinite-dimensional problem provided that the finite-dimensional trial space of vector fields is rich enough.

To discretize the control $\mathcal{V}$, we employ multivariate B-splines of degree 3 [47] generated on a regular grid that covers the hold-all domain D; see Figure 4.2. More precisely, we consider vector fields that can be written as

$$\mathcal{V}_N(\mathbf{x}) = \sum_{i=1}^{N} \begin{pmatrix} c_i^1 \\ c_i^2 \end{pmatrix} \mathrm{B}_i(\mathbf{x}), \quad c_i^1, c_i^2 \in \mathbb{R}, \tag{4.13}$$

where B_i denotes the scalar i-th multivariate B-spline of degree 3. A multivariate B-spline is the tensor product of univariate B-splines, which are piecewise polynomials with compact support. For example, the formula of a univariate cubic B-spline on the knot sequence $(0, 1, 2, 3, 4]$ reads

$$\begin{cases} x^3/6 & 0 < x \leq 1, \\ (-3x^3 + 12x^2 - 12x + 4)/6 & 1 < x \leq 2, \\ (3x^3 - 24x^2 + 60x - 44)/6 & 2 < x \leq 3, \\ (4 - x)^3/6 & 3 < x \leq 4. \end{cases}$$

This combination of features is excellent to access local sensitivity information and allows for an efficient implementation; see Section 3.3. More details on B-splines are given in Section 3.3.

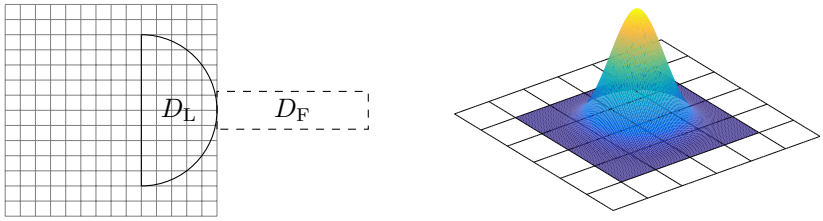

Figure 4.2.: *Left:* Grid used to generate multivariate B-splines of degree 3. *Right:* Multivariate B-spline of degree 3. Its support comprises 4×4 grid cells. The B-spline is polynomial in each cell.

To achieve good results, the number N of multivariate B-splines employed in the discretization (4.13) must be sufficiently big. Additionally, the evaluation of the shape functional $\mathcal{J}$ (4.10) is particularly expensive because it requires good approximations of the electric field E_z, which depends on $\mathcal{V}$. Therefore, we decide to employ deterministic algorithms to solve the optimization problem. More specifically, we rely on steepest descent direction algorithms [3, Ch. 3.4], which are simple optimization algorithms that allow to find local maxima.

Before giving the formula of the Fréchet derivative of the functional $\tilde{J}$, let us remark that the problem (4.12b) can be reformulated in a weak form as [59, Sect. 2]: Find $E_z \in H^1(\tilde{\Omega})$ such that

$$\int_{\tilde{\Omega}} \nabla E_z \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 E_z \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\tilde{\Omega}} \left(\mathrm{DtN}\left[E_z - E_z^{\mathrm{in}}\right] + \frac{\partial E_z^{\mathrm{in}}}{\partial \mathbf{n}} \right) \bar{v} \,\mathrm{d}S\,, \tag{4.14}$$

for all $v \in H^1(\tilde{\Omega})$, where $\mathrm{DtN} : H^{1/2}(\partial\tilde{\Omega}) \to H^{-1/2}(\partial\tilde{\Omega})$ denotes the *Dirichlet-to-Neumann* map [59, Sect. 2] and $\tilde{\Omega}$ is a ball that englobes both the hold-all domain D and the region of interest D_{F}.

Proposition 4.2.1. *The derivative in the direction $\mathcal{W}$ of the target functional $\tilde{\mathcal{J}}$ evaluated in $\mathcal{V}$ reads*

$$d\tilde{\mathcal{J}}(\mathcal{V}, \mathcal{W}) \coloneqq \mathrm{Re}\left\{ \int_D \nabla E_z \partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}} \nabla \bar{p} - k^2 E_z \bar{p}\, \partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \,\mathrm{d}\boldsymbol{x} \right\}, \tag{4.15}$$

where

$$\begin{aligned} \partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}} \coloneqq \det(\mathbf{D}T_{\mathcal{V}}) \Big(& \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}\mathcal{W}) \mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}T_{\mathcal{V}}^{-T} \\ & - \mathbf{D}T_{\mathcal{V}}^{-1} (\mathbf{D}T_{\mathcal{V}}^{-T} \mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W} \mathbf{D}T_{\mathcal{V}}^{-1}) \mathbf{D}T_{\mathcal{V}}^{-T} \Big)\,, \\ \partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \coloneqq \det(\mathbf{D}T_{\mathcal{V}}) & \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}\mathcal{W})\,. \end{aligned}$$

The scalar function p is the solution of the adjoint problem

$$\int_{\tilde{\Omega}} 2v \bar{E}_z \chi_{D_{\mathrm{F}}} + \nabla v \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{p} - k^2 v \bar{p} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\tilde{\Omega}} v \mathrm{DtN}^*[\bar{p}] \,\mathrm{d}S\,, \tag{4.16}$$

where $\chi_{D_{\mathrm{F}}}$ denotes the characteristic function of D_{F} and DtN^* *is the adjoint of* DtN *with respect to the bilinear form*

$$H^{1/2}(\partial\tilde{\Omega}) \times H^{-1/2}(\partial\tilde{\Omega}) \ni (v, w) \mapsto \int_{\partial\tilde{\Omega}} vw \,\mathrm{d}S\,.$$

Proof. We follow closely the steps in the proof of Proposition 2.1.2. First of all, note that E_z is the solution of (4.14) if and only if

$$\operatorname{Re}\left\{\int_{\tilde{\Omega}} \nabla E_z \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 E_z \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x}\right\} = \operatorname{Re}\left\{\int_{\partial\tilde{\Omega}} \left(\mathrm{DtN}\left[E_z - E_z^{\mathrm{in}}\right] + \frac{\partial E_z^{\mathrm{in}}}{\partial \mathbf{n}}\right) \bar{v} \,\mathrm{d}S\right\}, \quad (4.17)$$

for all $v \in H^1(\tilde{\Omega})$ (because v is allowed to attain values in $\mathbb{C}$).

We introduce the Lagrangian

$$\mathscr{L}(\mathcal{V}, E_z, v) \coloneqq \int_{D_{\mathrm{F}}} |E_z|^2 \,\mathrm{d}\boldsymbol{x} + \operatorname{Re}\left\{\int_{\tilde{\Omega}} \nabla E_z \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 E_z \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x}\right\} - \operatorname{Re}\left\{\int_{\partial\tilde{\Omega}} \left(\mathrm{DtN}\left[E_z - E_z^{\mathrm{in}}\right] + \frac{\partial E_z^{\mathrm{in}}}{\partial \mathbf{n}}\right) \bar{v} \,\mathrm{d}S\right\}. \quad (4.18)$$

To pursue the Lagrangian approach it is necessary to consider (4.17) instead of (4.14) because the Lagrangian functional $\mathscr{L}$ must be real valued.

By differentianting $\mathscr{L}$ with respect to the Lagrange multiplier v, we recover the constraint (4.17). The adjoint equation (4.16) is obtained by differentiating $\mathscr{L}$ with respect to E_z. Finally, Formula (4.15) can be obtained by differentiating $\mathscr{L}$ with respect to $\mathcal{V}$ and employing Lemma 3.4.1. Note that the variable $\mathcal{V}$ is compactly supported in the hold-all domain D and that D does not intersect neither D_{F} nor $\partial\tilde{\Omega}$. □

Optimization is carried out iteratively. First, we compute a descent direction $\mathcal{V}_N^{\mathrm{update}}$ as the solution of the linear variational problem

$$(\mathcal{V}_N^{\mathrm{update}}, \mathcal{W}_N)_{H^1} = d\tilde{\mathcal{J}}(\mathcal{V}_N, \mathcal{W}_N) \quad \forall\, \mathcal{W}_N \text{ as in (4.13)}, \quad (4.19)$$

where $(\cdot,\cdot)_{H^1}$ denotes the usual H^1-inner product [15, Ch. II, Sect. 1]. Then, we choose an optimization step length δ according to [63, Sect. 3.5]. Finally, we add the update $\delta\mathcal{V}_N^{\mathrm{update}}$ to the map $T_{\mathcal{V}_N} = \mathrm{I} + \mathcal{V}_N$. A pseudo-code of the optimization algorithm is listed in Section 3.4.

Clearly, to evaluate the shape gradient (4.15), it is necessary to replace the electric field E_z and the adjoint p with numerical approximations. We employ linear Lagrangian finite elements on quasi-uniform triangular meshes [15]. Firstly, we introduce a bounded domain Ω that encloses the hold-all domain

D and the region of interest D_{F}. Then we state the variational formulation of (4.12b) on Ω [15, Ch. 2, Sect. 2]. That is, E_z must satisfy

$$\int_{\Omega} \nabla E_z \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 E_z \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \frac{\partial E_z}{\partial \mathbf{n}} \bar{v} \,\mathrm{d}S\,, \quad \text{for all } v \in H^1(\Omega)\,, \tag{4.20}$$

where $H^1(\Omega)$ denotes the Sobolev space of square integrable functions with square integrable weak derivatives [15, Ch. 2, Def. 1.2]. Note that $\mathbf{M}_{\mathcal{V}}$ is equal to the identity matrix outside the hold-all domain D. Finally, we replace the right hand side of (4.20) by realizing the radiation condition (4.9) via a perfectly matched layer (PML) [12]. To do this, we sourround Ω with a layer Ω^{PML}, whose coordinates are linearly stretched in the complex plane; see Figure 4.3.

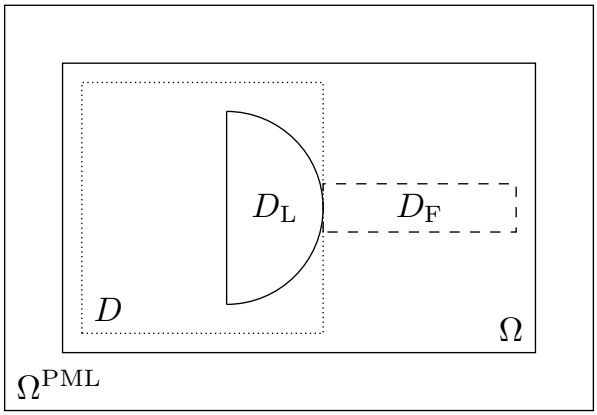

Figure 4.3.: The domain Ω is sourrounded by an absorbing layer Ω^{PML} that encloses the hold-all domain D and the region of interest D_{F}.

The radiation condition (4.9) can be mimicked by stating that the scattered field E_z^{s} must satisfy [60, Ch.13.5.3]

$$-\Delta E_z^{\mathrm{s}} - k^2 E_z^{\mathrm{s}} = 0 \quad \text{in } \Omega^{\mathrm{PML}}\,. \tag{4.21}$$

Furthermore, we impose $E_z^{\mathrm{s}} = 0$ on the exterior boundary of Ω^{PML}. Then, the weak form of (4.21) reads

$$\int_{\Omega^{\mathrm{PML}}} \nabla E_z^{\mathrm{s}} \cdot \nabla \bar{v} - k^2 E_z^{\mathrm{s}} \bar{v} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \frac{\partial E_z^{\mathrm{s}}}{\partial \mathbf{n}^{\mathrm{PML}}} \bar{v} \,\mathrm{d}S\,, \tag{4.22}$$

for all $v \in H^1(\Omega^{\mathrm{PML}})$ that vanish on the exterior boundary of Ω^{PML}. Note that the differential operator ∇ in (4.22) is with respect to complex coordinates and that $\mathbf{n}^{\mathrm{PML}}$ points outward from Ω^{PML}.

