Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Shape Optimization for Aerodynamic Problems

Dissertation

zur Erlangung des Grades eines Doktor der Naturwissenschaften
–Dr. rer. nat.–
der Fakultät für Mathematik, Informatik und Naturwissenschaften der Universität Hamburg
vorgelegt von
Sofiya Onyshkevych

GutachterInnen: Prof. Dr. Winnifried Wollner
Prof. Dr. Martin Siebenborn

Datum der Abgabe: 30.07.2025
Datum der Disputation: 23.10.2025

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Declaration on oath

I hereby declare upon oath that I have written the present dissertation independently and have not used further resources and aids than those stated in the dissertation.

Hamburg, den

_____

Sofiya Onyshkevych

Abstract

Shape optimization problems arise in many engineering applications of fluid dynamics, acoustics, and other fields. In such problems, one aims to find a shape that optimizes a certain objective while satisfying a given set of constraints. A persistent challenge is maintaining mesh quality during the optimization process. Poor mesh quality, such as distorted or overlapping elements, can lead to numerical instability and ultimately failure of the algorithm. Hence, it is of great importance to develop a robust method that produces meshes of high quality. Most importantly, one wants to ensure that the solvability of the problem itself is not lost in the process.

I consider the problem of minimizing drag in a flow tunnel, which is a typical shape optimization problem with partial differential equation constraints. In order to increase the set of reachable shapes, an extension operator is used – a mapping from the control variable on the boundary to the deformation field defined over the entire domain. Both linear and nonlinear formulations of the extension equation are investigated and their influence on the solution quality and robustness is assessed. The results demonstrate that appropriate extension strategies can significantly improve mesh quality without compromising the fidelity of the optimization process. It is shown that nonlinear extensions outperform linear ones in preserving element quality under large deformations.

Zusammenfassung

Optimierungsprobleme geometrischer Formen treten in vielen ingenieurwissenschaftlichen Anwendungen der Strömungsmechanik, Akustik und anderer Fachgebiete auf. Ziel solcher Probleme ist es, eine Geometrie zu finden, die ein gegebenes Zielkriterium optimiert und gleichzeitig eine Reihe von Nebenbedingungen erfüllt. Eine zentrale Herausforderung besteht darin, während des Optimierungsprozesses die Qualität des Gitters aufrechtzuerhalten. Eine schlechte Qualität des Gitters – etwa durch verzerrte oder sich überlappende Elemente – kann zu numerischen Instabilitäten und letztlich zum Versagen des Algorithmus führen. Daher ist es von großer Bedeutung, eine robuste Methode zu entwickeln, die Gitter von hoher Qualität erzeugt. Insbesondere muss sichergestellt werden, dass die Lösbarkeit des zugrunde liegenden Problems während der Optimierung nicht verloren geht.

Ich analysiere in dieser Dissertation das Problem der Minimierung des Strömungswiderstands in einem Strömungskanal – ein typisches Optimierungsproblem geometrischer Formen mit Nebenbedingungen in Form partieller Differentialgleichungen. Um die Menge der erreichbaren Formen zu vergrößern, verwende ich einen Erweiterungsoperator – eine Abbildung von der Steuerungsgröße am Rand auf das Verschiebungsfeld im gesamten Gebiet. Dazu untersuche ich sowohl lineare als auch nichtlineare Formulierungen der Erweiterungsgleichung und analysiere deren Einfluss auf die Lösungsgüte und die Robustheit des Verfahrens. Die Ergebnisse zeigen, dass geeignete Erweiterungsstrategien die Qualität des Gitters deutlich verbessern können, ohne die Güte des Optimierungsprozesses zu beeinträchtigen. Insbesondere erweisen sich nichtlineare Erweiterungen bei großen Deformationen als überlegen, da sie die Qualität der Gitterelemente besser erhalten.

List of publications

Published research articles

- Onyshkevych, S., Siebenborn, M. Mesh Quality Preserving Shape Optimization Using Nonlinear Extension Operators. J Optim Theory Appl 189, 291–316 (2021). https://doi.org/10.1007/s10957-021-01837-8

Peer-reviewed conference proceedings

- Onyshkevych, S., Siebenborn, M., Wollner, W. Preserving mesh quality in shape optimization. Proceedings in Applied Mathematics and Mechanics, 24, e202300146 (2024). https://doi.org/10.1002/pamm.202300146

Eigenanteilserklärung

I hereby declare that the results presented in this dissertation are the outcome of my own independent research, unless explicitly stated otherwise.

Ideas, results, or text passages originating from others or from jointly authored publications are clearly identified and appropriately cited.

All chapters of this thesis have been written by me. Chapter 2 and Chapter 3 summarize preliminary concepts and theoretical background, with all sources duly cited. The theoretical results in Chapter 4 are based on several published works, all of which are referenced; theorems and statements not developed by me are explicitly indicated. All numerical results presented in Chapter 5 were generated by me and are based on the ideas published in a joint paper with my supervisor [63] which is explicitly referred in the text of the thesis.

Acknowledgements

This dissertation would not have been possible without the support and encouragement of many people along the way.

To my parents, thank you for giving me the confidence to pursue my studies abroad and for supporting me through every step of that journey. That decision shaped everything that followed, and your trust in me made it possible.

To my sister, for never questioning my path, even when it twisted in unexpected ways. Thank you for standing beside me without question.

I am deeply grateful to Prof. Martin Siebenborn and Prof. Winnifried Wollner for their patience, persistence, for checking in with steady commitment when my pace faltered, and for always keeping the door open.

To Marta and Ira, my closest friends since high school, for being my mental scaffolding during the toughest days. Your late-night calls, humor, and constancy kept me upright more than once.

To the friends I met during my doctoral studies at Universität Hamburg: Eleonora, José, Nicolai, Lars, and Mae, thank you for the thoughtful conversations, the shared curiosity, and the sense of belonging you created.

To my colleagues at Aurubis, thank you for your understanding and patience as I worked toward finishing this dissertation while stepping into a new role. Your flexibility made the final steps possible.

And to my personal project manager, thank you Jonathan for knowing when to apply pressure and when to offer calm. Your balance of discipline and care has been invaluable and your presence made all the difference.

Contents

List of Algorithms

List of Figures

List of Tables

Abbreviations

Acronyms

AD automatic differentiation

a.e. almost everywhere

CAD Computer-aided design

CFD Computational fluid dynamics

FEM Finite Element Method

KKT Karush–Kuhn–Tucker conditions

NS Navier-Stokes

PDE Partial differential equation

UFL Unified Form Language

VI Variational inequality

1. Introduction

1.1. Motivation

Shape optimization is a fundamental problem in applied mathematics and engineering, where the objective is to determine an optimal shape of a domain to minimize or maximize a given functional subject to constraints.

These problems arise naturally in many physical and engineering contexts, where the performance or efficiency of a system depends on its shape. In general terms, a shape optimization problem seeks the best possible shape such that a given cost functional, which may represent energy, drag, compliance, or any other relevant physical quantity, is minimized or maximized under certain constraints. The domain itself becomes the variable in the optimization, and its influence is often indirect, described through the solution of a partial differential equation defined on it.

There is a wide range of problems which can be formulated in this framework from different fields and industries. In fluid dynamics, for instance, the objective might be to minimize drag acting on a solid body immersed in a flow, or to maximize lift in an airfoil configuration. In structural mechanics, one may aim to reduce stress concentrations or optimize stiffness for a given load, leading to more efficient and durable mechanical components. Problems involving thermal conduction may seek shapes that promote uniform temperature distributions or minimize heat loss. Shape optimization is also relevant in wave propagation and acoustics, where the geometry of a cavity or a waveguide affects resonance patterns or the reflection of sound. In electromagnetics, it is used in designing antennas and optical devices, where the shape determines the distribution and direction of electromagnetic fields.

Many of the most compelling applications of shape optimization are found in industrial contexts. In the aerospace industry, for example, the design of aircraft components such as wings, fuselages, or engine nacelles is highly dependent on aerodynamic performance. Optimizing the shape of a wing involves balancing lift and drag, ensuring structural integrity, and complying with manufacturing constraints. Simulations of compressible or incompressible flow, often governed by the Navier–Stokes equations, are used to evaluate performance metrics, and gradient-based optimization methods

iteratively refine the geometry. The same principles apply in rocket design, where nozzle shapes are optimized to maximize thrust efficiency while managing the thermal and mechanical stresses arising from extreme conditions.

In the automotive industry, shape optimization is used to reduce drag in vehicle bodies, thereby improving fuel economy and lowering emissions. Engineers also apply similar techniques to the internal flow design of engine components, aiming to improve flow efficiency or reduce noise and vibration. Structural aspects of vehicle design are also influenced by shape optimization: crash safety standards, for instance, require vehicle frames to deform in a controlled way during impact. This can be formulated as an optimization problem where the objective is to maximize energy absorption while minimizing weight and material use.

In biomedical engineering, it is used for designing implants and prosthetics that conform better to anatomical structures, leading to improved patient outcomes. In structural mechanics, shape optimization aids in designing lightweight and high-strength components for buildings and mechanical structures. In material science, it is applied to optimize the microstructure of composite materials to enhance their mechanical and thermal properties. Additionally, shape optimization plays an essential role in optics and photonics, where it is used to design lenses and waveguides with specific light propagation characteristics. In the field of energy systems, shape optimization is applied to improve the efficiency of heat exchangers and turbine blades, directly impacting energy savings and sustainability.

These examples all follow a similar mathematical structure: a shape-dependent PDE describes the system's behavior, and the optimization process explores geometric variations to achieve a better performance with respect to a given objective. This approach enables engineers and scientists to move beyond empirical adjustments and instead make informed decisions based on physical models and numerical simulations.

However, the methodology is not without challenges. From mathematical point of view, solving a shape optimization problem typically involves repeated evaluations of expensive numerical simulations on evolving geometries, leading to high computational costs. Further, the problems are frequently nonlinear and non-convex, which complicates the search for global optima. Even in case the problem is well posed, there is often no guarantee of converging to global solution. Representing and updating the geometry in a flexible and efficient way, whether through mesh deformation, level set methods, or parameterized CAD models, is itself a nontrivial task. Additionally, the optimal shapes must not only be high-performing in simulation but also manufacturable and robust with respect to uncertainties in loading conditions or material properties.

The mathematical formulation of shape optimization typically involves a physical setting often given by PDEs where the domain boundary itself is considered as a

design variable. Furthermore, the objective functional, which depends on the domain, also depends on the solution operator of the PDEs.

In this thesis I consider only the subset of the boundary to be deformed, representing an object inside a flow tunnel, which we will call an obstacle. The fluid flow in the tunnel is governed by Navier-Stokes equations, which is the physical setting of the problem. The surface of the obstacle is to be optimized for a given flow of the fluid.

This problem is well studied and its solution is investigated in [67, 68]. It is known, that the optimal configuration is a shape with two tips formed. However, this optimal setting is known to be not easily reached by solvers in case of large deformations. The main reason for this is mesh degradation or even mesh degeneration during iterative design updates. The obvious way to overcome this issue would be mesh refinement. In particular, one could refine specific areas of the mesh as the mesh quality gets worse. However, remeshing at every solution step leads to even higher computational complexity for already computationally challenging structure of the nonlinear system of the equations. Therefore, it is challenging to model mesh deformation in a way that is both not too computationally demanding and flexible enough to be applied to a range of problems without too much parameter-tuning.

To model mesh deformation in classical shape optimization, two predominant approaches exist for this problem: boundary variation methods and domain deformation methods. The former relies on the Hadamard-Zolésio structure theorem to compute shape derivatives by considering variations of the domain boundary [91]. This method is particularly useful if the changes in shape are small. The latter approach treats shape transformations as mappings of the entire domain, effectively translating the optimization problem into an optimal control problem in function spaces. The method of mappings, where an admissible transformation is applied to a reference domain, has been widely used for problems involving large deformation.

Therefore, in this thesis I follow this approach and reformulate the problem as an optimization over the set of transformations. In this case the problem is solved for the fixed reference domain for a sequence of transformations which depend on the deformation field.

In this framework, there is still an investigation required on how to effectively describe the mapping that propagated the deformation from the boundary to the whole domain. Traditionally, one uses linear-elasticity-based extension models. However, they can fail to preserve mesh quality over successive deformations, leading to element degeneration and numerical instabilities [38]. In addition to maintaining mesh quality, robust optimization algorithms must also handle constraints such as volume preservation, symmetry conditions, and manufacturing limitations, ensuring practical feasibility of the optimized designs.

To address this, there were proposed nonlinear extension operators that better accommodate large deformations by incorporating advection-like terms into the deformation process. The nonlinear extension ensures better mesh uniformity and prevents excessive element distortion, making this approach particularly suitable for problems in fluid dynamics and aerodynamic. Therefore, in this thesis I focus on the study of nonlinear equations to model deformations and show the effects of extension equation parameters on mesh quality.

1.2. General Problem Statement

In this thesis, I consider a class of shape optimization problems arising in fluid dynamics, in particular the minimization of drag acting on an obstacle placed in a flow tunnel governed by the Navier–Stokes equations. The surface of the obstacle, which represents the design boundary, is to be optimized while keeping the outer flow domain fixed. This problem is written in the following form:

$$\begin{aligned} \min_{\Omega \in G_{\text{adm}}} \quad & j(y, \Omega) \\ \text{s.t.} \quad & E(y, \Omega) = 0 \end{aligned} \tag{1.1}$$

In (1.1) j is a shape functional depending on a state variable y and the shape of the domain Ω. Thereby, y is a solution to the PDE E, which constraints the problem and itself depends on Ω. G_{adm} is a set of admissible shapes, i.e. shapes that satisfy constraints. The boundary of Ω is itself an optimization variable.

One of the main questions is an appropriate choice of the set of admissible shapes G_{adm}, in which optimization takes place. The existence theory for (1.1) is quite involved and complex. It is known that, in general, it admits a minimizer only if some geometrical or topological restrictions on the shapes are enforced.

This problem is discussed in more detail for a particular case of Navier-Stokes flow in Section 4.2.

Many approaches can be used to solve such problems. In this work, I use the method of mappings (see Section 3.3) to obtain an optimal control problem on a fixed reference domain. First, the optimization problem (1.1) is reformulated as an optimization problem over a set of admissible transformations $\mathcal{F}_{\text{adm}}$ as follows

$$\begin{aligned} \min_{F \in \mathcal{F}_{\text{adm}}} \quad & j(y, F(\Omega)) \\ \text{s.t.} \quad & E(y, F(\Omega)) = 0. \end{aligned} \tag{1.2}$$

Starting with some fixed reference domain Ω and optimize using transformations $F(\Omega)$ yet without explicitly performing mesh deformations. Further, the choice of the set of admissible shapes is now a choice of spaces in which we defined the deformation from reference to optimal domain: $G_{\text{adm}} := \{F(\Omega) \colon F \in \mathcal{F}_{\text{adm}}\}$

For more details about method of mappings please refer to [50, 80, 17].

1.3. Contributions

This thesis makes several contributions to the field of shape optimization, particularly in the context of stationary Navier–Stokes flow.
Firstly, it provides a detailed formulation of the optimality system for shape optimization problem in stationary Navier-Stokes flow, incorporating nonlinear extension operators for mesh deformation. The influence of different parameters in the nonlinear extension equation on the solution process is analyzed.
Secondly, the thesis presents a comparative analysis of linear and nonlinear extension strategies, highlighting their impact on mesh quality and solver performance. This analysis demonstrates the advantages of nonlinear extensions in handling complex geometries and large deformations.
Thirdly, the thesis includes a numerical implementation using the FEniCS framework, with extensive simulations that validate the effectiveness of the proposed approach. The implementation showcases the practical applicability of the theoretical contributions and provides insights into the computational aspects of shape optimization. Also advantages and drawbacks of the software are discussed.
Lastly, the thesis explores the influence of extension parameters and geometric constraints on optimization outcomes, offering a comprehensive study of how these factors affect the performance and results of shape optimization algorithms. Overall, the contributions of this thesis advance the understanding and application of shape optimization techniques in various engineering and scientific domains.

1.4. Thesis Structure

The outline of this thesis is as follows.

Chapter 2 reviews the literature on shape optimization, including classical methods, applications, and recent advances.

Chapter 3 introduces the mathematical background of shape optimization, including key concepts such as shape derivatives, domain variations, and the formulation of optimization problems constrained by partial differential equations.

I begin by recalling fundamental notions of differentiability and the basics of shape calculus. This is followed by an overview of the underlying physical model of fluid dynamics, its properties, and a brief discussion of the well-posedness of the governing equations. Next, a general constrained optimization problem is formulated and necessary and sufficient optimality conditions are presented. I introduce the control-to-state mapping, derive the adjoint formulation, and describe the Lagrangian approach. The chapter concludes with a discussion of discretization techniques, including finite element methods, and classical optimization algorithms such as gradient-based and Newton-type methods.

Chapter 4 addresses the investigation of a shape optimization problem governed by Navier–Stokes flow. I begin by deriving the weak formulation of the problem, which includes the flow model, the equations governing mesh deformation, and the geometric constraints. I then derive the optimality system and reformulate the problem in the reference domain.

I introduce the deformation operator S which maps the boundary control to the deformation field in the domain and explain the possible approaches to model mesh deformation and choices of this operator. Futher, some existence results for cases when S is linear and nonlinear conclude this chapter.

In Chapter 5 numerical realizations the the problem are discussed. I first introduce the software we used and different approaches how one could tackle the problem using it. It is explained why I chose the particular software and what limitations are coming with this decision. Next, the basic solution algorithm is introduced and its improvement based on decoupling the system is proposed. In final sections of this chapter I present numerical examples for different cases of extension operator, meshes and starting geometries and compare linear and nonlinear extension strategies.

Finally, Chapter 6 concludes the thesis by summarizing the main findings and outlining possible directions for future research.

2. Literature Review

In this chapter an overview of the research conducted in the areas of optimal design and shape optimization is presented. It is a brief look at the history of research related to a shape optimization problem of domain variation type as well as popular underlying physical models and industrial applications with a focus on fluid dynamic applications.

In addition, in this section, commonly used methodologies and the associated challenges are highlighted, and explanations for the choice of certain approaches are provided.

Design and specifically shape optimization are at the core of many engineering applications ranging from optimizing musical instruments and vehicles to electrical devices.

In acoustics, shape optimization is used to improve the design of acoustic horns for better impedance matching with the surrounding air [7, 73, 85], to optimize loudspeaker shapes for desired sound directivity [24], to control acoustic pressure in specific spatial regions [36], and to design brasswind instruments with enhanced performance [47].

In the context of fluid dynamics, shape optimization under steady Stokes flow is widely used in the design of microchannel structures and biomedical devices such as DNA chips [88], blood pumps [18, 20, 90], dielectrophoretic devices [51], and microfluidic nanoliter mixers [22].

In aeronautics and marine engineering, shape optimization plays a critical role in designing aircraft and spacecraft components [45, 2], as well as in the optimization of hull forms and control surfaces for marine vehicles [21, 43, 87, 31].

Unlike geometric shape optimization, where the topology of the domain remains fixed, structural shape optimization seeks the optimal form of a domain whose topology is not known a priori [62]. This distinguishes it from problems in classical geometry processing, as the optimization is formulated over domains subject to physical and structural constraints.

For comprehensive reviews of structural topology optimization, please refer to [10, 78, 23, 62, 9, 72]. Among notable contributions, the homogenization method proposed in [11] interprets topology optimization as a problem of optimal material distribution. Another widely adopted approach is the density-based method, introduced by Yang and Chuang [89], where the material density is treated as a continuous design variable. Moreover, black-and-white topology optimization methods, particularly for elasticity and multiphysics problems, have been extensively studied in [12].

The foundations of shape optimization for aerodynamic flows can be traced back to early 20th-century pioneers in fluid dynamics, such as Ludwig Prandtl, whose boundary layer theory (1904) and Theodore von Kármán, known for his work on turbulence in the 1930s, established essential conceptual frameworks for analyzing fluid–structure interactions. These theoretical advances coincided with significant experimental developments. One notable example is the emergence of wind tunnels in the 1930s, which enabled empirical testing of aerodynamic shapes, particularly in the context of aircraft and automotive design. The post–World War II jet age and the advent of supersonic flight in the 1950s introduced new challenges in aerodynamic optimization, as traditional design methods proved inadequate for handling high-speed flows and the thermal stresses encountered in missile and rocket development [28].

In a classical paper, Pironneau demonstrated that, for a three-dimensional body in a low Reynolds number flow, the unit-volume body with the smallest drag, obtained using variational methods in optimal control, has uniform vorticity [68]. The optimal shape is symmetric with respect to the axis aligned with the flow direction and resembles a rugby ball, featuring conical front and rear ends with an angle of 120°.

This study was later extended to higher Reynolds numbers in [67], where the authors considered laminar flow governed by the Navier–Stokes equations. These theoretical results for Stokes flow were subsequently confirmed numerically by Bourot [15] using quadratic minimization techniques, and the drag of the optimized body was computed. Following the arguments in [68], the problem of determining the optimal profile of a cylinder placed perpendicular to a uniform Stokes flow at low Reynolds numbers was addressed in [71].

When modeling shape optimization problems, it is often necessary to exclude trivial or degenerate solutions. A commonly imposed constraint is the preservation of the obstacle's volume, as employed in [67].

Concurrently, Hicks and Henne developed parametric airfoil shape parameterization methods that became industry standards, enabling systematic exploration of design spaces through control point adjustments [70]. The 1980s witnessed the rise of computational fluid dynamics capabilities, with Jameson pioneering adjoint-based optimization techniques that improved gradient calculation efficiency for Navier-

Stokes flows.

Further, in [76] a smoothing method for the surface mesh was used to satisfy the constant volume condition and the biharmonic equation was chosen as mesh relocation technique. In chemistry and colloidal science applications i.e drug delivery, biomimetic microrobots, one often needs to fix a particle surface area. This demonstrates the necessity to consider a fixed surface area as another important geometrical constraint.
By changing the geometrical constraint from Pironneau's fixed volume to fixed surface area in [59] was computed the 'alternative' optimal shape that has a surface vorticity proportional to the mean surface curvature. This optimal shape was shown to be almost twice as slender as the fixed-volume shape with the front and rear ends tangent to a cone of semi-angle 30.8°. In studies of the shape optimization for fluid dynamical applications, shape optimization algorithms to minimize the drag on an object under a constant volume condition have also been studied in [58, 38, 77].

While shape optimization has been actively researched in numerical methods and simulations, the theoretical foundations of the field have also been rigorously developed, particularly in areas such as elliptic PDEs, and structural mechanics. In the context of fluid–structure interaction and unsteady flow problems, however, the theoretical analysis remains quite challenging. Notable contributions in this direction include the work of Raymond and Vanninathan [69], who studied the existence and regularity of solutions to an unsteady FSI problem. Furthermore, Haubner et al. [39] investigated the continuity and Fréchet differentiability of solutions to an unsteady Navier–Stokes–Lamé system.

The problem of minimizing drag in Stokes flow using the adjoint variable method has been studied extensively in a series of foundational works [68, 15, 4], which laid the groundwork for modern shape optimization techniques. While much of the research in the field has focused on numerical simulations and computational strategies, several monographs provide a comprehensive theoretical background and survey of available methods [56, 37, 58, 81, 4]. Multiple approaches have been developed to address shape optimization problems, each with its own mathematical structure, computational requirements, and applicability to specific problem classes.

Domain transformation methods rely on mapping the physical domain onto a fixed reference domain via a family of transformations, which are updated throughout the optimization process. This allows for efficient treatment of complex geometries and constraints while simplifying computations and boundary conditions. This approach follows the framework originally introduced by Murat and Simon [60], in which the computational domain remains fixed and the design is represented through a deformation mapping from a reference domain. This reformulation naturally lends itself to the application of optimal control techniques.

The Hadamard–Zolésio method is one of the most classical tools in shape optimization. It is based on the concept of shape derivatives, which describe how small perturbations of the domain boundary affect a given objective functional. A key result in this theory is the Hadamard structure theorem, which shows that the shape derivative depends only on the normal component of the boundary variation. This insight greatly simplifies the analysis and computation of shape gradients, particularly in boundary-sensitive problems such as drag reduction and thermal optimization. In cases involving PDE constraints, the Eulerian derivative of the objective functional typically depends on the shape derivative of the state, which can be expressed in terms of an adjoint variable [56].

An alternative formulation is the method of mappings, also known as the perturbation of identity method. Here, the domain is parameterized by a bi-Lipschitz homeomorphism from a reference configuration, transforming the shape optimization problem into an optimal control problem in Banach spaces. Shape sensitivities can be computed through adjoint equations or via direct sensitivity analysis. Interestingly, Hadamard–Zolésio calculus can be rigorously derived from this mapping-based approach through integration by parts, thereby unifying the two perspectives [38, 17].

In contrast to these explicit deformation methods, level-set approaches represent the domain implicitly as the zero level of a scalar function defined in a higher-dimensional space. The evolution of the shape is governed by Hamilton–Jacobi-type equations, which allow for natural handling of topological changes such as merging and splitting of domains. This makes level-set methods well suited for problems where the optimal domain topology is unknown in advance [19]. Parametric methods, on the other hand, describe the domain explicitly via a finite number of parameters, such as control points of Bézier curves or splines. These methods provide high geometric fidelity and computational efficiency but are less flexible in dealing with topology changes.

Phase-field methods provide another powerful approach by representing the domain using a diffuse interface model. The transition between material and void is captured by a smooth phase-field function, making these methods particularly effective for topology optimization problems. They provide a stable numerical framework and are well suited to problems requiring smooth transitions and regularized formulations [14].

3. Theoretical Background

3.1. Preliminaries

I first set up the environment of the discussion: Lipschitz domains, some basic functional spaces, and a common notation for differentiation. The definitions and results in this section are primarily based on the monographs [41, 58, 27].

I start by formally defining Lipschitz continuity and Lipschitz domains. Lipschitz domains are commonly considered in shape optimization due to their well-behaved geometric and analytical properties. In particular perturbations of Lipschitz domains remain Lipschitz, preserving the structure of the problem.

In the following sections, I will consider shape optimization problems constrained by PDEs, such as the Laplace equation, the Navier–Stokes equations, and the equations of linear elasticity. The existence and stability of solutions under domain perturbations typically rely on certain regularity assumptions, such as Lipschitz continuity of the domain. Without such regularity, fundamental tools from PDE theory such as well-posedness of results and Sobolev space embeddings may no longer apply.

Definition 3.1.1. *The function φ is **Lipschitz continuous**, meaning there exists a constant $L > 0$ such that for all $x, y \in V$,*

$$|\varphi(x) - \varphi(y)| \leq L\|x - y\|_2,$$

where $\|\cdot\|_2$ denotes Euclidean norm on $\mathbb{R}^n$.

*A domain $\Omega \subset \mathbb{R}^n$ is called a **Lipschitz domain** if its boundary $\partial\Omega$ can be locally represented as a graph of a Lipschitz continuous function.*

3.1.1. Lebesgue and Integer Order Sobolev Spaces

Let $\Omega \subset \mathbb{R}^n$. I denote by $L^p(\Omega), 1 < p < \infty$ the space of absolutely integrable with p-th power real functions on Ω endowed with the norm

$$\|v\|_{L^p}(\Omega) = \left(\int_\Omega |v(x)|^p \, dx \right)^{\frac{1}{p}}$$

Further, I denote by $L^\infty(\Omega)$ the space of essentially bounded real functions with the norm

$$\|v\|_{L^\infty(\Omega)} = \operatorname*{ess\,sup}_\Omega (|v(x)|).$$

Definition 3.1.2. *The **Sobolev space** of index (k, p), where $1 \leq p < \infty$ and $k \in \mathbb{N}$ is defined by*

$$W^{k,p}(\Omega) \overset{def}{=} \{v \in L^p(\Omega) : D^\alpha v \in L^p(\Omega) \quad \forall |\alpha| \leq k\}$$

with a norm

$$\|v\|_{k,p,\Omega} \overset{def}{=} \left(\sum_{|\alpha| \leq k} \|D^\alpha v\|^p_{L^p(\Omega)} \right)^{\frac{1}{p}},$$

where D^α is denoting the α-th weak partial derivative.

When $p = 2, W^{k,2}(\Omega) = H^k(\Omega)$ is a Hilbert space with the inner product

$$(u, v)_{H^k(\Omega)} = \sum_{|\alpha| \leq k} (D^\alpha u, D^\alpha v)_{L^2(\Omega)}$$

and the corresponding norm $\|v\|_{k,\Omega} := \|v\|_{k,2,\Omega}$.

Theorem 3.1.3 (Sobolev embedding)**.** *Let $\Omega \subset \mathbb{R}^N$ be an open bounded set with Lipschitz continuous boundary. Then the following continuous embeddings hold:*

1. *if $p < N, W^{1,p}(\Omega) \hookrightarrow L^{p^*}(\Omega)$, for $\frac{1}{p^*} = \frac{1}{p} - \frac{1}{N}$,*
2. *if $p = N, W^{1,p}(\Omega) \hookrightarrow L^q(\Omega)$, for $1 \leq q < +\infty$,*
3. *if $p > N, W^{1,p}(\Omega) \hookrightarrow C^{0,1-\frac{N}{p}}(\overline{\Omega})$.*

3.1.2. Gâteaux- and Fréchet Differentiability

Let X, Y be Banach spaces, $U \subset X$ be an open nonempty set. Then for an operator $F : U \to Y$ I define the following derivatives:

Definition 3.1.4. *The **directional derivative** of F at $u \in U$ in the direction $h \in X$ is given by:*

$$dF(u;h) = \lim_{\tau \to 0^+} \frac{F(u+\tau h) - F(u)}{\tau}, \tag{3.3}$$

provided the limit exists.

Definition 3.1.5. *The function F is said to be **Gâteaux differentiable** at $u \in U$ if the directional derivative $dF(u;h)$ exists for all $h \in X$ and is bounded and linear, i.e.*

$$F'(u) \in \mathcal{L}(X, Y) \quad for \quad F'(u) : h \mapsto dF(u;h).$$

Definition 3.1.6. *The function F is said to be **Fréchet differentiable** at $u \in U$ if there exists a bounded linear operator $A \in \mathcal{L}(X, Y)$ such that:*

$$\lim_{\|h\|_X \to 0} \frac{\|F(u+h) - F(u) - Ah\|_Y}{\|h\|_X} = 0. \tag{3.4}$$

*In this case, A is called the **Fréchet derivative** of F at u, denoted by $F'(u)$.*

I mention some further properties of Fréchet derivative which are going to be used in the further sections.

Let $F : X_1 \times X_2 \to Y$ be a mapping between Banach spaces. For a fixed $u_2 \in X_2$, the **partial Fréchet derivative** of F with respect to u_1 at (u_1, u_2) is given by:

$$D_{u_1} F(u_1, u_2) \in \mathcal{L}(X_1, Y), \tag{3.5}$$

provided the limit

$$\lim_{\|h_1\|_{X_1} \to 0} \frac{\|F(u_1 + h_1, u_2) - F(u_1, u_2) - D_{u_1} F(u_1, u_2) h_1\|_Y}{\|h_1\|_{X_1}} = 0 \tag{3.6}$$

exists. A similar definition holds for the partial derivative $D_{u_2} F(u_1, u_2)$ with respect to u_2. Then, it holds

$$F'(u_1, u_2)(h_1, h_2) = D_{u_1} F(u_1, u_2) h_1 + D_{u_2} F(u_1, u_2) h_2, \tag{3.7}$$

for all $(h_1, h_2) \in X_1 \times X_2$.

Let $F : X \to Y$ and $G : Y \to Z$ be Fréchet differentiable operators between Banach spaces. Then the composition $G \circ F : X \to Z$ is also Fréchet differentiable, and its derivative satisfies the **chain rule**:

$$(G \circ F)'(u) = G'(F(u)) \circ F'(u). \tag{3.8}$$

Here, $G'(F(u)) \in \mathcal{L}(Y, Z)$ and $F'(u) \in \mathcal{L}(X, Y)$, so their composition is a bounded linear operator in $\mathcal{L}(X, Z)$.

3.1.3. General Optimization Problem

I consider a general optimization problem with PDE constraints as given in [41]:

$$\begin{aligned} &\min J(y, u), \\ &\text{s. t. } e(y, u) = 0. \end{aligned} \tag{3.9}$$

I assume that $J : Y \times U \to \mathbb{R}$ and $e : Y \times U \to Z$ are continuously Fréchet differentiable and that the following condition is satisfied:

$$e_y(\bar{y}, \bar{u}) \in \mathcal{L}(Y, Z) \quad \text{is a bijection.} \tag{3.10}$$

From implicit function theorem it follows that there exists (locally) a unique solution to the state equation $e(y, u) = 0$ in the neighbourhood of $(\bar{y}, \bar{u})$ and that the solution operator is continuously Fréchet differentiable (see e.g. [25]).

I first define the Lagrange function $L : Y \times U \times Z^* \to \mathbb{R}$.

Definition 3.1.7. ***The Lagrangian*** *associated with the general optimization problem given by (3.9) is defined as*

$$L(y, u, p) = J(y, u) + \langle p, e(y, u) \rangle_{Z^*, Z}, \tag{3.11}$$

where $p \in Z^*$ is a ***Lagrange multiplier****, and $\langle \cdot, \cdot \rangle_{Z^*, Z}$ denotes the duality pairing between Z^* and Z.*

This formulation allows one to treat the objective and constraint in a unified framework. The central importance of the Lagrangian lies in the derivation of first-order necessary conditions for optimality, extending classical Karush–Kuhn–Tucker (KKT) theory to infinite-dimensional settings.

In the present PDE-constrained optimization setting, the Lagrangian framework is essential for deriving adjoint equations as described in the next subsection.

3.1.4. Optimality Conditions

By differentiating the Lagrangian with respect to the state variable and enforcing stationarity, one obtains the adjoint problem, which provides an efficient way to compute derivatives of the cost functional with respect to the design or control variables. This adjoint-based approach decouples the derivative computation from the dimensionality of the state space and is fundamental in large-scale applications [41, 83].

