Andreas Oslandsbotn

Shape Optimisation with the Multimesh Formulation of the Stokes Equations

June 2019

NTNU
Norwegian University of
Science and Technology

Shape Optimisation with the Multimesh Formulation of the Stokes Equations

Andreas Oslandsbotn

Applied Physics and Mathematics	
Submission date:	June 2019
Supervisor:	Simon Wolfgang Funke
Co-supervisor:	Jørgen Dokken and Jon Andreas Støvneng

Norwegian University of Science and Technology
Department of Physics

Abstract

This thesis explores the possibilities for combining the recently proposed multimesh finite element method with gradient based shape optimisation. The incentive is to reduce the computational cost related to deformations of the computational domain, that is associated with the standard finite element method. The problem that is studied is the Stokes-drag problem, where the optimisation of energy dissipation into heat, with regards to the optimal placement of objects in a multi-body system, is considered. In the problem formulation, the Stokes equations are used as the constraining partial differential equations.

Multimesh FEM relies on multiple independent meshes that can be associated with separate parts in a multi-body system. This way the method avoids the computational cost of re-meshing. As such, the scheme is particularly well suited to handle the kind of problems that is studied in this thesis. However, in order to combine this scheme with gradient based shape optimisation it is necessary to develop a gradient representation that is compatible with the partitioning of the domain.

The novel contribution in this work is the analytical derivation of a gradient representation for the Stokes-drag problem, with basis in the multimesh scheme. The shape derivative of the drag functional is derived using the adjoint approach and corollaries from the Hadamard theorem. In the derivation all constraint equations are considered on their strong form. The computational domain is considered in a multi-domain setting where the interface conditions, from the multimesh FEM formulation of the Stokes equations, are introduced on their strong form. The resulting Hadamard formulation of the shape derivative is found to be equivalent to the expression found from a single-domain analysis. Numerical experiments conducted in this work, have verified the accuracy of the gradient in several different scenarios. In particular, the gradient is demonstrated to have acceptable precision around extrema points which is further improved by refining the mesh.

Using layout optimisation of a tidal turbine farm as a benchmark, the performance of the combined multimesh shape optimisation scheme is demonstrated to replicate results from the literature. The time consumption associated with gradient evaluations and mesh update is found to be negligible compared to other operations such as assembly of and solving the linear system. Several possible applications of the developed scheme are proposed and suggestions for future work are made.

SAMMENDRAG

Denne oppgåva utforsker moglegheitene av å kombinere den nyleg føreslått "multimesh finite element method" med gradientbasert formoptimering. Insentivet til dette er å redusere berekningskostnaden relatert til deformasjonar av berekningsdomenet, som typisk er assosiert med standardformuleringa av "The finite element method". Problemet som er studert er "The Stokes-drag problem", der funksjonen som beskriv friksjon i ei væske er optimert med tanke på den relative posisjonering av lekamar i eit mangepartikkel system. I problemformulasjonen er eit sett av partielle differensialligninger kjent som "The Stokes equations", brukt for å beskrive hastigheita til den omliggande væska.

"Multimesh FEM" er ei metode som nyttar seg av mange uavhengige maskeringar av berekningsdomenet. Desse maskeringane kan då bli assosiert med ulike komponentar i mangepartikkel systemet og på denne måten kan metoden unngå berekningskostnaden relatert til endring i geometrien av berekningsdomenet. I kraft av dette er "multimesh FEM"-metoden spesielt godt eigna til å løyse den type optimeringsproblem som er studert i denne oppgåva. For å kunne kombinere denne metoden med gradientbasert formoptimering er det nødvendig å finne ein gradient representasjon som er kompatibel med ei oppdeling av berekningsdomenet.

Det nye bidraget i denne oppgåva er den analytiske utrekning av ein gradient representasjon for "The Stokes-drag problem", som er kompatibel med "multimesh" formuleringa. Den deriverte av energi dissipasjons funksjonen er utrekna ved hjelp av «The adjoint approach» og ved å nytte Hadamard teoremet. I utrekninga er den sterke forma av alle partielle differensial ligningar brukt. Berekningsdomenet er oppdelt i mange deldomene og kontinuitetbetingelsa for hastigheit og trykk er innført over grensene mellom dei ulike domena. Hadamard formuleringa for eit oppdelt berekningsdomene er funnen å være lik Hadamard formuleringa frå eit enkelt berekningsdomene. Numeriske eksperiment i denne oppgåva har verifisert presisjonen til gradienten i mange ulike situasjonar. Det er vist at gradienten har akseptabel presisjon rundt ekstremalpunkt.

Ved å bruke eit system med turbinar i ein tidevannstrøm som testproblem, har metoden vist seg å reprodusere resultat frå litteraturen. Det er funne ut at tidsbruken assosiert med gradientevalueringa og oppdatering av domene-maskeringane, er neglisjerbar samanlikna med tida det tek å løyse det lineære systemet assosiert med "The stokes equations". Mange ulike applikasjonar for den nye metoden er føreslått.

Dedication and acknowledgements

I would like to dedicate this thesis
to Hugin and Munin
whom have been my companions on this journey.