To couple (4.20) and (4.22), we first introduce the variable u defined as

$$u(\mathbf{x}) \coloneqq \begin{cases} E_z(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega\,, \\ E_z^{\mathrm{s}}(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega^{\mathrm{PML}}\,. \end{cases}$$

By (4.20), (4.22), and the general interface conditions in electromagnetism, this function satisfies

$$\int_{\Omega \cup \Omega^{\mathrm{PML}}} \nabla u \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 u \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \frac{\partial E_z^{\mathrm{in}}}{\partial \mathbf{n}} \bar{v} \,\mathrm{d}S\,,$$

for all $v \in H_0^1(\Omega \cup \Omega^{\mathrm{PML}})$, that is, for all $v \in H^1(\Omega \cup \Omega^{\mathrm{PML}})$ that vanish on the exterior boundary of Ω^{PML}. Unfortunately, this function u does not belong to $H_0^1(\Omega \cup \Omega^{\mathrm{PML}})$ because it has a jump on $\partial\Omega$. To overcome this issue, we introduce a cut-off function u_{off} that satisfies

$$u_{\mathrm{off}} \in H^1(\Omega)\,, \quad u_{\mathrm{off}} = E_z^{\mathrm{in}} \quad \text{on } \partial\Omega\,, \quad u_{\mathrm{off}} = 0 \quad \text{in } \Omega^{\mathrm{PML}}\,.$$

Additionally, it is convenient to construct u_{off} so that $u_{\mathrm{off}} = 0$ in $D \cup D_{\mathrm{F}}$. Such a function can be constructed easily in the finite element space. Then, the function $\tilde{u} \coloneqq u - u_{\mathrm{off}}$ is in $H_0^1(\Omega \cup \Omega^{\mathrm{PML}})$ and satisfies

$$\int_{\Omega \cup \Omega^{\mathrm{PML}}} \nabla \tilde{u} \cdot \mathbf{M}_{\mathcal{V}} \nabla \bar{v} - k^2 \tilde{u} \bar{v} \det \mathbf{D}T_{\mathcal{V}} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \frac{\partial E_z^{\mathrm{in}}}{\partial \mathbf{n}} \bar{v} \,\mathrm{d}S - \int_{\Omega} \nabla u_{\mathrm{off}} \cdot \nabla \bar{v} - k^2 u_{\mathrm{off}} \bar{v} \,\mathrm{d}\boldsymbol{x}\,,$$

for all $v \in H_0^1(\Omega \cup \Omega^{\mathrm{PML}})$. Finally, note that $\tilde{u} = E_z$ in D and in D_{F} by construction. Thus, we can use $\tilde{u}$ to evaluate the shape gradient (4.15) and the target functional (4.10).

The scalar function p can be computed in an analogous way, but in this case there is no need to introduce a cut-off function.

Note that, employing the formulation (4.12), it is sufficient to construct an initial grid on $\Omega \cup \Omega^{\mathrm{PML}}$ to simulate any shape in $\mathcal{U}_{\mathrm{ad}}(D_{\mathrm{L},0})$, avoiding the need to generate a new FEM mesh for every design.

Replacing E_z and p with numerical approximations in (4.15) inevitably introduces an approximation error in the computation of the steepest descent direction. However, in light of Theorem 2.2.4, we can expect that this approximation error decreases quadratically with respect to the meshwidth h, and does not severely affect the performance of the steepest descent direction algorithm.

An optimization step comprises the computation of the finite element solution of (4.12b) and (4.16), and of the descent direction in (4.19). To compute E_z^h and p^h one needs to assemble a (sparse) stiffness matrix (the same can be used for both) and to solve the related linear system. The latter task can be performed efficiently exploiting sparsity, whilst the computational cost of the matrix assembly is directly proportional to the number of triangles of the mesh, and is only slightly larger than the assembly of the stiffness matrix related to the PDE constraint in (4.11). To compute the descent direction $\mathcal{V}_N^{\text{update}}$ it is necessary to evaluate $d\tilde{\mathcal{J}}(\mathcal{V},\cdot)$ (4.15) for all basis vector fields $\mathrm{B}_i(\mathbf{x})\mathbf{e}_x$ and $\mathrm{B}_i(\mathbf{x})\mathbf{e}_y$. Its computational cost is directly proportional to the number of triangles of the mesh because B-splines have compact support. A detailed explanation of an efficient implementation of the algorithm in MATLAB is provided in Section 3.5.

Without Formula (4.15) at hand it would be virtually impossible to employ a steepest descent direction algorithm. Approximations of the gradient $d\tilde{\mathcal{J}}$ for $2N$ design parameters (see (4.13)) with finite differences would require at least $2N+1$ evaluations of the functional $\tilde{\mathcal{J}}$ (and thus to solve (4.12b) $2N+1$ times), whereas Formula (4.15) requires only the solution of (4.16). Note that $2N+1$ evaluations of $\tilde{\mathcal{J}}$ only lead to a first order approximation of the gradient, which would be considerably less accurate than the proposed method.

4.3. Numerical experiments

We first start with an initial guess $D_{\mathrm{L},0}$ that is an ellipse with semi-minor and semi-major axes of length 4λ and 5λ, respectively. We illuminate $D_{\mathrm{L},0}$ from the left as shown in Fig. 4.1 and locate the region of interest D_{F}, a rectangle of sides $\lambda/2$ and 2λ, on the backside of $D_{\mathrm{L},0}$. The boundary of the grid used to generate B-splines is a square of sidelength 11.2λ as in Fig. 4.4. The mesh employed for finite element computations has 211313 nodes, 421184 triangles,

and width $h = 0.083\lambda$.

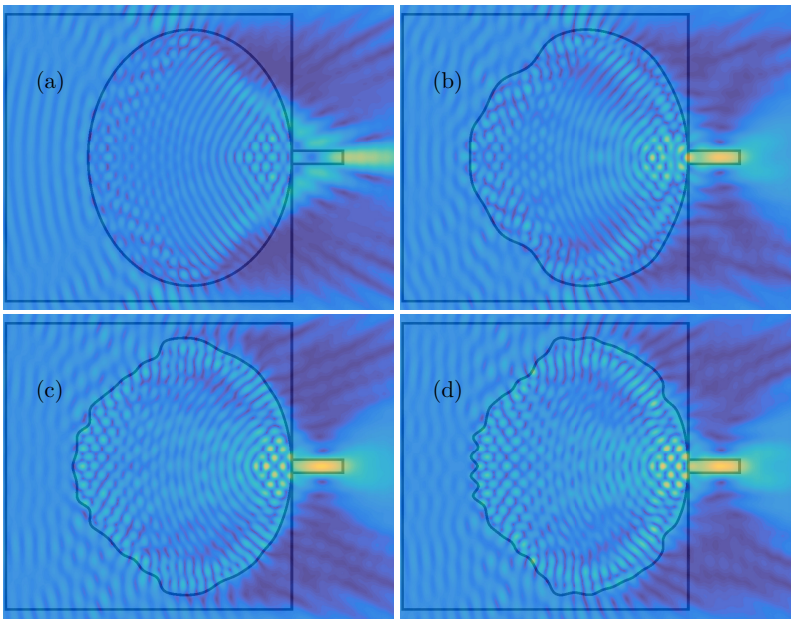

Figure 4.4.: *First numerical experiment*: Absolute value of E_z before (a) and after optimization with 49 (b), 289 (c), and 729 (d) multivariate B-splines. The optimized shapes are thicker in order to shift the focus closer to the lens surface.

In the *first numerical experiment* we choose the refractive index $n = 1.5$. Before optimization, the focus lies beyond the region of interest; see Fig. 4.4(a). We optimize the shape employing $7^2 = 49$ (b), $17^2 = 289$ (c), and $27^2 = 729$ (d) multivariate B-splines. The target functional $\tilde{\mathcal{J}}$ increases to 315%, 355%, and 359% of the initial value, respectively. Essentially, the optimized shapes are thicker than the original ellipse. This obviously shifts the focus closer to the lens surface. The three retrieved shapes differ slightly, which is a consequence of the optimization problem being ill-posed [26]. In particular, we observe that the front of the lens evolves into a more or less sinusoidal line, when sufficiently many B-splines are used.

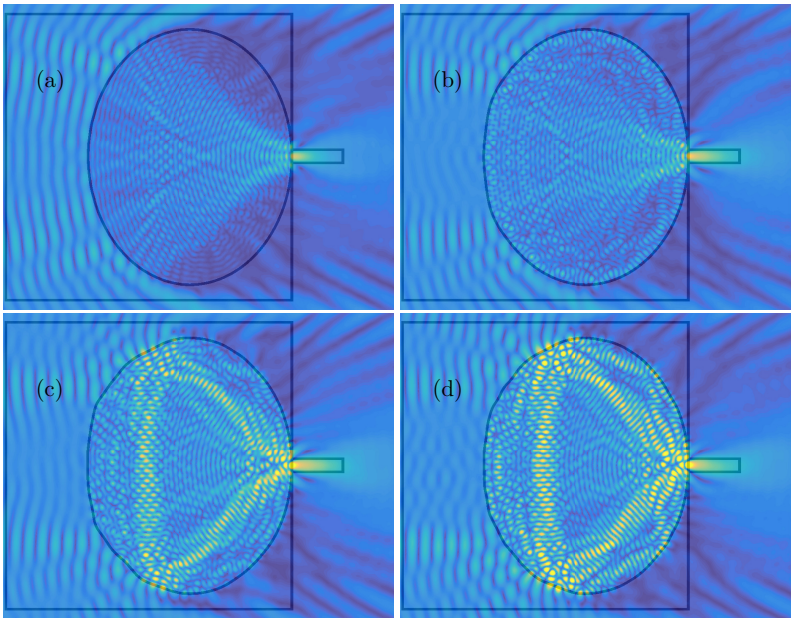

Figure 4.5.: *Second numerical experiment*: Absolute value of E_z before (a) and after (b,c,d) optimization. We observe a high sensitivity of the field distribution.

In the *second numerical experiment* we choose the refractive index $n = 2.7$. In this case, the initial shape is already almost optimal; see Fig. 4.5(a). As in the *first numerical experiment*, we optimize the shape employing 49 (b), 289 (c), and 729 (d) multivariate B-spline. Although the difference between the optimized and the initial shapes are hardly visible, we observe a high sensitivity of the field distribution. The target functional $\tilde{\mathcal{J}}$ increases to 162%, 203%, and 186% of the initial value, respectively. The improvement employing 789 B-splines is slightly less than the one for 289 B-splines. Again, this is a consequence of the ill-posedness of the optimization problem: there are many local minima close to the optimum, and steepest descent methods are not global minimizers [63].

In the *third numerical experiment* we choose the refractive index $n = 3.5$.

Before optimization, the focus lies inside the lens; see Fig. 4.6(a). As in the *first numerical experiment*, we optimize the shape employing 49 (b), 289 (c), and 729 (d) multivariate B-spline. This time, optimized shapes are thinner in order to shift the focus outside the lens. In this case, the high sensitivity of the field distribution with respect to the shape boundary influences the improvements of the target functional $\tilde{\mathcal{J}}$, which increases to 451%, 461%, and 570% of the initial value, respectively.

Finally, we perform a *fourth numerical experiment* to show that we can easily deal with nonsmooth shapes, too. The initial guess $D_{\mathrm{L},0}$ is a half-circle of radius 2λ with refractive index $n = 2$; see Fig. 4.6(e). The boundary of the grid used to generate B-splines is a square of side 5λ as in Fig. 4.6(e). The mesh employed for finite element computations has 211937 nodes, 422432 triangles, and width $h = 0.0817\lambda$. The optimized shape obtained employing 729 B-splines is displayed in Fig. 4.6(f). The target functional $\tilde{\mathcal{J}}$ increases to 217% of the initial value. Additionally, we observe that the retrieved shape reflects less. We repeat the experiment for $n = 1.5$ and $n = 3$. The optimized shapes are displayed in Fig. 4.6 ((g),(h), respectively). The target functional $\tilde{\mathcal{J}}$ increases to 183% and 482% of the initial value, respectively. For $n = 1.5$ the thickness of the lens in the region in front of D_{F} is increased, whilst for $n = 3$ optimality is achieved by corrugating the surface so that transmission increases (see Fig. 4.6(i)), i. e., the optimization runs towards a mixture with a Fresnel-type lens.

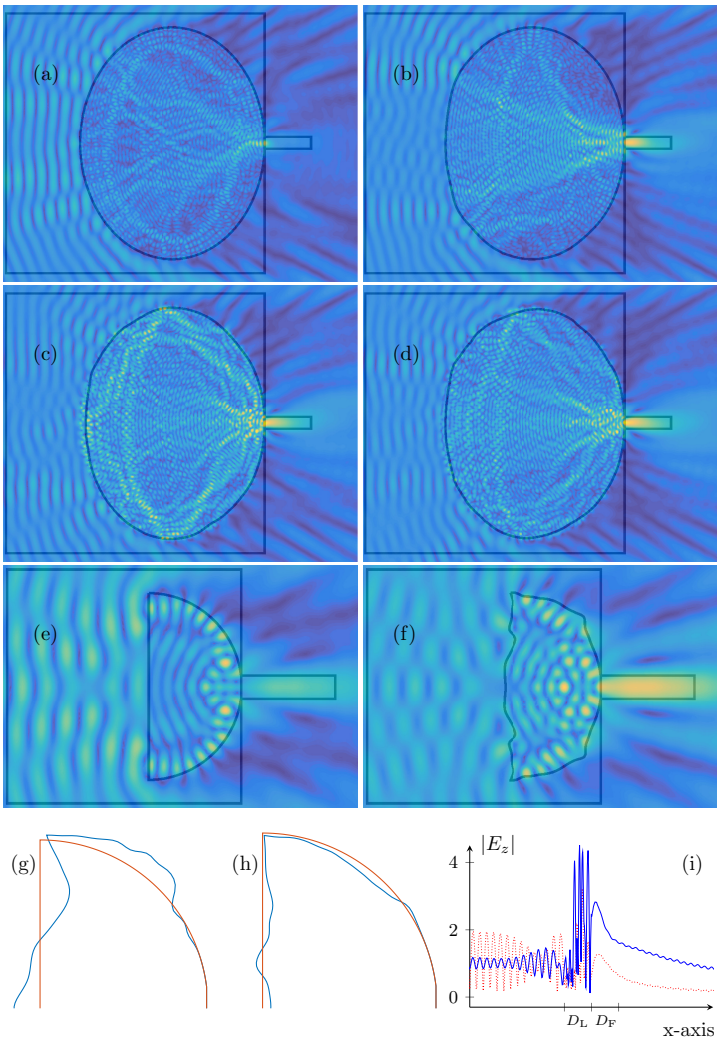

Figure 4.6.: *Third numerical experiment*: Absolute value of E_z before (a) and after (b,c,d) optimization. The optimized shapes are thinner in order to shift the focus outside the lens. *Fourth numerical experiment:*Absolute value of E_z before (e) and after (f) optimization for $n = 2$, and upper half of optimized and initial lens for $n = 1.5$ (g) and $n = 3$ (h). For the latter case, we plot $|E_z|$ along the x-axis before (red dotted line) and after (blue line) optimization (i). We observe that the optimum is achieved by drastically increasing transmission.