Theorem 3.1.8. *Let $(\bar{y}, \bar{u})$ be a local minimizer of the optimization problem (3.9). Assume that all involved mappings are Fréchet differentiable and that the partial derivative $e_y(\bar{y}, \bar{u}) : Y \to Z$ is surjective. Then there exists a unique adjoint variable $p \in Z^*$ such that the first-order necessary optimality conditions are given by the system*

$$\begin{aligned} e(\bar{y}, \bar{u}) &= 0, \\ L_y(\bar{y}, \bar{u}, p) &= 0, \\ L_u(\bar{y}, \bar{u}, p) &= 0. \end{aligned} \tag{3.12}$$

Proof. Because $(\bar{y}, \bar{u})$ is a local minimizer and all mappings are Fréchet differentiable, the first-order necessary condition for optimality implies

$$L_y(\bar{y}, \bar{u}, p)w = 0, \qquad L_u(\bar{y}, \bar{u}, p)h = 0$$

for all $w \in Y$ and $h \in U$. Differentiating the Lagrangian $L(y, u, p)$ with respect to y in the direction w yields

$$\begin{aligned} L_y(y, u, p)w &= J_y(y, u)w + \langle p, e_y(y, u)w \rangle_{Z^*,Z} \\ &= J_y(y, u)w + \langle e_y(y, u)^* p, w \rangle_{Y^*,Y}. \end{aligned} \tag{3.13}$$

The stationarity condition $L_y(\bar{y}, \bar{u}, p) = 0$ then leads to the adjoint equation

$$e_y(\bar{y}, \bar{u})^* p = J_y(\bar{y}, \bar{u}).$$

Surjectivity of $e_y(\bar{y}, \bar{u})$ guarantees that this equation admits a unique solution $p \in Z^*$, see [41] or [83].

Analogously, differentiating $L(y, u, p)$ with respect to u in the direction h leads to

$$\begin{aligned} L_u(y, u, p)h &= J_u(y, u)h + \langle p, e_u(y, u)h \rangle_{Z^*,Z} \\ &= J_u(y, u)h + \langle e_u(y, u)^* p, h \rangle_{U^*,U}. \end{aligned} \tag{3.14}$$

The stationarity condition $L_u(\bar{y}, \bar{u}, p) = 0$ then yields

$$e_u(\bar{y}, \bar{u})^* p = J_u(\bar{y}, \bar{u}).$$

Together with the state equation $e(\bar{y}, \bar{u}) = 0$, this completes the system (3.12). □

The Lagrangian formulation is particularly useful for nonlinear PDE-constrained optimization problems, where the adjoint equation is not straightforward to identify directly, see [27] for examples.

3.1.5. Reduced Formulation

An alternative and widely used method is the reduced formulation, which avoids introducing Lagrange multipliers explicitly. Assume that for every u in a neighborhood of $\bar{u}$ the state equation

$$e(y,u) = 0$$

admits a unique solution $y(u)$, and that $e_y(y(u),u)$ is continuously invertible. By the implicit function theorem this defines the **control-to-state map**

$$G : U \to Y, \qquad u \mapsto y(u) = G(u), \tag{3.15}$$

see [41, Sec. 1.6] or [83, Thm. 3.8].

The optimization problem (3.9) then reduces to

$$\min_{u\in U} f(u) := J(G(u),u). \tag{3.16}$$

Differentiating the state equation with respect to u yields

$$e_y(y(\bar{u}),\bar{u})y'(\bar{u})h + e_u(y(\bar{u}),\bar{u})h = 0,$$

where $y'(u)h$ denotes the derivative of G. If $\bar{u}$ is a local minimizer of (3.16), then the first-order necessary optimality condition gives

$$f'(\bar{u})h = J_y(y(\bar{u}),\bar{u})y'(\bar{u})h + J_u(y(\bar{u}),\bar{u})h = 0 \qquad \forall h \in U, \tag{3.17}$$

see [27].

Definition 3.1.9. *An element $p \in Z^*$ is called the **adjoint state** corresponding to $\bar{u}$ if it satisfies the adjoint equation*

$$e_y(y(\bar{u}),\bar{u})^* p = J_y(y(\bar{u}),\bar{u}). \tag{3.18}$$

Using this definition and the linearized state equation, the derivative representation (3.17) implies the following optimality system.

Theorem 3.1.10. *Let $\bar{u}$ be a local minimizer of the reduced problem (3.16), and let $y(\bar{u})$ be the associated state. If (3.18) admits a solution $p \in Z^*$, then the following conditions hold:*

$$\begin{aligned} e(y(\bar{u}),\bar{u}) &= 0, \\ e_y(y(\bar{u}),\bar{u})^* p &= J_y(y(\bar{u}),\bar{u}), \\ e_u(y(\bar{u}),\bar{u})^* p &= J_u(y(\bar{u}),\bar{u}). \end{aligned} \tag{3.19}$$

*System (3.19) is called the **optimality system** for $\bar{u}$.*

For complete proofs and further details, see [41, 83, 27]. The system (3.19) is analogous to the system derived earlier using Lagrangian given by (3.12). The reduced approach is implemented as a part of dolfin-adjoint package, which I use in some examples in Chapter 5.

3.2. Shape Optimization Problems

In this section, I give a brief overview of the basic notions and methods developed for shape optimization problems [52, 56, 81].

Informally, I consider the shape optimization problem which reads as follows: find a domain $\Omega \in \mathbb{R}^d$ that minimizes an objective function $\bar{J}$, where Ω is restricted to a set of admissible domains G_{adm} which may model geometric constraints. Formally, this problem can be written in the following way.

Problem 3.2.1. *Let $d \in \{2,3\}$ and $G_{adm} \subset \left\{\Omega \subset \mathbb{R}^d \mid \Omega \text{ is in a bounded domain}\right\}$ be a set of admissible domains. Furthermore, for arbitrary $\Omega \in G_{adm}$ let $Y(\Omega)$ and $Z(\Omega)$ be Banach spaces of functions defined on Ω. Let $\bar{J}$ be the objective function*

$$\bar{J} : \{(\tilde{y},\Omega) \mid \tilde{y} \in Y(\Omega), \Omega \in G_{adm}\} \to \mathbb{R}, \tag{3.20}$$

and $\bar{E}$ be an operator between the sets of admissible function spaces

$$\bar{E} : \{(\tilde{y},\Omega) \mid \tilde{y} \in Y(\Omega), \Omega \in G_{adm}\} \to \{\tilde{z} \mid \tilde{z} \in Z(\Omega)\}. \tag{3.21}$$

Then

$$min\ \bar{J}(\tilde{y},\Omega)\ s.t.\ \bar{E}(\tilde{y},\Omega) = 0,\ \Omega \in G_{adm} \tag{3.22}$$

*is called the **shape optimization problem**.*

In Problem 3.2.1, the expression $\bar{E}(\tilde{y},\Omega) = 0$ typically means that the state $\tilde{y}$ solves a partial or ordinary differential equation with the operator $\bar{E}$ including initial and boundary conditions.

In order to proceed with the analytical investigation of Problem 3.2.1 new theoretical concepts are needed, in particular the shape derivative and the reference domain.

3.3. Method of Mappings

The method of mappings for shape optimization problems was originally proposed in [79] and [60].

There are two common approaches to define shape derivatives in the context of the reference domain transformation:

- the velocity (speed) method [91];
- the perturbation of identity method ([60, 52] and references therein).

These to methods are shown to be equivalent in certain sense [52, 56].

In what follows, I give a general idea of the reference domain transformation approach and the perturbation of the identity method which I utilize later on to analyze the shape optimization problem for Navier-Stokes flow given in (4.59).

Firstly, I fix a bounded domain $\Omega_{\text{ref}} \in G_{\text{adm}}$, called the *reference domain* [52]. I assume that an admissible domain $\Omega \in G_{\text{adm}}$ can be characterized by a suitable associated transformation $\tau \in T_{\text{ad}} \subset T(\Omega_{\text{ref}})$ s.t. $\tau(\Omega_{\text{ref}}) = \Omega$. There

$$T(\Omega_{\text{ref}}) = \left\{ \tau : \mathbb{R}^d \to \mathbb{R}^d \mid (\tau - \text{id}) \in S \right\} \tag{3.23}$$

is an affine space of transformations defined on Ω_{ref} and S is a suitable Banach space. Further, the transformations $\tau \in T(\Omega_{\text{ref}})$ are restricted to a set T_{ad} of bi-Lipschitz functions where $(\tau - \text{id}) \in D \subset S$ and D is open in S. To link this approach to the set of admissible domains it is assumed that

$$T_{\text{ad}} = \{ \tau \in T(\Omega_{\text{ref}}) \mid \tau(\Omega_{\text{ref}}) \in G_{\text{adm}} \}. \tag{3.24}$$

Hence, Problem 3.2.1 takes the following form:

Problem 3.3.1. *Let $d \in \{2,3\}$ and $T_{ad} \subset T\left(\Omega_{ref}\right)$ be a set of admissible transformations of a reference domain $\Omega_{ref} \in \mathcal{G}_{adm}$. Furthermore, for all $\tau \in T_{ad}$ let $Y(\tau)$ and $Z(\tau)$ be Banach spaces of functions defined on $\tau\left(\Omega_{ref}\right)$. Let $\bar{J}$ be the objective function*

$$\bar{J} : \{ (\tilde{y}, \tau) \mid \tilde{y} \in Y(\tau), \tau \in T_{ad} \} \to \mathbb{R}, \tag{3.25}$$

and $\bar{E}$ be an operator between the sets of admissible function spaces

$$\bar{E} : \{(\tilde{y}, \tau) \mid \tilde{y} \in Y(\tau), \tau \in T_{ad}\} \to \{\tilde{z} \mid \tilde{z} \in Z(\tau)\}. \tag{3.26}$$

Then the problem

$$min\ \bar{J}(\tilde{y}, \tau)\ s.t.\ \bar{E}(\tilde{y}, \tau) = 0,\ \tau \in T_{ad} \tag{3.27}$$

is called the ***shape optimization problem with transformations.***

In order to make this approach well-defined, the following is also required. Let $\tau_1, \tau_2 \in T_{\text{ad}}$ with $\tau_1(\Omega_{\text{ref}}) = \tau_2(\Omega_{\text{ref}}) =: \hat{\Omega}$. Then for the solutions y_1 and y_2 of $\bar{E}(y_1, \tau_1) = 0$ and $\bar{E}(y_2, \tau_2) = 0$ it holds $y_1 \circ \tau_1^{-1} = y_2 \circ \tau_2^{-1} =: \hat{y}$.

For the complete reference on the construction of a suitable transformation space $T(\Omega_{\text{ref}})$ as well as the conservation of boundary smoothness and regularity of transformed functions I refer to [52] and references therein.

Finally, one can reformulate the optimization problem on a fixed domain Ω_{ref} by transforming the underlying PDE onto the domain Ω_{ref} [52].

Suppose the state equation is given in a variational form

$$\langle \bar{E}(\tilde{y}, \tau), \tilde{\phi} \rangle_{Z(\tau), Z(\tau)^*} = 0 \quad \forall\, \tilde{\phi} \in Z(\tau)^*, \tag{3.28}$$

where $Z(\tau)^*$ is a suitable space of test functions defined on $\Omega = \tau(\Omega_{\text{ref}})$ for $\tau \in T_{\text{ad}}$. I show how the state equation can be transformed in this setting to obtain a variational formulation with state and test functions defined on a fixed domain.

By defining the pullback $\phi(x) := \left(\tilde{\phi} \circ \tau\right)(x),\ x \in \Omega_{\text{ref}}$, with $\tilde{\phi}$ being the test function for the problem defined on $\Omega = \tau(\Omega_{\text{ref}})$ the test function in the variational formulation can be transported from Ω to Ω_{ref}. Further, for all $\tilde{y} \in Y(\tau)$, $\tilde{\phi} \in Z(\tau)^*$ and $\tau \in T_{\text{ad}}$, $E(y, \tau)$ is defined as

$$\langle E(\tilde{y} \circ \tau, \tau), \tilde{\phi} \circ \tau \rangle_{Z_{\text{ref}}, Z_{\text{ref}}^*} := \langle \bar{E}(\tilde{y}, \tau), \tilde{\phi} \rangle_{Z(\tau), Z(\tau)^*}. \tag{3.29}$$

Now denote $T_{\text{ref}} := T(\Omega_{\text{ref}})$ and suppose that the following holds

- Y_{ref} is a Banach space s. t. $\forall\, \tau \in T_{\text{ad}}$ and then $Y_{\text{ref}} \subset \{\tilde{y} \circ \tau \mid \tilde{y} \in Y(\tau)\}$.
- Z_{ref} is a Banach space s. t. $\forall\, \tau \in T_{\text{ad}}$ and then $Z_{\text{ref}}^* = \left\{\tilde{\phi} \circ \tau \mid \tilde{\phi} \in Z(\tau)^*\right\}$.
- The solution $\tilde{y}(\tau)$ of

  $$\bar{E}(\tilde{y}(\tau), \tau) = 0 \tag{3.30}$$

  satisfies $\tilde{y} \circ \tau \in Y_{\text{ref}}\ \forall\, \tau \in T_{\text{ad}}$.

Now let $\tau \in T_{\text{ad}}$ and define $y(\tau) := \tilde{y} \circ \tau$. Then the transformed state equation in variational form

$$\langle E(y,\tau), \phi \rangle_{Z_{\text{ref}}, Z_{\text{ref}}^*} = 0 \quad \forall \phi \in Z_{\text{ref}}^* \tag{3.31}$$

is equivalent [52] to Eq. (3.28).

Hence, for the operator $E : Y_{\text{ref}} \times T_{\text{ref}} \to Z_{\text{ref}}$ s. t. $\forall \tau \in T_{\text{ad}}$, $\forall \tilde{y} \in Y(\tau)$ it holds that

$$E(y,\tau) = 0 \iff \bar{E}(\tilde{y},\tau) = 0. \tag{3.32}$$

Similarly, for $y = \tilde{y} \circ \tau \in Y_{\text{ref}}$ and $\tau \in T_{\text{ad}}$ the objective function $J : Y_{\text{ref}} \times T_{\text{ref}} \to \mathbb{R}$ is defined with $J(y,\tau) = \bar{J}(\tilde{y},\tau)$.

Thus, Eq. (3.27) can be rewritten in Problem 3.3.1 as

Problem 3.3.2. *The problem of the form*

$$min\ J(y,\tau)\ s.t.\ E(y,\tau) = 0,\ \tau \in T_{ad}, \tag{3.33}$$

where $J : Y_{ref} \times T_{ref} \to \mathbb{R}$ *and* $E : Y_{ref} \times T_{ref} \to Z_{ref}$, *is called the* ***shape optimization problem with transformations on a fixed domain****.*

In this formulation the optimal control problem posed on fixed function spaces where τ is the control. This setup allows to apply standard tools from optimal control theory to derive first-order optimality conditions and to employ gradient-based optimization algorithms to solve the problem [52].

The approach described in this section can also be extended to compute shape derivatives of functionals defined over variable domains governed by partial differential equations. In such non-stationary settings, one typically maps functions from Sobolev spaces defined on a varying domain to corresponding Sobolev spaces on a fixed holdall domain. This parameterization enables the application of differential calculus on a fixed reference configuration as discussed in [56].

In particular, when the state equations are given by non-stationary incompressible Navier–Stokes equations, the admissible state space consists of vector-valued Sobolev functions that are divergence-free. To preserve the divergence-free condition under domain transformations, the Piola transformation is employed. This transformation ensures that volume-preserving properties are maintained when mapping vector fields between domains.

For a rigorous treatment and formal definition of the Piola transformation, I refer to [32] and [26].

3.4. Numerical Optimization Methods

I now recall some standard methods for solving general optimization problem before combining all previous considerations and formulating a method for solving shape optimization problem.

Consider again the most general optimization problem with PDE constraints on Hilbert space

$$\begin{aligned} &\min_{u \in U} J(y,u) \\ &\text{s. t. } e(y,u) = 0. \end{aligned} \tag{3.34}$$

In Eq. (3.34) U is a Hilbert space, $e(y,u)$ - PDE constraint, $y(\cdot)$ is a solution operator of the constraint. Further, it is assumed that J is smooth enough. At this point I do not consider any special type of constraints, i.e. geometric and shape constraints or other constraints on the control or state variables.

Most of the optimization methods rely on first and second derivative of the cost functional to find directions that lead to the decreased objective function values. Hence, the core questions are the choice of good initial guess for second order methods, step size and the directions. In this section, I am going to consider some basic strategies for selecting the suitable search directions and solving the problem (3.34).

3.4.1. Descent Method

I start with rewriting the optimization problem (3.34) in the reduced form:

$$\min_{u \in U} f(u), \tag{3.35}$$

where U is a Hilbert space and $f : U \to \mathbb{R}$ is continuously Fréchet differentiable. Starting from the initial u_0 one wants to construct an iterative process. The main idea of descent methods is to find a descent direction d_k that would satisfy the following

$$f(u_k + \alpha_k d_k) < f(u_k) \tag{3.36}$$

for some $\alpha_k > 0$. In order to find the steepest direction, one can approximate the the cost functional f via a first-order Taylor expansion:

$$f(u_k + \alpha_k d_k) \approx f(u_k) + \alpha_k (\nabla f(u_k), d_k)_U \tag{3.37}$$

Since U is a Hilbert space, in Eq.(3.37) Riesz representation theorem is used to obtain a gradient of f. Hence, the direction that minimizes the cost functional is obtained from the following optimization problem

$$d_k = \underset{\|d\|_U=1}{\arg\min} (\nabla f(u_k), d)_U \tag{3.38}$$

Depending on the definition of the norm and associated inner product in Eq. (3.38), one gets different algorithms. Using Cauchy-Schwarz inequality, for $d \in U$, for $\|d\|_U = 1$ one gets

$$(\nabla f(u_k), d)_U \geq -\|\nabla f(u_k)\|_U \|d\|_U \geq -\|\nabla f(u_k)\|_U$$

with equality only if $d = -\frac{\nabla f(u_k)}{\|\nabla f(u_k)\|_U}$. Since the aim is to minimize the functional f, which decreases in the direction $-\nabla f(u_k)$, I choose

$$d_k = -\nabla f(u_k) \tag{3.39}$$

as the direction. So I obtain the iterative method

$$u_{k+1} = u_k - \alpha_k \nabla f(u_k), \tag{3.40}$$

where α_k is some step size. Method given by Eq. (3.40) is called *steepest descent* method (or *gradient descent* method).

Remark 3.4.1. *I assume that descent direction d_k satisfies the angle condition*

$$-(\nabla f(u_k), d_k)_U \geq \eta \|\nabla f(u_k)\|_U \|d\|_U \quad \text{for } \eta \in (0,1). \tag{3.41}$$

This condition requires that the angles between the chosen search directions d_k and the directions of steepest descent $-\nabla f(u_k)$ are uniformly bounded away from 90°.
To ensure that the sequence of search directions $\{d_k\}_k$ is admissible, it is required that this condition holds [27].

When descent direction is chosen, another issue to address is how far one can move in this direction, i.e. one needs to determine a step size α_k (also called a *line search parameter*). In general, one obtains step size by solving the problem

$$\alpha_k = \underset{\alpha>0}{\arg\min} f(u_k + \alpha d_k) \tag{3.42}$$

which might be quite challenging in practice, hence usually one chooses different strategies to selecting step size.

Remark 3.4.2. *The sequence $\{\alpha_k\}_k$ of step sizes is called admissible if the following two conditions hold*

$$f(u_k + \alpha_k d_k) < f(u_k) \quad \forall k = 1, 2, \ldots \tag{3.43}$$

$$f(u_k + \alpha_k d_k) - f(u_k) \overset{k\to\infty}{\to} 0 \implies \frac{(\nabla f(u_k), d_k)_U}{\|d_k\|_U} \overset{k\to\infty}{\to} 0 \tag{3.44}$$

A globally convergent descent method is given by Algorithm 1.

Algorithm 1 A descent method.

1: Choose $u_0 \in U$ and set $k = 0$.
2: **repeat**
3: Choose descent direction d_k that satisfies (3.41).
4: Determine α_k such that (3.43) to (3.44) hold.
5: Set $u_{k+1} = u_k + \alpha_k d_k$ and $k = k + 1$.
6: **until** stopping criteria.
7: **return** u_k.

For further details on convergence, strategies to select step size and other details I refer to [27, 41, 16].

I now formulate the steepest descent algorithm for the general PDE-constrained optimization problem given in (3.9). Based on the expression for the derivative of the reduced cost functional provided in (3.17), the complete steepest descent algorithm can be stated as follows:

Algorithm 2 A gradient descent method for PDE-constrained problem.

1: Choose $u_0 \in U$.
2: Compute (y_0, p_0) by solving

$$e(y, u_0) = 0, \quad e_y(y_0, u_0)^* p = J_y(y_0, u_0).$$

3: Set $k = 0$.

4: **repeat**
5: Choose the descent direction $d_k = -\nabla f(u_k)$.
6: Determine α_k such that (3.43) to (3.44) hold.
7: Set $u_{k+1} = u_k + \alpha_k d_k$.
8: Compute (y_{k+1}, p_{k+1}) by sequentially solving

$$e(y, u_{k+1}) = 0, \quad e_y(y_{k+1}, u_{k+1})^* p = J_y(y_{k+1}, u_{k+1}).$$

9: Set $k = k + 1$.
10: **until** stopping criteria.
11: **return** u_k.

3.4.2. Newton Method

In contrast to the steepest descent method, which is based only on gradient information, the Newton method incorporates second-order derivative (Hessian) information to determine search directions, typically leading to faster local convergence.

Using second-order Taylor's expansion one gets

$$q(d) = f(u_k) + (\nabla f(u_k), d)_U + \frac{1}{2}(\nabla^2 f(u_k)d, d)_U, \tag{3.45}$$

where $\nabla^2 f(u_k)d$ is a Riesz representative of $f''(u_k)d$. The first order optimality condition for the minimizer of $q(d)$ reads as follows:

$$\nabla f(u_k) + \nabla^2 f(u_k) d_k = 0$$

Assuming that second derivative is smooth enough and invertible, one gets

$$d_k = -(\nabla^2 f(u_k))^{-1} \nabla f(u_k). \tag{3.46}$$

Therefore, the iterative method is obtained

$$u_{k+1} = u_k - (\nabla^2 f(u_k))^{-1} \nabla f(u_k). \tag{3.47}$$

Algorithm 3 Newton's method.

1: Choose $u_0 \in U$, set $k = 0$.
2: **repeat**
3: Set $u_{k+1} = u_k - (\nabla^2 f(u_k))^{-1} \nabla f(u_k)$.
4: Set $k = k + 1$.
5: **until** stopping criteria.
6: **return** u_k.

Consider now again a constrained optimization problem given by Eq.(3.9):

$$\begin{aligned} &\min J(y,u) \\ &\text{s. t. } e(y,u) = 0. \end{aligned}$$

Let $(\bar{y}, \bar{u})$ be the local optimal solution to the problem, $V(\bar{u})$ - neighborhood of $\bar{u}$. Further, it is assumed that $J : Y \times U \to \mathbb{R}$ and $e : Y \times U \to Z$ are twice continuously Fréchet differentiable with Lipschitz continuous second derivatives, and that

$$e_y(\bar{y}, \bar{u}) \text{ is a bijection in a neighbourhood of } (\bar{y}, \bar{u}).$$

Now I formulate Newton's method for PDE-constrained optimization problem as derived in [27] as given in Algorithm 4.

Remark 3.4.3 (Stopping Criteria)**.** *A critical component of Algorithms 1 to 4 is the choice of a stopping criterion, which determines when the iteration is considered sufficiently close to an optimal solution. Common stopping criteria include:*

- *Gradient norm tolerance:* $\|\nabla f(u_k)\| \leq \varepsilon$
- *Relative gradient reduction:* $\|\nabla f(u_k)\| / \|\nabla f(u_0)\| \leq \varepsilon_{rel}$
- *Small update norm:* $\|u_{k+1} - u_k\| \leq \varepsilon_{step}$
- *Objective stagnation:* $|f(u_{k+1}) - f(u_k)| \leq \varepsilon_{obj}$
- *Maximum number of iterations:* $k \geq k_{\max}$

In the numerical examples in Chapter 5, I used a combination of these conditions to ensure efficient termination.

Depending on the discretization level, I used objective stagnation or bound on maximum number of iterations. For a comprehensive discussion of stopping criteria in optimization algorithms, I refer to [61].

Algorithm 4 Newton's method for PDE-constrained optimization problem.

1: Choose $u_0 \in V(\bar{u})$.
2: Compute (y_0, u_0) by solving

$$e(y, u_0) = 0, \quad e_y(y_0, u_0)^* p = J_y(y_0, u_0).$$

3: Set $k = 0$.

4: **repeat**
5: Solve Newton system for $(\delta_y, \delta_u, \delta_\pi)$:

$$\begin{pmatrix} L''_{yy}(y_k, u_k, p_k) & L''_{yu}(y_k, u_k, p_k) & e'_y(y_k, u_k)^* \\ L''_{uy}(y_k, u_k, p_k) & L''_{uu}(y_k, u_k, p_k) & e'_u(y_k, u_k)^* \\ e'_y(y_k, u_k) & e'_u(y_k, u_k) & 0 \end{pmatrix} \begin{pmatrix} \delta_y \\ \delta_u \\ \delta_\pi \end{pmatrix} = \begin{pmatrix} 0 \\ e_u(y, u)^* p - J_u(y_k, u_k) \\ 0 \end{pmatrix} \tag{3.48}$$

6: Set $u_{k+1} = u_k + \delta_u$.
7: Compute (y_{k+1}, p_{k+1}) by sequentially solving

$$e(y, u_{k+1}) = 0, \quad e_y(y_{k+1}, u_{k+1})^* p = J_y(y_{k+1}, u_{k+1}).$$

8: Set $k = k + 1$.
9: **until** stopping criteria.
10: **return** u_k.

3.5. Discretization with Finite Elements and Mesh Quality

Optimization problems, and especially shape optimization problems, are challenging to solve, due to the nonlinearity of shape spaces. Furthermore in general, they are formulated on infinite dimensional spaces. I overcome this by first reformulating the problems from the shape space to standard Hilbert spaces, and then by discretizing the problem. There are two known approaches for discretizing optimization problems: the 'first-optimize-then-discretize' approach and the 'first-discretize-then-optimize' approach. In this thesis, I focus on the former one and refer to Section 5.2 and [56, 13] for further information on the latter one.

In 'first-optimize-then-discretize' approach one starts with the first order necessary optimality conditions for the optimization problem. Then all variables in the system are discretized, including state and adjoint variable, control variable and integrals. One is also free to choose different ansatz spaces for discretization of state and adjoint

variables. Please note that by choosing the same ansatz spaces for the state and the adjoint variable will lead to an optimality condition identical to the one obtained with 'first-discretize-then-optimize' method.

The optimality system derived from an optimization problem given by (3.12) consists of differential equations and algebraic conditions describing the necessary conditions for optimality. To be able to solve the problem numerically, one has to first rewrite this system from the continuous formulation to a discrete one. This process involves discretization of partial differential equations, where the domain of the problem is divided into a finite number of elements to approximate the solution, converting the continuous problem into a system of algebraic equations.

Through FEM, the continuous domain is represented by a mesh, consisting of interconnected elements (e.g., triangles, quadrilaterals, or tetrahedra), over which the equations are solved approximately. Consequently, transitioning from the continuous optimality system to practical numerical computation requires both the derivation of a discrete formulation via FEM and the preservation of mesh quality, which is crucial for accurately resolving the geometry and capturing the solution's behavior. A high-quality mesh ensures numerical stability, accuracy, and convergence of the solution. Once the problem is discretized, iterative optimization methods such as gradient descent or Newton-type algorithms can be applied efficiently to solve the resulting system.

The accuracy and efficiency of the solution depend heavily on the quality of the mesh used for discretization. A well-constructed mesh ensures that the solution captures the behavior of the system while minimizing computational effort. Mesh quality is influenced by factors such as element size, shape, and distribution, which must align with the problem's geometry and physical properties. Balancing computational cost with solution accuracy is critical in this step, as it directly impacts the reliability of the results.

3.5.1. Mesh Quality

One of the key challenges in shape optimization is maintaining mesh quality throughout the iterative optimization process, since geometric updates are intrinsic to shape optimization.

This issue becomes especially critical in problems involving large deformations, which are the primary focus of this thesis. In such settings, mesh quality can deteriorate significantly, and in some cases, elements may even become degenerate. The approach proposed in Section 4.6 is designed to address this challenge by preserving mesh quality throughout the optimization. To assess the effectiveness of such techniques, it

is essential to have appropriate metrics for evaluating mesh quality, which allow both qualitative and quantitative comparisons.

Since in the numerical experiments I use triangular and tetrahedral elements, I am focusing only on quality measures for evaluating triangular and tetrahedral elements.

3.5.2. Mesh Quality Measures

To quantify a mesh quality, I choose the metric that determines how far the element is from an ideal cell shape. For each measure, the highest values of the quality measure correspond to the best elements. These measures provide quantitative metrics for evaluating and improving the geometric quality of elements, resulting in more accurate and reliable numerical results. I am going to focus on one quality measure within this thesis to evaluate quality of meshes in the section with numerical examples (Section 5.2). For further information about other mesh quality measures and their comparison and evaluation please refer to [75].

Among various mesh quality measures, the *radius ratio* is commonly used to assess the geometric quality of elements, especially in triangular and tetrahedral meshes. The radius ratio is a dimensionless quantity that compares the inradius (radius of the largest inscribed circle or sphere) to the circumradius (radius of the smallest circumscribed circle or sphere) of an element.

Definition 3.5.1. *The radius ratio R_r for an element is defined as:*

$$R_r = \frac{r_{in}}{r_{circ}},$$

where:

- r_{in} *is the inradius of the element.*
- r_{circ} *is the circumradius of the element.*

For high-quality elements, the radius ratio approaches 1, whereas for poor-quality elements, often with sharp angles or highly skewed geometry, it approaches 0.

In addition to the radius ratio, other commonly used mesh quality measures include:

- Aspect Ratio: Measures the ratio of the longest to the shortest edge.
- Skewness: Evaluates the deviation of an element from an ideal shape.
- Jacobian Determinant: Ensures that the element is non-inverted and well-scaled.

The radius ratio is crucial in mesh generation and refinement, ensuring well-conditioned stiffness matrices, reducing numerical errors, and improving solver convergence and efficiency in FEM simulations.

3.6. Condition number

Since at the core of the optimization solution process is the FEM-based solver, it is reasonable to account typical issues and intricacies natural for solvers of such type.

In particular, one aims to keep the solver as accurate as possible while minimizing computational efforts. It translates to the goal of keeping the condition number of the stiffness matrix as small as possible. Matrices which are poorly conditioned are affecting linear equation solvers. Namely, bad condition number of the FEM stiffness matrix slows down solvers or often leads to large round-off errors into the results.

The condition number of the stiffness matrix is highly sensitive to the quality of the discretization mesh. Both the shape and size of the finite elements strongly influence matrix conditioning. In particular, triangular elements with small internal angles tend to produce large eigenvalues in the stiffness matrix, degrading both the numerical stability and the accuracy of the solution.

To mitigate this, it is critical to ensure that the internal angles of triangles in the mesh are well-behaved. As shown in [5], it is not sufficient to control only the element size; maintaining a lower bound on the minimum angle and avoiding angles approaching 180° are also essential. In fact, the accuracy of finite element solutions deteriorates when the mesh contains elements with excessively large or small angles.

3.6.1. Condition Number for Linear Systems

In this subsection I consider the condition number strictly from the perspective of solving linear systems of equation, i.e. the condition number of Jacobian. I define the condition number of a matrix and discuss how it is linked to solving linear systems.

Consider a standard $n \times n$ linear system of equations

$$Ax = b, \tag{3.49}$$

where $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n, x \in \mathbb{R}^n$. The question I aim to answer is: How sensitive is the solution to $Ax = b$ with respect to changes in b?

The condition number of a square invertible matrix $A \in \mathbb{R}^{n \times n}$ is defined as follows

$$\kappa(A) = \|A\| \cdot \left\|A^{-1}\right\|, \tag{3.50}$$

where $\|\cdot\|$ is some chosen matrix norm. Alternatively one can represent condition number of a matrix A in terms of its largest and smallest singular values $\sigma_{max}, \sigma_{min}$ as follows

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}. \tag{3.51}$$

If further matrix A is symmetric and positive definite, then

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}, \tag{3.52}$$

where $\lambda_{max}(A), \lambda_{min}(A)$ are the largest and smallest eigenvalues of A. In the following I will use the notion of a condition number given by Eq. (3.52).

From Eq. (3.52) it is known, that if matrix A is singular then its condition number is infinite, i.e $\kappa(A) = \infty$.

With regards to the problem (3.49), if A is a singular matrix and a solution x exists, then one can change the solution without changing b.

Therefore, a large condition number of a system indicates that the solution is highly sensitive to perturbations in data b.

3.6.2. Condition Number for Optimization Problems

When it comes to the optimization poor conditioning affects the quality of the direction and step size. Consider the simple minimization problem for quadratic function

$$J(x) = b^T x + \frac{1}{2} x^T A x,$$

where A is a positive definite matrix. The behaviour of J in the neighbourhood of a local minimum is given by the eigenvalues of matrix A, in particular, by condition number $\kappa(A)$.

Consider the level set of the function $J(x)$. The smallest and largest eigenvalues correspond to the principal axes of the ellipses. When $\kappa(A) = 1$, the contours of $J(x)$ are circular and as $\kappa(A)$ increases they become more elongated. So when $\kappa(A)$ is large, the cost function will be highly affected by the relative change with respect to the

norm of x. The directions that produce the largest and smallest changes in the function value are the eigenvectors corresponding to λ_{max} and λ_{min}, respectively.

Hence, by using the Hessian matrix at the solution one can measure the sensitivity, or conditioning, of the function value with respect to changes in x [35].

As condition number increases, the contours become more elongated, as shown in the second figure. If $\kappa(A)$ is large, the relative change in the objective function due to a perturbation of constant norm in the variables will vary radically depending on the direction of the perturbation.