A special thanks should be directed to my supervisor Jørgen Dokken who has guided me patiently and offered a great environment for me to learn. For this I am most grateful. I would also like to thank my other supervisor here at Simula Research Laboratory, Simon Wolfgang Funke, with whom I have had many great discussions and learned allot from. A thanks should also be directed Jon Andreas Støvneng, who has been my official supervisor at NTNU.

Finally, I would like to thank my family for proofreading this thesis.

Andreas Oslandsbotn
Oslo, 6. June 2019

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter gives an introduction to the work conducted in this thesis. The first part is a discussion that aims to motivate for the particular topics that are studied and to clarify the current status of those fields that form the foundation of this work. This discussion is followed by Section 1.2 which defines the scope of the thesis, formally defined in three objectives. Section 1.3 clarifies the contribution of the current work and lists several areas of application where this study can be of relevance. The chapter concludes with a short section on the structure of the text to help the reader manoeuvre through the thesis.

1.1 Background

The relevant fields for this thesis are simulation and optimisation and the coupling of the two. Simulation in the context of physical modelling and scientific/industrial computing, is a computer based approach to replicate the behaviour of a given system by approximating a mathematical model that describe the system. Optimisation in this context, and mathematical optimisation in particular, is a field at the intersection between applied mathematics and computer science. In optimisation problems the goal is typically to optimise for some quantity constrained by some criteria, while simultaneously keeping the computational cost low. A common scenario, that is the relevant context in this thesis, involves maximising or minimising an objective function by exploring possible input values from a restricted domain.

The coupling of simulation and optimisation is an economically cost efficient way to explore the properties of physical systems and is extensively used in both industry and academia. Where physical experiments often require extensive infrastructure to be conducted, a computer based road map is often preferable as a first approach. As well as being important in their own right, results from simulation and numerical optimisation experiments can often be used as guide-lines and benchmarks for implementations in the physical lab.

Of particular interest for this work are physical systems described by partial differential equations (PDEs). In this thesis, simulation using the finite element method (FEM) will be used to obtain the approximate solution of a PDE. The advantage of combining simulation with optimisation algorithms has lead to an increasing field of research and the evolution of several different techniques to handle a great variety of problems [47]. One such family of problems are related to shape optimisation [35, 46], in which the goal is to optimise the geometry of a system with regards to some objective function and where the system is subject to some physical constraints. There is a multitude of applications for shape optimisation ranging from aerodynamic design and general optimisation in fluid mechanics [34, 45], to structure mechanics and combinations of these. Examples are optimal design of aeroplanes with regards to drag minimisation [27] and optimal positioning of objects such as current carrying multi-cables [10, 19] or tidal turbines in a tidal turbine farm [15]. In Fig. 1.1 (c), the optimal configuration of a system with 256 tidal turbines is shown outside Stroma Island. The images in Fig. 1.1 are adapted from [15], where a coupled simulation optimisation experiment was conducted.

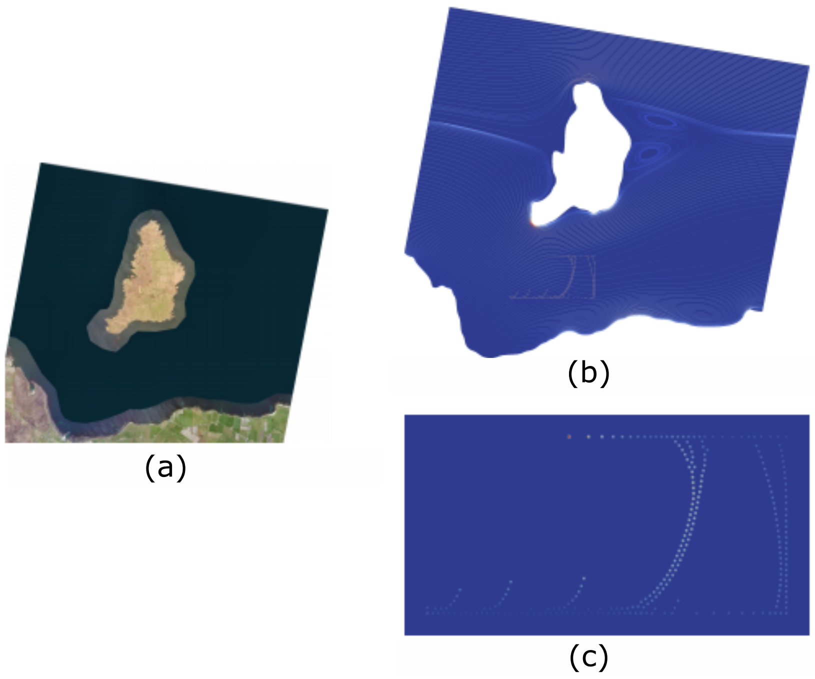

Figure 1.1: Illustration of an optimisation problem involving 256 tidal turbines in a tidal turbine farm. (a) shows a satelite image of a nearshore area. (b) shows the configuration of the tidal turbines after termination of the gradient-based optimisation algorithm, the white dots represents the individual turbines. (c) shows a zoom of the final configuration. The adjoint approach was used in the gradient evaluations. Image courtesy [15].