5. Conclusion and Outlook

We focused on PDE-constrained shape optimization and investigated several numerical aspects when the solution of the PDE constraint is approximated with the finite element method.

In Chapter 2, we studied the impact of finite element approximations on the approximation of shape gradients that are constrained to elliptic boundary value problems. For this framework, several authors suggested that the volume-based formulation of shape gradients is more accurate; cf. [13], [24, Ch. 10, Rmk. 2.3], and [40, Ch. 3.3.7]. We provided theoretical and numerical evidence that confirm this intuition. In particular, we showed that shape gradients can be approximated replacing the state and the adjoint variable with (linear Lagrangian) finite element solutions. The error introduced decays algebraically with rate 2 with respect to the meshwidth, when the volume formulation of the shape gradient is considered.

In Chapter 3, we developed an algorithm to perform shape optimization. We employed transformation techniques to recast the shape optimization problem as an optimal control problem. In contrast to [50] and [31], we did not rely on an explicit parametrization of the domain boundary. To tackle the resulting infinite-dimensional optimization problem, we applied a conforming Ritz discretization of the control variable and presented a steepest descent algorithm based on shape gradients. We assessed the performance of this algorithm in several numerical experiments. In Appendix A, we provided a complete ready-to-run MATLAB implementation.

In Chapter 4, we applied the shape optimization algorithm of Chapter 3 to engineer the shape of microlenses and maximize the energy in a rectangular output box behind the lens. We considered numerical models of microlenses with different indices of refraction and started from elliptical and semi-circular shapes. The results in Section 4.3 demonstrated the good performance of the shape optimization algorithm. Interestingly, the reflection of the microlenses was also reduced although this goal was not explicitly formulated. The reason for this is that increasing reflection will decrease the light intensity in the output box.

We conclude this work with a list of topics that might worth being further investigated.

- In most mesh-moving methods, the H^1-representative of shape gradients is computed with the same trial space employed for finite element discretizations. In the particular case of linear Lagrangian finite elements, Berggren [13] showed that the boundary and the volume formulation of shape gradients are (in general) not equivalent. However, the difference introduced by the discretization of the vector field $\mathcal{V}$ may converge to zero as the finite element mesh is refined. It could be interesting to quantify this error and to test wether the volume formulation of shape gradients is superior to its boundary counterpart. This result may be useful to investigate mesh-independency of the descent direction employed in steepest descent optimization algorithms for shape optimization.

- In Section 3.2 we showed how to reformulate PDE constrained shape optimization problems as problems of optimal control. If the control variable $\mathcal{V}$ is discretized with linear Lagrangian finite elements, the algorithm presented in Section 3.4 can be interpreted as a mesh-moving method. However, it differs from standard mesh-moving methods because it computes steepest descent directions with a formula that is similar but not equivalent to the shape gradient (see Remark 3.4.2). A comparison between this "new" mesh-moving method with standard ones would be desirable.

- At the end of Section 3.7, we sketched how to exploit the wavelet transform to regularize descent directions. This idea should be further investigated to extend insight into the "nature" of shape gradients.

- In Section 4, we applied the shape optimization algorithm from Section 3 to optimize the shape of microlenses. We considered only one shape functional, but the procedure can be repeated for other optimization goals. For instance, we could aim at maximum intensity gradients in the output box (which would be interesting for optical tweezers), or other shapes of the output box, or more than one output box, etc. For a proof of concept, only 2D lenses were considered, but the procedure can easily be extended to more realistic axisymmetric lenses or even, with some bigger numerical effort, to full three-dimensional lenses. Finally, let us remark that the algorithm can be readily used for various other

applications among which, for instance, plasmonic antennas and optical gratings.

A. Complete Matlab code of a shape optimization algorithm

We consider the shape optimization test case

$$\underset{\Omega\in\mathcal{U}_{\mathrm{ad}}(\Omega_0)}{\operatorname{argmin}}\ \mathcal{J}(\Omega,u) \coloneqq \int_\Omega u^2\,\mathrm{d}\boldsymbol{x} \quad \text{s.t.} \quad -\Delta u = f \text{ in } \Omega\,, \quad u = 0 \text{ on } \partial\Omega\,. \tag{A.1}$$

In parametric form, the test case (A.1) reads (see Example 3.2.1)

$$\underset{\mathcal{V}\in B_{1-\varepsilon}}{\operatorname{argmin}}\,\tilde{\mathcal{J}}(\mathcal{V},u) \coloneqq \int_{\Omega_0} u^2 \det \mathbf{D}T_{\mathcal{V}}\,\mathrm{d}\boldsymbol{x} \quad \text{subject to} \tag{A.2}$$

$$\begin{cases} -\operatorname{div}(\mathbf{M}_{\mathcal{V}}\,\mathbf{grad}\,u) &= (\det \mathbf{D}T_{\mathcal{V}})T_{\mathcal{V}}^*(f) \quad \text{in } \Omega_0\,, \\ u &= 0 \quad \text{on } \partial\Omega_0\,, \end{cases} \tag{A.3}$$

where $T_{\mathcal{V}}^*(f) \coloneqq f\circ T_{\mathcal{V}}$ and $\mathbf{M}_{\mathcal{V}} \coloneqq (\det \mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}$.

The Fréchet derivative of $\tilde{\mathcal{J}}$ from (A.2) reads (see Example 3.4.3)

$$\begin{aligned} d\tilde{\mathcal{J}}(\mathcal{V},u;\mathcal{W}) = \int_{\Omega_0} &(u^2 - fp)\partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \\ &- \mathbf{grad}\,f\cdot\mathcal{W}p\det \mathbf{D}T_{\mathcal{V}} + \mathbf{grad}\,p\cdot\partial_{\mathcal{W}}\mathbf{M}_{\mathcal{V}}\,\mathbf{grad}\,u\,\mathrm{d}\boldsymbol{x}\,, \end{aligned} \tag{A.4}$$

where

$$\begin{aligned} \partial_{\mathcal{W}}\mathbf{M}_{\mathcal{V}} \coloneqq \det(\mathbf{D}T_{\mathcal{V}})\Big(&\operatorname{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} \\ &-\mathbf{D}T_{\mathcal{V}}^{-1}(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}T_{\mathcal{V}}^{-T}\Big)\,, \end{aligned} \tag{A.5}$$

$$\partial_{\mathcal{W}}(\det \mathbf{D}T_{\mathcal{V}}) \coloneqq \det(\mathbf{D}T_{\mathcal{V}})\operatorname{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\,, \tag{A.6}$$

and where $p\in H_0^1(\Omega_0)$ is the solution to the adjoint problem

$$\begin{cases} -\operatorname{div}(\mathbf{M}_{\mathcal{V}}\,\mathbf{grad}\,p) &= -2u(\det \mathbf{D}T_{\mathcal{V}}) \quad \text{in } \Omega_0\,, \\ p &= 0 \quad \text{on } \partial\Omega_0\,. \end{cases} \tag{A.7}$$

The goal of this Chapter is to provide a complete MATLAB-code that

1. Starts with the initial configuration $\mathcal{V} = 0$,
2. Approximates the state variable u,
3. Evaluates the misfit functional $\mathcal{J}$,
4. Approximates the adjoint variable p,
5. Updates the control $\mathcal{V}$ employing $d\tilde{\mathcal{J}}$,
6. Approximates the state variable u in the new configuration,
7. Evaluates the misfit functional $\mathcal{J}$ in the new configuration.

The routines that allow to perform these computations are the basic tools for implementing optimization algorithms based on steepest descent updates. The following MATLAB-script calls these routines, which are further extensively documented in the following Sections.

```
clear all
%create Mesh for FEM computations
Mesh = createMesh;
QR = P3O3; %Gauss Quadrature Rule of order 3 (2D)

%initialize vector field info
SI = initializeSplineInfo;

%right-hand side and Dirichlet boundary conditions
SourceFct = @(x) ones(size(x,1),1);
DirFct = @(x) zeros(size(x,1),1);
%Note: SourceFct and DirFct must be constant for
%current implementation of dJ

%precompute affine transf. and multivar. B-Splines
[precompAffTrans,xQ_Transf] = ...
    precomputeAffineTransformation(Mesh,QR);
precompDV = precomputeSplineJacobian(SI,xQ_Transf);

%compute state solution
[U,A,DTa,DTb,DTc,DTd,detDT,FD] = ...
    computeState(Mesh,SI,precompDV,precompAffTrans,...
    QR,SourceFct,DirFct,xQ_Transf);

%evaluate misfit functional J
```

```
fprintf('J = %1.4f\n',evalJ(U,Mesh,QR,precompAffTrans,detDT))

%compute adjoint solution
P = computeAdjoint(Mesh,A,detDT,precompAffTrans,QR,U,FD);

%evaluate shape gradient
dJ = EulerianDerivative(Mesh,SI,QR,U,P,precompAffTrans,...
    precompDV,DTa,DTb,DTc,DTd,detDT,xQ_Transf,SourceFct);

%H1 representative
V_new = SteepestDescent(SI,dJ);

%update control
delta = 1; %optimization step length
SI.T_update.X = SI.T_update.X + delta*V_new.X;
SI.T_update.Y = SI.T_update.Y + delta*V_new.Y;

%test that the new vale of the control is feasible
assert(computeMinDet(SI.T_update,precompDV) >= 0.05)

%evaluate misfit functional J on updated control
[U,A,DTa,DTb,DTc,DTd,detDT,FD] = computeState(Mesh,SI,...
    precompDV,precompAffTrans,QR,SourceFct,DirFct,xQ_Transf);
fprintf('J = %1.4f\n',evalJ(U,Mesh,QR,precompAffTrans,detDT))
```

A.1. The function `createMesh`

The function **`createMesh`** generates a triangular mesh on the domain Ω_0, which, in this example, is a circle of radius 1 centered in the origin. This function is based on the MATLAB-function **`initmesh`** from the *Partial Differential Equation Toolbox*.

The function **`createMesh`** returns a structure array **`Mesh`**, which contains the fields **`Coordinates`**, **`Elements`** and **`BdNodes`**. The field **`Coordinates`** contains the coordinates of the nodes of the mesh. It is a matrix of dimension $n_p \times 2$, where n_p is the number of nodes in the mesh. The field **`Elements`** indicates the indexes (with respect to the field **`Coordinates`**) of the triangle vertices. It is a matrix of dimension $n_t \times 3$, where n_t is the number of triangles in the mesh. The field **`BdNodes`** is an array that contains the indexes (with respect to the field **`Coordinates`**) of the mesh nodes that lie on the boundary $\partial\Omega_0$.

```
function Mesh = createMesh
%create a Mesh using the Matlab command initmesh.
%the geometry is a disc of radius 1

R = 1; %radius

%Decomposed geometry matrix dl
dl(:,1)=[1 -R  R 0 0 1 0 0 0 R]';
dl(:,2)=[1  R -R 0 0 1 0 0 0 R]';

%generate the mesh
[p,e,t]=initmesh(dl);

%Create Mesh.Coordinates
Mesh.Coordinates= p';

%Create Mesh.Elements, Mesh.myEdges
Mesh.Elements = t(1:3,:)';
isBdEges = e(5,:)<=4;
BdEges = e(1:2,isBdEges)';
Mesh.BdNodes = unique(BdEges);

end
```

A.2. The function P3O3

The function `P3O3` belongs to the MATLAB-library `LehrFEM` developed at ETH Zurich. It returns a struct `QuadRule` that contains the weights (in the field `w`) and the coordinates (in the field `x`) of a 3 point Gauss quadrature in the reference triangle (0,0)-(1,0)-(0,1).

```
function QuadRule = P3O3()
% P3O3 2D Quadrature rule.
%
%   QUADRULE = P3O3() computes a 3 point Gauss quadrature rule
%   of order 3 (exact for all polynomials up to degree 2) on
%   the reference element.
%
%   QUADRULE is a struct containing the following fields:
%    w Weights of the quadrature rule
%    x Abscissae of the quadrature rule
```

```
%
%   Example:
%
%   QuadRule = P3O3();
%

%   Copyright 2005-2005 Patrick Meury
%   SAM - Seminar for Applied Mathematics
%   ETH-Zentrum
%   CH-8092 Zurich, Switzerland

  QuadRule.w = 1/6*ones(3,1);
  QuadRule.x = [  0 1/2; ...
                1/2 1/2;
                1/2   0];

return
```

A.3. The function `initializeSplineInfo`

The function `initializeSplineInfo` returns the struct `SI`, which contains the fields `spOrder`, `nBsplines`,`BSknots_x`, `BSknots_y`, `SplinesSupp`, `H`, `T_update`.