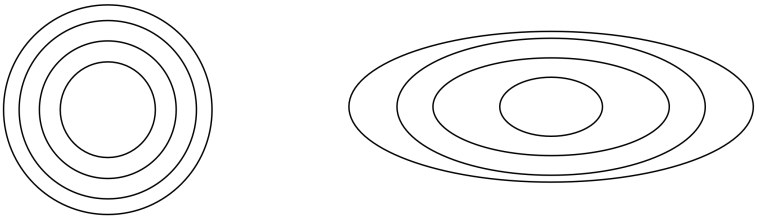


Figure 3.1.: Contours of a quadratic function with $\kappa(A) = 1$ and $\kappa(A) = 10$.

In general, one would have to compute all eigenvalues and corresponding eigenvectors to obtain full information about a Hessian matrix. However, many algorithms compute the Cholesky factorization of the matrix [35], which can be used to estimate λ_{max} and λ_{min} and their corresponding eigenvectors.

3.6.3. Condition Number Effects for Newton Method Convergence

If one examines the proof of quadratic convergence for Newton's method, it becomes clear that convergence relies on the non-singularity of the Hessian at the solution. Moreover, the size of the region in which quadratic convergence is guaranteed decreases as the condition number of the Hessian increases [35]. Therefore the convergence of a Newton-type solver will be degraded if the Hessian at the solution is ill-conditioned.

In the context of FEM, the Hessian of the objective function is often closely related to the stiffness matrix—particularly when the underlying problem involves energy minimization or PDE-constrained optimization. Therefore, the condition number of the stiffness matrix directly affects the condition number of the Hessian. Since mesh quality has a strong influence on the conditioning of the stiffness matrix, it follows that poorly shaped or highly distorted elements can lead to an ill-conditioned Hessian. This, in turn, reduces the effectiveness of Newton-type solvers by shrinking the region of rapid convergence and increasing the sensitivity of the solution to numerical errors.

Thus, careful mesh design is not only important for the accuracy of FEM discretization, but also plays a critical role in ensuring the robust and efficient convergence of Newton-based optimization algorithms. I illustrate this by numerical examples in Chapter 5.

4. Shape Optimization Problem in Navier-Stokes Flow

In this section I discuss Navier-Stokes equations and show how they can be transformed to the reference domain.

4.1. Navier-Stokes Equations

The incompressible Navier-Stokes equations represent one of the most fundamental models in fluid dynamics, which describe the velocity field and pressure distribution of a fluid with uniform density. In this section, I consider both two- and three-dimensional formulations, as the numerical results presented in Chapter 5 include simulations in both settings. Further, for simplicity, I restrict this work to the stationary equations.

Let Ω be a bounded Lipschitz domain in $\mathbb{R}^d, d = 2,3$ with a Lipschitz continuous boundary $\partial\Omega$. I consider the stationary NS equations

$$\begin{aligned} -\nu\Delta v + (v\cdot\nabla)v + \nabla p &= 0 && \text{in } \Omega \\ \operatorname{div} v &= 0 && \text{in } \Omega \\ v &= v_\infty && \text{on } \Gamma_{\text{in}} \\ v &= 0 && \text{on } \Gamma_{\text{obs}} \cup \Gamma_{\text{wall}} \\ pn - \nu\frac{\partial v}{\partial n} &= 0 && \text{on } \Gamma_{\text{out}}. \end{aligned} \tag{4.53}$$

In (4.53) $v = v(x) : \Omega \to \mathbb{R}^d$ is the velocity field, $p = p(x) : \Omega \to \mathbb{R}$ is the pressure of the flow at a point x, $n : \Gamma \to \mathbb{R}^d$ - outward-facing unit normal vector to Γ, ν - kinematic viscosity, v_∞ describes the velocity profile at the inflow boundary.

Since within this thesis I consider only laminar flows, I assume strong viscous effects modeled by high values of ν. As is known, $\nu = \frac{1}{Re}$, where Re is Reynolds number.

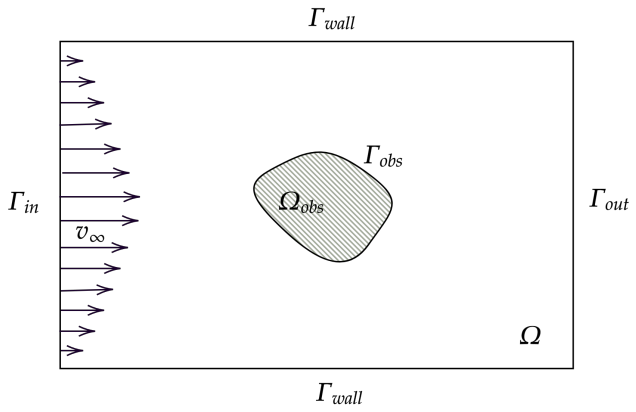


Figure 4.1.: Sketch of the holdall domain $G = \Omega \cup \Omega_{\text{obs}}$.

Therefore, in the present setting I consider only low *Re*.

Together with Γ_{obs} the fluid domain Ω is allowed to change, but the outer boundaries, i.e. Γ_{in}, Γ_{out} and Γ_{wall}, of the experiment are fixed.

Here I assume that Γ_{in} and Γ_{obs} have positive Lebesgue measure, $\Gamma_{\text{obs}} \cap (\Gamma_{\text{in}} \cup \Gamma_{\text{wall}} \cup \Gamma_{\text{out}}) = \emptyset$ holds during the entire optimization.

4.1.1. Weak Formulation

A classical solution for (4.53) is such that $v \in C^2(\Omega) \cap C(\overline{\Omega})$ and $p \in C^1(\Omega)$. In order to require less regularity and proceed with discretization, I derive weak formulation of the coupled Navier-Stokes (4.99).

Remark 4.1.1. *Consider a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^d$. By $Df \in \mathbb{R}^{d\times d}$ I denote the Jacobian matrix with the ordering defined as follows*

$$Df = \left(\frac{\partial f_i}{\partial x_j} \right)_{i,j=1,\dots,d} \tag{4.54}$$

To define weak formulation of the problem given by (4.53) I consider

$$\begin{aligned} V &:= \{v \in H^1(\Omega, \mathbb{R}^d) \colon \operatorname{div}(v) = 0, v|_{\Gamma_{\text{in}}} = v_\infty, v|_{\Gamma_{\text{wall}} \cup \Gamma_{\text{obs}}} = 0 \text{ a.e.}\}, \\ V_0 &:= \{v \in H^1(\Omega, \mathbb{R}^d) \colon \operatorname{div}(v) = 0, v|_{\Gamma_{\text{in}} \cup \Gamma_{\text{wall}} \cup \Gamma_{\text{obs}}} = 0 \text{ a.e.}\}, \\ Q &:= \{p \in L^2(\Omega)\} \end{aligned} \tag{4.55}$$

Then the weak formulation of the PDE constraint (4.53) reads as

Find $(v, p) \in V \times Q$ such that

$$\begin{aligned} \int_\Omega \nu Dv : D\delta_v + (Dv\, v) \cdot \delta_v - p \operatorname{Tr}(D\delta_v)\, dx = 0, \\ - \int_\Omega \delta_p \operatorname{Tr}(Dv)\, dx = 0 \end{aligned} \tag{4.56}$$

for all test functions $(\delta_v, \delta_p) \in V_0 \times Q$.

Note that within this thesis, I am using the symbol δ_* to denote test functions associated with a given variable $*$.

Remark 4.1.2. *In (4.56), the notation $A : B$ stands for the Frobenius inner product of two $d \times d$ matrices A and B, i.e.*

$$A : B = \sum_{i,j=1}^{d} A_{ij} B_{ij}. \tag{4.57}$$

In what follows, I also denote by

$$\| A \| = (A : A)^{\frac{1}{2}}$$

the associated Frobenius norm, where $d = 2, 3$.

4.1.2. Optimization Problem Statement

I consider a classical problem of optimal shape design for a viscous, incompressible fluid flow given by Navier-Stokes equations described in [58].

Let Ω be a d-dimensional, bounded domain with Lipschitz boundary illustrated in Fig. 4.1 for $d = 2$. I consider the minimization of the following energy dissipation functional

$$\min_{\Gamma_{\text{obs}}} j(v, \Gamma_{\text{obs}}) = \frac{\nu}{2} \int_\Omega \sum_{i,j=1}^{d} \left(\frac{\partial v_i}{\partial x_j} \right)^2 dx. \tag{4.58}$$

In (4.58) the boundary Γ_{obs} of the obstacle Ω_{obs} serves as the design variable. Additionally, I assume Ω_{obs} is non-empty, connected set and Γ_{obs} is a smooth and compact Riemannian manifold without boundary.

Since I assume the absence of gravity, there are only drag and lift forces present in the system. As mentioned in [58], for a symmetric body if its axis is aligned with the velocity at infinity then there is no lift and only drag is generated due to the velocity difference between the solid body and the fluid [44]. Therefore, one can alternate minimization of the energy of the system and minimization of the drag on the surface of the obstacle[34],[57]. In the case of non-symmetric body, I will choose the energy dissipation minimization as the objective function. I will refer to the objective function as either the drag or the energy dissipation interchangeably.

So the problem can now be formulated in a form (1.1) as follows

Problem 4.1.3.

$$\begin{aligned} \min_{\Gamma_{obs}} \; & j(v,\Gamma_{obs}) = \frac{\nu}{2}\int_{\Omega}\sum_{i,j=1}^{d}\left(\frac{\partial v_i}{\partial x_j}\right)^2 dx \\ s.t. \quad & E(y,F(\Omega)) = 0 \end{aligned} \tag{4.59}$$

In (4.59) the velocity v is a solution to Navier-Stokes equations given by (4.53), which is a part of constraints denoted by E. I will describe in detail the rest of the constraints which are part of the (4.59) in the further sections.

One class of constraints to consider is geometric constraints, which in case of the selected Navier-Stokes flow based problem make sure that trivial solutions are excluded.

4.1.3. Geometrical Constraints

Two classical and widely used examples for this are the bound on the change of the volume and constraint on the location of the barycenter of the shape, which I am going to discuss in more details further. Among others, the constraint on the surface of the shape is often considered as an alternative for the constraint on the volume. Another common constraint which often comes from design specifications or other regulations is that a shape should be located within a feasible region. However it is commonly implemented in a parametric sense, by bounding the location of nodes [64].

In case of the problem discussed in this work, i.e., for the shape optimization of a specimen Ω_{obs} with respect to functionals of type given by (4.58), it is essential to exclude trivial solutions. The obvious design improvement would be movement of the obstacle toward the walls Γ_{wall} or shrinking of an Ω_{obs} to a point which would represent undesired descent directions.

Since the considered problem is to find optimal shapes of an obstacle of a certain volume located in the flow tunnel it is natural to use geometrical constraints to fulfill these requirements. So the barycenter and volume of the obstacle Ω_{obs} are fixed using the constraints

$$\text{vol}(\Omega_{\text{obs}}) = \int_{\Omega_{\text{obs}}} 1 \, dx = \text{ const}, \tag{4.60}$$

$$\text{bc}(\Omega_{\text{obs}}) = \frac{1}{\text{vol}(\Omega_{\text{obs}})} \int_{\Omega_{\text{obs}}} x \, dx = \text{ const}. \tag{4.61}$$

Since the computation for the barycenter involves the volume of Ω_{obs} itself, these conditions are coupled in principle. Yet, if (4.60) is fulfilled, the term $\text{vol}(\Omega_{\text{obs}})^{-1}$ in (4.61) is constant and can thus be factored out. By further assuming that the barycenter of the specimen Ω_{obs} is $0 \in \mathbb{R}^d$, it is thus sufficient to require $\int_{\Omega_{\text{obs}}} x \, dx = 0$.

4.2. Admissible Boundary Deformations

Consider again the shape optimization problem with transformations on a fixed domain. Motivated by [38], I choose Banach spaces X, Y such that

$$D_{\text{ad}} \subset X \hookrightarrow C^1(\Gamma_{\text{obs}})^d, \qquad Y \hookrightarrow C^1(\overline{\Omega})^d,$$

and assume that the extension operator $S : X \to Y$ is linear and continuous.

Further, I consider continuous extension mapping $S : X \to Y$. Then the problem (3.33) can be written as follows

$$\begin{aligned} \min_{c \in D_{\text{ad}}} \quad & j(y, F(\Omega)) + \frac{\alpha}{2}\|c\|_X^2 \\ \text{s.t.} \quad & E(y, F(\Omega)) = 0 \\ & F = \text{id} + w \qquad\qquad \text{in } \Omega \\ & w = S(c), \end{aligned} \tag{4.62}$$

where $E(y, F(\Omega))$ represents a PDE constraint, c is a control variable and $F = \text{id} + w$ is a perturbation of identity mapping. I choose D_{ad} and S such that the following holds [38]:

A1 There exists $\eta > 0$ and an open neighbourhood U of Ω such that for all admissible controls $c \in D_{\text{ad}}$, there exists a C^1-diffeomorphism $F : U \to U$ such that $F|_\Omega = id + S(c)$ and $DF(x)$ has a condition number bounded by η for all $x \in U$.

A2 Let $c_1, c_2 \in D_{\text{ad}}$. Then $(\text{id} + S(c_1))(\Omega) = (\text{id} + S(c_2))(\Omega)$ iff $c_1 = c_2$ on Γ_{obs}.

Assumption A1 leads to the following lemma:

Lemma 4.2.1. *Let $\Omega \subset \mathbb{R}^d, d \in \{2,3\}$ be a smooth domain and assumption A1 is fulfilled. Then $(id + S(c))(\Omega)$ is a Lipschitz domain for all admissible $c \in D_{ad}$.*

Proof. Follows from [42, Thm. 4.1]. □

Remark 4.2.2. *In the framework of Lemma 4.2.1 one can even claim that $id + S(c)$ is a C^1 domain for all admissible $c \in D_{ad}$.*

I present here sufficient conditions for assumption A1 to be fulfilled.

Theorem 4.2.3. *Let $d \in \{2,3\}$, Ω be a bounded Lipschitz domain, $\eta_1 \in (0,1)$, X, Y be Banach spaces such that $Y \hookrightarrow C^1(\overline{\Omega})^d$. Further, it is assumed $S : X \to Y$ is linear and continuous. Then there exists $\eta_2 > 0$ and*

$$D_{ad} := \{c \in X : \det(D(id + S(c))) \geq \eta_1, \|c\|_X \leq \eta_2\}.$$

Then assumption A1 holds true.

Proof. For proof and more details I refer to [38]. □

Furthermore, Theorem 4.2.3 can be extended to the case of nonlinear mapping S. Following [38] I consider the additional Banach space $\tilde{X}$.

Theorem 4.2.4. *Let $d \in \{2,3\}$, Ω, U be bounded Lipschitz domains, with $\Omega \subset U$, $\eta_1 \in (0,1)$, and $\eta_2 > 0$, $X, \tilde{X}, Y$ be Banach spaces such that $Y \hookrightarrow C_c^1(U)^d$. Let further X be compactly embedded into $\tilde{X}$. Further, it is assumed $S : X \to Y$ is continuous with $S(0) = 0$ and that S can be extended to a continuous mapping $S : \tilde{X} \to C^1(\overline{U})^d$. Then for*

$$D_{ad} := \{c \in X : \det(D(id + S(c))) \geq \eta_1, \|c\|_X \leq \eta_2\}$$

the assumption A1 holds true.

Proof. For proof I refer to [38]. □

4.3. Transformation to the Reference Domain

In order to reformulate the optimization problem (4.58) to (4.61) as an optimal control problem in appropriate function spaces, I fix the domain Ω as a reference configuration following the methodology described in Section 3.3. In the following I use the subscript $(*)_w$ to denote the deformed state via the mapping F, i.e. $\Omega_w := F(\Omega)$.

Let

$$F : \Omega \mapsto \Omega_w,\ F = \mathrm{id} + w \tag{4.63}$$

such that F results in an admissible deformation for Ω. In (4.63) $w \in W^{1,\infty}(\Omega, \mathbb{R}^d)$ denotes the domain displacement.

I parameterize the domain Ω via the mapping given in (4.63) for some given sufficiently small displacement field w:

$$\Omega_w := \{x + w : x \in \Omega\} \tag{4.64}$$

So the set of admissible shapes $G_{\mathrm{adm}} := \{F(\Omega) \colon F \in \mathcal{F}_{\mathrm{adm}}\}$ can now be defined in terms of the perturbation of identity mapping as follows

$$G_{\mathrm{adm}} := \{F(\Omega) \colon F = \mathrm{id} + w, w \in W^{1,\infty}(\Omega, \mathbb{R}^d)\} \tag{4.65}$$

So now I can formulate (4.59) on the reference domain as an optimal control problem. I use the method of mappings [60] by writing down the state (4.56), objective (4.58) and the corresponding state variable v in terms of $F(\Omega)$ and obtain

$$\begin{aligned} \min_{c \in L^2(\Gamma_{\mathrm{obs}})} \quad & j(y, F(\Omega)) + \frac{\alpha}{2}\|c\|^2_{L^2(\Gamma_{\mathrm{obs}})} & \\ \text{s.t.} \quad & E(y, F(\Omega)) = 0 & \\ & F = \mathrm{id} + w & \text{in } \Omega \\ & \det(DF) \geq \eta_{\mathrm{det}} & \text{in } \Omega \\ & w = S(c). & \end{aligned} \tag{4.66}$$

In (4.66), I add the L^2-regularization term to the cost functional with regularization parameter $\alpha > 0$ to get a smoother optimal control, following the classical Tikhonov approach. This naturally raises the question of how the solution to the overall problem depends on the choice of α. I will explore this dependency in detail in Section 5.2.2. While this term helps stabilize the optimization problem and penalizes large control amplitudes, it does not directly enforce higher regularity such as continuity or differentiability of the control. Since the control c enters through a smoothing operator S

into the deformation $w = S(c)$, the overall regularity of w may still be ensured. Nevertheless, if higher control regularity is indeed necessary — for instance, to guarantee $w \in W^{1,\infty}$ and $w = c$ on Γ_{obs} — then a stronger regularization (e.g., H^1- or H^2-based) or domain-specific assumptions on S may be required. The precise role of α and the interplay between control regularization and deformation regularity will be explored further in Section 5.2.2.

4.3.1. Navier-Stokes on Reference Domain

Let $\Omega \subset \mathbb{R}^d$ be the reference domain and $F : \Omega \to \Omega_w := F(\Omega)$ be a sufficiently smooth invertible deformation mapping defining the current domain Ω_w.

Let (v_w, p_w) denote the velocity and pressure fields on the current domain Ω_w, which satisfy the Navier-Stokes equations there. Their counterparts on the reference domain Ω are denoted by (v, p).

The fields on the two domains are related by pullback and pushforward via F as follows:

$$v = v_w \circ F, \quad p = p_w \circ F, \tag{4.67}$$

or equivalently,

$$v_w = v \circ F^{-1}, \quad p_w = p \circ F^{-1}. \tag{4.68}$$

More explicitly, the mappings are

$$v_w : \Omega_w \to \mathbb{R}^d, \quad v : \Omega \to \mathbb{R}^d, \tag{4.69}$$

$$p_w : \Omega_w \to \mathbb{R}, \quad p : \Omega \to \mathbb{R}. \tag{4.70}$$

The spatial derivatives transform according to the chain rule:

$$Dv_w(y) = Dv(x)(DF(x))^{-1} \quad \text{for } y = F(x), \tag{4.71}$$

and similarly for the pressure gradient,

$$Dp_w(y) = Dp(x)(DF(x))^{-1}. \tag{4.72}$$

Since the divergence is the trace of the velocity gradient, it holds

$$\nabla_{x_w} \cdot v_w = \operatorname{Tr}(Dv_w) = \operatorname{Tr}\left(Dv(DF)^{-1}\right).$$

Therefore, the incompressibility constraint pulled back to the reference domain Ω reads

$$\operatorname{Tr}\left(Dv(DF)^{-1}\right) = 0. \tag{4.73}$$

Using the relations (4.69) to (4.72) and standard computations, the weak formulation of the shape optimization problem pulled back to the reference domain Ω reads:

$$\min_{F\in\mathcal{F}_{\text{adm}}} \quad j(v,F(\Omega)) = \frac{\nu}{2}\int_\Omega \left(Dv(DF)^{-1}\right) : \left(Dv(DF)^{-1}\right)\det(DF)\,dx \tag{4.74}$$

$$\text{s.t.} \quad \int_\Omega \Big[\nu\left(Dv(DF)^{-1}\right) : \left(D\delta_v(DF)^{-1}\right) + \left(Dv(DF)^{-1}v\right)\cdot\delta_v - p\,\mathrm{Tr}\left(D\delta_v(DF)^{-1}\right)\Big]\det(DF)\,dx = 0, \tag{4.75}$$

$$-\int_\Omega \delta_p\,\mathrm{Tr}\left(Dv(DF)^{-1}\right)\det(DF)\,dx = 0, \tag{4.76}$$

$$\int_{\Omega_{\text{obs}}} \det(DF) - 1\,dx = 0, \tag{4.77}$$

$$\int_{\Omega_{\text{obs}}} F\,\det(DF)\,dx = 0 \tag{4.78}$$

for all test functions $(\delta_v,\delta_p)\in V\times Q$.

The shape optimization problem (4.74) to (4.78) is formulated as optimization over a set of admissible mappings. In order to reformulate as the optimal control problem of form (4.66) the data on admissible transformation is needed. Thus, following the same approach as in [38] and further reformulating the constraint $\det(DF)\geq\eta_{\text{det}}$ as a penalty term.

Problem 4.3.1 (Optimal control problem for Navier-Stokes equations)**.**

$$\begin{aligned}\min_{c\in L^2(\Gamma_{obs})}\ & J(v,c) := j(v,F) + \frac{\alpha}{2}\int_{\Gamma_{obs}} c^2\,ds + \frac{\beta}{2}\int_\Omega \left((\eta_{det}-\det(DF))_+\right)^2\,dx\\ s.t.\ & \text{(4.75) to (4.78)}\\ & F = \mathrm{id}+w\\ & w = S(c),\end{aligned} \tag{4.79}$$

where $(\cdot)_+$ *denotes the positive-part function.*

To complete Problem 4.3.1 the extension equation has to be specified, namely the mapping S which links a scalar valued boundary control c to admissible deformation fields w. Several possible ways to model this are presented in the next section (Section 4.4).

4.4. Approaches to Choose the Extension Operator S

In transformation-based shape optimization, one typically controls the deformation of a reference domain Ω via a vector-valued displacement field w. However, since optimization is often carried out with scalar-valued design parameters defined only on a subset of the boundary (the design boundary Γ_{obs}), a crucial modeling step is to construct a suitable extension operator S that maps scalar boundary controls to domain-wide deformations.

This section discusses several strategies for defining the extension operator S, following the framework proposed in [38]. The central idea is to use a sequence of classical PDE-based operators to gradually lift a scalar-valued control variable c defined on Γ_{obs} to a vector-valued displacement field $w \in \Omega$. The construction of S must ensure sufficient regularity and injectivity of the resulting transformation, while also allowing for flexibility in boundary deformation (e.g., enabling tangential or normal displacements). This is particularly relevant in practical applications where mesh quality and numerical stability are critical.

The mapping is typically realized through the following chain:

$$c \overset{\text{I.}}{\mapsto} b \overset{\text{II.}}{\mapsto} z \overset{\text{III.}}{\mapsto} w, \tag{4.80}$$

where:

- **I.** The scalar control c is smoothed by solving a Laplace–Beltrami equation on Γ_{obs}, yielding a scalar-valued function b.
- **II.** This function b is extended to the interior of Ω by solving an elliptic PDE, producing either a scalar field z or directly a vector field.
- **III.** If needed, a mapping from scalar to vector field is performed to obtain the final deformation field w.

Let $\Omega \subset \mathbb{R}^d$ be a smooth domain with boundary Γ, and let $\Omega_{\text{obs}} \subset \Omega$ be a fixed obstacle with boundary $\Gamma_{\text{obs}} \subset \Gamma$, such that $\Gamma \setminus \Gamma_{\text{obs}} \neq \emptyset$. The following construction assumes sufficient smoothness of Ω and Γ_{obs}, and that Section 4.2 is satisfied to guarantee admissible deformations.

Auxiliary Operators

I first present some PDE-based operators which are needed later for the definition of an extension operator S.

Laplace–Beltrami smoothing on Γ_{obs}:

$$b - \Delta_{\Gamma_{\text{obs}}} b = f \quad \text{on } \Gamma_{\text{obs}}, \tag{4.81}$$

where $\Delta_{\Gamma_{\text{obs}}}$ denotes the Laplace–Beltrami operator. The associated solution operator is denoted by $S_{\Gamma_{\text{obs}}}$ in the scalar case and $S^d_{\Gamma_{\text{obs}}}$ in the vector-valued case.

Scalar elliptic extension in Ω:

$$\begin{aligned} \Delta z &= 0 \quad \text{in } \Omega, \\ z &= 0 \quad \text{on } \Gamma \setminus \Gamma_{\text{obs}}, \\ \nabla z \cdot n &= b \quad \text{on } \Gamma_{\text{obs}}, \end{aligned} \tag{4.82}$$

where n denotes the outward unit normal vector on Γ. The corresponding solution operator is denoted S_Ω. In (4.82) both b and z are scalar-valued.

Vector-valued elliptic extension in Ω:

$$\begin{aligned} \boldsymbol{\nabla} \cdot \left(Dz + Dz^T \right) &= 0 \quad \text{in } \Omega, \\ z &= 0 \quad \text{on } \Gamma \setminus \Gamma_{\text{obs}}, \\ \nabla z \cdot n &= b \quad \text{on } \Gamma_{\text{obs}}, \end{aligned} \tag{4.83}$$

where Dz denotes the Jacobian matrix of z. The associated solution operator S^d_Ω maps a vector-valued b to a vector-valued z.

The extension operator S can be constructed by composing the auxiliary operators defined above in different ways. Each approach begins with a scalar control c on the design boundary Γ_{obs}, which is first smoothed using the Laplace–Beltrami operator and then extended into the interior of the domain Ω via an elliptic PDE. The point at which the control becomes vector-valued, as well as the method by which this transition occurs, influences both the nature of the resulting deformations and the complexity of the implementation.

In the following, three distinct strategies for defining S are presented. These differ in the placement of the scalar-to-vector transition, the directional restrictions imposed on the deformation field, and the regularity of the resulting transformations.

Strategy 1 (S1): Scalar Extension with Normal Projection

This strategy constructs the displacement field using only scalar-valued intermediate fields. The final vector field is obtained by projecting the scalar result onto a smoothly

extended normal direction:

$$S_1(c) := S_\Omega(S_{\Gamma_{\text{obs}}}(c))n_{\text{ext}}, \tag{4.84}$$

where n_{ext} denotes a smooth extension of the unit normal vector from Γ_{obs} into Ω.

The operator $S_1(c)$ in a weak form is given via the following weak formulation of operators $S_{\Gamma_{\text{obs}}}$:

$$\int_{\Gamma_{\text{obs}}} b\delta_b + \nabla_{\Gamma_{\text{obs}}} b \cdot \nabla_{\Gamma_{\text{obs}}} \delta_b \, ds = \int_{\Gamma_{\text{obs}}} c\delta_b \, ds, \tag{4.85}$$

and S_Ω:

$$\int_\Omega \nabla z \cdot \nabla \delta_z \, dx = \int_{\Gamma_{\text{obs}}} b \cdot \delta_z \, ds,$$

where δ_b and δ_z are test functions from the corresponding spaces. Finally, since z is still a scalar-valued variable, it is mapped to vector-valued w following step $III.$ as follows:

$$\int_\Omega w \cdot \nabla \delta_n \, dx = \int_\Gamma z n_{ext} \cdot \delta_n \, dx, \tag{4.86}$$

for all corresponding test functions δ_n.

Displacements are limited to the normal direction of Γ_{obs}, which simplifies the modeling and is consistent with the structure of the shape derivative in many classical shape optimization formulations (like Hadamard–Zolésio theorem [91]).

Strategy 2 (S2): Vector-Valued Elliptic Extension

In this strategy, scalar operators are used in the initial stages, but the final deformation field is obtained directly by solving a vector-valued ellipticPDE, thereby avoiding the need for an explicit normal extension (step III.). An extension operator is defined as follows

$$S_2(c) := S^d_\Omega(S_{\Gamma_{\text{obs}}}(c) \cdot n), \tag{4.87}$$

with the corresponding weak formulation

$$\int_\Omega \left(Dw + Dw^T\right) : D\delta_w \, dx = \int_{\Gamma_{\text{obs}}} bn \cdot \delta_w \, dx. \tag{4.88}$$

for all suitable test functions δ_w together with the weak Laplace-Beltrami equation given in (S1) by (4.85). I am going to discuss the motivation behind this equation in Section 4.5. Further, in comparison with (S1) the equation (4.86) is not needed anymore.

This approach results in a smoother displacement field and enables more stable deformations, while still aligning the deformation with the normal direction through the scaled right-hand side.

Strategy 3 (S3): Fully Vector-Valued Extension

This strategy lifts the scalar control to a vector field at the earliest stage by interpreting the control as a scaled normal vector field:

$$S_3(c) := S_\Omega^d(S_{\Gamma_{\text{obs}}}^d(cn)), \tag{4.89}$$

Hence, as in the previous case then the deformation field w is obtained directly at a step III without using (4.86). In (4.89) n is the outward unit normal vector to Γ_{obs} and $S_{\Gamma_{\text{obs}}}^d$ denotes the solution operator to the vector-valued Laplace-Beltrami equation given by the following variational formulation:

$$\int_{\Gamma_{\text{obs}}} b \cdot \delta_b + D_{\Gamma_{\text{obs}}} b : D_{\Gamma_{\text{obs}}} \delta_b \, ds = \int_{\Gamma_{\text{obs}}} cn \cdot \delta_b \, ds. \tag{4.90}$$

In (4.90) δ_b is a test function corresponding to b from the suitable space. Further, S_Ω^d is a solution operator to vector-valued elliptic extension equation defined as before by (4.88) which now doesn't have to be scaled on the right-hand side by n:

$$\int_\Omega \left(Dw + Dw^T\right) : D\delta_w \, dx = \int_{\Gamma_{\text{obs}}} b \cdot \delta_w \, dx. \tag{4.91}$$

This formulation offers the most general setting, allowing both normal and tangential displacements. It also removes the need for any artificial normal projection, potentially resulting in more flexible deformations.

Table 4.1.: Comparison of extension strategies for operator S. In all cases c is scalar.

#	Interm. Fields	Projection	Direction	Regularity
S1	Scalar b, z	Yes (n_{ext})	Normal only	High
S2	Scalar b	No	Normal (RHS)	Higher than S1
S3	Vector b	No	Normal + Tangential	Highest

4.5. Linear Extension Equation

Following the approaches described in Section 4.4 I use the coupling of elliptic equation and Laplace-Beltrami equation to define the extension equation. As already discussed, some of the common linear choices of an elliptic equations are Laplace equation, linear elasticity equations [74, 29] or its modifications.

In this section I am going to discuss one of the exmples of the extension equation: the equation based on the linear elasticity equations as well as its simplifications which I already briefly mentioned in Section 4.4. In [74], motivated by physical considerations mesh deformation was suggested to be modeled by solving a linear elasticity problem with variable second Lamé parameter μ_{elas}.

$$\begin{aligned} \operatorname{div}(\sigma(w)) &= 0 && \text{in } \Omega \\ w &= 0 && \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \\ \sigma(w) \cdot n &= -c && \text{on } \Gamma_{\text{obs}}, \end{aligned} \tag{4.92}$$

where

$$\begin{aligned} \sigma &:= \lambda_{\text{elas}} \operatorname{Tr}(\epsilon) I + 2\mu_{\text{elas}} \epsilon, \\ \epsilon &:= \frac{1}{2}\left(\nabla\omega + \nabla\omega^T\right), \end{aligned} \tag{4.93}$$

are the stress and strain tensors, respectively. Here λ_{elas} and μ_{elas} denote the Lamé parameters, which are defined via Young's modulus E and Poisson's ratio ν as follows

$$\lambda_{\text{elas}} = \frac{\nu E}{(1+\nu)(1-2\nu)}, \tag{4.94}$$

$$\mu_{\text{elas}} = \frac{E}{2(1+\nu)}. \tag{4.95}$$

I derive weak form of elasticity problem (4.92) using sufficiently smooth test functions r:

Problem 4.5.1. *Find $w \in U$ such that*

$$\int_{\Omega} \sigma(\omega) : \nabla r \, dx - \int_{\Gamma_{obs}} d \cdot r \, ds = 0, \quad \forall r \in U, \tag{4.96}$$

where

$$U := \{u \in H^1(\Omega) : u = 0 \text{ on } \Gamma_{\text{wall}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}}\}. \tag{4.97}$$

In [74] some results are presented for $\lambda_{\text{elas}} = 0$ and $\mu_{\text{elas}} \in [\mu_{textmin}, \mu_{textmax}] = [1, 500]$ smoothly decreasing from Γ_{obs} to the outer boundaries. Therefore, it was required to solve an additional Poisson equation to determine μ as an initial step:

$$\begin{aligned} \Delta\mu_{\text{elas}} &= 0 && \text{in } \Omega \\ \mu_{\text{elas}} &= 1 && \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \\ \mu_{\text{elas}} &= 500 && \text{on } \Gamma_{\text{obs}}. \end{aligned} \tag{4.98}$$

Remark 4.5.2. *In [74] the linear elasticity equation was used as a Riesz-representation of the shape derivative, whereas in [29] a slight modification of this method was suggested to use the stresses as the design parameters for the problem. One can simplify it further and use Riesz representation of the control $c \in L^2(\Gamma_{obs})$, since the mesh deformation is a function of the control variable as given in (4.92).*

Despite this approach to be leading to quite robust results, a lot of parameter tuning and adjustments to the setting of the problem was required. Apart from the above mentioned additional solve of Poisson, the problem was heavily dependent on the geometric dimension. Therefore, it was later proposed to further simplify this model and choose μ to be of a constant value, namely $\mu = 1$.

This way, I obtain a simple elliptic equation with the symmetrized derivative $\left(Dw + Dw^T\right)$ which leads to better mesh qualities in comparison to using only Dw, as was illustrated in [38].