There are several kinds of optimisation algorithms. Common methods to name a few are;

iterative methods, where the iterates are proven to converge, and heuristics, which provide approximate solutions but where the iterates do not necessarily converge. Shape optimisation problems are typically solved using iterative methods. A commonly used iterative method is the gradient descent approach. For gradient-based optimisation strategies an important aspect is the development of efficient methods for calculating the sensitivity, i.e the derivative of the objective function with respect to the PDE constraint. An important step in this regard is the calculation of sensitivities using the adjoint approach as described in [20] and advocated in [4, 36, 41]. The main advantage with this approach, as opposed to methods where the sensitivities are calculated directly, is that the adjoint system is independent of the number of design parameters needed to describe the system. Therefore, the computational cost of evaluating the gradient will primarily be determined by the cost of solving the adjoint system which is roughly equivalent to the cost of solving the related PDE. The configuration of the tidal turbine farm shown in Fig. 1.1, was optimised using the adjoint approach to evaluate the gradient. The adjoint approach can be performed in two different ways, the "continuous approach" and the "discrete approach", which in general does not commute. In [4] it was shown that the two methods converge, under certain conditions, with decreasing mesh size. The "continuous approach" relies on a "differentiation-then-discretization" scheme. In [41] the "continuous approach", with the Hadamard formulation of the shape derivative, was advocated. From the Hadamard theorem [46] it follows that for sufficiently smooth boundaries the shape gradient is only non-zero at the boundary of the domain. The Hadamard formulation gives an analytical expression for the shape derivative which can be quickly evaluated once the PDE and corresponding adjoint equations are solved for. Therefore the gradient evaluations are very cost effective and independent of the number of design parameters. The disadvantage with the "continuous approach", compared to the "discrete approach", is that the gradient is discretely inconsistent. This issue occurs because information of the mesh structure in the perturbed domain is not included in the differentiation. In the "discrete approach", the system is first discretised and then differentiated, leading to sensitivities that are dependent on the mesh structure. This procedure ensures that the resulting sensitivity is discretely consistent as promoted in [36], but has the drawback that the mesh sensitivities are unsuitable to calculate for complex systems. As commented by [4], the "discrete approach" seem to be more reliable in obtaining the exact result, but as also commented, there seem to be some controversy as to which method is best to use. A more thorough comparison of these approaches can be found in [16].

When the geometry is changed during a shape optimisation procedure it is necessary to update the mesh to the new geometry in each iteration. Re-meshing and mesh deformation are often used for this purpose but at a high computational cost that slows down the optimisation algorithm. The multimesh finite element method (MMFEM) [24] has recently been proposed as an alternative way to deal with changing geometries. The MMFEM method is a discretisation scheme with multiple independent meshes. Fig. 1.2 illustrates the principle with abstract objects in a flow. In the figure, an independent mesh is associated with each object while a background mesh is used to represent the computational domain of the flow. Fig. 1.3 (b) offers a more concrete example in a 3D setting. Here an independent mesh is used to represent a house in an air flow, while the background mesh is again used to represent the flow. The image is adapted from [48]. The analysis of the MMFEM scheme is presented in [23], where convergence and stability of the method is proven. The multimesh finite element method is closely related to the cutFEM method first proposed in [17, 18]. The novelty with the MMFEM method is the use of non-interacting overlapping meshes to cut the elements of the underlying mesh. The method is also a generalisation that allows for multiple simultaneously overlapping meshes. In the MMFEM framework, naturally separate parts can be meshed independently. The individual meshes can then be moved and deformed separately, which greatly reduces the computational cost when only parts of the computational domain is subject to a changing geometry. To ensure continuity of the state variables across the interfaces, a weak enforcement scheme is used. This scheme is based on the work of Nitsche [37] and is therefore appropriately called Nitsche's method. This method was first adapted to enforce interface conditions on cut elements in [17, 18]. In [32] this approach was applied to the Stokes problem in a multimesh setting, proving that the method is stable, consistent and optimally convergent. The numerical implementation of this particular work can be found in [33].

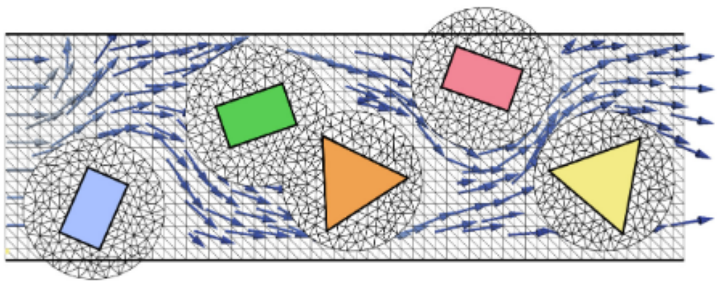

Figure 1.2: Illustration of the multimesh scheme. Abstract objects in a flow represented by independent meshes. Image courtesy [24].