We consider only B-Splines whose support is completely contained in the hold-all domain D. To ensure the approximation properties of the trial space in the vicinity of ∂D, we should introduce additional basis functions with support close to ∂D. However, we can choose the hold-all domain D so that the initial guess Ω_0 is compactly contained in D in it. Then, these additional basis functions have little impact on the optimization routine, because what matters are the values of the vector field $\mathcal{V}$ on Ω_0. Therefore, we omit these additional basis functions to simplify the code.

The field `spOrder` indicates the polynomial order of the B-Splines.

The field `nBsplines` indicates how many univariate B-Splines are taken into account. For simplicity, we assume that there are as many univariate B-Splines in the x-direction as in the y-direction.

The fields `BSknots_x` and `BSknots_y` contain the knots of the uniform univariate Splines in the x- and in the y-direction, respectively.

The field `SplinesSupp` contains information on the support of the multivariate B-Splines. See Subsection A.3.1

The field `H` contains the $H^1(D)$-Gramian matrix, which is necessary to compute the $H^1(D)$-representative of the Eulerian derivative $d\tilde{\mathcal{J}}$. See Subsection A.3.2

The field `T_update` is a struct that contains the coefficients of the B-Spline representation of the vector field $\mathcal{V}$. It is initialized to $\mathcal{V} = 0$.

```
function SI = initializeSplineInfo
%generates (and save in struct SI) quadratic B-Splines Info

%Hold-all domain D (a square)
HoldAll.pl = -[2 2];
HoldAll.pu = [2 2];

%order of B-Splines
SI.spOrder = 2; %quadratic B-Splines

%number of univariate B-Splines
SI.nBsplines = 18;

%1D arrays of Splines knots
SI.BSknots_x = linspace(HoldAll.pl(1),HoldAll.pu(1),...
    SI.nBsplines+(SI.spOrder+1));
SI.BSknots_y = linspace(HoldAll.pl(2),HoldAll.pu(2),...
    SI.nBsplines+(SI.spOrder+1));

%Support of all multivariate B-Splines
SI.SplinesSupp = ComputeSplineSupport(SI);

%Gramian matrix for H1 representative of dJ
SI.H = H1Bspline(SI);

%pre-allocate array for coefficients of vector field
SI.T_update.X = zeros(SI.nBsplines^2,1);
SI.T_update.Y = zeros(SI.nBsplines^2,1);

end
```

A.3.1. The function `ComputeSplineSupport`

The function `ComputeSplineSupport` returns the matrix `SplinesSupp`, which has dimension (`nBsplines`^{2}, 4). Each row indicates the support of a multivariate B-Spline. For instance, `SplineSupp(1,:)` $= [a_x, b_x, a_y, b_y]$, where (a_x, a_y)

denotes the lower left corner of the support, and (b_x, b_y) denotes the upper right corner of the support. The numbering of multivariate B-Splines is showed in Figure A.1.

```
function SplinesSupp = ComputeSplineSupport(SI)
%returns matrix with supports of multivariate B-Splines

nBsplines = SI.nBsplines;
BSknots_x = SI.BSknots_x;
BSknots_y = SI.BSknots_y;
spOrder = SI.spOrder;

%fix a numbering of the multivariate B-Splines
IdxDof=[kron((1:nBsplines).',ones(nBsplines,1)), ...
    kron(ones(nBsplines,1),(1:nBsplines).')];

%construct the support of each multivariate B-Spline
SplinesSupp = [BSknots_x(IdxDof(:,1)).', ...
    BSknots_x(IdxDof(:,1)+(spOrder+1)).', ...
    BSknots_y(IdxDof(:,2)).', ...
    BSknots_y(IdxDof(:,2)+(spOrder+1)).'];

end
```

A.3.2. The function `H1Bspline`

The function `ComputeSplineSupport` returns the $H^1(D)$-Gramian matrix `H`. The entries of `H` are

$$\mathtt{H}_{ij} = (B_i, B_j)_{H^1(D)}\,,$$

where $(B_i, B_j)_{H^1(D)}$ denotes the $H^1(D)$-inner product between the i^{th} and the j^{th} multivariate B-Spline. For simplicity, we assume that the hold-all domain D is a square. To construct `H` we take the Kronecker product of its 1D counterpart.

The function `ComputeSplineSupport` calls the `LehrFEM` function `gauleg`, which implements a 1D Gauss-Legendre quadrature rule to evaluate the 1D integrals involved; see Subsection A.3.2.1

```
function H = H1Bspline(SI)
%exploit the tensor product structure to
```

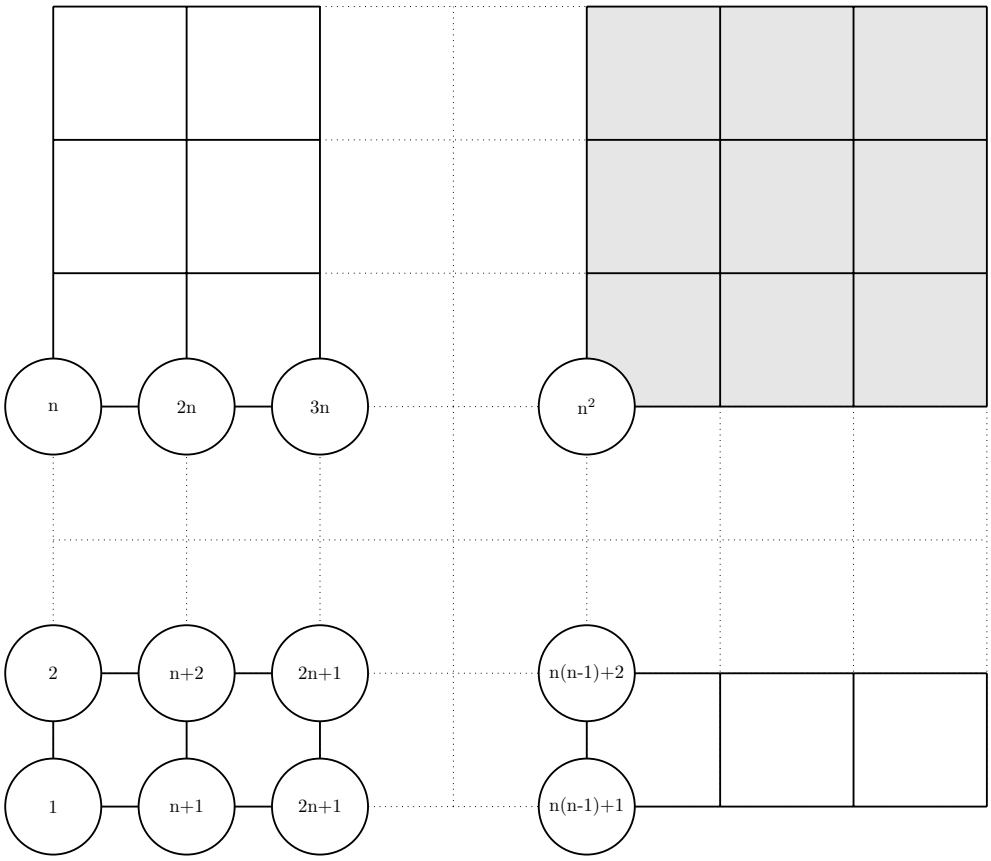

Figure A.1.: Multivariate B-Spline numbering used in the function `ComputeSplineSupport`. The grey square indicates the support of a quadratic multivariate B-Spline.

```
%compute H1-Gramian matrix for multivariate B-Splines
%the support of a B-spline is assumed to be a square
%(use only BSknots_x)

spOrder = SI.spOrder;
assert(spOrder==2); %just for quadratic B-Splines

nBsplines = SI.nBsplines;
BSknots = SI.BSknots_x;

%n-point Gauss-Legendre quadrature rule, order 2n-1 (LehrFEM)
%has to integrate piecewise pol. of order 2*spOrder
n = spOrder + 1;

%repeat and shift quadrature nodes to cover interval
%[BSknots(1),BSknots(4)] (supp. of a univariate quad. B-Spline)
%note that we use uniform B-splines
QR = gauleg(BSknots(1),BSknots(2),n);
QRx = bsxfun(@plus,QR.x,(BSknots(2)-BSknots(1))*[0 1 2]);
QRx = QRx(:);

%evaluate B-Spline and its derivative in QRx
[B, Bdiff] = myfastsplineAndDiff(QRx,BSknots(1),...
    BSknots(spOrder+2),spOrder);

%split values of B-Splines where they are polynomials
B1 = B(1:n);
B2 = B((n+1):2*n);
B3 = B((2*n+1):3*n);

B1diff = Bdiff(1:n);
B2diff = Bdiff((n+1):2*n);
B3diff = Bdiff((2*n+1):3*n);

%create the local assembly matrix

%L2-norm contribution
Aloc = zeros(3,3);
Aloc(1,1) = QR.w'*(B1.*B1);
Aloc(1,2) = QR.w'*(B1.*B2);
Aloc(1,3) = QR.w'*(B1.*B3);
Aloc(2,2) = QR.w'*(B2.*B2);
Aloc(2,3) = QR.w'*(B2.*B3);
Aloc(3,3) = QR.w'*(B3.*B3);
%Aloc is symmetric
```

```
Aloc = Aloc + triu(Aloc,1).';

%H1-seminorm contribution
AlocDiff = zeros(3,3);
AlocDiff(1,1) = QR.w'*(B1diff.*B1diff);
AlocDiff(1,2) = QR.w'*(B1diff.*B2diff);
AlocDiff(1,3) = QR.w'*(B1diff.*B3diff);
AlocDiff(2,2) = QR.w'*(B2diff.*B2diff);
AlocDiff(2,3) = QR.w'*(B2diff.*B3diff);
AlocDiff(3,3) = QR.w'*(B3diff.*B3diff);
AlocDiff = AlocDiff + triu(AlocDiff,1).';

%local to global map
Iloc = [-2 -2 -2 -1 -1 -1 0 0 0]';
Jloc = [-2 -1 0 -2 -1 0 -2 -1 0]';
Sloc = [Aloc(3,3) Aloc(3,2) Aloc(3,1)...
    Aloc(2,3) Aloc(2,2) Aloc(2,1)...
    Aloc(1,3) Aloc(1,2) Aloc(1,1)]';
SlocDiff = [AlocDiff(3,3) AlocDiff(3,2) AlocDiff(3,1)...
    AlocDiff(2,3) AlocDiff(2,2) AlocDiff(2,1)...
    AlocDiff(1,3) AlocDiff(1,2) AlocDiff(1,1)]';

%create indeces for sparse assembly
Iloc = bsxfun(@plus,Iloc,1:(nBsplines+2));
Jloc = bsxfun(@plus,Jloc,1:(nBsplines+2));
Sloc = repmat(Sloc,1,(nBsplines+2));
SlocDiff = repmat(SlocDiff,1,(nBsplines+2));

Iloc = Iloc(:);
Jloc = Jloc(:);
Sloc = Sloc(:);
SlocDiff = SlocDiff(:);

%exclude B-Splines whose support exits the hold-all domain
subidx = Iloc>0 & Jloc>0 & Iloc<=nBsplines & Jloc<=nBsplines;

Iloc = Iloc(subidx);
Jloc = Jloc(subidx);
Sloc = Sloc(subidx);
SlocDiff = SlocDiff(subidx);

%assemble 1D stiffness and mass matrix
M_L2 = sparse(Iloc,Jloc,Sloc,nBsplines,nBsplines);
M_semiH1 = sparse(Iloc,Jloc,SlocDiff,nBsplines,nBsplines);
```

```
%assemble 2D stiffnes and mass matrix
H = kron(M_L2,M_L2)+ kron(M_L2,M_semiH1)+kron(M_semiH1,M_L2);

end
```