4.6. Nonlinear Extension Equation

It is clear, that the structure of a shape space is highly nonlinear, since one can not easily define sums of shapes or linear combinations of them. Nevertheless, I am using the method of mappings with linear extension equation described in Section 4.5 to locally approximate the set of admissible shapes by linear function space of admissible deformations. Whereas linear extension equation successfully preserves mesh quality for many problems it is not suitable for solving optimization problems admitting large deformations. It turns out that the linearity of the approximation is a limiting factor for the linear extension equation. Therefore one aims to find such extension equation, and therefore the extension operator S that would extend the set of admissible shapes. The nonlinear operator suitable for modelling of large deformations was proposed in [63]. In this section I am going to explain the idea behind nonlinear extension operator and different modification of it.

I start by adopting the Strategy 1 (S1) described in Section 4.4 but instead of a linear elliptic equation I introduce an equation with a nonlinear advection term. Therefore, I define S in terms of the solution operator of the following coupled PDEs

$$\begin{aligned} b - \Delta_{\Gamma_{\text{obs}}} b &= cn && \text{on } \Gamma_{\text{obs}}, \\ -\operatorname{div}(\nabla w + \nabla w^\top) + \eta_{\text{ext}}(w \cdot \nabla) w &= 0 && \text{in } \Omega, \\ (\nabla w + \nabla w^\top) \cdot n &= b && \text{on } \Gamma_{\text{obs}} \\ w &= 0 && \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \end{aligned} \tag{4.99}$$

In the equation above $\Delta_{\Gamma_{\text{obs}}}$ denotes the vector-valued Laplace-Beltrami operator, $\eta_{\text{ext}} \geq 0$ a scalar which controls how much influence has the nonlinear term. A detailed discussion about the choice of η_{ext} follows in Section 5.2.5. By solving (4.99) the scalar-valued control $c \in L^2(\Gamma_{\text{obs}})$ is mapped to a vector-valued quantity $b \in H^2(\Gamma_{\text{obs}})$.

The main idea is that having an advective term in the extension equation allows for larger deformation, as it admits the movement of the nodes on the boundary in the direction of the flow, as shown in Section 5.2.5.

4.6.1. Weak Form

The weak formulation of Laplace-Beltrami equation, i.e. the first equation of (4.99) is given by (4.90).

For the second equation in (4.99), which specifies the mapping from vector-valued function b to the domain deformation, I follow the argumentation in [38] and obtain the weak formulation in the space

$$W := \left\{ w \in H^{\frac{7}{2}}(\Omega, \mathbb{R}^d) : w|_{\Gamma_{\text{in}} \cup \Gamma_{\text{wall}} \cup \Gamma_{\text{out}}} = 0 \,\text{a.e.} \right\}$$

as follows:

Find $w \in W$ such that

$$\int_\Omega (Dw + Dw^\top) : D\delta_w + \eta_{\text{ext}}(Dw\, w) \cdot \delta_w \, dx = \int_{\Gamma_{\text{obs}}} b\delta_w \, ds \tag{4.100}$$

for all $\delta_w \in W$ and in terms of $\eta_{\text{ext}} \geq 0$.

Here $D_{\Gamma_{\text{obs}}}$ denotes the derivative tangential to Γ_{obs}. In (4.90) the scalar-valued boundary control c is multiplied with the outer normal vector field n to Ω at Γ_{obs}. Then a vector-valued Laplace-Beltrami equation is solved over Γ_{obs}. This is coupled with (4.100) where the influence of the advection term is controlled via η_{ext}.

It is important to emphasize that the deformation field w, used to define the domain transformation $\mathcal{F} = \mathrm{id} + w$, must possess sufficient regularity to ensure that the transformed domain $\mathcal{F}(\Omega)$ remains a valid computational domain. Specifically, the transformation $\mathcal{F}$ must be at least $\mathcal{C}^1$-smooth in order for the Jacobian determinant $\det(D\mathcal{F})$ to be well-defined and continuous. This is essential for enforcing geometric constraints such as volume and barycenter preservation, and for avoiding mesh degeneration during the optimization process.

To ensure $\mathcal{C}^1$-regularity of $\mathcal{F}$, it is required that $w \in H^s(\Omega)^d$ for some $s > \frac{d}{2} + 1$, where $d \in \{2, 3\}$ denotes the spatial dimension. By Sobolev embedding (Theorem 3.1.3), this guarantees $w \in \mathcal{C}^1(\Omega)^d$. Choosing $s = \frac{7}{2}$ satisfies this condition in both two and three dimensions and provides a sufficient regularity margin. This justifies the use of the space $H^{7/2}(\Omega)^d$ in the formulation, as proposed in [38]. Moreover, this choice is compatible with elliptic regularity theory, which underpins the PDE-based construction of the deformation field from lower-regularity control variables.

In the setting of the method of mappings, the intermediate variable b arises as the solution to a Laplace-Beltrami equation on the design boundary Γ_d. Since the control variable c is taken from $L^2(\Gamma_d)$, elliptic regularity results on smooth, compact manifolds imply that the corresponding solution satisfies $b \in H^2(\Gamma_d)$. This additional regularity is crucial because b serves as boundary data for an elliptic extension equation in the domain Ω, used to generate the deformation field w. In order to obtain $w \in H^{\frac{7}{2}}(\Omega)^d$, and thereby ensure that the transformation $\mathcal{F}$ is at least $\mathcal{C}^1$, it is necessary that b lies in $H^2(\Gamma_d)$. This approach provides a practical way to achieve the required smoothness while avoiding the use of H^2-conforming finite elements, which are computationally demanding. Instead, regularity is introduced through elliptic smoothing via PDEs, enabling an efficient and flexible implementation.

4.6.2. Optimality System

In this section I derive the optimality system using the Lagrangian approach. I start with defining the Lagrangian of Problem 4.3.1 on the reference domain and then derive corresponding optimality conditions.

I proceed utilizing the *optimize-then-discretize* approach. I am following Jameson [46] in adopting the Lagrange multiplier viewpoint for design optimization because of its connection to constrained optimization and optimal control theory.

I am using nonlinear extension equation for the computations in this section since, as mentioned earlier, it would also include linear case, if extension factor is zero.

Following the discussion of Section 3.1.4, I now define the Lagrangian for the system:

$$
\begin{aligned}
&\mathcal{L}(v,p,w,b,c,\psi_v,\psi_p,\psi_w,\psi_b,\psi_{\text{vol}},\psi_{\text{bc}}) = \\
&\qquad \frac{\nu}{2}\int_\Omega \left(Dv(DF)^{-1}\right) : \left(Dv(DF)^{-1}\right)\det(DF)\,dx + \frac{\alpha}{2}\int_{\Gamma_{\text{obs}}} c^2\,ds \\
&\qquad + \frac{\beta}{2}\int_\Omega ((\eta_{\text{det}} - \det(DF))_+)^2\,dx \\
&\qquad - \int_\Omega \Big[\nu\left(Dv(DF)^{-1}\right) : \left(D\psi_v(DF)^{-1}\right) + (Dv(DF)^{-1}v)\cdot\psi_v \\
&\qquad - p\,\text{Tr}\left(D\psi_v(DF)^{-1}\right)\Big]\det(DF)\,dx \\
&\qquad + \int_\Omega \psi_p\,\text{Tr}\left(Dv(DF)^{-1}\right)\det(DF)\,dx \\
&\qquad - \int_\Omega (Dw + Dw^\top) : D\psi_w + \eta_{\text{ext}}(Dw\,w)\cdot\psi_w\,dx + \int_{\Gamma_{\text{obs}}} b\cdot\psi_w\,ds \\
&\qquad - \int_{\Gamma_{\text{obs}}} b\cdot\psi_b + D_{\Gamma_{\text{obs}}}b : D_{\Gamma_{\text{obs}}}\psi_b\,ds + \int_{\Gamma_{\text{obs}}} cn\cdot\psi_b\,ds \\
&\qquad - \psi_{\text{bc}}\cdot\int_\Omega (x+w)\det(DF)\,dx - \psi_{\text{vol}}\int_\Omega \det(DF) - 1\,dx,
\end{aligned}
\tag{4.101}
$$

where $\psi_\cdot$ are the Lagrange multipliers. For simplicity, in the following derivations I will use the notation $\mathcal{L}$ for $\mathcal{L}(v,p,w,b,c,\psi_v,\psi_p,\psi_w,\psi_b,\psi_{\text{vol}},\psi_{\text{bc}})$. I also denote by $\mathcal{L}_*$ the Gâteaux derivative of Lagrangian $\mathcal{L}$ in the direction $*$.

Note that for barycenter and volume condition (4.60) and (4.61) the multipliers are $\psi_{\text{vol}} \in \mathbb{R}$ and $\psi_{\text{bc}} \in \mathbb{R}^d$. However there is no associated variable with them since they are finite dimensional.

Further, I calculate directional derivatives with respect to all variables in Lagrangian. Denote $\langle\cdot:\cdot\rangle$ the Frobenius inner product of two matrices, $\langle\cdot,\cdot\rangle$ the inner product of two vectors, and $(\cdot,\cdot)$ the H^1 inner product.

Lemma 4.6.1. *The first-order optimality system associated with the Lagrangian $\mathcal{L}$ defined in (4.101), at the point $(v,p,w,b,c,\psi_v,\psi_p,\psi_w,\psi_b,\psi_{vol},\psi_{bc})$ is characterized by the derivatives*

$\mathcal{L}_v, \mathcal{L}_p, \mathcal{L}_w, \mathcal{L}_b, \mathcal{L}_{\psi_v}, \mathcal{L}_{\psi_p}, \mathcal{L}_{\psi_w}, \mathcal{L}_{\psi_b}, \mathcal{L}_c, \mathcal{L}_{\psi_{vol}}, \mathcal{L}_{\psi_{bc}}$ *as follows*

$$\begin{aligned}\mathcal{L}_v\delta_v =& \nu \int_\Omega \left(D\delta_v(DF)^{-1}\right) : \left(Dv(DF)^{-1}\right)\det(DF)\,dx \\ &- \nu \int_\Omega \left(D\delta_v(DF)^{-1}\right) : \left(D\psi_v(DF)^{-1}\right)\det(DF)\,dx \\ &- \int_\Omega D\delta_v(DF)^{-1}v)\cdot\psi_v \det(DF)\,dx \\ &- \int_\Omega (Dv(DF)^{-1}\delta_v)\cdot\psi_v \det(DF)\,dx \\ &+ \int_\Omega \psi_p \operatorname{Tr}\left(D\delta_v(DF)^{-1}\right)\det(DF)\,dx \\ =&0,\end{aligned} \tag{4.102}$$

$$\begin{aligned}\mathcal{L}_{\psi_v}\delta_{\psi_v} =& -\nu \int_\Omega \left(Dv(DF)^{-1}\right) : \left(D\delta_{\psi_v}(DF)^{-1}\right)\det(DF)\,dx \\ &- \int_\Omega (Dv(DF)^{-1}v)\cdot\delta_{\psi_v}\det(DF)\,dx \\ &+ \int_\Omega p \operatorname{Tr}\left(D\delta_{\psi_v}(DF)^{-1}\right)\det(DF)\,dx \\ =&0,\end{aligned} \tag{4.103}$$

$$\begin{aligned}\mathcal{L}_p\delta_p =& -\int_\Omega \delta_p \operatorname{Tr}\left(D\psi_v(DF)^{-1}\right)\det(DF)\,dx \\ =&0,\end{aligned} \tag{4.104}$$

$$\begin{aligned}\mathcal{L}_{\psi_p}\delta_{\psi_p} =& \int_\Omega \delta_{\psi_p} \operatorname{Tr}\left(Dv(DF)^{-1}\right)\det(DF)\,dx \\ =&0,\end{aligned} \tag{4.105}$$

$$\begin{aligned}\mathcal{L}_b\delta_b =& \int_{\Gamma_{obs}} \delta_b\cdot\psi_w\,ds - \int_{\Gamma_{obs}} \delta_b\cdot\psi_b + D_{\Gamma_{obs}}\delta_b : D_{\Gamma_{obs}}\psi_b\,ds \\ =&0,\end{aligned} \tag{4.106}$$

$$\begin{aligned}\mathcal{L}_{\psi_b}\delta_{\psi_b} =& -\int_{\Gamma_{obs}} b\cdot\delta_{\psi_b} + D_{\Gamma_{obs}}b : D_{\Gamma_{obs}}\delta_{\psi_b}\,ds + \int_{\Gamma_{obs}} cn\cdot\delta_{\psi_b}\,ds \\ =&\,0,\end{aligned} \tag{4.107}$$

$$\mathcal{L}_c\delta_c = \alpha\int_{\Gamma_{obs}} c\delta_c\,ds + \int_{\Gamma_{obs}} \delta_c n\cdot\psi_b\,ds = 0, \tag{4.108}$$

$$\begin{aligned}\mathcal{L}_{\psi_{vol}}\delta_{\psi_{vol}} =& -\delta_{\psi_{vol}}\int_\Omega \det(DF) - 1\,dx \\ =&\,0,\end{aligned} \tag{4.109}$$

$$\begin{aligned}\mathcal{L}_{\psi_{bc}}\delta_{\psi_{bc}} &= -\delta_{\psi_{bc}} \cdot \int_\Omega (x+w)\,\det(DF)\,dx \\ &= 0,\end{aligned} \tag{4.110}$$

I derive the derivatives of Lagrangian with respect of w term by term:

$$\begin{aligned}&\langle D_w \frac{\nu}{2}\int_\Omega \Big(Dv(DF)^{-1}\Big) : \Big(Dv(DF)^{-1}\Big)\det(DF)\,dx,\ \delta_w\rangle \\ =&-\nu\int_\Omega (Dv(DF)^{-1}) : (Dv(DF)^{-1}D\delta_w(DF)^{-1})\det(DF)\,dx \\ &+\frac{\nu}{2}\int_\Omega (Dv(DF)^{-1}) : (Dv(DF)^{-1})\,\mathrm{Tr}\Big((DF)^{-1}D\delta_w\Big)\det(DF)\,dx\end{aligned} \tag{4.111}$$

$$\begin{aligned}&\langle D_w \frac{\beta}{2}\int_\Omega ((\eta_{det}-\det(DF))_+)^2\,dx,\ \delta_w\rangle \\ &=-\beta\int_\Omega (\eta_{det}-\det(DF))_+\,\mathrm{Tr}\Big((DF)^{-1}D\delta_w\Big)\det(DF)\,dx\end{aligned} \tag{4.112}$$

$$\begin{aligned}&\langle D_w \int_\Omega -\nu\Big(Dv(DF)^{-1}\Big) : \Big(D\psi_v(DF)^{-1}\Big)\det(DF)\,dx,\ \delta_w\rangle \\ =&-\nu\int_\Omega (Dv(DF)^{-1}) : (D\psi_v(DF)^{-1})\,\mathrm{Tr}\Big((DF)^{-1}D\delta_w\Big)\det(DF)\,dx \\ &-\nu\int_\Omega (Dv(DF)^{-1}) : (-D\psi_v(DF)^{-1}D\delta_w(DF)^{-1})\det(DF)\,dx \\ &-\nu\int_\Omega (Dv(-(DF)^{-1})D\delta_w(DF)^{-1}) : (D\psi_v(DF)^{-1})\det(DF)\,dx\end{aligned} \tag{4.113}$$

(4.114)

$$\begin{aligned}&\langle D_w \int_\Omega -(Dv(DF)^{-1}v)\cdot\psi_v\det(DF)\,dx,\ \delta_w\rangle \\ =&-\int_\Omega (Dv(DF)^{-1}v)\cdot\psi_v\,\mathrm{Tr}\Big((DF)^{-1}D\delta_w\Big)\det(DF)\,dx \\ &-\int_\Omega (Dv(-(DF)^{-1})D\delta_w(DF)^{-1}v)\cdot\psi_v\det(DF)\,dx\end{aligned} \tag{4.115}$$

$$\langle D_w \int_\Omega p \operatorname{Tr}\left(D\psi_v (DF)^{-1}\right) \det(DF)\, dx,\ \delta_w \rangle \tag{4.116}$$

$$= \int_\Omega p \operatorname{Tr}\left(D\psi_v (DF)^{-1}\right) \operatorname{Tr}\left((DF)^{-1} D\delta_w\right) \det(DF)\, dx \tag{4.117}$$

$$- \int_\Omega p \operatorname{Tr}\left(D\psi_v (DF)^{-1} D\delta_w (DF)^{-1}\right) \det(DF)\, dx$$

$$\langle D_w \int_\Omega \psi_p \operatorname{Tr}\left(Dv (DF)^{-1}\right) \det(DF)\, dx,\ \delta_w \rangle$$

$$= \int_\Omega \psi_p \operatorname{Tr}\left(Dv (DF)^{-1}\right) \operatorname{Tr}\left((DF)^{-1} D\delta_w\right) \det(DF)\, dx \tag{4.118}$$

$$- \int_\Omega \psi_p \operatorname{Tr}\left(Dv (DF)^{-1} D\delta_w (DF)^{-1}\right) \det(DF)\, dx$$

$$\langle D_w \int_\Omega -(Dw + Dw^\top) : D\psi_w\, dx,\ \delta_w \rangle$$

$$= - \int_\Omega (D\delta_w + D\delta_w{}^\top) : D\psi_w\, dx \tag{4.119}$$

$$\langle D_w \int_\Omega -\eta_{ext}(Dw\, w) \cdot \psi_w\, dx,\ \delta_w \rangle$$

$$= - \int_\Omega (\eta_{ext}((D\delta_w\, w) + (Dw\, \delta_w)) \cdot \psi_w\, dx \tag{4.120}$$

$$\langle D_w \left(-\psi_{bc} \cdot \int_\Omega (x + w) \det(DF)\, dx\right),\ \delta_w \rangle$$

$$= -\psi_{bc} \cdot \int_\Omega \delta_w \det(DF) + (x + w) \operatorname{Tr}\left((DF)^{-1} D\delta_w\right) \det(DF)\, dx \tag{4.121}$$

$$\langle D_w \left(-\psi_{vol} \int_\Omega \det(DF) - 1\, dx\right),\ \delta_w \rangle$$

$$= -\psi_{vol} \int_\Omega \operatorname{Tr}\left((DF)^{-1} D\delta_w\right) \det(DF)\, dx \tag{4.122}$$

$$
\begin{aligned}
\mathcal{L}_w \delta_w = &- \nu \int_\Omega (Dv(DF)^{-1}) : (Dv(DF)^{-1} D\delta_w (DF)^{-1}) \det(DF)\, dx \\
&+ \frac{\nu}{2} \int_\Omega (Dv(DF)^{-1}) : (Dv(DF)^{-1}) \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&- \beta \int_\Omega (\eta_{det} - \det(DF))_+ \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&+ \nu \int_\Omega (Dv(DF)^{-1} D\delta_w (DF)^{-1}) : (D\psi_v (DF)^{-1}) \det(DF)\, dx \\
&+ \nu \int_\Omega (Dv(DF)^{-1}) : (D\psi_v (DF)^{-1} D\delta_w (DF)^{-1}) \det(DF)\, dx \\
&- \nu \int_\Omega (Dv(DF)^{-1}) : (D\psi_v (DF)^{-1}) \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&+ \int_\Omega (Dv(DF)^{-1} D\delta_w (DF)^{-1} v) \cdot \psi_v \det(DF)\, dx \\
&- \int_\Omega (Dv(DF)^{-1} v) \cdot \psi_v \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&- \int_\Omega p \operatorname{Tr}\Big(D\psi_v (DF)^{-1} D\delta_w (DF)^{-1}\Big) \det(DF)\, dx \\
&+ \int_\Omega p \operatorname{Tr}\Big(D\psi_v (DF)^{-1}\Big) \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&+ \int_\Omega \psi_p \operatorname{Tr}\Big(Dv(DF)^{-1} D\delta_w (DF)^{-1}\Big) \det(DF)\, dx \\
&- \int_\Omega \psi_p \operatorname{Tr}\Big(Dv(DF)^{-1}\Big) \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&- \int_\Omega (D\delta_w + D\delta_w{}^\top) : D\psi_w + \eta_{ext}((D\delta_w\, w) + (Dw\, \delta_w)) \cdot \psi_w\, dx \\
&+ \beta \int_\Omega (\eta_{det} - \det(DF))_+ \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&- \psi_{bc} \cdot \int_\Omega \delta_w \det(DF) + (x + w) \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&- \psi_{vol} \int_\Omega \operatorname{Tr}\Big((DF)^{-1} D\delta_w\Big) \det(DF)\, dx \\
&= 0,
\end{aligned}
\tag{4.123}
$$

$$\begin{aligned}\mathcal{L}_{\psi_w}\delta_{\psi_w} = &-\int_\Omega (Dw + Dw^\top) : D\delta_{\psi_w} + \eta_{ext}(Dw\, w)\cdot\delta_{\psi_w}\, dx \\ &+ \int_{\Gamma_{obs}} b\cdot\delta_{\psi_w}\, ds \\ &= 0,\end{aligned} \tag{4.124}$$

for all test functions $\delta_w \in W$, $\delta_v \in V$, $\delta_p \in Q$, $\delta_b \in H^2(\Gamma_{obs})$, $\delta_{\psi_w} \in W$, $\delta_{\psi_v} \in V_0$, $\delta_{\psi_p} \in Q$, $\delta_{\psi_b} \in H^2(\Gamma_{obs})$, $\delta_c \in L^2(\Gamma_{obs})$, $\delta_{\psi_{vol}} \in \mathbb{R}$, *and* $\delta_{\psi_{bc}} \in \mathbb{R}^d$.

Proof. Firstly, I derive some basic auxiliary derivatives, using some of the properties listed in Appendix A

$$\langle [\det(DF)]_\omega, \delta_w\rangle = \frac{\partial \det(DF)}{\partial w}\delta_w = \det(DF)\,\mathrm{Tr}\Big((DF)^{-1}\, D\delta_w\Big), \tag{4.125}$$

$$\langle [DF_\omega]_\omega, \delta_w\rangle = D\delta_w, \tag{4.126}$$

$$\left\langle \left[(DF)^{-1}\right]_\omega, \delta_w\right\rangle = \frac{\partial (DF)^{-1}}{\partial w}\delta_w = -(DF)^{-1}\,(D\ \delta_w)\,(DF)^{-1}, \tag{4.127}$$

$$\begin{aligned}\left[\left\langle \left[(DF)^{-1}\right]_\omega, \delta_w\right\rangle\right]^T &= \left\langle \left[-(DF)^{-1}\,(D\delta_w)\,(DF_\omega)^{-T}\right]_\omega, \delta_w\right\rangle \\ &= -\Big((DF)^{-1}\, D\delta_w\,(DF_\omega)^{-1}\Big)^T \\ &= -\Big(\nabla\delta_w\,(DF)^{-1}\Big)^T (DF)^{-T} \\ &= -(DF)^{-T}\,(D\delta_w)^T\,(DF)^{-T},\end{aligned} \tag{4.128}$$

$$\begin{aligned}\left\langle \left[\mathrm{Tr}\Big((DF)^{-1}\, D\delta_w\Big)\right]_\omega, \mu_w\right\rangle &= -D{\delta_w}^T : (DF)^{-1} D\mu_w (DF)^{-1} \\ &= \mathrm{Tr}\Big(-D\delta_w (DF)^{-1} D\mu_w (DF)^{-1}\Big) \\ &= \mathrm{Tr}\Big(-(DF)^{-1} D\mu_w (DF)^{-1} D\delta_w\Big).\end{aligned} \tag{4.129}$$

Further, to derive the derivative of the penalty term I use the following

$$\begin{aligned}\frac{\partial}{\partial w}((\eta_{\det} - \det(DF))_+)^2(w)\,\delta_w &= 2(\eta_{\det} - \det(DF))_+\,\chi_{\{\eta_{\det} > \det(DF)\}}\,\frac{\partial}{\partial w}\det(DF)(w)\delta_w \\ &= 2(\eta_{\det} - \det(DF))_+\,\mathrm{Tr}\Big((DF)^{-1}\delta_w\Big)\det(DF).\end{aligned} \tag{4.130}$$

Using expressions (4.125) to (4.130), standard rules of differentiation and applying definition of a domain transformation mapping $F = \mathrm{id} + w$ following Section 3.3, I obtain the derivatives given by Eqs. (4.102) to (4.124).

□

As a result of Lemma 4.6.1, I obtain a system of nonlinear partial differential equations characterizing the stationary points of the Lagrangian. This system can then be discretized and solved using a suitable numerical method. In this thesis, I employ the FEM for discretization and numerical solution due to its flexibility in handling complex geometries and boundary conditions.

The theoretical framework developed in this chapter provides the foundation for the computational approach that follows. In particular, it establishes the structure of the optimization problem, the necessary optimality conditions, and the analytical tools required to ensure well-posedness and consistency.

In the next chapter, I focus on the numerical implementation of the proposed model. I describe the discretization strategy, discuss software choices, and present simulation results that illustrate the practical performance of the method. Through these numerical experiments, I aim to validate the theoretical insights and demonstrate the effectiveness of the approach in solving shape and control optimization problems governed by PDEs.

5. Numerical Simulation of Shape Optimization Problems

5.1. FEniCS Project

In this section, I describe the key components of the FEniCS Project and give a brief overview of its main features. For a comprehensive introduction, I refer the reader to the FEniCS book [55] and the UFL documentation [1].

The FEniCS Project is a widely used open-source platform for solving PDEs using the FEM. One of its distinguishing features is the seamless integration of automatic differentiation (AD), enabling efficient and accurate computation of derivatives of variational forms. This capability is particularly valuable in applications such as PDE-constrained optimization, inverse problems, and sensitivity analysis.

As noted by the developers [53], the FEniCS Project was founded in 2003 with the ambitious goal of automating the entire finite element pipeline for computational modeling. This includes:

- Discretization of variational forms,
- Numerical solution of discrete systems,
- A posteriori error estimation and control,
- High-level modeling of PDEs, and
- Optimization and control of PDE-constrained systems.

Figure 5.1 displays the most important components of FEniCS and their relations. Solid lines denote dependencies and dashed lines represent data flow. The workflow is the following: Variational forms written in UFL are passed to the compiler FFC and then generated into UFC code based on C++ code, which is then used by DOLFIN for assembly of linear systems. Further, for FFC the code generation is done using finite element backend FIAT.

DOLFIN has two interfaces: One is implemented as a traditional C++ library, and another interface is implemented as a standard Python module. Both interfaces

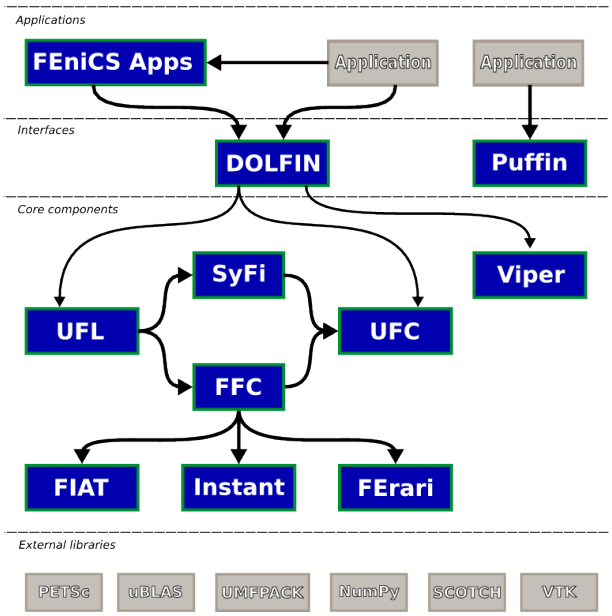


Figure 5.1.: FEniCS system architecture with DOLFIN as the main user interface component[53]

support parallel computations using multiple threads on a single node, using multiple nodes communicating via MPI and their combination.

Further, DOLFIN incorporates calls to external linear algebra software libraries such as PETSc [6], SCOTCH [65], uBLAS[84], etc.

DOLFIN provides a range of linear algebra objects and functionality, including vectors, dense and sparse matrices, direct and iterative linear solvers and eigenvalues solvers, and does so via a simple and consistent interface. It provides classes such as Matrix, Vector, Mesh, FiniteElement, FunctionSpace and Function.

5.1.1. Unified Form Language

At the core of FEniCS Project is the Unified Form Language (UFL)- a domain specific language for defining variational forms of differential equations in a notation similar to the standard analytical derivation [1]. It provides a level of abstraction that allows quickly and effortlessly translating underlying mathematics into the source code.

UFL consists of a set of operators and atomic expressions that can be used to represent variational forms and functionals. The library allows creating abstract syntax tree representations of variational problems, which enables automatic differentiation of

forms and expressions and allows other software libraries to generate concrete low-level implementations.

It is built on top of the Python language, and any Python code is valid in the definition of a form. UFL is a part of the FEniCS Project and is usually used in combination with other components from this project to compute solutions to PDEs. Since UFL does not provide a problem solving environment, it is tightly linked with the form compiler, e.g FFC or SFC [8] for generation of low-level code using UFL as its end-user interface, producing implementations of the UFC interface as an output.

UFL has been developed as a richer form language, especially for expressing nonlinear PDEs which would allow automatic differentiation of expressions and forms. Further it inherits the typical mathematical operations that are performed on variational forms, as well as basic algebraic operators such as transpose, determinant, inverse, trigonometric functions and elementary functions.

Additionally, UFL has already predefined names for the most common finite elements and allows to represent general hierarchies of mixed finite elements. Finite elements are defined by a *family*, *cell* or *mesh* and *polynomial degree*. The argument *family* is a string which defines the type of basis used, i.e. "Lagrange" or "CG" represent standard scalar Lagrange elements. UFL also supports notation from the Periodic Table of Finite Elements [3]. Apart from a scalar FiniteElement, UFL supports a vector-valued VectorElement and a TensorElement for rank 2 tensors. Further one can combine standard elements to create mixed elements, since DOLFIN allows the generation of arbitrarily nested mixed function spaces. Some examples are given in Listing 5.1, Listing 5.2.

```
V = VectorElement("CG", mesh.ufl_cell(), 2)
P = FiniteElement("CG", mesh.ufl_cell(), 1)
TH = FunctionSpace(mesh, MixedElement([V, P]))
```

Listing 5.1: Examples of UFL standard finite elements declarations and definition of a mixed finite element.

```
element = FiniteElement("Lagrange", triangle, 1)
v = TestFunction(element)
u = TrialFunction(element)
f = Coefficient(element)

a = dot(grad(v), grad(u))*dx #bilinear form
L = v*f*dx #linear form
```

Listing 5.2: UFL definition of variational form for Poisson equation.

5.1.2. FIAT

The Finite element Automatic Tabulator (FIAT) [49, 48] is a Python library for the automatic tabulation of finite element basis functions over polynomial function spaces up to three spatial dimensions. Basically it allows finite elements with very complicated bases to be constructed automatically. FIAT automatically tabulates the basis functions required for finite element discretizations at given sets of points. Further, FIAT provides the basis function back-end for FFC and enables high-order H^1, $H(\mathrm{div})$ and $H(\mathrm{curl})$ elements.

Also FIAT evaluates the basis functions and their derivatives at points, associates the basis functions (or degrees of freedom) with topological facets of the domain such as its vertices, edges and its placement on the edges, provides rules for evaluating the degrees of freedom applied to arbitrary functions (needed for Dirichlet boundary conditions), computes quadrature schemes. [53].

5.1.3. Dolfin

DOLFIN (Dynamic Object-oriented Library for Finite element computation) is a C++/Python library that functions as the main user interface of FEniCS [54]. It hosts a large part of the computational infrastructure, making it the central problem-solving environment for finite element simulations.

It is a core component of FEniCS that contains the essential functionality for the FEM-based simulations of PDEs. It serves as the problem-solving environment within FEniCS, enabling users to define and solve complex mathematical models described by PDEs in an efficient and user-friendly manner.

DOLFIN defines and manages data structures required for handling computational meshes, geometries, and finite element assembly. It provides the necessary tools for assembling the discrete linear and nonlinear forms that arise during the finite element discretization of PDEs. Moreover, DOLFIN manages communication between FEniCS components, wrapping low-level functionality behind a clean, consistent interface. This enables users to focus on modeling and analysis rather than the underlying computational complexity.

5.1.4. Automatic Differentiation and Dolfin-adjoint

Since the optimization procedure is often dependent on gradient-based algorithms, one always needs to compute derivatives. However in case of complex problems, deriving the shape derivative manually can become very complicated and error-prone, especially for time-dependent or nonlinear PDEs. To overcome this issue one often utilizes the automated differentiation techniques. The idea is to consider a model as a sequence of elementary linear algebra operation (in case of low level AD tools) or consider each variational problem in the model as a single operation, which is the strategy adopted by Dolfin-adjoint.

Dolfin-Adjoint is a computational framework designed for the efficient solution of PDE-constrained optimization problems, leveraging algorithmic differentiation within the FEniCS Project. This tool uses pyadjoint to differentiate FEM models written in Dolfin and Firedrake [30]. The tool is automatically deriving adjoint models through high-level algorithmic differentiation, where the forward model is considered as a sequence of variational problems. Starting with the model represented in UFL, dolfin-adjoint computes first and second order shape derivatives, using the adjoint method for first-order derivatives, and a combination of the tangent linear method and the adjoint method for the second order derivatives.

Dolfin-Adjoint constructs a computational graph representing the sequence of operations used to solve a PDE problem which includes assembly of finite element matrices, application of boundary conditions, solver calls for obtaining the state variable.

5.1.5. Preconditioners

When solving linear systems arising from PDE-constrained optimization for incompressible flow problems, one frequently encounters saddle point problems of the form:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{5.131}$$

where $A \in \mathbb{R}^{n\times n}$ is typically symmetric positive-definite, $B \in \mathbb{R}^{m\times n}$ encodes constraints (e.g. divergence-free condition), and the block structure corresponds to a constrained optimization or incompressible flow setting. These systems are often indefinite and poorly conditioned, particularly as the mesh is refined or parameters vary, making preconditioning essential for efficient iterative solvers.

A common strategy is to use block preconditioners that exploit the structure of the saddle point system. One classical choice is the block diagonal or block triangular

preconditioner:

$$P = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} A & B^T \\ 0 & -S \end{bmatrix}, \tag{5.132}$$

where $S = BA^{-1}B^T$ is the (negative) Schur complement. If A is large, assembling or inverting S exactly is usually infeasible, and approximations are used instead. A common approximation for S in incompressible flow problems is the pressure mass matrix or scaled Laplacian.