In [10], the MMFEM method was used in combination with shape optimisation for several state equations and the computational cost saving of this method, compared to traditional re-meshing, was demonstrated. The work compared both the "continuous" and the "discrete approach" and found that the discrete method was infeasible for the multimesh formulation. For this reason the "continuous approach" with the Hadamard formulation was suggested as the method of choice, for shape optimisation in combination with multimesh FEM.

The Navier–Stokes equations are extremely important as they describe the physics of many phenomena of scientific and engineering interest. The equations are widely used to model fluids and gasses and as such they are important for many engineering purposes including aircraft and car design and optimisation of wind and tidal turbines [3, 21, 55]. The Stokes equations, see Definition 1.1, are a linearization of the Navier-Stokes equations and is best used to describe

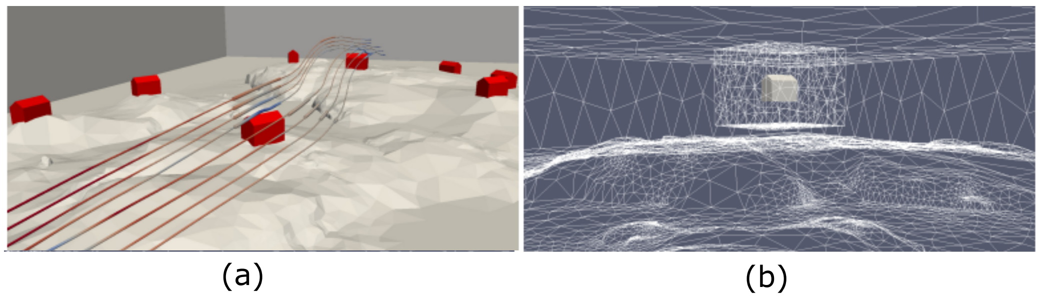

Figure 1.3: Optimisation of the position of houses in air flow with respect to wind reduction and view aspects. (a) illustrates how the house positions are subject to among other things the geometrical constraints of the landscape. (b) illustrates how the multimesh scheme is used to represent houses and air flow with independent meshes. Image curtesy [48].

slow flow of incompressible fluids. In this regard they are relevant for many important fields that involves fluids with high viscosity. Examples are the viscous environment of microorganisms [28, 54], plate tectonics [13] and micro sensors for biomedical purposes, with blood flow and respiratory monitoring [44]. The Stokes equations have also been used to simulate the flow of air, in an optimisation problem involving the optimal placement of buildings [48]. Fig. 1.3 (a) is adapted from [48] and illustrates the setup with multiple houses spread over a landscape. The right image, Fig. 1.3 (b) illustrates how the Stokes flow and the houses are represented by the multimesh scheme.

The Stokes equations are besides a popular starting point as a test problem for the design of FEM based algorithms. One reason for this is that the Stokes equations have many similarities with the Navier-Stokes equations and therefore works as a natural first step in the development of algorithms for the more complex Navier-Stokes system. In particular, several papers have been dedicated to the development of a stable MMFEM formulation for the Stokes equations [22, 32] and work is currently ongoing to develop a similar formulation for the Navier-Stokes equations.

Definition 1.1. The Stokes problem. Let $\Omega \in \mathbb{R}^2$ be the domain of interest with a piece-wise smooth boundary $\partial\Omega = \partial\Omega_D \bigcup \partial\Omega_N$ where $\partial\Omega_D$ and $\partial\Omega_N$ are the Dirichlet and Neumann boundaries respectively. Let $\mathbf{u} : \Omega \longrightarrow \mathbb{R}^2$ correspond to the velocity of the fluid and let the pressure be given by $p : \Omega \longrightarrow \mathbb{R}$. The Stokes equations are then given as,

$$-\mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \qquad \text{on } \Omega, \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{on } \Omega, \tag{1.2}$$

$$\nabla\mathbf{u} \cdot \boldsymbol{n} - p\boldsymbol{n} = \mathbf{g}_N \qquad \text{on } \partial\Omega_N, \tag{1.3}$$

$$\mathbf{u} = \mathbf{g}_D \qquad \text{on } \partial\Omega_D. \tag{1.4}$$

where $\boldsymbol{n}$ is the unit normal on $\partial\Omega$, $\boldsymbol{f}$ is a source term and both $\boldsymbol{g}_D$ and $\boldsymbol{g}_N$ are vector valued functions specific for a given problem. The constant μ is the dynamic viscosity of the fluid.

For shape optimisation and optimisation in general, a crucial step is to decide which criteria, or objective function, to optimise for. One such criteria which is commonly used is the drag, or energy dissipation into heat. For aeroplanes and cars the goal is typically to minimize this parameter, while for wind turbines and tidal turbines it could be to maximise the drag. For an incompressible fluid like the Stokes flow, the energy dissipation can be modelled with the drag functional J as defined in Lemma 1.1.