A.3.2.1. The function gauleg

The function **gauleg** belongs to the MATLAB-library **LehrFEM** developed at ETH Zurich. It returns a struct **QuadRule** that contains the weights (in the field **w**) and the coordinates (in the field **x**) of a **n** point Gauss-Legendre quadrature rule in the interval (**a**,**b**).

```
function QuadRule = gauleg(a,b,n,tol)
% GL_RULE 1D Gauss-Legendre quadrature rule.
%
%   QUADRULE = GAULEG(A,B,N,TOL) computes the N-point
%   Gauss-Legendre quadrature rule on the interval [A,B]
%   up to the prescribed tolerance TOL. If no tolerance
%   is prescribed GAULEG uses the machine precision EPS.
%
%   Note that all quadrature rules obtained from GAULEG
%   are of order 2*N-1.
%
%   The struct QUADRULE contains the following fields:
%    W N-by-1 matrix specifying the weights of the quad. rule.
%    X N-by-1 matrix specifying the abscissae of the q. rule.
%
%   Example:
%
%   QuadRule = gauleg(0,1,10,1e-6);

%   Copyright 2005-2005 Patrick Meury
%   SAM - Seminar for Applied Mathematics
%   ETH-Zentrum
%   CH-8092 Zurich, Switzerland

  % Check for right number of arguments

  if(nargin == 3)
    tol = eps;
  end

  % Initalize variables
```

```
m = floor((n+1)/2);
xm = (b+a)/2;
xl = (b-a)/2;

for i = 1:m

  % Initial guess of root (starting value)

  z = cos(pi*(i-1/4)/(n+1/2));

  delta = tol+1;

  while(delta > tol)

    p1 = 0;
    p2 = 1;

    for k = 0:(n-1)

      % Computing value of n-th Legendre polynomial at point
      % z using the recursion:
      %
      %   (j+1)*P_(j+1)(z) = (2*j+1)*z*P_(j)(z)-j*P_(j-1)(z)

      p3 = ((2*k+1)*z*p2-k*p1)/(k+1);

      % Computing value of first derivative of n-th Legendre
      % polynomial at point z using the recursion:
      %
      %   (1-z^2)*P'_(j)(z) = j*[z*P_(j)(z)-P_(j-1)(z)]

      dp = n*(z*p3-p2)/(z^2-1);
      p1 = p2;
      p2 = p3;

    end

    % Performing Newton update

    z_old = z;
    z = z_old-p3/dp;

    delta = abs(z-z_old);
```

```
    end

    % Computing weights and abscissae

    x(i) = xm-xl*z;
    x(n+1-i) = xm+xl*z;
    w(i) = 2*xl/((1-z^2)*dp^2);
    w(n+1-i) = w(i);

  end

  % Assign output arguments

  QuadRule.w = w(:);
  QuadRule.x = x(:);

return
```

A.4. The function precomputeAffineTransformation

The function `precomputeAffineTransformation` returns the struct `precompAffTrans` and the matrix `xQ_Transf`.

The struct `precompAffTrans` contains the fields `det_BK`, `iBKa`,`iBKb`,`iBKc`,`iBKd`. The field `det_BK` is an array of lenght n_t (the number of triangles in the mesh). Each entry `det_BK(i)` corresponds to the determinant of the affine transformation that maps the reference triangle (0,0)-(1,0)-(0,1) to the `i`$^{\text{th}}$ triangle of the mesh.

The fields `iBKa`,`iBKb`,`iBKc`,`iBKd` are array of lenght n_t. The matrix

```
[iBKa(i), iBKb(i); iBKc(i), iBKd(i)]
```

corresponds to the transposed inverse of the Jacobian of the affine transformation that maps the reference triangle (0,0)-(1,0)-(0,1) to the `i`$^{\text{th}}$ triangle of the mesh.

The matrix `xQ_Transf` has dimension $(n_q n_t) \times 2$, where $n_q = 3$ because we employ the 3 point quadrature rule `P3O3`. It contains the mapped quadrature points. For instance, the vector

```
[xQ_Transf(i); xQ_Transf(i+1); xQ_Transf(i+2)]
```

contains the 3 quadrature points mapped to the `i`$^{\text{th}}$ triangle of the mesh.

```
function [precompAffTrans,xQ_Transf] = ...
    precomputeAffineTransformation(Mesh,QuadRule)

nPts = size(QuadRule.x,1); %number of quadrature points
nElements = size(Mesh.Elements,1);

%Affine transformation
bK = Mesh.Coordinates(Mesh.Elements(:,1),:);
BK_row1 = Mesh.Coordinates(Mesh.Elements(:,2),:)-bK;
BK_row2 = Mesh.Coordinates(Mesh.Elements(:,3),:)-bK;
det_BK = BK_row1(:,1).*BK_row2(:,2) ...
    - BK_row1(:,2).*BK_row2(:,1);
inv_BK_row1 = [BK_row2(:,2) -BK_row1(:,2)]./det_BK(:,[1 1]);
inv_BK_row2 = [-BK_row2(:,1) BK_row1(:,1)]./det_BK(:,[1 1]);

iBKa = (inv_BK_row1(:,1))';
iBKb = (inv_BK_row1(:,2))';
iBKc = (inv_BK_row2(:,1))';
iBKd = (inv_BK_row2(:,2))';

precompAffTrans.det_BK = det_BK;
precompAffTrans.iBKa = iBKa;
precompAffTrans.iBKb = iBKb;
precompAffTrans.iBKc = iBKc;
precompAffTrans.iBKd = iBKd;

%map quadrature points
BKa = transpose(BK_row1(:,1));
BKb = transpose(BK_row1(:,2));
BKc = transpose(BK_row2(:,1));
BKd = transpose(BK_row2(:,2));

QuadRule_x_1 = QuadRule.x(:,ones(nElements,1));
QuadRule_x_2 = QuadRule.x(:,2*ones(nElements,1));

xQ_1 = BKa(ones(nPts,1),:).*QuadRule_x_1 ...
    + BKc(ones(nPts,1),:).*QuadRule_x_2;
xQ_2 = BKb(ones(nPts,1),:).*QuadRule_x_1 ...
    + BKd(ones(nPts,1),:).*QuadRule_x_2;

xQ_1 = xQ_1 + transpose(bK(:,ones(nPts,1)));
xQ_2 = xQ_2 + transpose(bK(:,2*ones(nPts,1)));

xQ_Transf = [xQ_1(:) xQ_2(:)];
```

```
end
```

A.5. The function precomputeSplineJacobian

The function `precomputeSplineJacobian` returns the struct `precompDV`, which contains the fields `V`, `V_dX`, `V_dY`. To compute the entries of these fields it calls the function `myfastsplineAndDiff`.

The field `V` is a matrix of size $(n_q n_t) \times$ `nBsplines`^{2}. Each column contains the values of a multivariate B-Splines evaluated on the mapped quadrature points `xQ_Transf`.

The field `V_dX` is a matrix of size $(n_q n_t) \times$ `nBsplines`^{2}. Each column contains the value of the derivative (with respect to the x-component) of a multivariate B-Spline evaluated on the mapped quadrature points `xQ_Transf`.

The field `V_dY` is a matrix of size $(n_q n_t) \times$ `nBsplines`^{2}. Each column contains the value of the derivative (with respect to the y-component) of a multivariate B-Spline evaluated on the mapped quadrature points `xQ_Transf`.

```
function  precompDV = precomputeSplineJacobian(SI,xQ_Transf)
%precompute multivar. B-Splines in mapped quadr. points

spOrder = SI.spOrder;
SplinesSupp = SI.SplinesSupp;

%support of B-splines
ax = SplinesSupp(:,1).';
bx = SplinesSupp(:,2).';
ay = SplinesSupp(:,3).';
by = SplinesSupp(:,4).';

%pre-evaluate multivariate B-Splines
[SPLINE_X,DIFFSPLINE_X] = ...
    myfastsplineAndDiff(xQ_Transf(:,1),ax,bx,spOrder);
[SPLINE_Y,DIFFSPLINE_Y] = ...
    myfastsplineAndDiff(xQ_Transf(:,2),ay,by,spOrder);

precompDV.V = SPLINE_X.*SPLINE_Y;
precompDV.V_dX = DIFFSPLINE_X.*SPLINE_Y;
precompDV.V_dY = SPLINE_X.*DIFFSPLINE_Y;

end
```

A.5.1. The function `myfastsplineAndDiff`

The function **`myfastsplineAndDiff`** evaluates (according to (3.11)) all univariate quadratic B-Splines (and their derivatives) with support specified by the row vectors `a` and `b` in the points listed in the columns vector `x`. These values are returned in the two matrices `Y` and `dY`, which have size `length(x)` $\times$ `length(a)`.

```
function [Y,dY]=myfastsplineAndDiff(x,a,b,spOrder)
% Evaluate in x all univariate quadratic B-splines with support
% specified by (a,b)

assert(min(b-a)>=0)
assert(spOrder == 2) %only for quadratic B-splines

%shift and scale x to the interval [0,3]
BminusA=b-a;

length_a = length(a);
length_x = length(x);

X = bsxfun(@minus,x,a);
X = (spOrder+1)*bsxfun(@rdivide,X,BminusA);

%indeces of B-Splines for which X is in the interval [0,1]
idx = X>0&X<=1;
X_idx01 = X(idx);
[I_01,J_01] = find(idx);
%indeces of B-Splines for which X is in the interval [1,2]
idx = X>1&X<=2;
X_idx12 = X(idx);
[I_12,J_12] = find(idx);
%indeces of B-Splines for which X is in the interval [2,3]
idx = X>2&X<=3;
X_idx23 = X(idx);
[I_23,J_23] = find(idx);

clear X idx

%evaluate and store the values of X^2 for each subinterval
X2_idx01 = X_idx01.*X_idx01;
```

```
X2_idx12 = X_idx12.*X_idx12;
X2_idx23 = X_idx23.*X_idx23;

I = [I_01;I_12;I_23];
clear I_01 I_12 I_23
J = [J_01;J_12;J_23];
clear J_01 J_12 J_23

%evaluate univariate B-Splines
Y = [X2_idx01/2;...
    (-2*(X2_idx12)+6*(X_idx12)-3)/2;...
    ((X2_idx23)-6*(X_idx23)+9)/2];

%evaluate derivative of univariate B-Splines
dY = [(X_idx01);...
    -2*(X_idx12)+3;...
    (X_idx23)-3];

clear -regexp ^X %clear all variables that start with X

%assemble sparse matrix Y and dY
Y = sparse(I,J,Y,length_x,length_a);
if length(BminusA) == 1
    dY=(spOrder+1)*dY./(BminusA(J));
else
    dY=(spOrder+1)*dY./transpose(BminusA(J));
end
dY = sparse(I,J,dY,length_x,length_a);

return
```

A.6. The function computeState

The function `computeState` computes the finite element solution of (A.3). Additionally, it returns the stiffness matrix `A` of (A.3) (which can be recycled to compute the finite element solution of (A.7)) and the arrays `DTa`, `DTb`, `DTc`, `DTd`, `detDT`, and `FreeDofs`.

The arrays `DTa`, `DTb`, `DTc`, `DTd`, `detDT` have length $n_q n_t$, where $n_q = 3$ (because we employ the 3 point quadrature rule `P3O3`) and n_t is the number of triangles in the mesh. The matrix

```
[DTa(i), DTb(i); DTc(i), DTd(i)]
```

corresponds to the Jacobian $\mathbf{D}T_{\mathcal{V}}$ evaluated in the quadrature point whose coordinates correspond to the `i`$^{\text{th}}$ row of the matrix `xQ_Transf`. The determinant of this matrix is stored in `detDT(i)`.