The FEniCS Project provides basic support for block-structured preconditioners through PETSc and external packages like Hypre and MUMPS. However, custom preconditioners, particularly ones targeting the Schur complement, are not natively supported without modifying the solver stack.

In the experiments, I observed that applying GMRES with the default PETSc preconditioners leads to slow convergence due to poor conditioning and lack of structural awareness. In contrast, direct solvers like MUMPS perform better for moderately sized problems, as they handle the full coupling between blocks implicitly. However, they are not scalable to very large systems due to memory and computational demands. For more details I refer to Section 5.2.

The choice of preconditioner has a critical impact on the convergence rate of iterative solvers such as GMRES or MINRES. Without a tailored preconditioner, these solvers suffer from stagnation or slow convergence due to the indefiniteness of the system and the presence of near-zero eigenvalues.

In saddle point problems, effective preconditioning reduces the condition number of the system and clusters eigenvalues, enabling rapid convergence. For large-scale problems, the development of scalable and structure-aware preconditioners, especially those approximating the inverse of the Schur complement, is an active area of research.

For the problem investigated in this thesis, one could utilize the block structure not once, but two times: the forward system and the adjoint system, since the smooth part (4.123) is split from the other equations and therefore the solver can be tailored for this structure.

5.2. Numerical Results

In this section I will present numerical simulations for the Navier-Stokes shape optimization problem stated in the Section 3.2. The idea is to illustrate the proposed

approach to model mesh deformation using a non-trivial case study, without requiring excessive parameter tuning. This example is challenging because of a coupling between the velocity and pressure field and nonlinearity of the equation.

For each deformation of the domain $F(\Omega)$, the stationary incompressible Navier–Stokes equations described in Section 4.1 are solved using a finite element discretization. The analytical properties of the stationary Navier–Stokes equations (existence of weak solutions, uniqueness for sufficiently small data, and regularity) are classical, see [33, 82].

I will start by presenting the well-know results for the benchmark shape optimization problem for large deformations obtained with the usage of linear elasticity equation and discuss what challenges usually arise while employing the strategy of modelling deformation field using linear equation.

Then I will demonstrate the results for nonlinear extension operator and show how the optimal solution differs depending on the choice of parameters of the extension operator. In particular, I will vary the nonlinear factor to demonstrate how it affects the mesh and allows to reach shapes which were not easily accessible for the system with the linear operator.

To prove this point I will present the study of the mesh quality of the result for different initial shapes and parameters.

I will further discuss how the solution process of the overall system was affected by introducing this additional nonlinearity in the system.

Finally, I explain how the local injectivity can be extended to globally injective transformation mappings by adding an artificial volume to the aerodynamic specimen based on the examples given in [63].

5.2.1. General Setting

I consider flows in $\Omega = \{\Omega_{2d}, \Omega_{3d}\}$ corresponding to 2-dimensional and 3-dimensional (2D and 3D) settings, respectively, with the obstacle in the tunnel as described in Section 4.1.

To reduce the effect of the boundary on the obstacle, it is located far enough from the walls of the tunnel. As a result of preliminary simulations, the chosen setting was found to be suitable:

$$\Omega_{2d} = [-14, 14] \times [-14, 14],$$

$$\Omega_{3d} = \{x \in \mathbb{R}^3 : -7 \leq x_1 \leq 7, \left(x_2^2 + x_3^2\right)^{1/2} \leq 3\}.$$

In all examples the initial velocity of the inlet Γ_{in} is a parabolic flow:

$$v_\infty = (v_{max} - \frac{\|x\|_2^2}{\delta^2}, 0, ..., 0) \in \mathbb{R}^d, \tag{5.133}$$

where $v_{max} = 1.0$, δ is a diameter of the flow tunnel Ω. The obstacle Γ_{obs} is a circle of radius $r = 1$ or a sphere of radius $r = 0.5$ located in the centre of the tunnel, i.e. $\mathrm{bc}(\Omega_{\mathrm{obs}}) = 0 \in \mathbb{R}^d$.

The kinematic viscosity, unless stated otherwise, is set to $\nu = 0.02$, which corresponds to the Reynolds number

$$Re = \frac{v_{\mathrm{mean}} L}{\nu} = \frac{\frac{2}{3} \cdot 2}{0.02} \approx 67, \tag{5.134}$$

taking a fluid density of $\rho = 1.0$ into account. There, the characteristic length of the flow configuration is the diameter of the object perpendicular to the flow direction: $L = 2 \cdot r = 2$. I also consider gravity-free setting in all examples.

5.2.2. Optimization Approach

To solve the problem, I follow the so-called 'first-optimize-then-discretize' approach. Based on the finite element discretization, in this approach one starts with deriving first-order necessary optimality conditions for the problem. Then the discretization of every variable related to state, control and functionals is performed. In comparison with the 'first-discretize-then-optimize' approach, where the discretization of the adjoint variable is determined by the test space of the discrete state [40], following the 'first-optimize-then-discretize' approach I am free to choose the discretization for the adjoint variable. However, since I aim to have symmetry property, I nevertheless choose the same test spaces for state and adjoint variables, leading eventually to the same optimality system as in the 'first-discretize-then-optimize' approach. Therefore, I start with the earlier-derived optimality conditions (4.102) to (4.124) and discretize it.

The optimality system (4.123) to (4.110) is solved for a sequence of different values of Tikhonov regularization parameter α. Consider again the nonlinear shape optimization problem (4.79). As I already mentioned in (4.79) α is a regularization parameter. I consider a decreasing sequence of α_k, starting from $\alpha_0 = \alpha_{\mathrm{init}}$ until the desired level α_{target} is reached. This way I approximate the problem by a sequence of simpler nonlinear optimization problems for each k, where every solution $k-1$ serves as initial guess to the problem k. This procedure is carried until

the desirable very small value of α is reached. The information of the known values $y_k := (w, v, p, b, c, \psi_w, \psi_v, \psi_p, \psi_b, \psi_{\text{vol}}, \psi_{\text{bc}})_k$ is passed through the sequence of $k+1$ iterations starting from a linear problem for $k = 0$. I summarize this method in Algorithm 5.

Depending on the structure of the problem, physical properties and mesh size one needs to adjust parameters $\alpha_{\text{init}}, \alpha_{\text{dec}}$ and α_{target}. I describe the strategies and consideration about how they are chosen in Section 5.2.4 and illustrate their influence on the solution of the problem.

Since parts of the optimality system are non-differentiable, in particular the ones corresponding to the determinant condition (4.66) the semismooth Newton's method is applied. For the details about the method, please refer to [41, 86].

Algorithm 5 Direct optimization algorithm

Require: $0 < \alpha_{\text{target}} \leq \alpha_{\text{init}}$, $0 < \alpha_{\text{dec}} < 1$

$y_0 \leftarrow 0$
$k \leftarrow 0$
$\alpha_k \leftarrow \alpha_{\text{init}}$
while $\alpha_k \geq \alpha_{\text{target}}$ **do**
    Solve (4.123) to (4.110) for y_{k+1} with semismooth Newton's method with y_k as initial guess and regularization parameter α_k
    $\alpha_{k+1} \leftarrow \alpha_{\text{dec}} \alpha_k$
    $k \leftarrow k + 1$
end while

5.2.3. Iterative Optimization Algorithm

Whereas direct optimization algorithms like the one given by Algorithm 5 work well for most of the problems, they require a large number of solves for nonlinear, non-smooth optimality system. In addition, at every solution step one would need to solve a large system of linear equations arising from discretized PDEs. A common challenge encountered is their substantial memory consumption, especially for three-dimensional problems or fine discretizations. One would often need to factorize large matrices (especially when applying direct linear solvers, like MUMPS), which can lead to memory requirements that exceed the available resources, resulting in out-of-memory exceptions. This issue becomes particularly critical in high-resolution simulations or while considering time-dependent models.

Hence, it was proposed in [63] to utilize the structure of the problem and decouple the optimality system (4.123) to (4.110) as described in Algorithm 6.

The idea is to separate the solution process of the parts of the problem corresponding to state (4.103) and (4.105), adjoint (4.102) and (4.104) and deformation (4.106) to (4.110), (4.123) and (4.124) and solve them independently.

Algorithm 6 Iterative optimization algorithm

Require: $0 < \alpha_{\text{target}} \leq \alpha_{\text{init}}$, $0 < \alpha_{\text{dec}} < 1$, $0 < \epsilon$

$y_0 \leftarrow 0$
$k \leftarrow 0, \ell \leftarrow 0$
$\alpha_k \leftarrow \alpha_{\text{init}}$
while $\alpha_k \geq \alpha_{\text{target}}$ **do**
    repeat
        Set y_ℓ as initial guess
        Solve (4.103) and (4.105) for $(v,p)_{\ell+1}$
        Solve (4.102) and (4.104) for $\left(\psi_v, \psi_p\right)_{\ell+1}$
        Solve (4.106) to (4.110), (4.123) and (4.124) for $(w,b,c,\psi_w,\psi_b,\psi_{\text{vol}},\psi_{\text{bc}})_{\ell+1}$ with semismooth Newton's method and regularization parameter α_k
        $\ell \leftarrow \ell + 1$
    until $\frac{\|c_{\ell+1} - c_\ell\|_{L^2(\Gamma_{\text{obs}})}}{\|c_{\ell+1}\|_{L^2(\Gamma_{\text{obs}})}} < \epsilon$
    $\alpha_{k+1} \leftarrow \alpha_{\text{dec}} \alpha_k$
    $k \leftarrow k + 1$
end while

I remark that, although the subproblems are solved sequentially, they are not entirely independent due to the coupling inherent in the problem structure. As a result, the subproblem solves cannot be executed in parallel without affecting correctness. As outlined in Algorithm 6, the procedure begins with the solution of the direct and adjoint systems. The information obtained from these solves is then used to compute the update to the geometry.

However, the decoupling allows us to reuse existing solvers for the state equation and embed them into the shape optimization framework. Further, since now the underlying linear systems of each of the subproblems are much smaller, the memory requirement for linear solvers is significantly reduced. Moreover, in some cases, the subproblems can now be solved using iterative solvers with preconditioners as discussed in Section 5.1.5 since they do not require as good of initial guess as the general large problem.

The idea of Algorithm 6 is to have a fixed-point like approach. Analogously to Algorithm 5 I iterate over a regularization parameter α decreasing it by α_{dec} at every step and the solutions for the optimization problem k are used as the initial guess for the nonlinear solver in iteration $k+1$. However, whereas in the direct approach I solved each of the problem k directly, now I approximate each of them by a fixed-point

iteration. So now, for each problem k I construct an approximation which consists of three subproblems (corresponding to each part of the decoupled optimality system) and solve them one by one.

The numerical experiments and comparisons of the computational times for both full system obtained via Algorithm 5 and decoupled one from Algorithm 6 are presented in Section 5.2.

5.2.4. Linear Elasticity

As discussed in Section 4.5, one of the common ways to represent shape derivative is by using the linear elasticity equation as its Riesz representation.

Following the procedure in Algorithm 5, I aim to solve the problem described in Section 5.2.1 starting from the "simplest" optimization problem with a large regularization parameter α_{init}. I choose $\alpha_{\text{init}} = 1, \alpha_{\text{dec}} = 0.5, \beta = 0$.

For the numerical experiment the domain is discretized into 5136 elements. The problem is then solved for the stationary Navier-Stokes flow with $\nu = 0.04$. The velocity field of the state equation over the reference and final domains are shown in Figure 5.2.

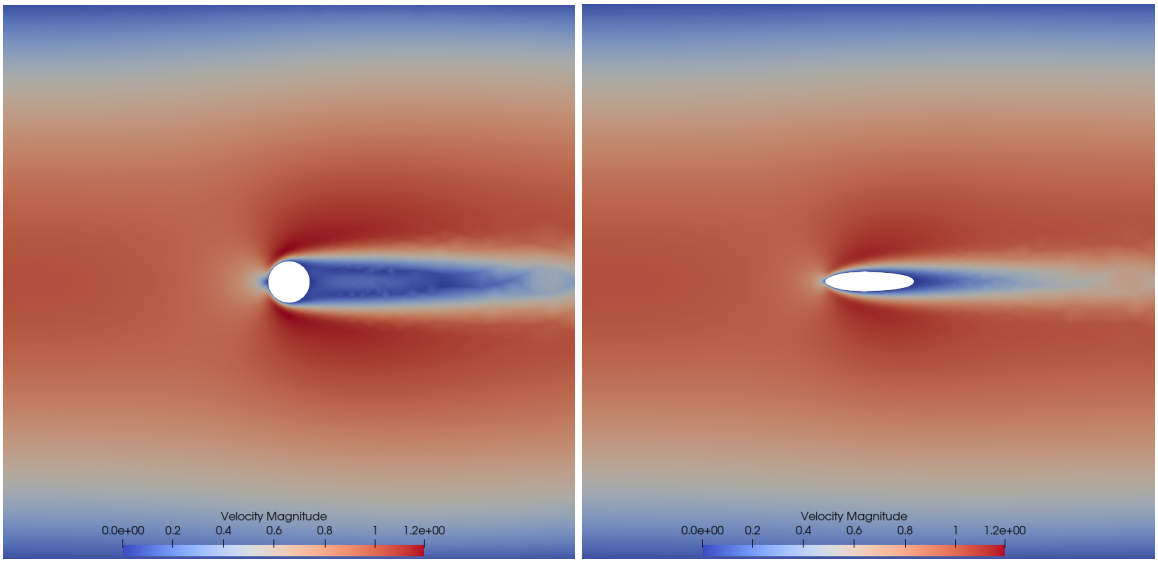


Figure 5.2.: Magnitude of velocity v computed on reference Ω (left) and final $F(\Omega)$ (right) domains for linear elasticity extension equation, Section 5.2.4.

The objective is to solve the Problem 3.3.2 for large deformation, i.e. small ν. However,

if the viscosity is chosen to be small, the problem becomes more advection-dominated, which results in more elongated shape of the obstacle. So with every iteration, boundary nodes move significantly, causing boundary triangles to deform, which is then propagated to the rest of the domain with the linear elasticity equation. This results in highly compressed triangles at the tips of the obstacle. Finally, at some point the mesh degenerates to the point the solver can not proceed with the iteration process anymore.

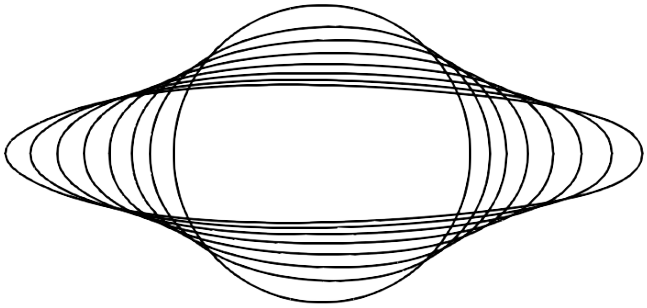


Figure 5.3.: Domain Ω for each F using the linear extension equation, Section 5.2.4.

The solution process was terminated after 7 iterations (0-6) without reaching the desired tolerance of 10^{-5} due to mesh degeneration, as shown in Figure 5.4. The left figure shows the mesh at the last iteration, with elongated triangles near the tip, and the right figure shows the corresponding element quality. The deformations at each iteration step are shown in Figure 5.3.

Figure 5.4.: Mesh and mesh quality on the last iteration step ($it = 6$) for the linear extension equation, Section 5.2.4.

It is known that the optimal solution for this problem features two kinks, which are not visible in Figure 5.2, as the optimization process was terminated prematurely before reaching the optimal shape.

Nevertheless, the drag was successfully reduced from 6.461945×10^{-1} to 5.583938×10^{-1} during the optimization, corresponding to a relative improvement of 13.59%,

as detailed in Table 5.1. At each iteration, a halving step size $\alpha = 2^{-\text{it}}$ was applied, and the computational time per iteration stabilized around 10 seconds after the first iteration.

The mesh quality, quantified by the minimum radius ratio (RR), gradually deteriorated as the optimization progressed, as expected for this case of large deformation. The RR increased from 1.6241 in the initial mesh to 9.4018 by the final iteration, at which point element distortion became significant and overlapping at the tip of the obstacle began to occur.

Table 5.1.: Cost functional values for each optimization step using the linear extension equation, Section 5.2.4.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	6.461945e-01	0.00	15.77	1.6241
1	2^{-1}	6.287758e-01	2.70	10.76	1.6250
2	2^{-2}	6.098816e-01	5.62	10.55	1.6262
3	2^{-3}	5.922243e-01	8.35	10.45	1.8967
4	2^{-4}	5.774753e-01	10.63	10.38	2.1064
5	2^{-5}	5.662434e-01	12.37	10.41	3.5385
6	2^{-6}	5.583938e-01	13.59	10.47	9.4018

This loss in mesh quality ultimately limited the number of admissible optimization steps and prevented convergence to the true optimal shape. To address this issue and allow for larger shape updates while maintaining mesh integrity, the nonlinear extension equation was introduced as an improved mesh deformation strategy. The following section presents the corresponding results and highlights the improvements achieved in terms of both mesh quality and optimization performance.

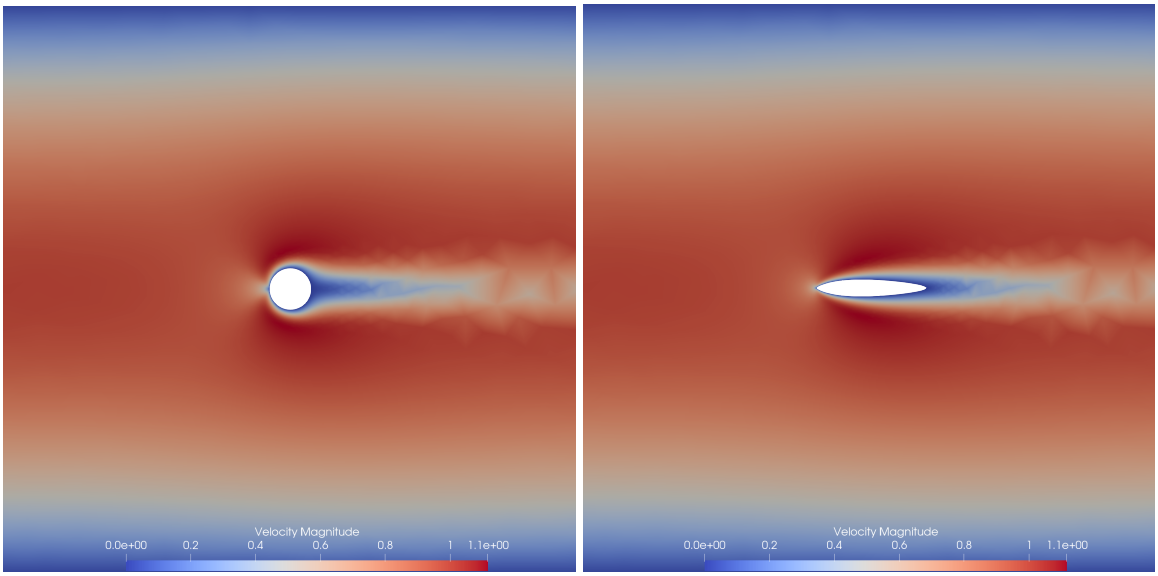
5.2.5. Nonlinear Equation

This subsection investigates how the solution is affected by introducing a nonlinear term into the extension equation.

The nonlinear term to the extension equation is added as described in Section 4.4 using the extension factor of $\eta_{\text{ext}} = 3.0$. All other parameters are kept consistent with the setting outlined in Section 5.2.4: the computational domain consists of 5136 elements, $\alpha_{\text{init}} = 1, \alpha_{\text{dec}} = 0.5, \beta = 0$, and a viscosity coefficient of $\nu = 0.04$.

The results obtained under this setting are presented below. Figure 5.5 shows the magnitude of the velocity field v computed on both the initial and the deformed domains. Since the viscosity coefficient is sufficiently small to allow for large deformations, yet

large enough to ensure a laminar flow regime, the resulting velocity field is consistent with classical solutions to the Navier–Stokes benchmark problem.



(a) Initial domain Ω (b) Deformed domain $F(\Omega)$

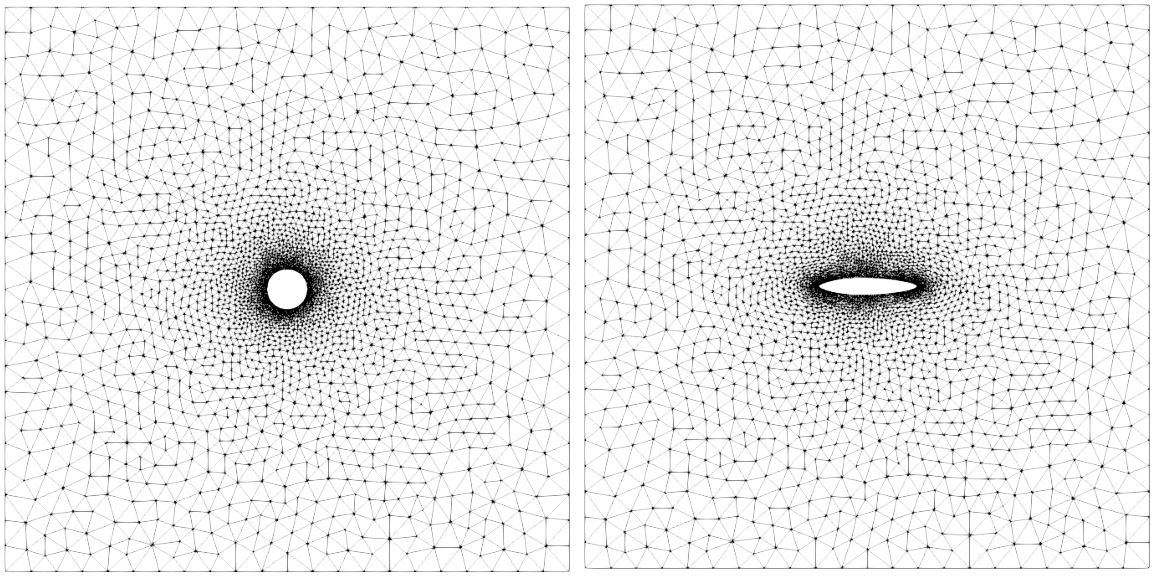
Figure 5.5.: Magnitude of velocity v computed on reference domain Ω (left) and deformed $F(\Omega)$ (right) using the nonlinear extension equation, Section 5.2.5.

The corresponding initial and final meshes are shown in Figure 5.6. Unlike in the linear extension case presented in Section 5.2.4, the characteristic kinks at the tip and rear of the obstacle clearly form in the final optimized shape, indicating that the nonlinear extension approach allows the optimization to progress further toward the expected optimal configuration.

It is important to note that all meshes presented in this thesis are generated in a post-processing step. Specifically, the mapping operator F, which is computed during the optimization procedure, is applied to the initial domain to obtain the final, deformed mesh. This strategy eliminates the need to recompute and remesh the domain at each iteration of the solution process, thereby significantly reducing computational cost and complexity.

As a consequence, the mesh deformations shown on Figure 5.3, Figure 5.8, Figure 5.19 do not represent a sequence of shape optimization iterates, as is common in classical approaches that apply the descent direction iteratively to the mesh. Instead, this methodology produces no intermediate shapes between the initial and final domains. The final deformed configuration is obtained directly by applying the mapping operator and the corresponding displacement field w.

Starting from an initially circular domain, the shape becomes increasingly elongated



(a) Reference grid (b) Deformed grid in $F(\Omega)$

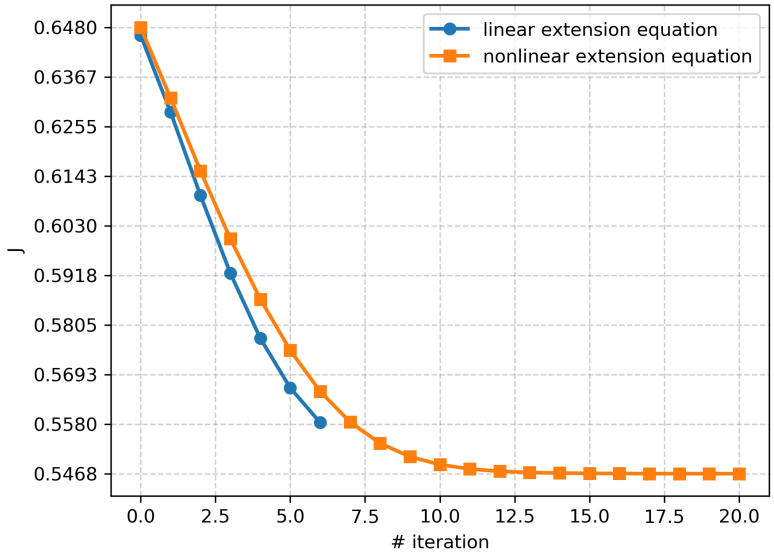
Figure 5.6.: Mesh deformation associated with the nonlinear extension equation, Section 5.2.5.

with each optimization iteration, similar to the behavior observed in the case of linear deformation. However, in the present setting, tangential displacements of boundary nodes are permitted. This is achieved by incorporating a nonlinear term in the extension equation, which effectively introduces an advection-like behavior into the mesh deformation process. As a result, boundary nodes are allowed to slide along the surface and accumulate near the tip and rear of the domain. This adaptive redistribution helps preserve mesh quality, particularly in regions with high curvature or sharp features, and prevents element distortion as the shape evolves.

In case of this experiment, the optimization procedure converged after 20 iteration steps of Algorithm 5. The value of the cost functional was reduced from 0.646194 to 0.546801 which corresponds to the improvement of 15.61%. The details about objective function value at every iteration step are given in Figure 5.7.

Figure 5.9 illustrates the difference in the final configurations for examples of linear vs nonlinear extension equation. Starting from the same initial configuration, the solution process for the linear case stops before reaching the optimum, whereas in the mesh corresponding to the last iteration of the nonlinear extension, the desired kinks at the front and back of the shape were generated.

A closer look at the problematic area around the tip of the shape for the final shape of nonlinear case is shown on Figure 5.10.



(a) Convergence plot illustrating the decrease in the cost functional.

it	α_{it}	J
0	2^{0}	0.646 194
1	2^{-1}	0.631 990
2	2^{-2}	0.615 419
3	2^{-3}	0.600 037
4	2^{-4}	0.586 374
5	2^{-5}	0.574 760
6	2^{-6}	0.565 449
7	2^{-7}	0.558 496
8	2^{-8}	0.553 701
9	2^{-9}	0.550 657
10	2^{-10}	0.548 866
11	2^{-11}	0.547 875
12	2^{-12}	0.547 350
13	2^{-13}	0.547 079
14	2^{-14}	0.546 941
15	2^{-15}	0.546 871
16	2^{-16}	0.546 836
17	2^{-17}	0.546 819
18	2^{-18}	0.546 810
19	2^{-19}	0.546 805
20	2^{-20}	0.546 803

(b) Tabulated values at each iteration step.

Figure 5.7.: Convergence analysis of the cost functional during the shape optimization using the nonlinear extension equation, Section 5.2.5.

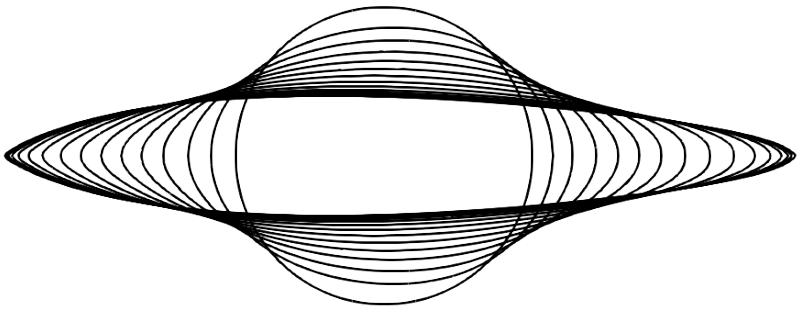


Figure 5.8.: Domain Ω for each of F with nonlinear extension equation, Section 5.2.5.

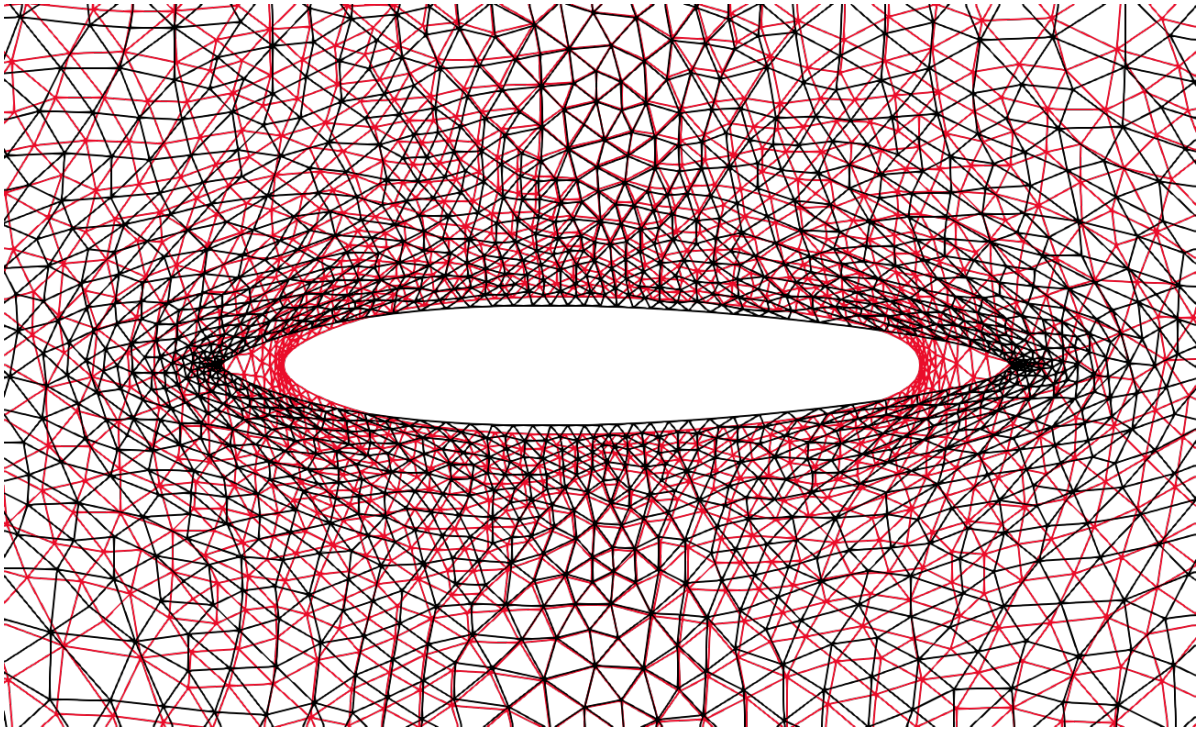


Figure 5.9.: Comparison of the meshes for linear (red, Section 5.2.4) and nonlinear (black, Section 5.2.5) extension equation at the final iteration.

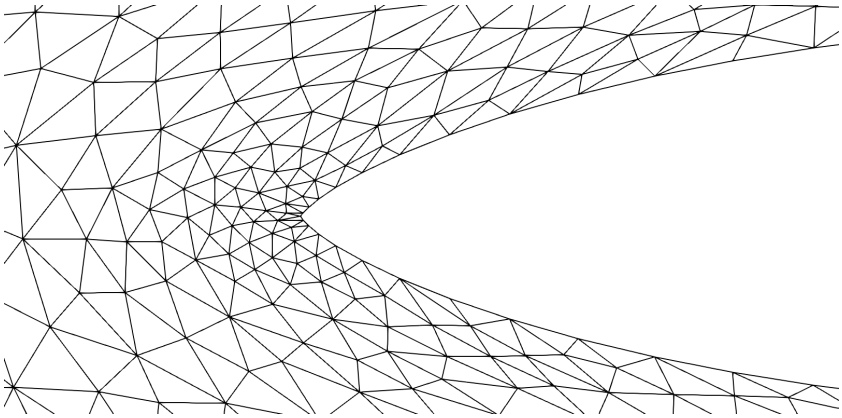


Figure 5.10.: Closeup for mesh at the tip of the shape in the final grid, Section 5.2.5.

Furthermore, the meshes at the final iteration step for linear case and for the same iteration step for the nonlinear case are given on Figure 5.11 and Figure 5.12, respectively.

The corresponding to these examples values of radius ratio mesh quality measure are given in Table 5.2

Table 5.2.: Values of RR for optimization steps for linear ($\eta_{\text{ext}} = 0$, , Section 5.2.4) and nonlinear ($\eta_{\text{ext}} = 3$, Section 5.2.5) extension equations.

it	RR_{lin}	RR_{nonlin}
0	1.6241	1.5723
1	1.6250	1.5984
2	1.6262	1.6288
3	1.8967	1.8663
4	2.1064	2.2781
5	3.5385	3.0084
6	9.4018	4.2616

To assess the impact of spatial discretization on the quality and stability of the optimization results, a mesh refinement study was performed and the results for coarser and finer meshes and different values of η_{ext} are provided in Appendix C. Several meshes with increasing resolution were considered, ranging from a coarse mesh with 962 cells to a refined mesh with over 13000 elements. For each mesh, the optimization procedure was executed using the same solver settings and stopping criteria, allowing direct comparison of objective values, convergence rates, and mesh quality metrics.