Lemma 1.1. ***(Drag in an incompressible fluid)****.*
For an incompressible fluid, the dissipation of energy into heat can be expressed as,

$$J(\boldsymbol{u},\Omega) = \mu \int_\Omega \sum_{i,j=1}^{2} \left(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_j}\right)^2 dx \tag{1.5}$$

where $\boldsymbol{u}$ *is a velocity field satisfying the state equation chosen for the problem. In this case the Stokes equations from Definition 1.1 will take this role. The constant* μ *is the dynamic viscosity from Eq.* (1.1).

Proof: *The proof can be found in [41] and follows from taking the time derivative of the expression for kinetic energy in the fluid and utilise the Navier-Stokes equations along with the incompressibility assumption that the density does not change with time.*

1.2 Overview and Scope

This work is based on two pillars. The first pillar concerns simulation of Stokes flow using the finite element method with particular focus on the multimesh finite element scheme. The second pillar concerns shape optimisation with focus on gradient based optimisation and the Hadamard formulation of the shape derivative. The optimisation problem that will be studied is defined in Definition 1.2 and will be referred to as "The Stokes-drag problem". The analysis in this thesis will be restricted to two dimensions, but the experimental setup along with the overall theory developed in this work can easily be generalised to three dimensions.

The central concern in this work is to investigate how the two pillars can be merged to form a combined multimesh-based shape optimisation scheme for the Stokes-drag problem. In this context, it is the goal to derive a gradient representation in a multimesh setting. The incentive is that to combine the multimesh setup with gradient-based shape optimisation, it is first necessary to find a representation of the gradient that is consistent with multiple independent meshes.

Definition 1.2. The Stokes-drag problem. Let $J(\boldsymbol{u},\Omega)$ be the objective function defined in Lemma 1.1 and let $\boldsymbol{g}(\boldsymbol{u},\Omega) = 0$ be the abstract form of the Stokes equations from Definition 1.1. The optimisation problem that will be studied in this work is then,

$$\max_{\boldsymbol{u},\Omega} \quad J(\boldsymbol{u},\Omega) \qquad \text{on } \Omega \tag{1.6}$$

$$\text{subject to } \boldsymbol{g}(\boldsymbol{u},p,\Omega) = 0 \qquad \text{on } \Omega \tag{1.7}$$

where $\Omega \subset \mathbb{R}^2$ is the computational domain over which the problem is defined.

The main advantage with the multimesh scheme is the computational cost saving with regards re-meshing. It must therefore be demanded of the gradient representation to also be computationally cheap so as to not diminish the advantage with the multimesh setup. In this context, the findings from [10] will be used as a starting point. In [10], the Hadamard representation of the gradient was suggested as the method of choice when doing shape optimisation with the multimesh scheme. This work will therefore use the "continuous approach" and the Hadamard theorem [46] to derive the gradient. In particular, it will be a concern to investigate how the interface conditions, emerging as a consequence of the multimesh scheme, should be treated when deriving the Hadamard form.

Numerical experiments are used to investigate the proficiency of the Hadamard representation in combination with the multimesh scheme. In this context several relevant aspects are considered including; computational cost savings for the combined scheme, proficiency of the gradient at critical points and convergence properties of the separate parts. As a benchmark model for the combined optimisation scheme a system of tidal turbines is used. This allows for a comparison of the optimal configurations found here, with similar work from the literature [15, 19]. To get a better overview of the work, the scope is formally summarised in three main objectives. A schematic of how these objectives are related is given in Fig. 1.4 at the end of this chapter.

Objective 1: The first objective in this work is dedicated to the implementation of a multimesh finite element solver for the Stokes problem in Definition 1.1. As a part of this, it is the goal to verify the correctness of the implementation and demonstrate the performance of the method. In particular, the goal is to test the performance of the method in terms of mesh update with increasing mesh complexity and compared the results to similar tests in the literature.

Objective 2: The second objective in this work has been to derive the Hadamard formulation of the shape derivative, for the Stokes-drag problem, in the context of a multi-domain partitioning. The intention is to perform the derivation using the "continuous approach" and treat all constraint equations on their strong form to see if this reproduces the result from a single-domain analysis. To the best of the writers knowledge this derivation has not been performed before in the context of the multimesh formulation and must be considered as the novel contribution in this work. The derivation of this formulation is made in Section 5.4. In the extension of this, it is the goal to verify the correctness of the formulation by numerical experiments. In particular, to see to what extent the simplifications made in the derivation will affect the precision of the gradient.

Objective 3: The last objective has been to experimentally investigate the performance of the combined multimesh-based shape optimisation scheme by building on the models developed for Objective 1 and Objective 2. The incentive is to see how the combined scheme perform in practise and to explore its suitability as a framework for solving shape optimisation problems. The performance will be tested by optimising the positions of turbines in a tidal turbine farm and benchmark the results to similar experiments from the literature.

1.3 Contribution

The shape optimisation part of this work, with the derivation of the Hademard formulation of the shape derivative, has a foundation in the work performed in [41, 43] where several results from the literature, regarding the Hademard formulation, are listed and proven. In [43] the Hadamard form for several objective functions are derived with the Navier-Stokes equations as state constraints. While in [41], the Hadamard form of the Stokes-drag problem is derived and shown to be independent of the adjoint state. This is a known results from the literature and is also derived in [35]. What distinguish the derivation performed in the present work from the works listed above, is the derivation of the Hadamard form with basis in a multi-domain partitioning of the computational domain.