The array `FreeDofs` contains the indexes of the mesh nodes that are not on the boundary $\partial\Omega_0$.

```
function [U,A,DTa,DTb,DTc,DTd,detDT,FreeDofs] = ...
    computeState(Mesh,SI,precompDV,precompAffTrans,QR,...
    SourceFct,DirFct,xQ_Transf)

%Assemble Stiffness Matrix
[A,DTa,DTb,DTc,DTd,detDT] = ...
    assemMat_LFE_withMAP(Mesh,precompDV,SI,QR,precompAffTrans);

%preallocate solution vector
nCoordinates = length(Mesh.Coordinates);
U = zeros(nCoordinates,1);

%assemble RHS
RHS = assemLoad_LFE_state(Mesh,QR,SourceFct,precompAffTrans,...
    xQ_Transf);

%Dirichlet Boundary Condition
U(Mesh.BdNodes) = DirFct(Mesh.Coordinates(Mesh.BdNodes,:));
FreeDofs = setdiff(1:nCoordinates,Mesh.BdNodes);

%Adjust RHS
RHS = RHS-A*U;

%compute Solution
U(FreeDofs) = A(FreeDofs,FreeDofs)\RHS(FreeDofs);
end
```

A.6.1. The function `assemMat_LFE_withMAP`

The function `assemMat_LFE_withMAP` returns the stiffness matrix `A` of (A.3) and the arrays `DTa`, `DTb`, `DTc`, `DTd`, `detDT`, which will belong to the output of the function `computeState`. It calls the function `grad_shap_LFE` of the MATLAB-library `LehrFEM` and the function `myMatrixMult2D`.

```
function [A,DTa,DTb,DTc,DTd,detDT] = ...
```

```
    assemMat_LFE_withMAP(Mesh,precompDV,SI,QR,precompAffTrans)

nElem = length(Mesh.Elements);
nPts = size(QR.w,1);
grad_N = grad_shap_LFE([0 0]);

%Affine transformation
det_BK = precompAffTrans.det_BK;
iBKa = precompAffTrans.iBKa;
iBKb = precompAffTrans.iBKb;
iBKc = precompAffTrans.iBKc;
iBKd = precompAffTrans.iBKd;

%Bsplines diffusion matrix
DTa = precompDV.V_dX*SI.T_update.X + ones(nPts*nElem,1);
DTb = precompDV.V_dY*SI.T_update.X;
DTc = precompDV.V_dX*SI.T_update.Y;
DTd = precompDV.V_dY*SI.T_update.Y + ones(nPts*nElem,1);

detDT = DTa.*DTd-DTb.*DTc;

%C = (DT)^(-1) * (DT)^(-T) * detDT
Ca = (DTd.^2+DTb.^2)./abs(detDT);
Cb = (-DTd.*DTc-DTa.*DTb)./abs(detDT);
Cc = Cb;
Cd = (DTa.^2+DTc.^2)./abs(detDT);

Ca = reshape(Ca,nPts,nElem);
Cb = reshape(Cb,nPts,nElem);
Cc = reshape(Cc,nPts,nElem);
Cd = reshape(Cd,nPts,nElem);

%perform quadrature
CaQ = QR.w.'*Ca;
CbQ = QR.w.'*Cb;
CcQ = QR.w.'*Cc;
CdQ = QR.w.'*Cd;

%combine (TK = det_BK*transpose(inv_BK)*CQ*inv_BK)
[TKa,TKb,TKc,TKd] = ...
    myMatrixMult2D(CaQ,CbQ,CcQ,CdQ,iBKa,iBKb,iBKc,iBKd);
[TKa,TKb,TKc,TKd] = ...
    myMatrixMult2D(iBKa,iBKc,iBKb,iBKd,TKa,TKb,TKc,TKd);

TKa = det_BK'.*TKa;
```

```
TKb = det_BK'.*TKb;
TKc = det_BK'.*TKc;
TKd = det_BK'.*TKd;

vectMatvect = @(v,Ma,Mb,Mc,Md,w) v(1)*(Ma*w(1)+Mb*w(2)) ...
    + v(2)*(Mc*w(1)+Md*w(2));

Aloc_11 = vectMatvect(grad_N(1:2),TKa,TKb,TKc,TKd,grad_N(1:2));
Aloc_12 = vectMatvect(grad_N(1:2),TKa,TKb,TKc,TKd,grad_N(3:4));
Aloc_13 = vectMatvect(grad_N(1:2),TKa,TKb,TKc,TKd,grad_N(5:6));
Aloc_22 = vectMatvect(grad_N(3:4),TKa,TKb,TKc,TKd,grad_N(3:4));
Aloc_23 = vectMatvect(grad_N(3:4),TKa,TKb,TKc,TKd,grad_N(5:6));
Aloc_33 = vectMatvect(grad_N(5:6),TKa,TKb,TKc,TKd,grad_N(5:6));

% Update lower triangular part
Aloc_21 = Aloc_12;
Aloc_31 = Aloc_13;
Aloc_32 = Aloc_23;

A = [Aloc_11; Aloc_21; Aloc_31; Aloc_12; Aloc_22; Aloc_32; ...
    Aloc_13; Aloc_23; Aloc_33];

I = reshape(Mesh.Elements(:,[1 2 3 1 2 3 1 2 3])',9*nElem,1);
J = reshape(Mesh.Elements(:,[1 1 1 2 2 2 3 3 3])',9*nElem,1);
A = sparse(I,J,A(:));
end
```

A.6.1.1. The function `grad_shap_LFE`

The function **`grad_shap_LFE`** belongs to the MATLAB-library **`LehrFEM`** developed at ETH Zurich. It evaluates the gradient of the hat functions associated to the vertices of the reference element (0,0)-(1,0)-(0,1) evaluated in the points whose coordinates are specified in the rows of the matrix **`x`**. The output **`grad_shap`** is a matrix of size **`length(x)`**$\times$ 6. The first and the second columns contains the values of the gradient of the hat function associated to the vertex (0,0). The third and the fourth columns contains the values of the gradient of the hat function associated to the vertex (1,0). The fifth and the sixth columns contains the values of the gradient of the hat function associated to the vertex (0,1).

```
function grad_shap = grad_shap_LFE(x)
```

```
% GRAD_SHAP_LFE Gradient of shape functions.
%
%   GRAD_SHAP = GRAD_SHAP_LFE(X) computes the values of the
%   gradient of the shape functions for the Lagrangian finite
%   element of order 1 at the quadrature points X.
%
%   Example:
%
%   grad_shap = grad_shap_LFE([0 0]);
%
%   See also shap_LFE.

%   Copyright 2005-2005 Patrick Meury and Kah Ling Sia
%   SAM - Seminar for Applied Mathematics
%   ETH-Zentrum
%   CH-8092 Zurich, Switzerland

  % Initialize constants

  nPts = size(x,1);

  % Preallocate memory

  grad_shap = zeros(nPts,6);

  % Compute values of gradients

  grad_shap(:,1:2) = -ones(nPts,2);
  grad_shap(:,3) = ones(nPts,1);
  grad_shap(:,6) = ones(nPts,1);

return
```

A.6.1.2. The function myMatrixMult2D

The function **myMatrixMult2D** perform the standard matrix multiplication for matrices that are input entrywise.

```
function [Ma,Mb,Mc,Md] = myMatrixMult2D(Xa,Xb,Xc,Xd,Ya,Yb,Yc,Yd)

Ma = Xa.*Ya + Xb.*Yc;
Mb = Xa.*Yb + Xb.*Yd;
Mc = Xc.*Ya + Xd.*Yc;
```

```
Md = Xc.*Yb + Xd.*Yd;

end
```

A.6.2. The function `assemLoad_LFE_state`

The function **`assemLoad_LFE_state`** returns the load vector of the state problem of (A.3). It calls the function **`shap_LFE`** of the MATLAB-library **`LehrFEM`**.

```
function L = assemLoad_LFE_state(Mesh,QR,SourceFct,...
    precompAffTrans,xQ_Transf)
%assemble the load vector for the state problem

nPts = length(QR.w);
det_BK = precompAffTrans.det_BK;
nCoordinates = length(Mesh.Coordinates);
nElements = length(Mesh.Elements);

% Precompute shape functions
N = shap_LFE(QR.x);

vidx = Mesh.Elements;

%prepare integrand
FVal = SourceFct(xQ_Transf);
FVal = reshape (FVal,nPts,nElements);
FVal = transpose(FVal);

%evaluate integrals
FValb1Q = (FVal*(QR.w.*N(:,1))).*det_BK;
FValb2Q = (FVal*(QR.w.*N(:,2))).*det_BK;
FValb3Q = (FVal*(QR.w.*N(:,3))).*det_BK;

%fast assembly
FValbQ = [FValb1Q; FValb2Q; FValb3Q];
L = accumarray(vidx(:), FValbQ,[nCoordinates 1]);

end
```

A.6.2.1. The function shap_LFE

The function **shap_LFE** belongs to the MATLAB-library **LehrFEM** developed at ETH Zurich. It evaluates the hat functions associated to the vertices of the reference element (0,0)-(1,0)-(0,1) evaluated in the points whose coordinates are specified in the rows of the matrix **x**. The output **shap** is a matrix of size **length(x)**$\times$ 3. The first column contains the values of the hat function associated to the vertex (0,0). The second column contains the values of the hat function associated to the vertex (1,0). The third column contains the values of the hat function associated to the vertex (0,1).

```
function shap = shap_LFE(x)
% SHAP_LFE Shape functions.
%
%   SHAP = SHAP_LFE(X) computes the values of the shape
%   functions for the Lagrangian finite element of order
%   1 at the quadrature points X.
%
%   Example:
%
%   shap = shap_LFE([0 0]);
%
%   See also grad_shap_LFE.

%   Copyright 2005-2005 Patrick Meury and Kah Ling Sia
%   SAM - Seminar for Applied Mathematics
%   ETH-Zentrum
%   CH-8092 Zurich, Switzerland

  shap = zeros(size(x,1),3);

  shap(:,1) = 1-x(:,1)-x(:,2);
  shap(:,2) = x(:,1);
  shap(:,3) = x(:,2);

return
```

A.7. The function evalJ

The function **evalJ** evaluates the misfit functional (A.2).

```
function J = evalJ(U,Mesh,QR,precompAffTrans,detDT)

%pre-evaluate a hat function in quadrature points
N = shap_LFE(QR.x);

%evaluate state variable in quadrature points;
%size(UxQ) = [nPts,nElements]
idx = Mesh.Elements;
UxQ = [U(idx(:,1)) U(idx(:,2)) U(idx(:,3))]*transpose(N);
UxQ = transpose(UxQ);

%evaluate the integrand on the quadrature points;
%size(integrand) = [nPts,nElements]
detDT = reshape(detDT,size(UxQ,1),size(UxQ,2));
integrand = (UxQ).^2.*detDT;

%integrate
J = (QR.w'*integrand)*precompAffTrans.det_BK;

end
```

A.8. The function `computeAdjoint`

The function **`computeAdjoint`** computes the finite element solution of (A.7).

```
function P = ...
    computeAdjoint(Mesh,A,detDT,precompAffTrans,QR,U,FreeDofs)

%preallocate array
nCoordinates = length(Mesh.Coordinates);
P = zeros(nCoordinates,1);

%assemble RHS
RHS = assemLoad_LFE_adjoint(Mesh,QR,U,detDT,precompAffTrans);

%compute Solution
P(FreeDofs) = A(FreeDofs,FreeDofs)\RHS(FreeDofs);
end
```

A.8.1. The function assemLoad_LFE_adjoint

The function **assemLoad_LFE_adjoint** assembles the load vector of the adjoint problem (A.7).

```
function L = ...
    assemLoad_LFE_adjoint(Mesh,QR,U,detDT,precompAffTrans)

det_BK = precompAffTrans.det_BK;
nCoordinates = length(Mesh.Coordinates);

%pre-evaluate a hat function in quadrature points
N = shap_LFE(QR.x);

%evaluate state variable in quadrature points
%size(UxQ) = [nElements,nPts]
vidx = Mesh.Elements;
UxQ = [U(vidx(:,1)) U(vidx(:,2)) U(vidx(:,3))]*transpose(N);

%evaluate part of integrand on the quadrature points
%size(integrand) = [nPts,nElements]
detDT = reshape(detDT,size(UxQ,2),size(UxQ,1));
FVal = -2*UxQ.*transpose(detDT);

%compute integrals
FValb1Q = (FVal*(QR.w.*N(:,1))).*det_BK;
FValb2Q = (FVal*(QR.w.*N(:,2))).*det_BK;
FValb3Q = (FVal*(QR.w.*N(:,3))).*det_BK;

%fast assembly of load vector
FValbQ = [FValb1Q; FValb2Q; FValb3Q];
L = accumarray(vidx(:), FValbQ,[nCoordinates 1]);

end
```

A.9. The function EulerianDerivative

The function **EulerianDerivative** returns the structure **dJ**, which contains the values of the Eulerian derivative (A.4) for all vector fields of the form $\mathcal{W} = \mathbf{e}_x B_j$ (in the field **X**) and $\mathcal{W} = \mathbf{e}_y B_j$ (in the field **Y**), where $\mathbf{e}_x = (1,0)^T$, $\mathbf{e}_y = (0,1)^T$, and B_j denotes the j^{th} multivariate B-Spline. The functions u and p

are replaced by finite element approximations. In the current implementation it is assumed that function f (the right-hand side of the state problem (A.3)) is constant, so that the term $-\mathbf{grad}\, f \cdot \mathcal{W}p \det \mathbf{D}T_{\mathcal{V}}$ in (A.4) vanishes.