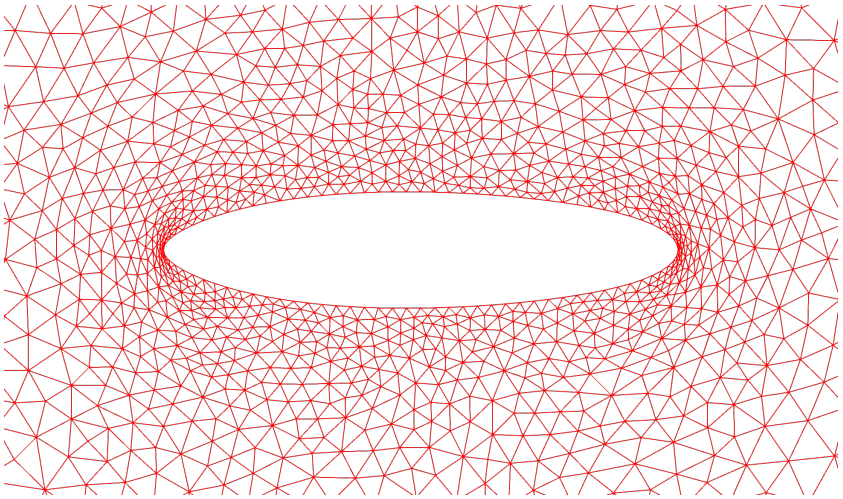


Figure 5.11.: Mesh for linear extension equation at final iteration step $it = 6$, Section 5.2.4.

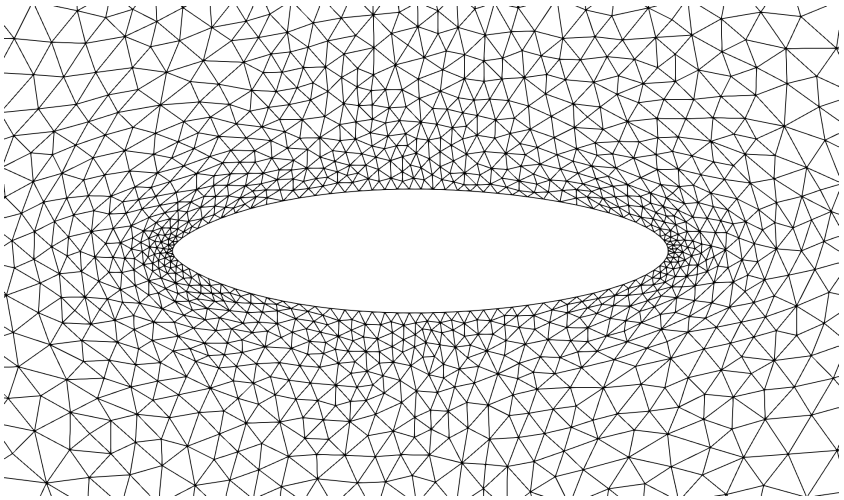


Figure 5.12.: Mesh for nonlinear extension equation at $it = 6$, Section 5.2.5.

5.2.6. Influence of the Extension Factor

Following the introduction of the nonlinear term in the extension equation, the question arises how much nonlinearity would be optimal for such problems, i.e. what would be the most feasible value of the extension factor η_{ext}.

To investigate this, I conduct a comparative study illustrated in Table 5.3, examining how the optimal shape evolves as the extension factor η_{ext} varies. All primary parameters are kept consistent with the previous example discussed in Section 5.2.4: $\alpha_{\text{init}} = 1$, $\alpha_{\text{dec}} = 0.5$ and a viscosity coefficient of $\nu = 0.04$. One important modification in this setup is the mesh: to better observe the influence of different extension factors, a finer mesh with a higher density of boundary elements is used. The refinement is concentrated around the obstacle to ensure comparability of results, resulting in a mesh with 7928 elements.

Moreover, unlike in the previous example in Section 5.2.5, where the determinant condition was inactive, this case includes a moderately active determinant penalty to discourage excessive tangential motion of boundary elements. Specifically, the parameters are set to $\eta_{\text{det}} = 0.2$ and $\beta = 5$. For reference, results for a similar configuration with inactive determinant condition disabled are provided in Appendix D.

The mesh deformations, resulting from the optimal solution $F = \text{id} + w$, are visualized in Figure 5.13. Each of the subfigures illustrates the impact of the extension factor η_{ext} on the deformation of the mesh, focusing on the tip of the shape during the optimization process. One can observe that by increasing extension factor η_{ext} I get the decrease in the value of the objective function J as shown in Table 5.3. For more details about objective function values for all iteration steps please refer to Appendix B.

In the following paragraphs I analyse the results illustrated on Table 5.3. For $\eta_{\text{ext}} = 0$, the deformation led to the compression of triangles around the tip of the shape. One can observe the deformation of elements further from the tip is minimal since the deformation equation in this setting is linear.

As η_{ext} increases to 0.25 and 0.3, subtle improvements of the mesh become visible, with slight stretching and adjustments in the mesh a bit further from the tip allowing more movement for boundary elements in the direction opposite to the fluid flow. The elements are still highly compressed around the tip. However, the degeneration is less prominent indicating the onset of nonlinear deformation which comes with the addition of the nonlinear term in the mesh deformation equation. Nevertheless, as I observe, this factor is still too small to overcome the effects of the large deformation of the shape.

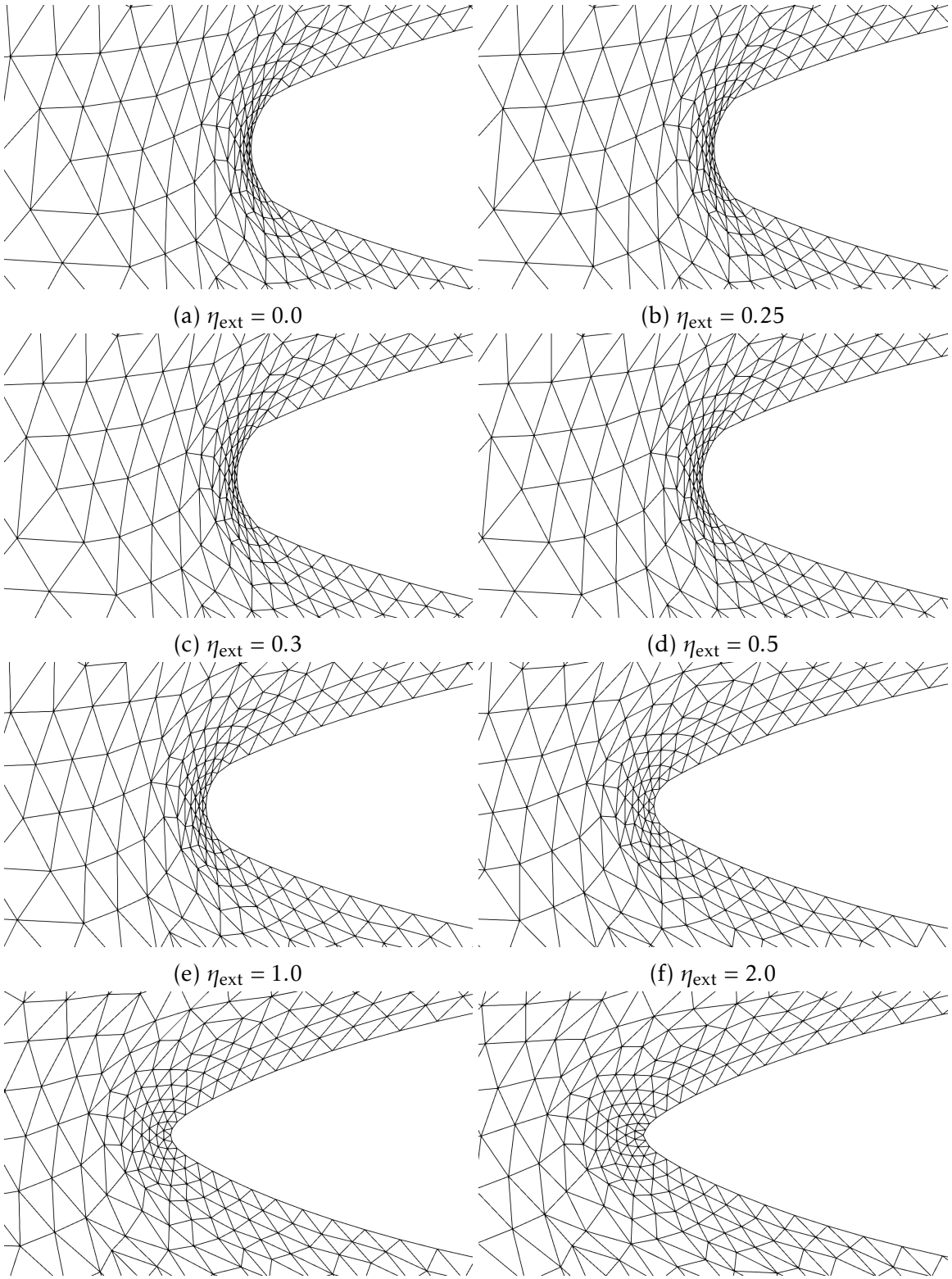
For $\eta_{\text{ext}} = 0.5$ and $\eta_{\text{ext}} = 1.0$, the mesh undergoes more pronounced adjustments, with the mesh becoming more and more regular around the tip.

For higher values of the extension parameter, such as $\eta_{ext} = 2$ and $\eta_{ext} = 3$, the deformations become extensive, with substantial stretching of the tip and surrounding regions, demonstrating the robustness of the nonlinear extension operator in maintaining mesh quality and avoiding mesh collapse. By avoiding mesh entanglement during the early stages of optimization, the method enables the continuation of the optimization process toward more complex and sharper shapes, allowing convergence to a solution that is closer to the true optimal configuration. The tip of the shape deforms significantly more, allowing the shape to become more elongated. Despite these large deformations, the mesh elements remain well-distributed, confirming the method's capacity to preserve mesh quality. These results emphasize the efficacy of the nonlinear extension approach in accommodating significant geometric changes while maintaining computational stability and accuracy.

Finally, I check the deformation for $\eta_{\text{ext}} = 4.0$. As shown on Figure 5.13h, the further increase of the extension factor does not bring further improvement, as triangles around the tip become more elongated in the direction of the flow. This way I can deduce that the best choice of the η_{ext} for the current setting is $\eta_{\text{ext}} = 3.0$.

I also remark that even though the choice of the extension factor depends on the physical setting of the problem, it is not sensitive w.r.t. initial discretization. As I solved this problem on a range of meshes with different coarseness, the optimal choice of extension factor remained at around $\eta_{\text{ext}} = 3.0$. For details about numerical experiments for different mesh coarseness I refer to the supporting material in Appendix C.

So the extension factor directly influences mesh quality. In particular, employing the nonlinear extension equation leads to significantly improved preservation of mesh



(a) $\eta_{\text{ext}} = 0.0$ (b) $\eta_{\text{ext}} = 0.25$

(c) $\eta_{\text{ext}} = 0.3$ (d) $\eta_{\text{ext}} = 0.5$

(e) $\eta_{\text{ext}} = 1.0$ (f) $\eta_{\text{ext}} = 2.0$

(g) $\eta_{\text{ext}} = 3.0$ (h) $\eta_{\text{ext}} = 4.0$

Figure 5.13.: Comparison of the front tip of the shape for different extension factors, Section 5.2.6.

Table 5.3.: Optimal objective function values J, relative objective value with respect to the initial configuration and radius ratio depending on η_{ext}, Section 5.2.6.

η_{ext}	J	$\frac{J}{J_{\text{init}}}$	RR
0	0.279 034	0.801 264	13
0.25	0.278 867	0.800 785	9.2
0.3	0.278 837	0.800 698	8.6
0.5	0.278 722	0.800 368	6.6
1.0	0.278 504	0.799 742	4.0
2.0	0.278 323	0.799 222	2.9
3.0	0.278 291	0.799 131	2.2
4.0	0.278 286	0.799 116	2.2

quality throughout the deformation process.

In Figure 5.14, I quantify this effect by evaluating the mesh quality for the deformations shown in Figure 5.13. The visualizations display the radius ratio quality measure, previously introduced in Section 3.5, using a color scale to highlight local variations in element quality across the domain.

5.2.7. Influence of the Determinant Condition

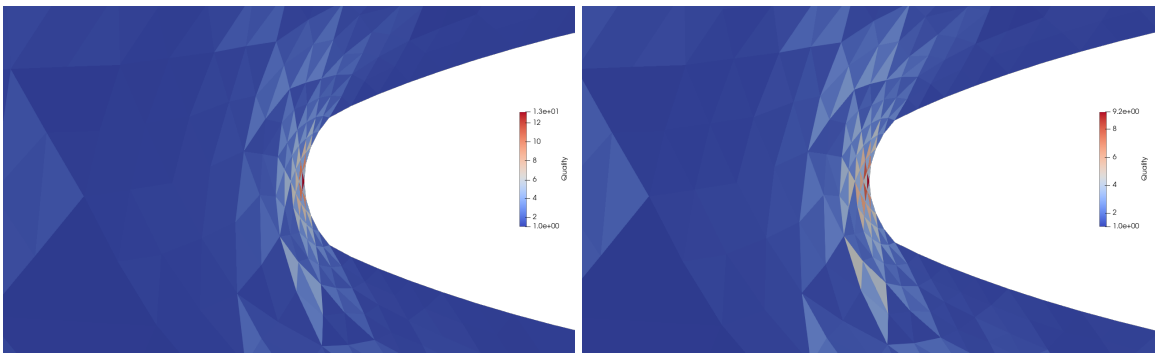
In shape optimization, admissible domain deformations are typically constrained by enforcing a lower bound on the Jacobian determinant of the transformation. This condition ensures that the mapping remains locally injective and orientation-preserving. Mathematically, for a deformation mapping F, the constraint

$$\det(DF) \geq \eta_{\text{det}}$$

is imposed to avoid self-intersections and to guarantee that the transformed domain remains a valid shape. This lower bound serves as a safeguard against mesh degeneration and topological changes by limiting the amount of local volume compression in Ω, especially when large deformations are involved. The constraint is commonly incorporated into the definition of the admissible set of transformations or penalized via a regularization term in the optimization functional.

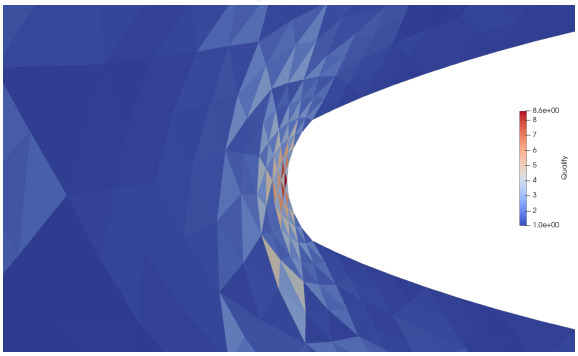
In this section, the influence of the parameter η_{det} on the optimization process is investigated. Starting with a relatively strict lower bound, the value of η_{det} is gradually reduced, thereby relaxing the constraint and allowing for increasingly flexible deformations.

In the underlying experiment the viscosity is again chosen to be $\nu = 0.02$ and I fix $\eta_{\text{ext}} = 3.0$ in all examples. The discretized domain Ω consists of 7928 triangles. Further,

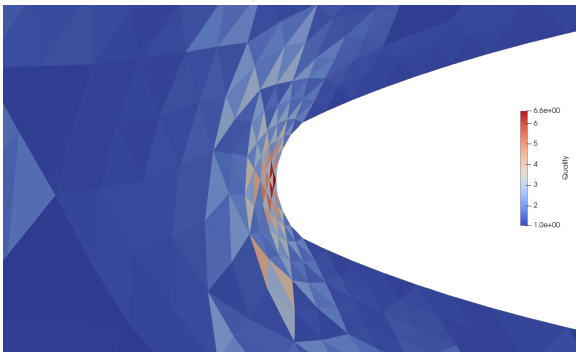


(a) $\eta_{\text{ext}} = 0.0$

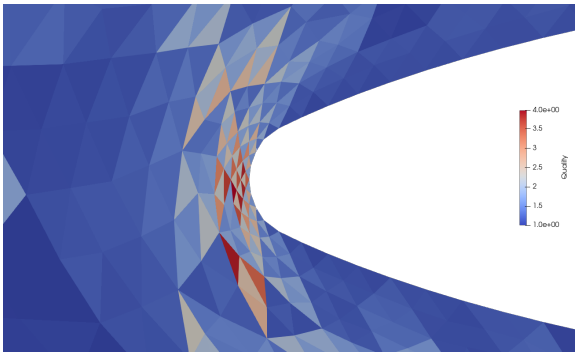
(b) $\eta_{\text{ext}} = 0.25$



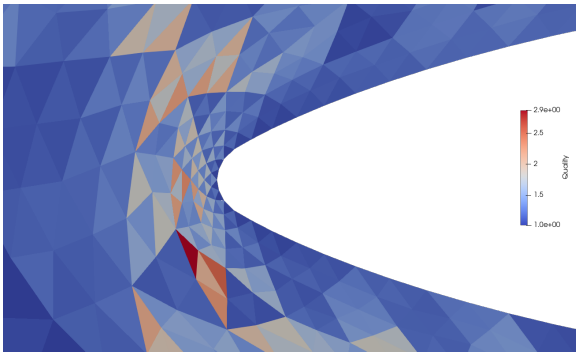
(c) $\eta_{\text{ext}} = 0.3$



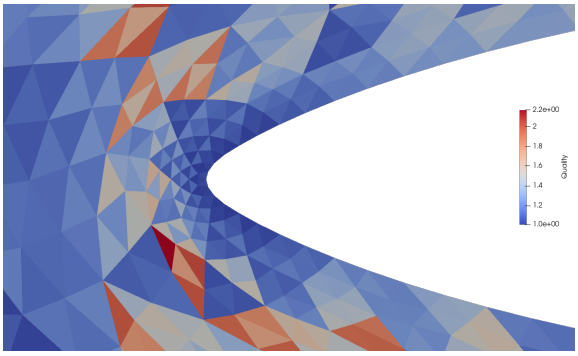
(d) $\eta_{\text{ext}} = 0.5$



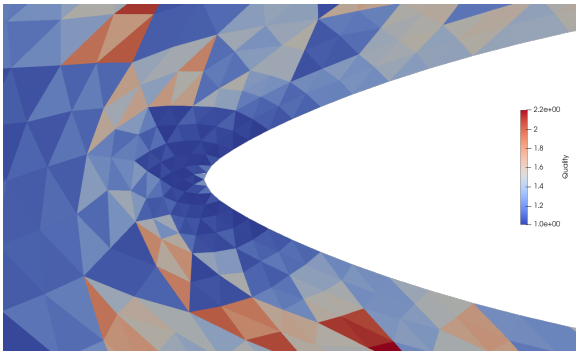
(e) $\eta_{\text{ext}} = 1.0$



(f) $\eta_{\text{ext}} = 2.0$



(g) $\eta_{\text{ext}} = 3.0$



(h) $\eta_{\text{ext}} = 4.0$

Figure 5.14.: Comparison of the mesh quality of front wedge of the meshes for different extension factors, Section 5.2.6.

I choose $\alpha_{\text{init}} = 10, \alpha_{\text{dec}} = 0.5$ and fix the penalty term for determinant condition $\beta = 2$.

Starting with $\eta_{\text{det}} = 0.6$, I decrease the value of the parameter and investigate how this change affects the set of admissible shapes $\mathcal{F}_{\text{adm}}$. For reference, the initial value of the cost functional, i.e. J evaluated for the reference shape with $c = 0$, and thus $F = \text{id}$, is $J_{\text{init}} = 0.348256$.

Table 5.4.: Optimal objective function values depending on η_{det}, Section 5.2.7.

η_{det}	0.6	0.5	0.3	0.25	0.2	0.1
J	0.283 454	0.280 542	0.278 430	0.278 330	0.278 291	0.278 277
$\frac{J}{J_{\text{init}}}$	0.813 926	0.805 562	0.799 498	0.799 210	0.799 098	0.799 05

In this experiment it turns out that in the last computation with $\eta_{\text{det}} = 0.1$ the condition is inactive.

With decreasing η_{det} one can observe a decrease in the objective function J as shown in Table 5.4. I also remark that despite the shape looking more elongated with the decreasing value of η_{det}, the volume remains unchanged according to the geometrical constraint (4.60).

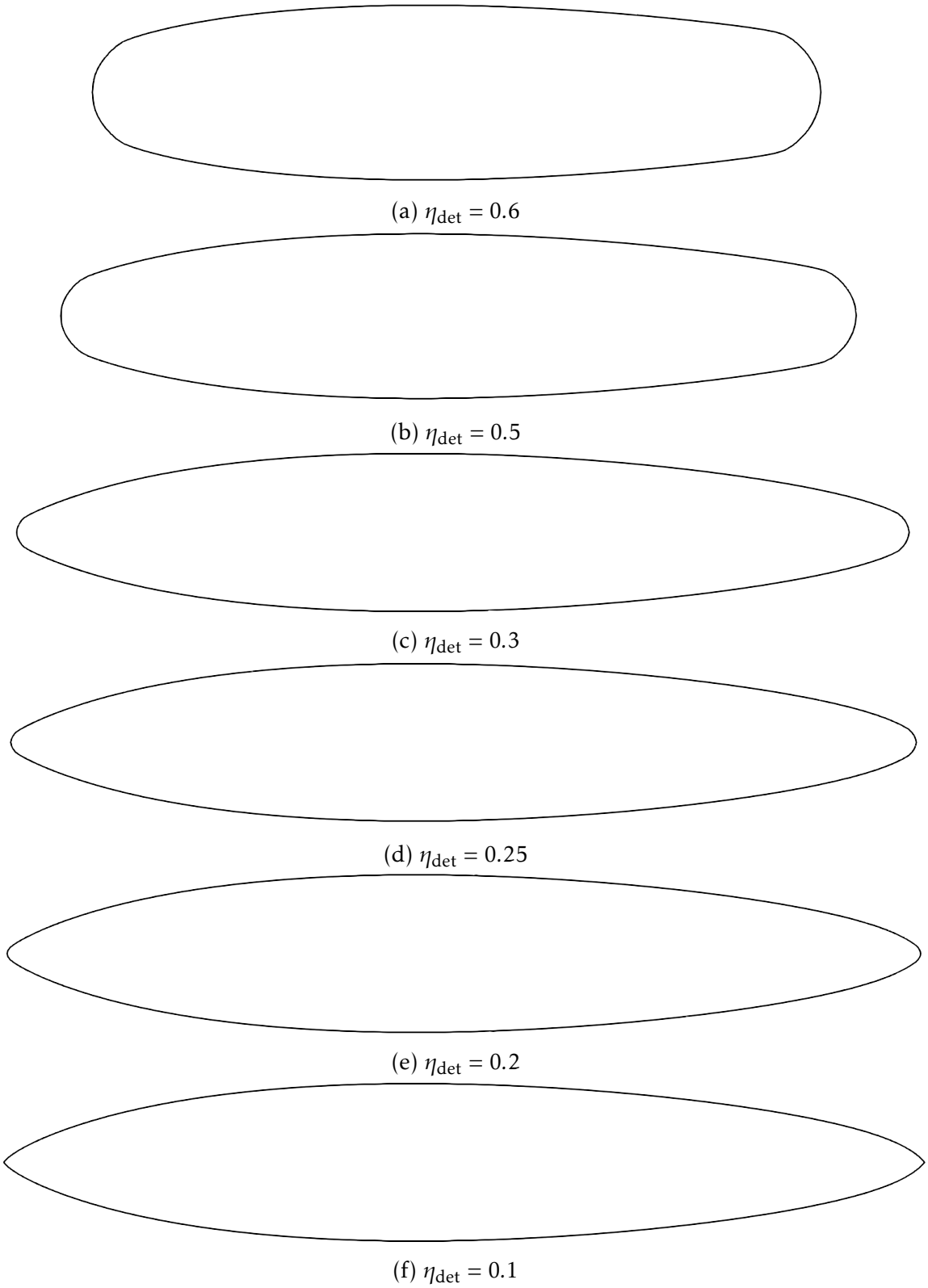
5.2.8. Case of Non-convex Shapes with Large Deformations

Let us consider a case in which the underlying initial domain is not convex. I demonstrate the behavior of the solution process with the linear and nonlinear extension equations for large deformations.

I choose the domain Ω with an obstacle given by a B-spline curve Γ_{obs} given by 7 control points, as illustrated by Figure 5.16.

The associated meshes are illustrated on Figure 5.17. I would like to remark again that there was no actual mesh deformation between the iterations, the meshes were computed in post-processing for better visual interpretation of the results.

In this experiment the viscosity of the fluid is chosen as $\nu = 0.02$, the domain Ω consists of 11 653 triangles with the barycenter and volume of Ω_{obs} in the reference configuration given by $\text{bc}(\Omega_{\text{obs}}) = (-0.214\,979, 0.022\,546)^\top$ and $\text{vol}(\Omega_{\text{obs}}) = 783.678\,335$. Since in the experiments the optimal shape is fixed to be located in the origin, i.e. $\text{bc}(F(\Omega_{\text{obs}})) = 0$, the optimal shape also shifted a bit during the optimization procedure.



(a) $\eta_{\text{det}} = 0.6$

(b) $\eta_{\text{det}} = 0.5$

(c) $\eta_{\text{det}} = 0.3$

(d) $\eta_{\text{det}} = 0.25$

(e) $\eta_{\text{det}} = 0.2$

(f) $\eta_{\text{det}} = 0.1$

Figure 5.15.: Comparison of the shape for different values of η_{det}, Section 5.2.7.

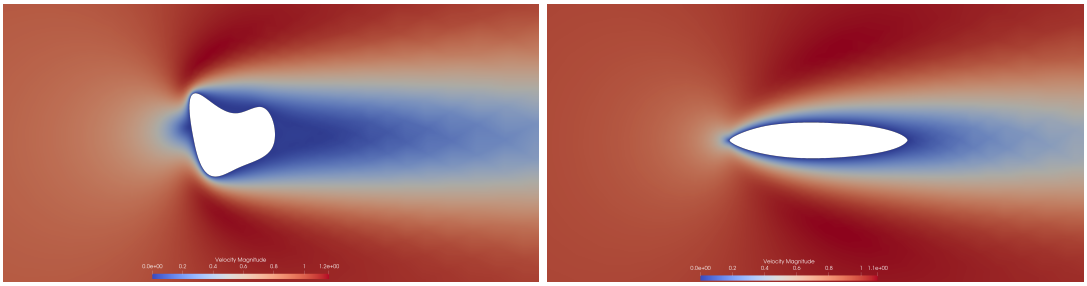


Figure 5.16.: Magnitude of velocity v computed on concave reference domain Ω (left) and deformed $F(\Omega)$ (right) for nonlinear extension equation as described in Section 5.2.8.

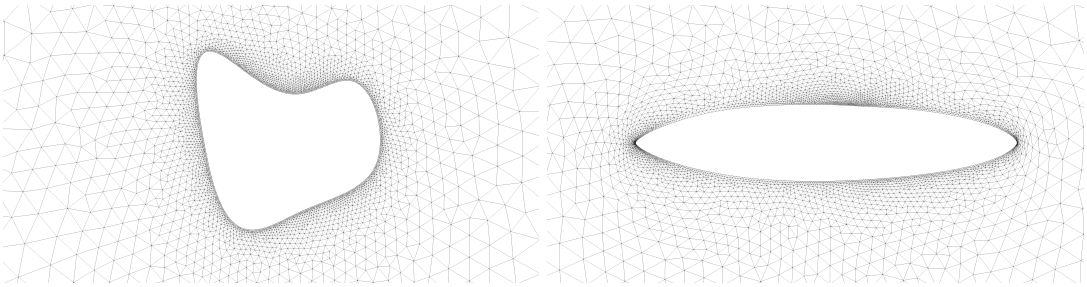


Figure 5.17.: Reference and deformed grids for concave domain, Section 5.2.8.

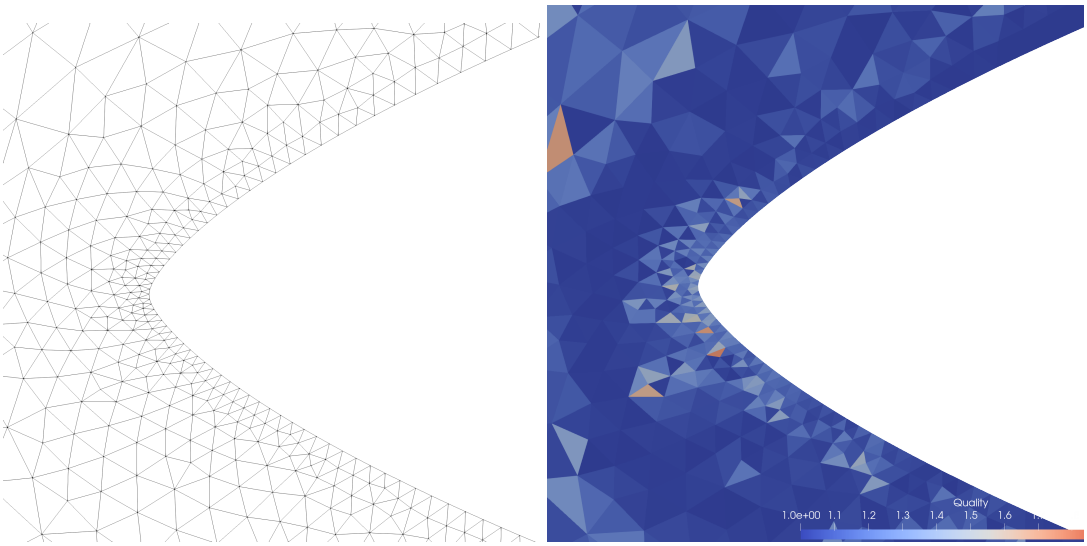


Figure 5.18.: Closeup at a tip of the concave shape: mesh and radius ratio values, Section 5.2.8.

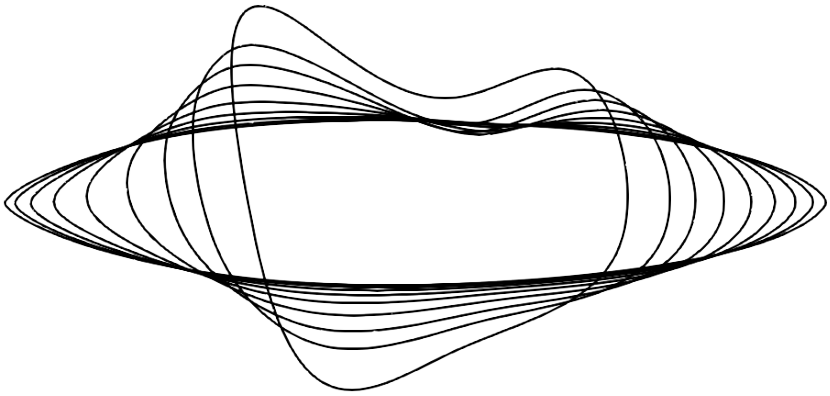


Figure 5.19.: Domain Ω for each of F with nonlinear extension equation, Section 5.2.8.

This case is interesting to investigate since there is no symmetry of the initial shape anymore. So unlike in previously considered cases with the geometry of the reference domain being circular or ellipse, the normal vector field n to the boundary of the obstacle is not pointing homogeneously in all directions. It is relevant for out approach with extension equation mapping scalar-valued control variable to the deformation field variable via n as discussed in (4.99). So it is expected that in comparison with the experiment with the homogeneously distributed vector field n one might observe changes in the way different discretization elements are deformed during the iterations process.

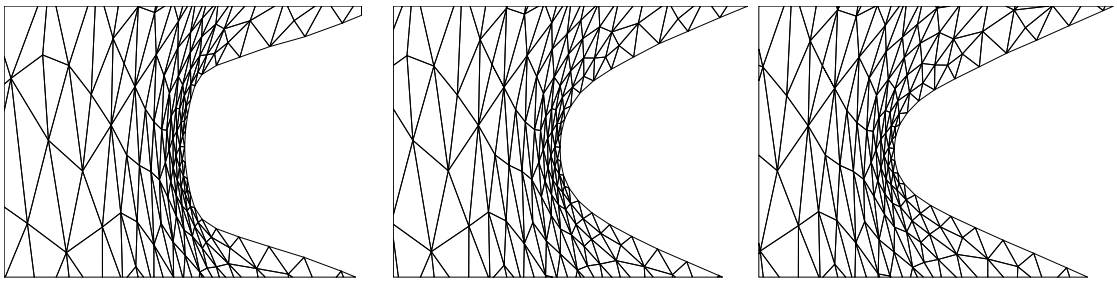
The magnitude of velocity v, computed in the undeformed reference domain Ω (i.e. when $c = 0$), is shown on the left-hand side of Figure 5.16. The right-hand side illustrates the velocity field after deformation, where the domain is mapped according to the optimized shape transformation $F = \mathrm{id} + w$. The corresponding meshes for this experiment are depicted in Figure 5.17 with the initial mesh, corresponding to the undeformed configuration, shown on the left and the deformed mesh, associated with the final optimized shape, on the right.

Despite the asymmetry in the initial configuration, the optimal mapping F results in a well-distributed and relatively uniform mesh. Notably, even in the absence of explicit geometric information such as the surface normal vector n, the optimization process successfully resolves sharp features, such as the kinks at the front and rear tips of the shape. This indicates the capability of the method to capture and preserve important geometric features purely through the optimization dynamics, without requiring additional constraints or enhancements.

Figure 5.18 shows a detailed view of the mesh around the tip along with the mesh quality visualization.

5.2.9. Effects on Condition Number of Hessian

An important aspect in the analysis of the optimization problem is the numerical conditioning of the reduced Hessian. The condition number provides a measure of how sensitive the solution of the linearized system is to perturbations in data or discretization errors. In particular, in gradient-based methods or second-order optimization schemes, a poorly conditioned Hessian can lead to slow convergence, inaccurate search directions, or numerical instability. By investigating the condition number of the reduced system's Hessian, insights can be gained into the regularity of the problem and the influence of modeling choices, such as the lower bound η_{det}, the choice of extension operator S, and mesh quality. Moreover, understanding the conditioning behavior helps in designing appropriate preconditioning strategies or regularization terms to improve robustness and efficiency of the numerical solution process.



(a) $\eta_{\text{ext}} = 0.0$, $\kappa(H) \approx 79.7340$ (b) $\eta_{\text{ext}} = 0.25$ $\kappa(H) \approx 24.1355$ (c) $\eta_{\text{ext}} = 0.5$ $\kappa(H) \approx 16.6265$

(d) $\eta_{\text{ext}} = 1.0$, $\kappa(H) \approx 4.8447$ (e) $\eta_{\text{ext}} = 2.0$, $\kappa(H) \approx 3.5318$ (f) $\eta_{\text{ext}} = 3.0$, $\kappa(H) \approx 15.2386$

Figure 5.20.: Condition Number of Hessian $\kappa(H)$ for different η_{ext}, Section 5.2.9.