To solve the Stokes problem, see Definition 1.1, the multimesh variational form is derived and implemented. This form is taken from [22, 32] where the variational form of the Stokes problem with two meshes, a background mesh and an overlapping sub-mesh, is used. The current thesis generalises this variational form to multiple non interacting sub-meshes.

The use of multiple sub-meshes for the Stokes problem has already been done in [10]. In [10] an optimisation algorithm with basis in the Hadamard and multimesh formulations was tested for the Stokes problem. What distinguish the work done in the current thesis from [10], is the actual derivation of the Hadamard formulation in a multimesh setting. In addition, this thesis focus on testing the shape derivative for the Stokes problem in terms of displacement of the object positions, while [10] tested this in terms of rotation and deformation of the object domain.

1.3.1 Relevance and applications

The results from studying the three objectives above will be of importance with regards to both simulation involving Stokes flow, and optimisation in viscous environments where the energy dissipation is a relevant parameter to optimise for. Even though the analysis in this work focus on tidal turbines, the tidal turbine setup can easily be generalised to more abstract objects in Stokes flow. Therefore, the findings in this thesis will be important to a wide range of optimisation problems. Possible applications are:

- Optimise the flow of micro-sensors in blood flow by minimising the drag.
- Minimise drag on clusters of buildings. Fig. 1.3 illustrates a typical scenario.
- Minimise drag with regards to lubrication of machine components.

It is important to be aware that the explicit results derived for the Stokes-drag problem does not directly generalise to other state equations and objective functionals. However, by investigating how to best combine shape optimisation with the multimesh scheme, this work forms a foundation for future work utilising this combined scheme. Of particular relevance is the extension of this scheme to the Navier-Stokes equations. In this context this work can be of relevance for a wider range of applications, whose typical flow conditions are well described by the Navier-Stokes equations and where optimisation of the drag is crucial. Examples of such areas of application are:

- Optimisation of swarm behaviour, drone swarms etc.
- Optimisation of wind-turbine/tidal-turbine farms. Fig. 1.1 illustrates a typical scenario.
- Optimisation of the aerodynamic shape of aeroplanes, cars and similar.

1.4 Thesis structure

This thesis is divided into three main parts. The structure is illustrated in Fig. 1.4 which also highlights the main objectives. The green boxes represents objective 1, the blue boxes represents objective 2 and the red box corresponds to the final objective where the results from objective 1 and 2 are merged to test the combined scheme.

Part I is dedicated to simulation theory with focus on the multimesh finite element method. The relevant finite element terminology is presented in Chapter 2. Similarly, Chapter 3 introduces the necessary concept of the MMFEM scheme. Chapter 3 concludes with a derivation and motivation of the multimesh weak form of the Stokes equations.

The second part, Part II, is concerned with optimisation theory and in particular with gradient based shape optimisation which is the main concern for this work. Chapter 4 starts with a short introduction to the general principles of gradient based optimisation. Chapter 5 then gives an analytical derivation of the Hadamard form of the shape derivative in a multi-domain setting.

The last part, Part III, is dedicated to implementation and numerical experiments. Chapter 6 formulates the combined multimesh shape optimisation problem that will be studied in the numerical analysis. Chapter 7 explains how a multimesh finite element solver has been implemented in python and follow up with convergence tests to check the proficiency of the implementation. Chapter 8 gives a similar overview of the implemented optimisation algorithm and again tests the implementation using a simple benchmark model and convergence tests.