The main issue to evaluate Formula (A.4) is to compute

$$\int_{\Omega_0} \mathbf{grad}\, p \cdot \partial_{\mathcal{W}} \mathbf{M}_{\mathcal{V}} \mathbf{grad}\, u \, \mathrm{d}\boldsymbol{x} \tag{A.8}$$

efficiently. Replacing the matrix $\partial_{\mathcal{W}}\mathbf{M}_{\mathcal{V}}$ by its definition given in Equation (A.5), the integrand of (A.8) reads

$$\begin{aligned}\mathbf{grad}\, p \cdot \det(\mathbf{D}T_{\mathcal{V}}) \Big(&\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} \\ &-\mathbf{D}T_{\mathcal{V}}^{-1}(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}T_{\mathcal{V}}^{-T}\Big)\, \mathbf{grad}\, u\,,\end{aligned}$$

or, equivalently,

$$\begin{aligned}\det(\mathbf{D}T_{\mathcal{V}})(\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, p) \cdot \Big(&\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W}) \\ &-(\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\Big)\, (\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, u)\,,\end{aligned}$$

which we rewrite as

$$\begin{aligned}&\Big((\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, p) \cdot (\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, u)\Big)\, \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\mathcal{W}) \\ -&(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, p) \cdot (\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1})\Big)\, (\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, u)\,.\end{aligned} \tag{A.9}$$

Formula (A.9) suggests to begin with the computation of

$$\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, p \quad \text{and} \quad \mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, u\,,$$

which we store in the variables `detDTDTinvtranspGradP` and `DTinvtranspGradU`, respectively. The size of these variables is $(n_q n_t)\times 2$. Moreover, it is convenient to store the term

$$\Big(\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, p\Big) \cdot \Big(\mathbf{D}T_{\mathcal{V}}^{-T}\, \mathbf{grad}\, u\Big)$$

in the variable

`DTinvtranspGradU_times_DTinvtranspGradP`,

(which has size $(n_q n_t) \times 1$) because it is independent of the direction $\mathcal{W}$. Additionally, we store the (pointwise) multiplication of any combination of the x- and the y-components of $\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-T}\,\mathbf{grad}\,p$ and $\mathbf{D}T_{\mathcal{V}}^{-T}\,\mathbf{grad}\,u$ in the matrix `integrand_2` (which has size $(n_q n_t) \times 4$). This is useful to evaluate the integrand in the second line of (A.9).

To efficiently evaluate the integrand in the first line of (A.4), we store the values of $u^2 - fp$ in the variable `U2minusfp` (which has size $(n_q n_t) \times 1$).

Once we have computed and stored these variables, we can evaluate the Eulerian derivative by calling the function `EulerianDerivativeIntegrate`.

```
function dJ = ...
    EulerianDerivative(Mesh,SI,QuadRule,U,P,precompAffTrans,...
    precompDV,DTa,DTb,DTc,DTd,detDT,xQ_Transf,SourceFct)

T_update = SI.T_update;
nElements = length(Mesh.Elements);
nPts = size(QuadRule.w,1);
nBsplines = length(T_update.X);

%update position of quadrature points according to
%B_SPline transformation
xQ_moved = xQ_Transf ...
    + [precompDV.V*SI.T_update.X, precompDV.V*SI.T_update.Y];


%pre-evaluate hat functions in each triangle
N= shap_LFE(QuadRule.x);
vidx = Mesh.Elements;

%evaluate u,p and source function in quadrature points
U_xQ = [U(vidx(:,1)) U(vidx(:,2)) U(vidx(:,3))]*transpose(N);
U_xQ = transpose(U_xQ);
U_xQ = U_xQ(:);

P_xQ = [P(vidx(:,1)) P(vidx(:,2)) P(vidx(:,3))]*transpose(N);
P_xQ = transpose(P_xQ);
P_xQ = P_xQ(:);

f_xQ = SourceFct(xQ_moved);

%factor in front of dWdetDTV
U2minusfP = U_xQ.^2-f_xQ.*P_xQ;
```

```
%pre-evaluate gradient of hat functions in each triangle
%size(gradU) = [nElements,2]
grad_N = grad_shap_LFE([0 0]);
gradU = U(Mesh.Elements(:,1))*grad_N(1:2)...
       +U(Mesh.Elements(:,2))*grad_N(3:4)...
       +U(Mesh.Elements(:,3))*grad_N(5:6);
gradP = P(Mesh.Elements(:,1))*grad_N(1:2)...
       +P(Mesh.Elements(:,2))*grad_N(3:4)...
       +P(Mesh.Elements(:,3))*grad_N(5:6);

%adjust gradU and gradP with Jacobian of affine transformation
det_BK = precompAffTrans.det_BK;
iBKa = transpose(precompAffTrans.iBKa);
iBKb = transpose(precompAffTrans.iBKb);
iBKc = transpose(precompAffTrans.iBKc);
iBKd = transpose(precompAffTrans.iBKd);

iBKgradU = [iBKa.*gradU(:,1) + iBKb.*gradU(:,2),...
    iBKc.*gradU(:,1) + iBKd.*gradU(:,2)];
iBKgradP = [iBKa.*gradP(:,1) + iBKb.*gradP(:,2),...
    iBKc.*gradP(:,1) + iBKd.*gradP(:,2)];

%B-Splines diffusion matrix
DTinv.a = DTd./detDT;
DTinv.b = -DTb./detDT;
DTinv.c = -DTc./detDT;
DTinv.d = DTa./detDT;

detDT_DTinv.a = DTd;
detDT_DTinv.b = -DTb;
detDT_DTinv.c = -DTc;
detDT_DTinv.d = DTa;

%replicate nPts times iBKgradU and iBKgradP
%save each component separately
iBKgradUE_1 = transpose(iBKgradU(:,ones(nPts,1)));
iBKgradUE_1 = iBKgradUE_1(:);
iBKgradUE_2 = transpose(iBKgradU(:,2*ones(nPts,1)));
iBKgradUE_2 = iBKgradUE_2(:);
iBKgradPE_1 = transpose(iBKgradP(:,ones(nPts,1)));
iBKgradPE_1 = iBKgradPE_1(:);
iBKgradPE_2 = transpose(iBKgradP(:,2*ones(nPts,1)));
iBKgradPE_2 = iBKgradPE_2(:);
```

```
%multiply iBKgradU and iBKgradP with Jacobian of B-Spline transf.
DTinvtranspGradU = [DTinv.a.*iBKgradUE_1 + DTinv.c.*iBKgradUE_2,...
    DTinv.b.*iBKgradUE_1 + DTinv.d.*iBKgradUE_2];
DTinvtranspGradP = [DTinv.a.*iBKgradPE_1 + DTinv.c.*iBKgradPE_2,...
    DTinv.b.*iBKgradPE_1 + DTinv.d.*iBKgradPE_2];

DTinvtranspGradU_times_DTinvtranspGradP = ...
    sum(DTinvtranspGradP.*DTinvtranspGradU,2);

detDTDTinvtranspGradP = detDT(:,[1 1]).*DTinvtranspGradP;

integrand_2 = ...
    [detDTDTinvtranspGradP(:,1).*DTinvtranspGradU(:,1),...
    detDTDTinvtranspGradP(:,1).*DTinvtranspGradU(:,2),...
    detDTDTinvtranspGradP(:,2).*DTinvtranspGradU(:,1),...
    detDTDTinvtranspGradP(:,2).*DTinvtranspGradU(:,2)];

emptyMat = sparse([],[],[],nPts*nElements,nBsplines );

dJloc = @(DVa,DVb,DVc,DVd) EulerianDerivativeIntegrate(Mesh,...
    DVa,DVb,DVc,DVd,QuadRule,DTinv,detDT_DTinv,...
    DTinvtranspGradU_times_DTinvtranspGradP,integrand_2,det_BK,...
    U2minusfP);

%integrand 1 (Bsplines in x dir)
dJ.X = dJloc(precompDV.V_dX,precompDV.V_dY,emptyMat,emptyMat);

%integrand 2 (Bsplines in y dir)
dJ.Y = dJloc(emptyMat,emptyMat,precompDV.V_dX,precompDV.V_dY);
```

A.9.1. The function `EulerianDerivativeIntegrate`

The function `EulerianDerivativeIntegrate` performs the integration that appears in the Eulerian derivative formula (A.4).

First, it computes (A.6) and stores it in the matrix `dWdetDTV`, which has size $n_q n_t \times \texttt{nBsplines}^2$.

Then, it computes the first line of (A.4) partly (it remains to multiply these values with the determinant of the Jacobian of the affine transformation that maps the reference triangle (0,0)-(1,0)-(0,1) to the triangles of the mesh). To perform this computation efficiently it employs the function `computeSpMatTimesRepVec`.

Then, it computes the second line of (A.4) partly. The matrix `integrand_1`

corresponds to the first line of (A.9). Its dimension is $n_q n_t \times \texttt{nBsplines}^2$. The matrices `M1a`, `M1b`, `M1c`, `M1d` contain the compontents of $\mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1}$. Each of these matrices has dimension $n_q n_t \times \texttt{nBsplines}^2$. The matrices `M2a`, `M2b`, `M2c`, `M2d` contain the compontents of $\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\mathcal{W}^T + \mathbf{D}\mathcal{W}\mathbf{D}T_{\mathcal{V}}^{-1}$. Each of these matrices has dimension $n_q n_t \times \texttt{nBsplines}^2$. The matrix `integrand_2b` corresponds to the second line of (A.9). Its dimension is $n_q n_t \times \texttt{nBsplines}^2$.

```
function dJloc = ...
    EulerianDerivativeIntegrate(Mesh,DWa,DWb,DWc,DWd,QR,DTinv,...
    detDT_DTinv,DTinvtranspGradU_times_DTinvtranspGradP,...
    integrand_2,det_BK,U2minusfP)

nBsplines2 = size(DWa,2);
nPts = size(QR.w,1);

nElements = length(Mesh.Elements);
QR.w = sparse(QR.w);

%Frechet derivative of map V -> detDT_V
%size(dWdetDTV) = [nPts*nElements,nBsplines^2]
dWdetDTV = computeSpMatTimesRepVec(DWa,detDT_DTinv.a)...
         + computeSpMatTimesRepVec(DWb,detDT_DTinv.c)...
         + computeSpMatTimesRepVec(DWc,detDT_DTinv.b)...
         + computeSpMatTimesRepVec(DWd,detDT_DTinv.d);

%compute first part of integral (partly)
U2minusfPdWdetDTV_Q = computeSpMatTimesRepVec(dWdetDTV,U2minusfP);
U2minusfPdWdetDTV_Q = ...
    reshape(U2minusfPdWdetDTV_Q,nPts,nElements*nBsplines2);
U2minusfPdWdetDTV_Q = QR.w'*U2minusfPdWdetDTV_Q;
U2minusfPdWdetDTV_Q = ...
    reshape(U2minusfPdWdetDTV_Q,nElements,nBsplines2);

%compute second part of integral (partly)
integrand_1 = computeSpMatTimesRepVec(dWdetDTV,...
    DTinvtranspGradU_times_DTinvtranspGradP);

M1a = computeSpMatTimesRepVec(DWa,DTinv.a) ...
    + computeSpMatTimesRepVec(DWb,DTinv.c);
M1b = computeSpMatTimesRepVec(DWa,DTinv.b) ...
    + computeSpMatTimesRepVec(DWb,DTinv.d);
M1c = computeSpMatTimesRepVec(DWc,DTinv.a) ...
    + computeSpMatTimesRepVec(DWd,DTinv.c);
```

```
M1d = computeSpMatTimesRepVec(DWc,DTinv.b) ...
    + computeSpMatTimesRepVec(DWd,DTinv.d);

M2a = 2*M1a;
M2b = M1b+M1c;
M2c = M2b;
M2d = 2*M1d;

integrand_2b = computeSpMatTimesRepVec(M2a,integrand_2(:,1))...
             + computeSpMatTimesRepVec(M2b,integrand_2(:,2))...
             + computeSpMatTimesRepVec(M2c,integrand_2(:,3))...
             + computeSpMatTimesRepVec(M2d,integrand_2(:,4));

integrand = integrand_1 - integrand_2b;

integrand = reshape(integrand,nPts,nElements*nBsplines2);
integrand = QR.w'*integrand;
integrand = reshape(integrand,nElements,nBsplines2);

%sum two contributions of integral
integrand_final = integrand + U2minusfPdWdetDTV_Q;

%multiply with Jacobian of affine transformation
dJloc = computeSpMatTimesRepVec(integrand_final,det_BK);
dJloc = transpose(sum(dJloc,1));

end
```

A.9.1.1. The function computeSpMatTimesRepVec

```
function M = computeSpMatTimesRepVec(M,v)
%create a diagonal matrix with diagonal v
%and multiply it with the matrix M

idx = 1:length(v);
v = sparse(idx,idx,v);
M = v*M;

end
```

A.10. The function `SteepestDescent`

The function **`SteepestDescent`** solves (3.20) to compute the coefficients of the $H^1(D)$-representative of the Eulerian derivative (A.4).

```
function T_update_new = SteepestDescent(SI,W_dJ)

H = SI.H;

%compute H1 optimal descent direction

W_X = W_dJ.X; %get non-zero values of dJ
W_Y = W_dJ.Y;

%optimal directions subject to constraint of H1 type
Hnorm = sqrt((W_X).'*(H\W_X)+(W_Y).'*(H\W_Y));
T_X = H\W_X/Hnorm;
T_Y = H\W_Y/Hnorm;

%normalize (in H1)
T_X = T_X*sign(-W_X.'*T_X);
T_Y = T_Y*sign(-W_Y.'*T_Y);

T_update_new.X = T_X;
T_update_new.Y = T_Y;


end
```

A.11. The function `computeMinDet`

The function **`computeMinDet`** evaluates the determinant of Jacobian of the transformation $T_{\mathcal{V}}$ on the quadrature nodes and returns the minimal value.

```
function minDetDt = computeMinDet(T_update,precompDV)

height = size(precompDV.V_dX,1);

%DT_V = I + DV
DTa = precompDV.V_dX*T_update.X + ones(height,1);
DTb = precompDV.V_dY*T_update.X;
```

```
DTc = precompDV.V_dX*T_update.Y;
DTd = precompDV.V_dY*T_update.Y + ones(height,1);

detDT = DTa.*DTd-DTb.*DTc;
minDetDt = min(detDT);

end
```

References

[1] R. A. Adams and J. J. F. Fournier. *Sobolev spaces*. Elsevier/Academic Press, Amsterdam, second edition, 2003.