In this work, the computation of the Hessian of the reduced objective functional with respect to the shape deformation was carried out using the dolfin-adjoint library which is briefly described in Section 5.1.4. The overall approach involved two main steps: first, the reduced optimization problem was defined and solved, obtaining the solution of the state equations for a given deformation; second, the Hessian of the reduced functional was computed using automatic differentiation tools provided by dolfin-adjoint.

One can observe on the results in Figure 5.20, that there is some correlation between mesh quality and the computed condition number. However, it is important to note that the computed Hessians are highly sensitive to the underlying mesh. In particular, it was observed that small changes in mesh resolution or quality lead to significant variations in the spectrum and structure of the Hessian.

5.2.10. Decoupling NS Equations from Elasticity Equations

Let us illustrate the performance of the algorithm given by Algorithm 6.

Let's consider the basic example again, for $\nu = 0.02, \eta_{\text{ext}} = 3, \alpha_{\text{init}} = 10, \alpha_{\text{dec}} = 0.5, \eta_{\text{det}} = 0.01, \beta = 2$ solved on a mesh with 7928 elements.

The computational time comparison is given in Table 5.5, where the first column is number of iteration, T_{full} is the solution time of the undecoupled system, j_{full} corresponding to T_{full} value of the objective function, $j_{\text{decoupled}}$ - value of the objective function at the iteration step it in case of solving the decoupled problem, T_{forward}, T_{adjoint}, $T_{\text{deformation}}$ - time solving the forward system, the adjoint system and deformation part correspondingly. These results were obtained by using Newton solver and MUMPS for linear solve steps.

In this example for the classical case, i.e. without using the decoupling technique, the problem converged after 35 iterations with value $j_{\text{full}} = 0.2782$, and already after the iteration $it = 17$ there was almost no change to the objective function (see Table B.1). However, for the case of the decoupled problem, after reaching the shape close to the optimal at iteration $it = 14$, the problem started to diverge.

I remark that this behavior can be avoided by choosing a larger decrement of regularization parameter α_{dec} such that the transition between problems would have been smoother. For the sake of this experiment, I decided against increasing α_{dec} to better highlight differences in computational time for a smaller amount of iterations.

However, for the completeness of the study, it is important to notice that solving the decoupled problem with $\alpha_{\text{dec}} = 0.8$ ended up diverging at $it = 55$ with $\alpha = 4.6768e{-}05$ and $j_{\text{decoupled}} \approx 0.2619$.

Having all three parts of the system partially decoupled from each other allows the use of different solvers for each part and have more control over the procedure. This provides an opportunity to build preconditioners for each part and parallelize the code. I tested this approach by using GMRES as a linear solver for both direct and adjoint solve while keeping the non-iterative MUMPS for the deformation part.

Table 5.5.: Comparison of the solution time for standard vs decoupled problem and corresponding values of the objective function, Section 5.2.10.

it	T_{full}	J_{full}	$J_{decoupled}$	T_{forward}	T_{adj}	$T_{\text{deformation}}$
1	36.2062	0.3457	0.3440	8.7618	2.5853	6.2197
2	36.4930	0.3412	0.3401	1.8739	0.7498	1.9727
3	49.8679	0.3341	0.3378	1.8379	0.7734	2.1034
4	49.4882	0.3243	0.3337	1.9061	0.8178	2.7456
5	48.2206	0.3132	0.3267	1.7950	0.7485	2.6282
6	47.7823	0.3027	0.3160	1.8145	0.7565	2.608
7	48.0959	0.2943	0.3038	1.8005	0.7690	2.6112
8	48.0532	0.2882	0.2945	1.8577	0.7753	2.5990
9	48.0943	0.2842	0.2875	1.8471	0.7656	2.6106
10	48.2612	0.2816	0.2830	1.8014	0.7499	2.5858
11	36.1988	0.2800	0.2788	1.8045	0.7620	2.5934
12	36.6835	0.2792	0.2774	1.8210	0.7560	1.9862
13	36.9373	0.2787	0.2728	1.8124	0.7466	1.9788
14	36.6471	0.2785	0.2656	1.7894	0.7683	2.5829

The results are illustrated in Table 5.6. Note that at every iteration step, Newton converges in one step, the amount of linear solves for forward and adjoint problem solves are given in a table. Further, after $it = 14$ the optimal solution is found.

It is important to note that the underlying PDE-constrained optimization problem exhibits a saddle-point structure, which is known to be challenging for iterative solvers. As a result, the performance of GMRES is noticeably inferior to that of the direct solver in this case. The relatively slow convergence can be attributed to the use of the default preconditioning strategy provided by FEniCS, which does not exploit the specific block structure of the system. Ideally, a preconditioner that incorporates knowledge of the coupled state-adjoint system, such as a block-diagonal or Schur complement-based approach, would significantly improve convergence behavior. Unfortunately, due to limitations in the FEniCS Project, particularly in the 2019.1.0 version used for this work, it was not feasible to integrate such customized preconditioning techniques. As a result, the reported results reflect the raw performance of GMRES without advanced acceleration strategies, and should be interpreted as a baseline rather than an optimized implementation.

Future work could address these limitations by extending the solver infrastructure to allow for more flexible preconditioning, potentially leveraging external libraries such as PETSc or incorporating low-rank or multigrid-based preconditioners tailored to the structure of the optimization problem.

Table 5.6.: Solution time for each part of the decoupled problem, Section 5.2.10.

it	$J_{\text{decoupled}}$	T_{forward}	# lin. it. forward	T_{adj}	# lin. it. adj.	$T_{\text{deformation}}$
1	1.3969	24.7836	19266	9.0757	6698	2.2256
2	1.3870	5.9759	4288	4.7917	3649	2.1914
3	1.3823	5.5065	4034	4.4535	3522	2.0808
4	1.3743	5.9576	4729	4.8456	3861	2.6999
5	1.3626	7.0433	5349	5.3718	4242	2.7035
6	1.3488	7.6503	6100	5.9274	4660	2.7298
7	1.3365	8.3769	6899	6.2753	5122	2.0918
8	1.3276	9.8563	7456	7.3688	5563	2.2141
9	1.3215	10.8034	7913	7.2341	5850	2.0175
10	1.3153	10.5097	8319	8.0357	5913	2.1541
11	1.3140	10.8879	8402	7.0239	5578	2.1104
12	1.3130	10.3100	8189	5.9287	4836	2.0939
13	1.3133	9.5354	7811	4.9920	4076	2.0366
14	1.3127	7.6287	5683	4.7020	3156	2.1349
15	1.3126	5.7633	4599	3.6120	2772	2.0790
16	1.3126	4.7676	3607	3.1449	2265	2.1897

5.2.11. Results for 3-Dimensional Case

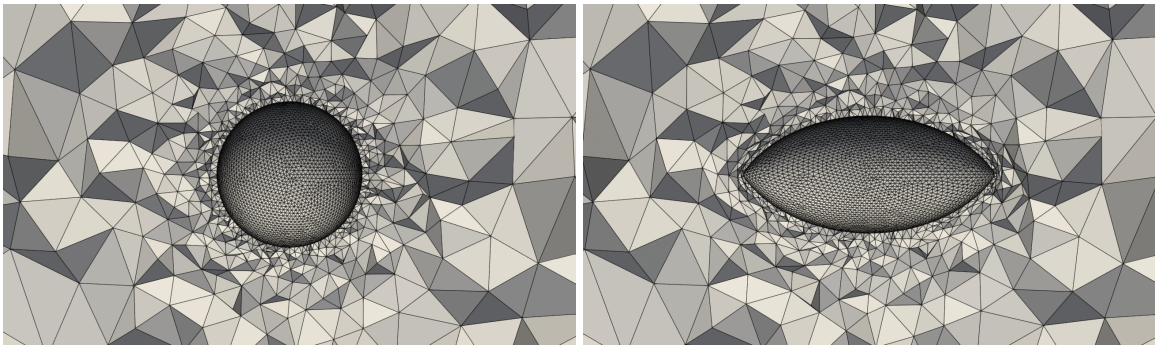
Now let me present some results for a 3D domain. In this section I do not aim to present a comprehensive study. These results are added for completeness but are not necessary for analyzing the performance and setup of presented algorithms.

I start by considering the mesh with 57481 tetrahedral elements. I choose the $\alpha_{\text{init}} = 10, \eta_{\text{det}} = 0.01, \eta_{\text{ext}} = 20$ and $\beta = 2$. Further, to ensure the smooth change between iterations, I choose $\alpha_{\text{dec}} = 0.8$. In this experiment the aim is to observe the behavior for moderate deformation, so the viscosity coefficient is chosen to be $\nu = 1$.

Initially, the objective was to compute the solution for both the full coupled system using Algorithm 5 and the decoupled system using the proposed iterative method (see Algorithm 6) and compare the computational times. However, already the computation of the first linear solve for the full system on this mesh exceeded the available memory resources. In this case, the direct linear solver MUMPS was employed. Due to these memory limitations, I proceed by solving only the decoupled system in the following analysis.

The initial and final meshes are presented on Figure 5.21. One can observe that in this case the kinks at tip and back of the obstacle got formed. The optimal shape was reached after $it = 37$ iterations. The corresponding value of the objective function at

the final iteration step is $J = 136.5173$.



(a) Reference grid for $d = 3$ (b) Deformed grid in $F(\Omega)$ for $d = 3$

Figure 5.21.: Mesh deformation for $\nu = 1, \eta_{\text{ext}} = 20$, Section 5.2.11.

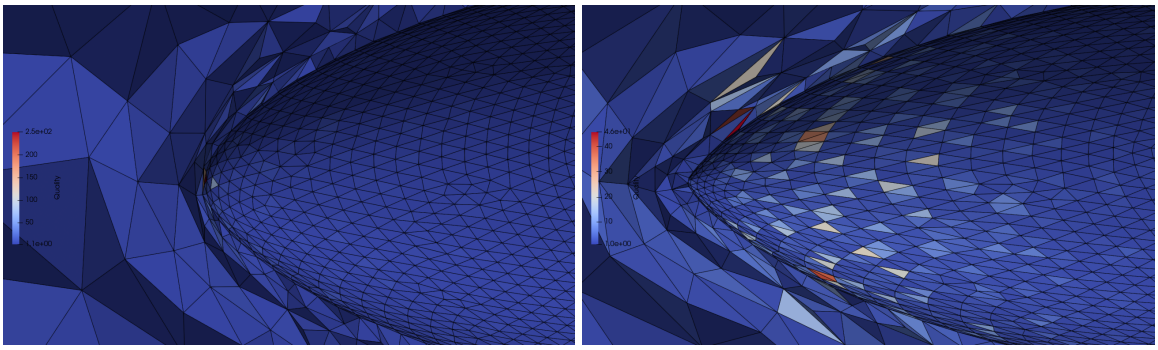
Next, I choose larger deformation: $\nu = 0.04$. I compute this for different values of extension factor $\eta_{\text{ext}} = 10, 20, 30$. In case of $\eta_{\text{ext}} = 10$, the effect of extension factor was not prominent enough, and the mesh degenerated. The solution process stopped at iteration step $it = 24$ with value of cost functional $J = 6.8332$. Since in 3D geometry it is difficult to illustrate which exact element caused the issue, I only comment that the radius ratio mesh quality value for the final mesh is $RR = 250.1929$. The closeup on the tip of the shape is illustrated on Figure 5.22.

One can notice that the problematic element causing solver infeasibility is indeed around the tip for $\eta_{\text{ext}} = 10$. The elements are stacked around this area - the behavior which I have already observed and discussed for the 2D case. In the case of $\eta_{\text{ext}} = 20$ the shape became more elongated and even at the later iteration steps one can observe regular elements around the tip. The worst element in radius ratio is not even at the tip, but a bit further to the side of the shape which is rather the effect of the initial meshing, than of the deformation. I note that the mesh quality at the first iteration step was $RR = 3.2554$.

The final shape is presented in Figure 5.23.

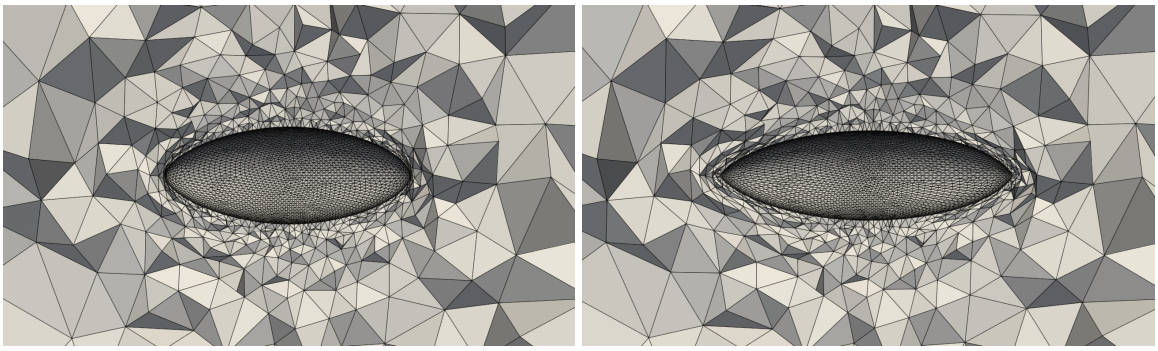
In two other cases with higher extension factor the optimal shape was reached after 55 iterations for the chosen earlier stopping criterion - a difference between objective function value of order 10^5. The corresponding value of the objective function at the final iteration step is $J = 6.8132$ for both cases.

However, one may notice that the resulting meshes are not exactly the same, especially in the area around tips. To illustrate the differences, I compute mesh quality for these two cases. The radius ratio value for $\eta_{\text{ext}} = 20$ is $RR = 98.2741$ and for $\eta_{\text{ext}} = 30$ is $RR = 45.7205$. Further, the visualization of mesh quality for the whole shape in the same scale is given on Figure 5.24.



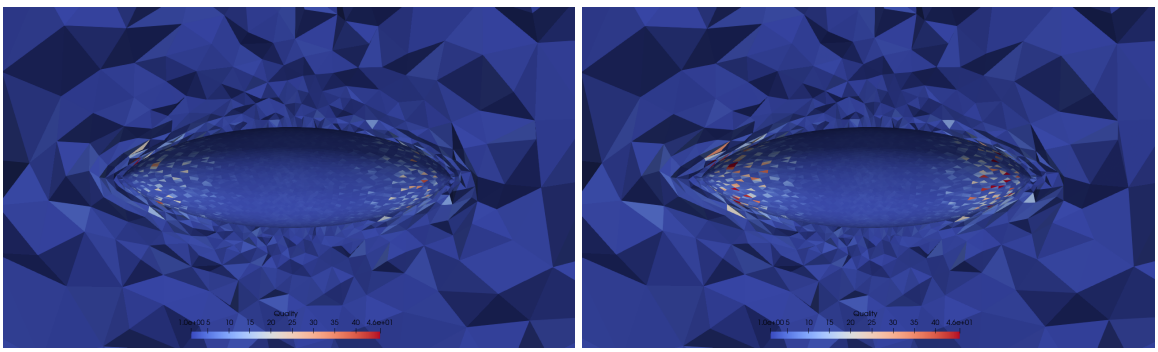
(a) Deformed grid for $\eta_{\text{ext}} = 10$. (b) Deformed grid for $\eta_{\text{ext}} = 20$.

Figure 5.22.: Closeup on the tip of the obstacle with visualized mesh quality for $\nu = 0.04$, Section 5.2.11.



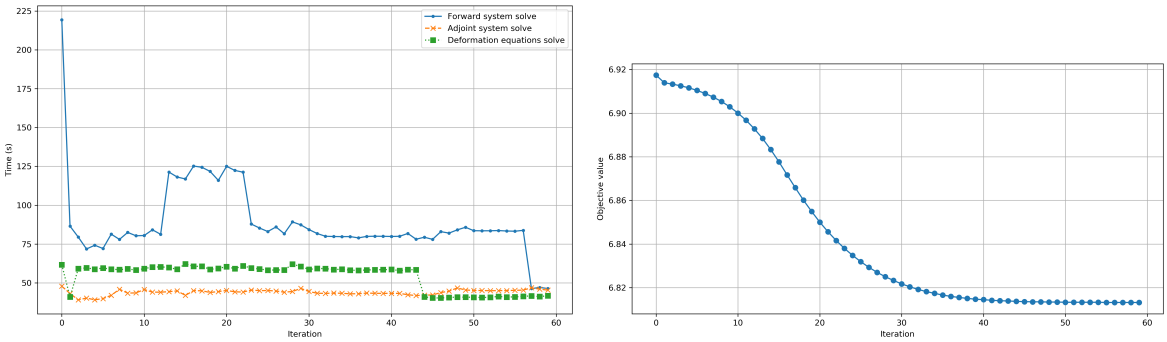
(a) Deformed grid for $\eta_{\text{ext}} = 10$. (b) Deformed grid for $\eta_{\text{ext}} = 20$.

Figure 5.23.: Mesh deformation for $\nu = 0.04$, Section 5.2.11.



(a) Deformed grid for $\eta_{\text{ext}} = 20$. (b) Deformed grid for $\eta_{\text{ext}} = 30$.

Figure 5.24.: Mesh quality for $\nu = 0.04$, Section 5.2.11.



(a) Computation times (in seconds) of the decoupled system.

(b) Values of the objective function J, Section 5.2.11.

Figure 5.25.: Solution details of the decoupled problem for the 3D case.

To provide insight into the performance of the optimization process, the objective function value and the computational times (in seconds) for the main components of the decoupled system are reported at every 5th iteration in Table Table 5.7. This selection offers a representative overview of the convergence behavior and computational cost without overwhelming the main text. The complete iteration data for all iteration steps is included in Appendix Appendix B for reference. Furthermore, the visualization of these results is presented in Figure 5.25.

Table 5.7.: Objective value and computational times (in seconds) for selected iterations, Section 5.2.11.

it	$J_{\text{decoupled}}$	T_{forward}	T_{adj}	$T_{\text{deformation}}$
0	6.9174	219.3453	47.8397	61.6476
5	6.9104	72.1228	39.8750	59.5125
10	6.9000	80.4657	45.7105	59.1582
15	6.8777	116.8806	41.9680	62.2114
20	6.8500	125.0129	45.0961	60.4684
25	6.8319	83.0466	45.0646	58.1354
30	6.8217	84.3168	44.4146	58.6313
35	6.8166	79.7923	42.9408	58.1470
40	6.8145	79.8686	43.2621	58.6021
45	6.8136	78.0026	42.1939	40.2766
50	6.8133	83.5745	45.1225	40.7086
55	6.8132	83.2357	45.1982	40.9755

These results demonstrate the effectiveness and robustness of the proposed optimization framework, even when applied to complex domains and nonlinear extensions. The convergence of the objective function, along with manageable computational times and acceptable mesh quality, confirms the usability of the method for practical

applications. While some sensitivity to mesh quality and regularization parameters was observed, the overall performance remains consistent.

6. Conclusion

In this thesis, shape optimization problems arising in aerodynamics were investigated, with a particular focus on minimizing drag in a flow governed by the Navier–Stokes equations. The central mathematical and computational challenge in such problems lies in controlling the deformation of the domain in a way that preserves mesh quality, ensures numerical stability, and allows for efficient optimization.

To address this, the method of mappings was adopted, the problem was reformulated on a fixed reference domain and shape changes were interpreted as transformations governed by deformation fields. This approach enabled the application of optimal control theory and the derivation of optimality conditions using adjoint-based methods. A key aspect of the work has been the modeling of the extension operator, which maps boundary variations to deformations of the entire domain. In this work, I explored both linear elasticity-based and nonlinear formulations of this operator, highlighting their respective advantages and limitations.

Through numerical experiments, I demonstrated the impact of the extension operator on mesh quality, condition number and therefore on convergence behavior, and the overall effectiveness of the optimization process. The results clearly showed that nonlinear extensions offer improved robustness for handling large deformations and preserving mesh regularity. These findings are particularly relevant for practical applications involving complex shapes and flow conditions, such as those encountered in aerodynamics.

This work contributes to the ongoing development of reliable and flexible tools for PDE-constrained shape optimization. Several challenges remain open for future investigation. For instance, extending the method to unsteady flows or incorporating more complex geometrical and physical constraints (e.g., multi-objective optimization or uncertainty quantification) would be natural next steps. Furthermore, exploring adaptive remeshing strategies or alternative formulations that allow for topological changes could significantly broaden the range of feasible applications.

Overall, the results presented in this thesis emphasize the importance of robust numerical strategies for shape optimization and provide a foundation for further developments in both theory and computational practice.

A. Some Basic Properties from Linear Algebra

We list here some matrix properties used in optimality system derivation in Section 4.6.2 following [66].

A.1. Trace Properties

$$tr(A) = \sum_i A_{ii} \tag{A.135}$$

$$tr(A) = tr(A^T) \tag{A.136}$$

$$tr(AB) = tr(BA) \tag{A.137}$$

$$tr(A+B) = tr(A) + tr(B) \tag{A.138}$$

$$tr(ABC) = tr(BCA) = tr(CAB) \tag{A.139}$$

$$tr(A^T B) = tr(AB^T) = tr(B^T A) = tr(BA^T) \tag{A.140}$$

$$tr(A^T B) = \sum_{ij} A_{ij} B_{ij} = A : B \tag{A.141}$$

$$\nabla \cdot v = tr(\nabla v) \tag{A.142}$$

B. Full Iteration Data

In this appendix detailed iteration data is presented including computational times for each part of the decoupled system (in seconds) and corresponding objective function values.

Table B.1.: Objective value and computational times for each step.

it	$J_{\text{decoupled}}$	T_{forward}	T_{adj}	$T_{\text{deformation}}$
0	6.9174	219.3453	47.8397	61.6476
1	6.9139	86.4915	43.1424	40.9045
2	6.9133	79.4681	39.0287	59.1328
3	6.9125	71.8218	40.2373	59.6379
4	6.9116	74.1993	39.0814	58.8134
5	6.9104	72.1228	39.8750	59.5125
6	6.9090	81.3479	41.9643	58.8368
7	6.9073	77.9746	45.9022	58.4707
8	6.9053	82.4753	43.3966	58.9726
9	6.9029	80.3163	43.5412	58.2152
10	6.9000	80.4657	45.7105	59.1582
11	6.8967	84.1600	44.0309	60.1306
12	6.8928	81.2566	44.0041	60.2926
13	6.8884	121.3217	44.3280	59.9097
14	6.8833	118.1391	44.8157	58.7214
15	6.8777	116.8806	41.9680	62.2114
16	6.8717	125.1917	44.9868	60.6111
17	6.8658	124.3994	44.8892	60.6674
18	6.8601	121.7516	43.8240	58.7118
19	6.8549	115.9715	44.3876	59.2415
20	6.8500	125.0129	45.0961	60.4684
21	6.8456	122.3874	44.2438	59.1351
22	6.8416	121.2346	44.0721	60.9719
23	6.8380	87.8530	45.4095	59.5046
24	6.8348	85.2886	44.9677	58.9555
25	6.8319	83.0466	45.0646	58.1354
26	6.8293	85.9797	44.7734	58.3160

Table B.1 (continued)

it	$J_{\text{decoupled}}$	T_{forward}	T_{adj}	$T_{\text{deformation}}$
27	6.8270	81.6487	44.0026	58.2990
28	6.8250	89.2347	44.4301	62.0060
29	6.8233	87.3411	46.4493	60.5530
30	6.8217	84.3168	44.4146	58.6313
31	6.8204	81.7492	43.3198	59.2677
32	6.8192	79.9852	43.2334	59.1725
33	6.8182	79.8351	43.4047	58.5886
34	6.8174	79.7598	43.2929	58.7345
35	6.8166	79.7923	42.9408	58.1470
36	6.8160	78.9464	43.0196	58.0318
37	6.8155	79.9051	43.4114	58.2669
38	6.8151	80.0231	43.2831	58.4604
39	6.8147	79.9634	43.2290	58.4723
40	6.8145	79.8686	43.2621	58.6021
41	6.8142	79.9647	43.2233	57.8892
42	6.8140	81.8038	42.3478	58.5844
43	6.8139	78.0834	41.8317	58.3429
44	6.8137	79.3760	42.1270	40.9863
45	6.8136	78.0026	42.1939	40.2766
46	6.8135	82.9558	43.7580	40.2956
47	6.8135	82.0389	44.6704	40.5828
48	6.8134	84.1096	46.7270	40.8303
49	6.8134	85.7828	45.4007	40.7723
50	6.8133	83.5745	45.1225	40.7086
51	6.8133	83.5048	45.0677	40.5782
52	6.8133	83.5398	44.9390	40.7221
53	6.8133	83.6664	45.0158	41.2605
54	6.8133	83.3793	45.0176	40.8760
55	6.8132	83.2357	45.1982	40.9755
56	6.8132	83.8058	45.3337	41.3179
57	6.8132	46.3062	46.8132	41.6292
58	6.8132	47.0310	45.9425	41.1458
59	6.8132	46.3632	45.3195	41.7410

C. Numerical Experiments for Different Meshes

In this section the results for meshes of different coarseness are presented for various values of η_{ext}. The improvement percentage reported in the table is computed relative to the initial objective value at iteration 0 as follows:

$$\text{Improvement (\%)} = 100 \cdot \frac{J_0 - J_k}{J_0}$$

where J_0 is the objective value at iteration 0 and J_k is the objective value at iteration k.

Although the primary stopping criterion is based on the convergence of the objective function, I also enforce a maximum number of iterations. This safeguard ensures the algorithm terminates within a bounded number of steps in cases where convergence is slow or the problem becomes numerically stiff. In practice, the method often converges well before reaching this upper bound.

The mesh quality is measured in terms of radius ratio measure (RR).

C.1. Coarse Mesh

The mesh used in this example was loaded from `mesh/coarse_final_mesh.xml`, containing 962 cells and 520 vertices, with a total volume of approximately 780.87. Key simulation parameters included:

- Viscosity $\nu = 0.04$
- Initial regularization parameter $\alpha = 1.0$
- Decay factor $\alpha_{\text{dec}} = 0.5$
- Stopping tolerance of 10^{-5}
- Determinant bound $\eta_{\text{det}} = 0.2$
- Penalty parameter $\beta = 5.0$
- Maximum iterations: 80

Table C.1.: Optimization results for $\eta_{\text{ext}} = 3$ and coarse mesh over 19 iterations showing the evolution of the objective functional, improvement, and mesh quality for a decreasing sequence of $\alpha = 2^{-n}$, with initial $\alpha = 1.0$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	6.509132e-01	0.00	2.77	1.4512
1	2^{-1}	6.346553e-01	2.50	1.90	1.4978
2	2^{-2}	6.172352e-01	5.17	1.98	1.5141
3	2^{-3}	6.011632e-01	7.64	1.94	1.5589
4	2^{-4}	5.871795e-01	9.79	1.86	1.6042
5	2^{-5}	5.754807e-01	11.59	1.89	1.6501
6	2^{-6}	5.661876e-01	13.02	1.88	1.7427
7	2^{-7}	5.592918e-01	14.08	1.83	2.0790
8	2^{-8}	5.545630e-01	14.80	1.89	2.4676
9	2^{-9}	5.515759e-01	15.26	1.84	2.9876
10	2^{-10}	5.498270e-01	15.53	1.83	3.6045
11	2^{-11}	5.488633e-01	15.68	1.84	6.0945
12	2^{-12}	5.483539e-01	15.76	2.63	6.2185
13	2^{-13}	5.480914e-01	15.80	1.84	7.4617
14	2^{-14}	5.479580e-01	15.82	1.45	8.3010
15	2^{-15}	5.478908e-01	15.83	1.45	8.7940
16	2^{-16}	5.478571e-01	15.83	1.05	9.0622
17	2^{-17}	5.478402e-01	15.84	1.05	9.2021
18	2^{-18}	5.478317e-01	15.84	1.05	9.2736

Table C.2.: Optimization results for $\eta_{\text{ext}} = 1$ and coarse mesh. The table reports the objective functional, improvement vs. initial objective, computational time, and mesh quality.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	6.495636e-01	0.00	2.75	1.4512
1	2^{-1}	6.313526e-01	2.80	1.92	1.5186
2	2^{-2}	6.104021e-01	6.03	1.94	1.5383
3	2^{-3}	5.910127e-01	9.01	1.88	1.5432
4	2^{-4}	5.757809e-01	11.36	1.86	1.5475
5	2^{-5}	5.649708e-01	13.02	1.89	1.6019
6	2^{-6}	5.578068e-01	14.13	1.88	1.6996
7	2^{-7}	5.533321e-01	14.81	1.84	1.8087
8	2^{-8}	5.506940e-01	15.22	1.44	1.9070
9	2^{-9}	5.492186e-01	15.45	1.45	1.9951
10	2^{-10}	5.484273e-01	15.57	1.44	2.0629
11	2^{-11}	5.480151e-01	15.63	1.50	2.1143
12	2^{-12}	5.478042e-01	15.67	1.45	2.1479
13	2^{-13}	5.476975e-01	15.68	1.44	2.1676
14	2^{-14}	5.476438e-01	15.69	1.05	2.1785
15	2^{-15}	5.476168e-01	15.69	1.04	2.1842
16	2^{-16}	5.476033e-01	15.70	1.05	2.1871
17	2^{-17}	5.475966e-01	15.70	1.09	2.1886

Table C.3.: Optimization results for $\eta_{\text{ext}} = 0$ and coarse mesh. Objective value, improvement from initial, and mesh quality are reported. After 6 iteration the mesh started overlapping.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	6.496366e-01	0.00	2.74	1.4512
1	2^{-1}	6.325293e-01	2.63	1.93	1.5278
2	2^{-2}	6.134500e-01	5.57	1.90	1.5593
3	2^{-3}	5.953279e-01	8.36	1.87	1.5841
4	2^{-4}	5.800564e-01	10.71	1.86	1.5983
5	2^{-5}	5.683698e-01	12.51	1.87	1.7728
6	2^{-6}	5.601777e-01	13.77	1.89	3.3677

C.2. Fine Mesh

The following results were obtained using a high-resolution mesh loaded from `mesh/final_mesh_13350.xml`. The mesh consists of 13350 cells and 7175 vertices, with a total volume of approximately 780.86. The optimization process employed the FEniCS default Newton solver, and the regularization parameter α was initialized at 1.0 and halved at each iteration. The following parameters were used:

- Viscosity $\nu = 0.04$
- Initial $\alpha = 1.0$
- Decay factor $\alpha_{\text{dec}} = 0.5$
- Stopping tolerance 10^{-5}
- Determinant bound $\eta_{\text{det}} = 0.2$
- Penalty parameter $\beta = 5.0$
- Maximum iterations: 80

Table C.4.: Optimization results using a fine mesh for $\eta_{\text{ext}} = 3$. Objective value, improvement, runtime, and mesh quality are reported per iteration.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	6.482732e-01	0.00	41.57	1.5753
1	2^{-1}	6.322408e-01	2.47	28.85	1.5724
2	2^{-2}	6.156521e-01	5.03	28.53	1.5973
3	2^{-3}	6.002533e-01	7.41	28.28	1.7546
4	2^{-4}	5.865731e-01	9.52	28.27	2.1189
5	2^{-5}	5.749514e-01	11.31	28.26	2.7573
6	2^{-6}	5.656424e-01	12.75	28.66	3.7958
7	2^{-7}	5.586919e-01	13.82	28.81	5.3675
8	2^{-8}	5.538973e-01	14.56	28.47	7.4805
9	2^{-9}	5.508516e-01	15.03	28.40	10.3389
10	2^{-10}	5.490590e-01	15.30	28.20	13.0817
11	2^{-11}	5.480668e-01	15.46	28.39	35.2375
12	2^{-12}	5.475405e-01	15.54	28.73	69.1369
13	2^{-13}	5.472688e-01	15.58	34.45	123.6280

Table C.5.: Optimization results using a fine mesh for $\eta_{\text{ext}} = 1$. Objective value, improvement, runtime, and mesh quality are reported per iteration.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	0.6463987	0.00%	41.68	1.7530
1	2^{-1}	0.6277301	2.89%	28.95	1.8560
2	2^{-2}	0.6070405	6.09%	28.74	1.8906
3	2^{-3}	0.5884704	8.96%	28.98	1.9105
4	2^{-4}	0.5740639	11.19%	28.60	2.1420
5	2^{-5}	0.5638123	12.78%	28.21	2.3989
6	2^{-6}	0.5569522	13.84%	28.20	2.6668
7	2^{-7}	0.5526187	14.51%	28.32	3.0681
8	2^{-8}	0.5500380	14.91%	28.95	3.4883
9	2^{-9}	0.5485845	15.13%	22.82	3.8351
10	2^{-10}	0.5478021	15.25%	22.30	4.0806
11	2^{-11}	0.5473941	15.32%	22.36	4.2332
12	2^{-12}	0.5471854	15.35%	22.36	4.3197
13	2^{-13}	0.5470798	15.36%	22.22	4.3660
14	2^{-14}	0.5470267	15.37%	22.89	4.3899
15	2^{-15}	0.5470001	15.38%	22.79	4.4022
16	2^{-16}	0.5469867	15.38%	16.75	4.4083
17	2^{-17}	0.5469801	15.38%	17.12	4.4114

Table C.6.: Optimization results using a fine mesh for $\eta_{\text{ext}} = 0$. Objective value, improvement, runtime, and mesh quality are reported per iteration. After 6 iteration the mesh started overlapping.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	0.6464806	0.00%	41.48	1.7530
1	2^{-1}	0.6290544	2.70%	28.88	1.8683
2	2^{-2}	0.6101465	5.62%	28.48	1.9197
3	2^{-3}	0.5924685	8.35%	28.37	1.9625
4	2^{-4}	0.5776974	10.64%	28.63	2.0040
5	2^{-5}	0.5664470	12.38%	28.77	2.6633
6	2^{-6}	0.5585836	13.60%	29.12	4.2838

D. Case of Inactive Bound on Determinant

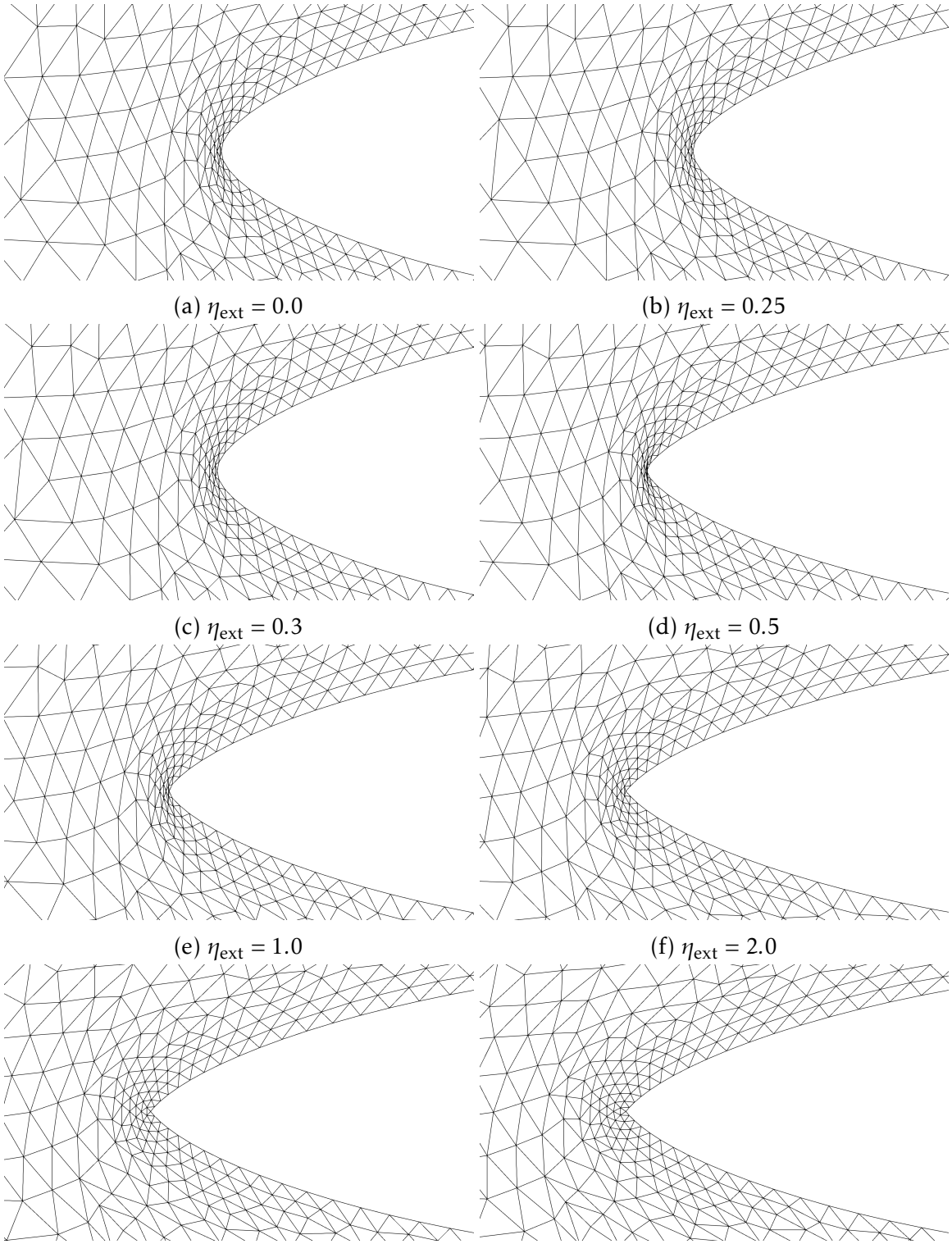
This section presents the results for the shape optimization problem using the nonlinear extension equation on a circular domain. All primary parameters are kept consistent with the example described in Section 5.2.6, ensuring comparability of the results.