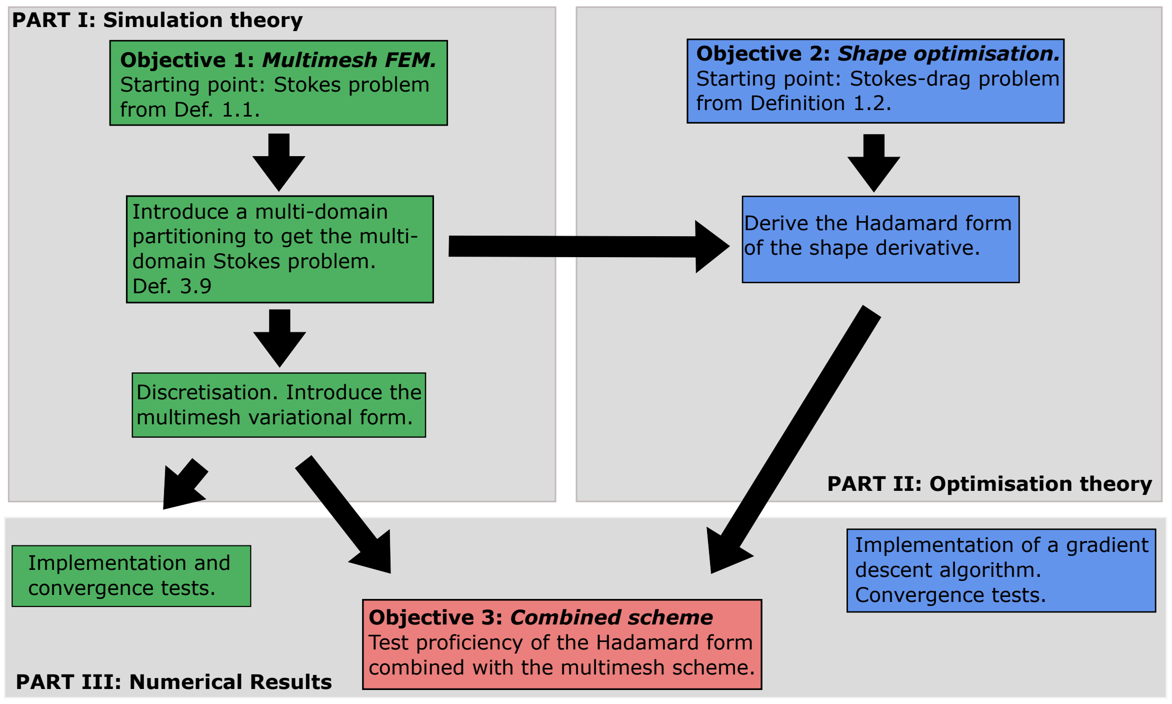
PART I: Simulation theory
Objective 1: *Multimesh FEM.* Starting point: Stokes problem from Def. 1.1.
Introduce a multi-domain partitioning to get the multi-domain Stokes problem. Def. 3.9
Discretisation. Introduce the multimesh variational form.
Objective 2: *Shape optimisation.* Starting point: Stokes-drag problem from Definition 1.2.
Derive the Hadamard form of the shape derivative.
PART II: Optimisation theory
Implementation and convergence tests.
Objective 3: *Combined scheme* Test proficiency of the Hadamard form combined with the multimesh scheme.
Implementation of a gradient descent algorithm. Convergence tests.
PART III: Numerical Results

Figure 1.4: Schematic of how the thesis is structured. The thesis is divided in three main parts concerned with each of the three main objectives in the work. The green boxes corresponds to Objective 1, the blue boxes corresponds to Objective 2 and the red box corresponds to the final objective, Objective 3.

Chapter 9 contains the actual numerical experiments. Here the multimesh scheme is tested and compared to previous work. In addition, several tests are conducted to check how the expression for the shape gradient from Chapter 5 perform in combination with the multimesh scheme. The chapter concludes by optimising the positions of several tidal turbine systems of increasing complexity.

Some comments on notation: To avoid confusion it is necessary with some short comments on the notation used in this work. This section aims to explain the most essential parts. In addition, Appendix A.1 will define the remaining conventions.

Boldface letters $\boldsymbol{v}$ will be used to highlight vectors in Euclidean space $\mathbb{R}^2$. The components of the vectors will be denoted as v_i for $i = 1,2$. Similarly, matrices will be expressed with bold capital letters $\boldsymbol{A}$ and their corresponding elements as A_{ij}. The exception is the Jacobian matrix. For a vector function $\boldsymbol{z}$ the Jacobian will be denoted $D\boldsymbol{z}$, without boldface. The notation $[\![\boldsymbol{z}]\!] = \boldsymbol{z}_0 - \boldsymbol{z}_i$ and $\langle\!\langle \boldsymbol{z} \rangle\!\rangle = (\boldsymbol{z}_0 + \boldsymbol{z}_i)/2$ will be extensively used in the text. Here $\boldsymbol{z}_0$ and $\boldsymbol{z}_i$ are vector functions on either side of an artificial interface in the computational domain $\Omega \subset \mathbb{R}$. The notation $[\![\cdot]\!]$ is the jump in $\boldsymbol{z}$ across the interface while $\langle\!\langle \cdot \rangle\!\rangle$ is the average. The dotproduct between these expressions will be written as $[\![\boldsymbol{z}]\!]\langle\!\langle \boldsymbol{z} \rangle\!\rangle := [\![\boldsymbol{z}]\!] \cdot \langle\!\langle \boldsymbol{z} \rangle\!\rangle$ etc., Where "$\cdot$" is suppressed to simplify the notation.

Part I

Simulation Theory

CHAPTER 2

THE FINITE ELEMENT METHOD

This chapter will briefly present the finite element method (FEM). In this regard the Stokes problem from Definition 1.1 will be used as a working example. For a more thorough introduction to the basic principles of the finite element method the reader is referred to Appendix C which again is based on a more comprehensive discussion in [6].

2.1 Overview

The finite element method is an approximation scheme designed to approximate the solution to boundary value problems. Starting with the strong form of the Stokes problem from Definition 1.1, the finite element method can be formulated in the following steps,

Step 1. State the problem in its weak form.

Step 2. Introduce a finite dimensional function space so that the problem can be solved as a linear system.

Step 3. Choose a basis. In the finite element method, piece-wise continuous basis functions defined over finite elements (a mesh) are used. The mesh introduces a spatial discretisation.

Step 4. Solve the linear system.