[2] G. Allaire. *Shape optimization by the homogenization method*. Springer-Verlag, New York, 2002.

[3] G. Allaire. *Conception optimale de structures*. Springer-Verlag, Berlin, 2007.

[4] G. Allaire, C. Dapogny, and P. Frey. Topology and geometry optimization of elastic structures by exact deformation of simplicial mesh. *C. R. Math. Acad. Sci. Paris*, 349(17-18):999–1003, 2011.

[5] G. Allaire, C. Dapogny, and P. Frey. Shape optimization with a level set based mesh evolution method. *Comput. Methods Appl. Mech. Engrg.*, 282:22–53, 2014.

[6] G. Allaire, F. Jouve, and A.-M. Toader. A level-set method for shape optimization. *C. R. Math. Acad. Sci. Paris*, 334(12):1125–1130, 2002.

[7] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.*, 194(1):363–393, 2004.

[8] P. F. Antonietti, A. Borzì, and M. Verani. Multigrid shape optimization governed by elliptic PDEs. *SIAM J. Control Optim.*, 51(2):1417–1440, 2013.

[9] F. Ballarin, A. Manzoni, G. Rozza, and S. Salsa. Shape optimization by free-form deformation: existence results and numerical solution for Stokes flows. *J. Sci. Comput.*, 60(3):537–563, 2014.

[10] K. Bandara, F. Cirak, G. Of, O. Steinbach, and J. Zapletal. Boundary element based multiresolution shape optimisation in electrostatics. *J. Comput. Phys.*, 297:584–598, 2015.

[11] D. Begis and R. Glowinski. Application de la méthode des éléments finis à l'approximation d'un problème de domaine optimal. Méthodes de résolution des problèmes approchés. *Appl. Math. Optim.*, 2(2):130–169, 1975/76.

[12] J.-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, 114(2):185–200, 1994.

[13] M. Berggren. A unified discrete-continuous sensitivity analysis method for shape optimization. In *Applied and numerical partial differential equations*, volume 15 of *Comput. Methods Appl. Sci.*, pages 25–39. Springer, New York, 2010.

[14] M. Blumenfeld. The regularity of interface-problems on corner-regions. In *Singularities and constructive methods for their treatment (Oberwolfach, 1983)*, volume 1121 of *Lecture Notes in Math.*, pages 38–54. Springer, Berlin, 1985.

[15] D. Braess. *Finite elements*. Cambridge University Press, Cambridge, third edition, 2007.

[16] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*. Springer, New York, third edition, 2008.

[17] M. Burger and W. Mühlhuber. Iterative regularization of parameter identification problems by sequential quadratic programming methods. *Inverse Problems*, 18(4):943–969, 2002.

[18] Z. Chen, A. Taflove, and V. Backman. Photonic nanojet enhancement of backscattering of light by nanoparticles: a potential novel visible-light ultramicroscopy technique. *Opt. Express*, 12(7):1214–1220, 2004.

[19] D. Chenais and E. Zuazua. Controllability of an elliptic equation and its finite difference approximation by the shape of the domain. *Numer. Math.*, 95(1):63–99, 2003.

[20] P. G. Ciarlet. *The finite element method for elliptic problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

[21] D. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory*. Springer, New York, third edition, 2013.

[22] M. Dambrine and M. Pierre. About stability of equilibrium shapes. *M2AN Math. Model. Numer. Anal.*, 34(4):811–834, 2000.

[23] M. C. Delfour and J.-P. Zolésio. Velocity method and Lagrangian formulation for the computation of the shape Hessian. *SIAM J. Control Optim.*, 29(6):1414–1442, 1991.

[24] M. C. Delfour and J.-P. Zolésio. *Shapes and geometries*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2011.

[25] G. Doğan, P. Morin, R. H. Nochetto, and M. Verani. Discrete gradient flows for shape optimization and applications. *Comput. Methods Appl. Mech. Engrg.*, 196(37-40):3898–3914, 2007.

[26] K. Eppler and H. Harbrecht. Coupling of FEM and BEM in shape optimization. *Numer. Math.*, 104(1):47–68, 2006.

[27] K. Eppler and H. Harbrecht. Efficient treatment of stationary free boundary problems. *Appl. Numer. Math.*, 56(10-11):1326–1339, 2006.

[28] K. Eppler and H. Harbrecht. Shape optimization for free boundary problems—analysis and numerics. In *Constrained optimization and optimal control for partial differential equations*, volume 160 of *Internat. Ser. Numer. Math.*, pages 277–288. Birkhäuser/Springer Basel AG, Basel, 2012.

[29] K. Eppler, H. Harbrecht, and R. Schneider. On convergence in elliptic shape optimization. *SIAM J. Control Optim.*, 46(1):61–83 (electronic), 2007.

[30] L. C. Evans. *Partial differential equations*. American Mathematical Society, Providence, RI, second edition, 2010.

[31] I. Fumagalli, N. Parolini, and M. Verani. Shape optimization for stokes flows: a finite element convergence analysis. *ESAIM: M2AN*, 49(4):921–951, 2015.

[32] S. Funken, D. Praetorius, and P. Wissgott. Efficient implementation of adaptive P1-FEM in Matlab. *Comput. Methods Appl. Math.*, 11(4):460–490, 2011.

[33] G. Galilei and S. Drake. *Discoveries and opinions of Galileo*. Doubleday, New York, 1957. Translated with an introduction and notes by Stillman Drake.

[34] I. M. Gelfand and S. V. Fomin. *Calculus of variations*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.

[35] D. Gilbarg and N. S. Trudinger. *Elliptic partial differential equations of second order*. Springer-Verlag, Berlin, 2001.

[36] J. Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Mémoires présentés par divers savants à l'Académie des sciences de l'Institut de France: Éxtrait. Imprimerie nationale, 1908.

[37] C. Hafner. Boundary methods for optical nano structures. *physica status solidi (b)*, 244(10):3435–3447, 2007.

[38] H. Harbrecht and J. Tausch. On the numerical solution of a shape optimization problem for the heat equation. *SIAM J. Sci. Comput.*, 35(1):A104–A121, 2013.

[39] J. Haslinger and R. A. E. Mäkinen. *Introduction to shape optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.

[40] E. J. Haug, K. K. Choi, and V. Komkov. *Design sensitivity analysis of structural systems*. Academic Press, Inc., Orlando, FL, 1986.

[41] A. Henrot and M. Pierre. *Variation et optimisation de formes*. Springer, Berlin, 2005.

[42] R. Herzog and K. Kunisch. Algorithms for PDE-constrained optimization. *GAMM-Mitt.*, 33(2):163–176, 2010.

[43] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE constraints*. Springer, New York, 2009.

[44] R. Hiptmair and A. Paganini. Shape optimization by pursuing diffeomorphisms. *Comput. Methods Appl. Math.*, 15(3):291–305, 2015.

[45] R. Hiptmair, A. Paganini, and S. Sargheini. Comparison of approximate shape gradients. *BIT*, 55(2):459–485, 2015.

[46] K. Höllig. *Finite element methods with B-splines*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.

[47] K. Höllig and J. Hörner. *Approximation and modeling with B-splines*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013.

[48] J. Jacobsen. As flat as possible. *SIAM Rev.*, 49(3):491–507, 2007.

[49] M.-S. Kim, T. Scharf, S. Mühlig, C. Rockstuhl, and H. P. Herzig. Engineering photonic nanojets. *Opt. Express*, 19(11):10206–10220, 2011.

[50] B. Kiniger and B. Vexler. A priori error estimates for finite element discretizations of a shape optimization problem. *ESAIM Math. Model. Numer. Anal.*, 47(6):1733–1763, 2013.

[51] V. A. Kondrat′ev. Boundary value problems for elliptic equations in domains with conical or angular points. *Trudy Moskov. Mat. Obšč.*, 16:209–292, 1967.

[52] E. Laporte and P. Le Tallec. *Numerical methods in sensitivity analysis and shape optimization*. Birkhäuser Boston, Inc., Boston, MA, 2003.

[53] T. Lassila and G. Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Comput. Methods Appl. Mech. Engrg.*, 199(23-24):1583–1592, 2010.

[54] A. Laurain and K. Sturm. Domain expression of the shape derivative and application to electrical impedance tomography. Technical Report 1863, Weierstrass Institute for Applied Analysis and Stochastics, 2013.

[55] A. Laurain and K. Sturm. Distributed shape derivative via averaged adjoint method and applications. *ESAIM: M2AN*, 2015.

[56] J. Li, J. M. Melenk, B. Wohlmuth, and J. Zou. Optimal a priori estimates for higher order finite elements for elliptic interface problems. *Appl. Numer. Math.*, 60(1-2):19–37, 2010.

[57] J. R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, Ltd., Chichester, 1999.

[58] MATLAB and P. D. E. Toolbox. *Version 8.5.0.197613 (R2015a)*. The MathWorks Inc., Natick, Massachusetts, 2015.

[59] J. M. Melenk and S. Sauter. Convergence analysis for finite element discretizations of the Helmholtz equation with Dirichlet-to-Neumann boundary conditions. *Math. Comp.*, 79(272):1871–1914, 2010.

[60] P. Monk. *Finite element methods for Maxwell's equations*. Oxford University Press, New York, 2003.

[61] P. Morice. Une méthode d'optimisation de forme de domaine. Application à l'écoulement stationnaire à travers une digue poreuse. In *Control theory, numerical methods and computer systems modelling (Internat. Sympos., IRIA LABORIA, Rocquencourt, 1974)*, pages 454–467. Lecture Notes in Econom. and Math. Systems, Vol. 107. Springer, Berlin, 1975.

[62] J. W. Neuberger. *Sobolev gradients and differential equations*. Springer-Verlag, Berlin, second edition, 2010.

[63] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, New York, second edition, 2006.

[64] A. Paganini. Approximate shape gradients for interface problems. In *New Trends in Shape Optimization*, volume 166 of *Internat. Ser. Numer. Math.*, pages 217–227. Springer International Publishing, 2015.

[65] A. Paganini, S. Sargheini, R. Hiptmair, and C. Hafner. Shape optimization of microlenses. *Opt. Express*, 23(10):13099–13107, 2015.

[66] O. Pironneau. *Optimal shape design for elliptic systems*. Springer-Verlag, New York, 1984.

[67] M. Primbs. New stable biorthogonal spline-wavelets on the interval. *Results Math.*, 57(1-2):121–162, 2010.

[68] S. A. Sauter and C. Schwab. *Boundary element methods*. Springer-Verlag, Berlin, 2011.

[69] V. H. Schulz. A Riemannian view on shape optimization. *Found. Comput. Math.*, 14(3):483–501, 2014.

[70] V. H. Schulz, M. Siebenborn, and K. Welker. A novel Steklov-Poincaré type metric for efficient PDE constrained optimization in shape spaces. arXiv:1506.02244v4 [math.OC], 2015.

[71] J. A. Sethian. *Level set methods and fast marching methods.* Cambridge University Press, Cambridge, second edition, 1999.

[72] Y. Shen, L. V. Wang, and J.-T. Shen. Ultralong photonic nanojet formed by a two-layer dielectric microsphere. *Opt. Lett.*, 39(14):4120–4123, 2014.

[73] J. Sokołowski and J.-P. Zolésio. *Introduction to shape optimization.* Springer-Verlag, Berlin, 1992.

[74] M. Souli and J.-P. Zolésio. Shape derivative of discretized problems. *Comput. Methods Appl. Mech. Engrg.*, 108(3-4):187–199, 1993.

[75] R. Udawalpola, E. Wadbro, and M. Berggren. Optimization of a variable mouth acoustic horn. *Internat. J. Numer. Methods Engrg.*, 85(5):591–606, 2011.

[76] K. Urban. *Wavelet methods for elliptic partial differential equations.* Oxford University Press, Oxford, 2009.