The computational mesh, loaded from `mesh/circle_domain.xml`, consists of 7928 cells and 4084 vertices. The key simulation parameters are listed below:

- Viscosity $\nu = 0.04$
- Initial regularization parameter $\alpha = 1.0$
- Decay factor $\alpha_{\text{dec}} = 0.5$
- Stopping tolerance of 10^{-5}
- Determinant bound $\eta_{\text{det}} = 0.1$
- Penalty parameter $\beta = 0.0$
- Maximum iterations: 80

Since the penalty parameter β is set to zero, the determinant constraint is inactive in this configuration.

In the next tables information about mesh quality, objective function value and computation time per iteration is given for different extension factors. RR denotes radius ratio measure. The initial mesh quality is 1.5583. For cases Tables D.1 to D.4 the optimization procedure was interrupted due to mesh degeneration. For cases Tables D.5 to D.7 the procedure converged to the optimal solution with the tolerance of 10^{-5}.



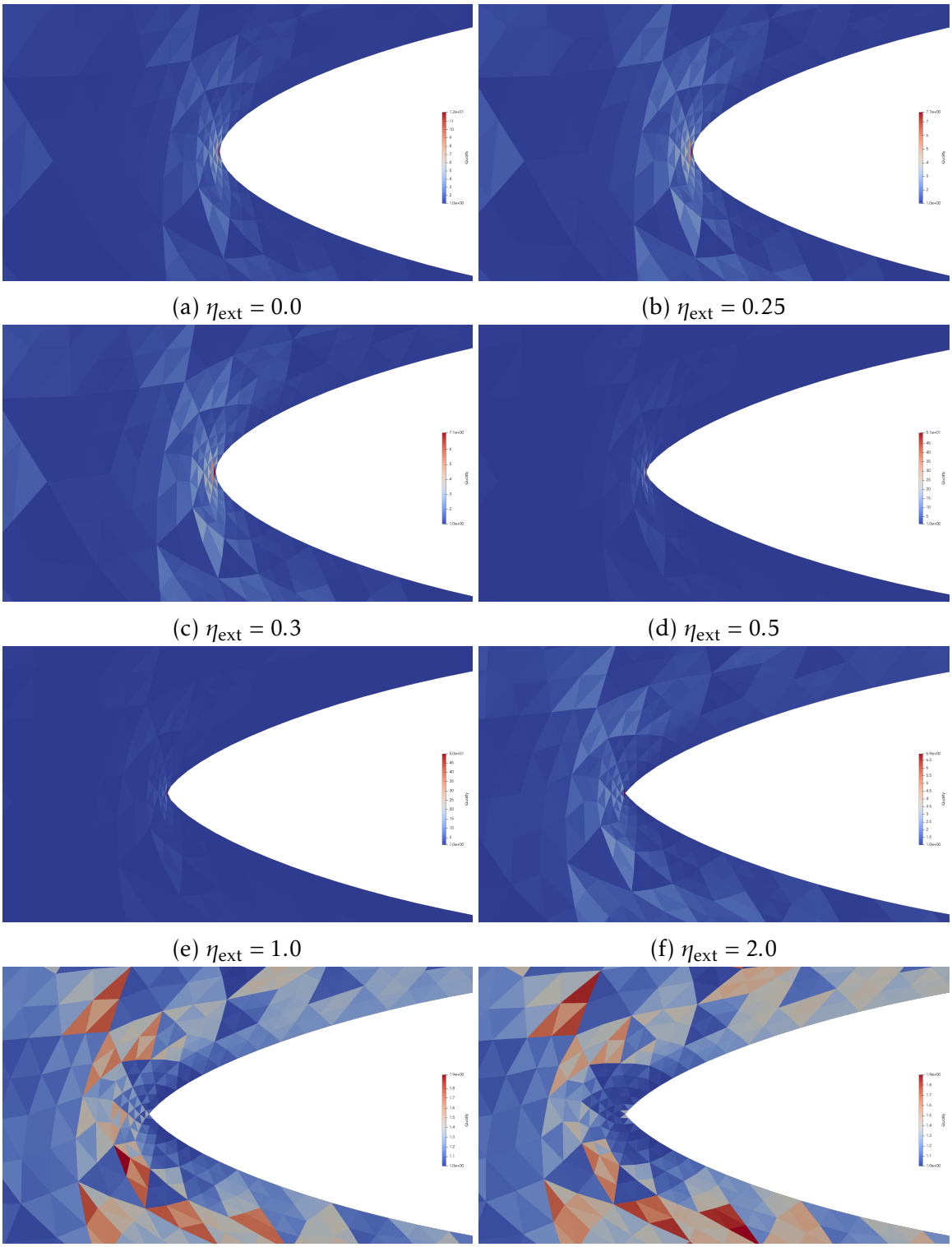
(a) $\eta_{\text{ext}} = 0.0$ (b) $\eta_{\text{ext}} = 0.25$

(c) $\eta_{\text{ext}} = 0.3$ (d) $\eta_{\text{ext}} = 0.5$

(e) $\eta_{\text{ext}} = 1.0$ (f) $\eta_{\text{ext}} = 2.0$

(g) $\eta_{\text{ext}} = 3.0$ (h) $\eta_{\text{ext}} = 4.0$

Figure D.1.: Comparison of the front wedge of the meshes for different extension factors.



(a) $\eta_{\text{ext}} = 0.0$ (b) $\eta_{\text{ext}} = 0.25$

(c) $\eta_{\text{ext}} = 0.3$ (d) $\eta_{\text{ext}} = 0.5$

(e) $\eta_{\text{ext}} = 1.0$ (f) $\eta_{\text{ext}} = 2.0$

(g) $\eta_{\text{ext}} = 3.0$ (h) $\eta_{\text{ext}} = 4.0$

Figure D.2.: Comparison of the mesh quality of front tip of the shape for different extension factors.

Table D.1.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 0$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^0	4.884440e-01	0.00	25.58	1.7010
1	2^{-1}	4.777717e-01	2.18	17.46	1.7617
2	2^{-2}	4.670459e-01	4.38	17.36	1.8179
3	2^{-3}	4.575949e-01	6.32	17.14	1.9709
4	2^{-4}	4.501066e-01	7.85	17.20	2.1974
5	2^{-5}	4.447062e-01	8.95	17.41	2.5869
6	2^{-6}	4.411439e-01	9.68	17.45	12.1432

Table D.2.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 0.25$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^0	4.884068e-01	0.00	25.54	1.7015
1	2^{-1}	4.776568e-01	2.20	17.53	1.7507
2	2^{-2}	4.668362e-01	4.42	17.21	1.8009
3	2^{-3}	4.573207e-01	6.36	17.02	1.9790
4	2^{-4}	4.498250e-01	7.90	17.32	2.2091
5	2^{-5}	4.444680e-01	9.00	17.55	2.4879
6	2^{-6}	4.409724e-01	9.71	17.29	7.7085

Table D.3.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 0.5$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^0	4.883713e-01	0.00	25.63	1.7020
1	2^{-1}	4.775449e-01	2.22	17.56	1.7642
2	2^{-2}	4.666315e-01	4.45	17.43	1.8219
3	2^{-3}	4.570548e-01	6.41	17.45	1.9871
4	2^{-4}	4.495558e-01	7.95	17.39	2.2207
5	2^{-5}	4.442443e-01	9.04	17.11	2.4642
6	2^{-6}	4.408142e-01	9.74	17.02	5.2468
7	2^{-7}	4.387734e-01	10.16	17.09	64.2224
8	2^{-8}	4.376364e-01	10.39	35.82	239.0493

Table D.4.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 1$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	4.883056e-01	0.00	25.60	1.7029
1	2^{-1}	4.773318e-01	2.25	17.46	1.7664
2	2^{-2}	4.662416e-01	4.52	17.42	1.8254
3	2^{-3}	4.565572e-01	6.50	17.44	2.0028
4	2^{-4}	4.490664e-01	8.04	17.24	2.2426
5	2^{-5}	4.438511e-01	9.10	16.95	2.4874
6	2^{-6}	4.405449e-01	9.78	17.02	2.9626
7	2^{-7}	4.386096e-01	10.18	13.39	7.4149
8	2^{-8}	4.375440e-01	10.40	16.94	49.8104

Table D.5.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 2$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	4.881986e-01	0.00	25.70	1.7042
1	2^{-1}	4.769626e-01	2.30	17.49	1.7695
2	2^{-2}	4.655814e-01	4.63	17.26	1.8300
3	2^{-3}	4.557734e-01	6.64	17.00	2.0284
4	2^{-4}	4.483698e-01	8.16	17.01	2.2748
5	2^{-5}	4.433491e-01	9.19	13.20	2.5212
6	2^{-6}	4.402324e-01	9.83	13.23	2.7324
7	2^{-7}	4.384325e-01	10.19	13.28	2.8879
8	2^{-8}	4.374487e-01	10.40	13.28	2.9882
9	2^{-9}	4.369309e-01	10.50	13.26	3.0466
10	2^{-10}	4.366645e-01	10.56	13.27	3.7274
11	2^{-11}	4.365293e-01	10.58	13.30	4.7534
12	2^{-12}	4.364611e-01	10.60	13.35	5.6590
13	2^{-13}	4.364269e-01	10.60	13.25	6.3055
14	2^{-14}	4.364098e-01	10.61	13.20	6.7016
15	2^{-15}	4.364012e-01	10.61	9.63	6.9225

Table D.6.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 3$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	2^{0}	4.881297e-01	0.00	25.65	1.7047
1	2^{-1}	4.766989e-01	2.34	17.53	1.7704
2	2^{-2}	4.651631e-01	4.71	17.34	1.8301
3	2^{-3}	4.553943e-01	6.71	17.10	2.0395
4	2^{-4}	4.481512e-01	8.19	17.04	2.2856
5	2^{-5}	4.432730e-01	9.19	13.45	2.5390
6	2^{-6}	4.402284e-01	9.81	13.29	2.7694
7	2^{-7}	4.384487e-01	10.18	13.76	2.9512
8	2^{-8}	4.374638e-01	10.38	13.76	3.0756
9	2^{-9}	4.369404e-01	10.49	13.53	3.1511
10	2^{-10}	4.366697e-01	10.54	13.39	3.1933
11	2^{-11}	4.365318e-01	10.57	13.32	3.2157
12	2^{-12}	4.364622e-01	10.58	13.36	3.2273
13	2^{-13}	4.364272e-01	10.59	13.79	3.2332
14	2^{-14}	4.364097e-01	10.60	9.92	3.2361
15	2^{-15}	4.364009e-01	10.60	9.62	3.2376

Table D.7.: Values of cost functional and mesh quality for each optimization step using the nonlinear extension equation with $\eta_{\text{ext}} = 4$.

it	$\alpha = 2^{-\text{it}}$	J	Improvement (%)	Time (s)	RR
0	1.000000	4.881052e-01	0.00	25.63	1.7044
1	0.500000	4.765739e-01	2.36	17.45	1.7686
2	0.250000	4.650559e-01	4.72	17.24	1.8273
3	0.125000	4.554610e-01	6.69	17.10	2.0356
4	0.062500	4.483765e-01	8.14	17.18	2.2819
5	0.031250	4.435446e-01	9.13	17.43	2.5516
6	0.015625	4.404581e-01	9.76	13.65	2.8171
7	0.007812	4.386066e-01	10.14	13.41	3.0438
8	0.003906	4.375587e-01	10.36	13.36	3.2097
9	0.001953	4.369928e-01	10.47	13.30	3.3156
10	0.000977	4.366972e-01	10.53	13.25	3.3767
11	0.000488	4.365458e-01	10.56	13.21	3.4097
12	0.000244	4.364691e-01	10.58	13.28	3.4269
13	0.000122	4.364306e-01	10.59	13.44	3.4357
14	0.000061	4.364112e-01	10.59	9.64	3.4401
15	0.000031	4.364015e-01	10.59	9.56	3.4423

Bibliography

[1] M. S. Alnæs et al. "Unified Form Language: A domain-specific language for weak formulations of partial differential equations". In: *ACM Transactions on Mathematical Software* 40 (2014). https://dl.acm.org/doi/10.1145/2566630.

[2] J.J. Alonso, P. LeGresley, and V. Pereyra. "Aircraft design optimization". In: *Mathematics and Computers in Simulation* 79.6 (2009). https://doi.org/10.1016/j.matcom.2007.07.001.

[3] D. N. Arnold and A. Logg. "Periodic Table of the Finite Elements". In: *SIAM News* 47 (2014). https://api.semanticscholar.org/CorpusID:117949506.

[4] H. Azegami. *Shape Optimization Problems of Domain Variation Type*. Springer, 2020. https://doi.org/10.1007/978-981-15-7618-8.

[5] I. Babuska and A. K. Aziz. "On the Angle Condition in the Finite Element Method". In: *SIAM Journal on Numerical Analysis* 13.2 (1976). https://doi.org/10.1137/0713021.

[6] S. Balay et al. *PETSc Web page*. https://petsc.org/.

[7] E. Bängtsson, D. Noreland, and M. Berggren. "Shape optimization of an acoustic horn". In: *Computer Methods in Applied Mechanics and Engineering* 192.11 (2003). https://doi.org/10.1016/S0045-7825(02)00656-4.

[8] C. Bauer, A. Frink, and R. Kreckel. "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language". In: *Journal of Symbolic Computation* 33.1 (2002). https://doi.org/10.1006/jsco.2001.0494.

[9] M. P Bendsøe. "Recent Developments in Topology Design of Continuum Structures". In: *Proc. NAFEMS Seminar on Topology Optimization*. NAFEMS-Vertretung, c/o CAD-FEM GmbH, 1997. https://doi.org/10.1051/proc:2002029.

[10] M. P. Bendsøe. *Optimization of Structural Topology, Shape, and Material*. Springer Berlin Heidelberg, 1995. https://doi.org/10.1007/978-3-662-03115-5.

[11] M. P. Bendsøe and N. Kikuchi. "Generating optimal topologies in structural design using a homogenization method". In: *Computer Methods in Applied Mechanics and Engineering* 71.2 (1988). https://doi.org/10.1016/0045-7825(88)90086-2.

[12] O. Bendsøe M.and Sigmund. "Material interpolation schemes in topology optimization". In: *Archive of Applied Mechanics* 69 (1999). https://doi.org/10.1007/s004190050248.

[13] R. Bergmann et al. "Shape optimization: what to do first, optimize or discretize?" In: *PAMM* 19.1 (2019). https://doi.org/10.1002/pamm.201900067.

[14] B. Bourdin and A. Chambolle. "Design-dependent loads in topology optimization". In: *European Series in Applied and Industrial Mathematics (ESAIM)* 9 (2003). https://doi.org/10.1051/cocv:2002070.

[15] J.-M. Bourot. "On the numerical computation of the optimum profile in Stokes flow". In: *Journal of Fluid Mechanics* 65.3 (1974). https://doi.org/10.1017/S0022112074001510.

[16] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. https://doi.org/10.1017/CBO9780511804441.

[17] C. Brandenburg et al. *A Continuous Adjoint Approach to Shape Optimization for Navier-Stokes Flow*. Ed. by K. Kunisch et al. Vol. 160. Internat. Ser. Numer. Math. Birkhäuser, Basel, 2009. https://doi.org/10.1007/978-3-7643-8923-9_2.

[18] J. L. Bull, A. J. Hunt, and E. Meyhöfer. "A theoretical model of a molecular-motor-powered pump". In: *Biomed. Microdevices* 7.1 (2005). https://doi.org/10.1007/s10544-005-6168-6.

[19] M. Burger and S. J. Osher. "A survey on level set methods for inverse problems and optimal design". In: *European Journal of Applied Mathematics* 16.2 (2005). https://doi.org/10.1017/S0956792505006182.

[20] G. Burgreen and J. Antaki. "CFD-based design optimization of a three-dimensional rotary blood pump". In: *American Society of Mechanical Engineers, Bioengineering Division* 39 (2012). https://doi.org/10.2514/6.1996-4185.

[21] E. Campana et al. "Shape optimization in ship hydrodynamics using computational fluid dynamics". In: *Computer Methods in Applied Mechanics and Engineering* 196 (2006). https://doi.org/10.1016/j.cma.2006.06.003.

[22] C. F. Chen et al. "A microfluidic nanoliter mixer with optimized grooved structures driven by capillary pumping". In: *Journal of Micromechanics and Microengineering* 16.7 (2006). https://doi.org/10.1088/0960-1317/16/7/033.

[23] D. Chenais. "On the existence of a solution in a domain identification problem". In: *J. Math. Anal. Appl.*, 52 (1975). https://doi.org/10.1016/0022-247X(75)90091-8.

[24] S. T. Christensen and N. Olhoff. "Shape Optimization of a Loudspeaker Diaphragm with Respect to Sound Directivity Properties". In: *Control and Cybernetics* 27 (1998).

[25] P. G. Ciarlet. *Linear and Nonlinear Functional Analysis with Applications*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2013. https://doi.org/10.1137/1.9781611972597.

[26] "Chapter 1 Geometrical and Other Preliminaries". In: Studies in Mathematics and Its Applications 20 (1988). Ed. by Philippe G. Ciarlet. https://doi.org/10.1016/S0168-2024(08)70058-4.

[27] J. De los Reyes. *Numerical PDE-Constrained Optimization*. Springer, 2015. https://doi.org/10.1007/978-3-319-13395-9.

[28] *Discover Engineering*. https://www.discoverengineering.org/aerodynamic-shape-optimization/.

[29] J. S. Dokken, S. K. Mitusch, and S. W. Funke. "Automatic shape derivatives for transient PDEs in FEniCS and Firedrake". In: (2020). arXiv: 2001.10058 [math.OC]. https://doi.org/10.48550/arXiv.2001.10058.

[30] J. S. Dokken, S. K. Mitusch, and S. W. Funke. "Automatic shape derivatives for transient PDEs in FEniCS and Firedrake". In: (2020). eprint: arXiv:2001.10058. https://doi.org/10.48550/arXiv.2001.10058.

[31] R. Duvigneau and M. Visonneau. "Hybrid genetic algorithms and artificial neural networks for complex design optimization in CFD". In: *International Journal for Numerical Methods in Fluids* 44.11 (2004). https://doi.org/10.1002/fld.688.

[32] M. Fischer et al. "Fréchet differentiability of unsteady incompressible Navier-Stokes flow with respect to domain variations of low regularity by using a general analytical framework". In: *SIAM J. Control Optim.* 55.5 (2017). https://doi.org/10.1137/16M1089563.

[33] Giovanni P. Galdi. *An introduction to the mathematical theory of the Navier–Stokes equations*. Springer, 2011.

[34] A. K. Gautesen. "A Force Relationship for Two-Dimensional Viscous Flow Past a Semi-Infinite Plate and an Obstacle". In: *SIAM J. Appl. Math.* 22.3 (1972). https://doi.org/10.1137/0122043.

[35] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2019. https://doi.org/10.1137/1.9781611975604.

[36] A. Habbal. "Nonsmooth Shape Optimization Applied to Linear Acoustics". In: *SIAM J. Optim.* 8 (1998). https://doi.org/10.1137/S105262349529581.

[37] J. Haslinger and R. A. E. Makinen. *Introduction to Shape Optimization: Theory, Approximation, and Computation*. USA: Society for Industrial and Applied Mathematics, 2003. https://doi.org/10.1137/1.9780898718690.

[38] J. Haubner, M. Siebenborn, and M. Ulbrich. "A Continuous Perspective on Shape Optimization via Domain Transformations". In: *SIAM Journal on Scientific Computing* 43.3 (2021). https://doi.org/10.1137/20m1332050.

[39] J. Haubner, M. Ulbrich, and S. Ulbrich. "Analysis of shape optimization problems for unsteady fluid-structure interaction". In: *Inverse Problems* 36.3 (2020). https://doi.org/10.1088/1361-6420/ab5a11.

[40] M. Hinze and F. Tröltzsch. "Discrete concepts versus error analysis in PDE-constrained optimization". In: *GAMM-Mitteilungen* 33 (2010). https://doi.org/10.1002/gamm.201010012.

[41] M. Hinze et al. *Optimization with PDE Constraints*. Mathematical Modelling: Theory and Applications, Vol. 23. Springer, 2009. https://doi.org/10.1007/978-1-4020-8839-1.

[42] S. Hofmann, M. Mitrea, and M. Taylor. "Geometric and transformational properties of Lipschitz domains, Semmes-Kenig-Toro domains, and other classes of finite perimeter domains". In: *Journal of Geometric Analysis* 17 (2007). https://doi.org/10.1007/BF02937431.

[43] J. C. Newman III et al. *Computational design optimization using RANS*. in: 24th Symp, 2002.

[44] A. L. Rogers J. F. Manwell J. G. McGowan. *Wind Characteristics and Resources*. John Wiley & Sons, Ltd, 2009. https://doi.org/10.1002/9781119994367.

[45] A. Jameson, L. Martinelli, and N. A. Pierce. "Optimum Aerodynamic Design Using the Navier-Stokes Equations". In: *Theoretical and Computational Fluid Dynamics* 10 (1998). https://doi.org/10.1007/s001620050060.

[46] Antony Jameson and Kui Ou. *Optimization Methods in Computational Fluid Dynamics*. American Cancer Society, 2010.

[47] W. Kausel. "Optimization of Brasswind Instruments and its Application in Bore Reconstruction". In: *Journal of New Music Research* 30.1 (2001). https://doi.org/10.1076/jnmr.30.1.69.7117.

[48] R. Kirby. "A general approach to transforming finite elements". In: *SMAI Journal of Computational Mathematics* 4 (2017). https://smai-jcm.centre-mersenne.org/articles/10.5802/smai-jcm.33/.

[49] R. C. Kirby. *FIAT: Numerical Construction of Finite Element Basis Functions*. Ed. by A. Logg, K.-A. Mardal, and G. N. Wells. Vol. 84. Lecture Notes in Computational Science and Engineering. Springer, 2012. Chap. 13. https://doi.org/10.1007/978-3-642-23099-8_13.

[50] K. Kunisch and G. Peichl. "Numerical gradients for shape optimization based on embedding domain techniques". English. In: *Comput. Optim. Appl.* 18.2 (2001). https://doi.org/10.1023/A:1008779803348.

[51] H. Li and R. Bashir. "On the Design and Optimization of Micro-Fluidic Dielectrophoretic Devices: A Dynamic Simulation Study". In: *Biomedical Microdevices* 6.4 (2004). https://doi.org/10.1023/B:BMMD.0000048561.26086.1a.

[52] F. K. H. Lindemann. "Theoretical and Numerical Aspects of Shape Optimization with Navier-Stokes Flows". In: (2012). https://mediatum.ub.tum.de/1108217.

[53] A. Logg. "Automating the Finite Element Method". In: *Archives of Computational Methods in Engineering - ARCH COMPUT METHOD ENG* 14 (2011). https://doi.org/10.1007/s11831-007-9003-9.

[54] A. Logg and G. N. Wells. "DOLFIN: Automated Finite Element Computing". In: *ACM Transactions on Mathematical Software* 37 (2010). https://doi.org/10.1145/1731022.1731030.

[55] A. Logg, G. N. Wells, and J. Hake. *DOLFIN: a C++/Python Finite Element Library*. Ed. by A. Logg, K.-A. Mardal, and G. N. Wells. Vol. 84. Lecture Notes in Computational Science and Engineering. Springer, 2012. Chap. 10. https://doi.org/10.1007/978-3-642-23099-8.

[56] M.C. Delfour and J.-P. Zolésio. *Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization*. 2nd. Vol. 22. Advances in Design and Control. SIAM, 2001. https://doi.org/10.1137/1.9780898719826.

[57] James Manwell, Jon Mcgowan, and A Rogers. *Wind Energy Explained: Theory, Design and Application, Second Edition*. Vol. 30. 2006. https://doi.org/10.1260/030952406778055054.

[58] B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids*. Oxford university press, 2010. https://doi.org/10.1093/acprof:oso/9780199546909.001.0001.

[59] T. D. Montenegro-Johnson and E. Lauga. "The other optimal Stokes drag profile". In: *Journal of Fluid Mechanics* 762 (2015). https://doi.org/10.1017/jfm.2014.673.

[60] F. Murat and J. Simon. "Etude de problèmes d'optimal design". In: (1976). Ed. by J. Cea. https://doi.org/10.1007/3-540-07623-9_279.

[61] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. Springer, 2006. https://doi.org/10.1007/978-0-387-40065-5.

[62] N. Olhoff. "On optimum design of structures and materials". In: *Meccanica* 31 (1996). https://doi.org/10.1007/BF00426257.

[63] S. Onyshkevych and M. Siebenborn. "Mesh Quality Preserving Shape Optimization Using Nonlinear Extension Operators". In: *Journal of Optimization Theory and Applications* 189 (2021). https://doi.org/10.1007/s10957-021-01837-8.

[64] F. Paganini A.and Wechsung and P. E. Farrell. "Higher-Order Moving Mesh Methods for PDE-Constrained Shape Optimization". In: *SIAM Journal on Scientific Computing* 40.4 (2018). https://doi.org/10.1137/17M1133956.

[65] F. Pellegrini. *Scotch*. http://www.labri.fr/perso/pelegrin/scotch.

[66] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Technical University of Denmark, 2012. http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html.

[67] O. Pironneau. "On optimum design in fluid mechanics". In: *Journal of Fluid Mechanics* 64.1 (1974). https://doi.org/10.1017/S0022112074002023.

[68] O. Pironneau. "On optimum profiles in Stokes flow". In: *Journal of Fluid Mechanics* 59.1 (1973). https://doi.org/10.1017/S002211207300145X.

[69] J.-P. Raymond and M. Vanninathan. "A fluid-structure model coupling the Navier-Stokes equations and the Lamé system". In: *Journal de Mathématiques Pures et Appliquées* 102.3 (2014). https://doi.org/10.1016/j.matpur.2013.12.00.

[70] T. A. Reist et al. "Cross Validation of Aerodynamic Shape Optimization Methodologies for Aircraft Wing-Body Optimization". In: *AIAA Journal* 58.6 (2020). https://doi.org/10.2514/1.J059091.

[71] S Richardson. "Optimum profiles in two-dimensional Stokes flow". en. In: *Proc., Math. Phys. Sci.* 450.1940 (1995). https://www.jstor.org/stable/52775.

[72] G. I. N. Rozvany. *Topology Optimization in Structural Mechanics. CISM Courses and Lectures*. 1997. https://doi.org/10.1007/978-3-7091-2566-3.

[73] S. Schmidt, E. Wadbro, and M. Berggren. "Large-Scale Three-Dimensional Acoustic Horn Optimization". In: *SIAM Journal on Scientific Computing* 38 (2016). https://doi.org/10.1137/15M1021131.

[74] V. Schulz and M. Siebenborn. "Computational comparison of surface metrics for PDE constrained shape optimization". In: *Computational Methods in Applied Mathematics* 16.3 (2016). https://doi.org/10.1515/cmam-2016-0009.

[75] J. Shewchuk. "What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures". In: *Proceedings of the 11th International Meshing Roundtable* 73 (2002). https://hal.science/hal-04614934v1.

[76] K. Shinohara et al. "Shape optimization using adjoint variable method for reducing drag in Stokes flow". In: *International Journal for Numerical Methods in Fluids* 58 (2008).

[77] M. Siebenborn and K. Welker. "Algorithmic Aspects of Multigrid Methods for Optimization in Shape Spaces". In: *SIAM Journal on Scientific Computing* 39.6 (2017). https://doi.org/10.1137/16m1104561.

[78] O. Sigmund and J. Petersson. "Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima". In: *Structural Optimization* 16 (1998). https://doi.org/10.1007/BF01214002.

[79] J. Simon. "Sur le contrôle par un domaine géométrique". In: *Laboratoire d'Analyse Numérique de l'Université de Paris VI* (1976). https://api.semanticscholar.org/CorpusID:124497750.

[80] T. Slawig. "Shape Optimization for Semi-Linear Elliptic Equations Based on an Embedding Domain Method". In: *Applied Mathematics and Optimization* 49.2 (2004). https://doi.org/10.1007/s00245-003-0787-1.

[81] J. Sokolowski and J.-P. Zolesio. *Introduction to Shape Optimization: Shape Sensitivity Analysis*. Vol. 16. Springer Science & Business Media, 2012. https://doi.org/10.1007/978-3-642-58106-9.

[82] Roger Temam. *Navier–Stokes Equations: Theory and Numerical Analysis*. AMS, 2001.

[83] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*. Vol. 112. Graduate Studies in Mathematics. American Mathematical Society, 2010. https://doi.org/10.1090/gsm/112.

[84] *uBLAS. Software package*. http://www.boost.org/libs/numeric/ublas/d.

[85] R. Udawalpola and M. Berggren. "Optimization of an acoustic horn with respect to efficiency and directivity". In: *International journal for numerical methods in engineering* 73.11 (2008). https://doi.org/10.1002/nme.2132.

[86] M. Ulbrich. "Constrained optimal control of Navier–Stokes flow by semismooth Newton methods". In: *Systems & Control Letters* 48 (2003). Optimization and Control of Distributed Systems. https://doi.org/10.1016/S0167-6911(02)00274-8.

[87] M. Valorani, D. Peri, and E. Campana. "Sensitivity Analysis Methods to Design Optimal Ship Hulls". In: *Optimization and Engineering* 4 (2003). https://doi.org/10.1023/B:OPTE.0000005391.23022.3b.

[88] Y. Yamaguchi et al. "A method for DNA detection in a microchannel: Fluid dynamics phenomena and optimization of microchannel structure". In: *Talanta* 68.3 (2006). https://doi.org/10.1016/j.talanta.2005.05.010.

[89] R.J. Yang and C.H. Chuang. "Optimal topology design using linear programming". In: *Computers & Structures* 52.2 (1994). https://doi.org/10.1016/0045-7949(94)90279-8.

[90] H. Yu, G. Janiga, and D. Thévenin. "Computational fluid dynamics-based design optimization method for Archimedes screw blood pumps". In: *Artif. Organs* 40.4 (2016). https://doi.org/10.1111/aor.12567.

[91] J.-P. Zolésio. "The material derivative (or speed) method for shape optimization variational formulation for incompressible Euler equation". In: *Optimization of distributed parameter structures*. Ed. by E.-J. Haug and J." Cea. Ser. E Appl. Sci. NATO Adv. Sci. Inst., 1981. Chap. 50. https://api.semanticscholar.org/CorpusID:124819846.