In the third step in this recipe it is necessary to chose an explicit basis. When using the finite element method, the strategy is to chose piece-wise continuous basis functions and a corresponding mesh structure. The motivation for this choice is that it greatly reduces the computational cost of solving the problem. A more thorough explanation to these concepts will be covered shortly.

2.2 Step 1. The weak formulation of the Stokes problem

The first concept to introduce, is the weak formulation of the problem. This will be done using the Stokes problem from Definition 1.1 as a working example. The weak formulation has the advantage that it puts weaker continuity constraints on the solution. Therefore, where a solution to the strong form must be up to two times continuously differentiable, the solution $(\boldsymbol{u}, p)$ to the weak form only needs the first order derivatives to be continuously integrable. In this regard the solution $(\boldsymbol{u}, p)$ is sought from an infinite dimensional Sobolov space such that $(\boldsymbol{u}, p) \in V \otimes Q$ where

$$
\begin{aligned}
V &= [\mathscr{H}^1(\Omega) : \boldsymbol{u} = g_D \text{ on } \Gamma_D], \\
Q &= [\mathscr{H}^0(\Omega)].
\end{aligned} \tag{2.1}
$$

See Appendix B.1.1 for a definition of these Sobolov spaces.

The strong form of the Stokes problem consists of the momentum equation, Eq. (1.1), and the continuity equation, Eq (1.2). The strong form of the momentum equation is converted to its weak form by taking the inner product between the residual $\boldsymbol{e}_m = -\nabla^2 \boldsymbol{u} + \nabla p - \boldsymbol{f}$, and the vector function $\boldsymbol{v} \in V$ and then integrate all second order derivatives by parts. Similarly the continuity equation is converted by taking the inner product between the residual $e_c = \nabla \cdot \boldsymbol{u}$ and the scalar fuction $q \in Q$. The residuals are then demanded to be orthogonal to the function spaces V and Q such that the inner products are zero for all $\boldsymbol{v} \in V$ and $q \in Q$. The vector functions $(\boldsymbol{u}, p)$ are typically referred to as the trial functions and the vector functions $(\boldsymbol{v}, q)$, that takes the innerproduct with the residuals, are referred to as the test functions.

For the Galerkin method that is employed here, the test and trial functions are chosen over the same function spaces. The motivation for this procedure is given in Appendix C.3. The short version of the motivation is related to the step covered in Section 2.3. When seeking the approximate solution from the finite dimensional spaces $V_h \subset V$ and $Q_h \subset Q$, the orthogonality condition ensures that the best minimiser of the residual errors $\boldsymbol{e}_m, e_c$ is chosen. This minimiser is at the same time the best approximation to the solution $(\boldsymbol{u}, p) \in V \otimes Q$ from the subspace $V_h \otimes Q_h$. More details are found in Section C.3 in the appendix.

Therefore, let $(\boldsymbol{u}, p) \in V \otimes Q$ where V and Q are as defined above. Taking the inner product then gives for the momentum equation,

$$
-\int_\Omega (\nabla^2 \boldsymbol{u}) \cdot \boldsymbol{v} dx + \int_\Omega (\nabla p) \cdot \boldsymbol{v} dx = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} dx \quad \forall \boldsymbol{v} \in V \tag{2.2}
$$

and for the continuity equation,

$$
\int_\Omega \nabla \cdot \boldsymbol{u} q = 0 \quad \forall q \in Q. \tag{2.3}
$$

To remove the second order derivative on the trial function $\boldsymbol{u}$, the Laplacian term in Eq. (2.2) is integrated by parts. The term involving ∇p is also integrated by parts, as this allows the Neumann boundary condition to be included. The result is,

$$
\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} dx - \int_{\partial\Omega} \nabla \boldsymbol{u} \cdot \boldsymbol{v} \cdot \boldsymbol{n} dS + \int_{\partial\Omega} p \boldsymbol{v} \cdot \boldsymbol{n} dS - \int_\Omega p \nabla \cdot \boldsymbol{v} dx = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} dx. \tag{2.4}
$$

The Neumann boundary condition can then be introduced in a natural way by substituting $\nabla\boldsymbol{u}\cdot\boldsymbol{n}$ with Eq. (1.3). The boundary integrals over the Dirichlet boundaries are removed since the Dirichlet conditions will be applied directly on the matrix element of the final linear system. Combining Eq. (2.4) with Eq. (2.3) gives,

$$\int_\Omega \nabla\boldsymbol{u}:\nabla\boldsymbol{v}dx - \int_\Omega q\nabla\cdot\boldsymbol{u} - \int_\Omega p\nabla\cdot\boldsymbol{v}dx = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}dx + \oint_{\partial\Omega_N}\boldsymbol{g}_N\cdot\boldsymbol{v}dS. \tag{2.5}$$

Here q has been substituted for by $-q$, a choice that will be motivated at the end of this discussion. With these equations at hand, the problem of solving the Stokes equations can be stated in the weak form in the following way. Find $(\boldsymbol{u},p)\in V\otimes Q$ such that,

$$\boldsymbol{A}(\boldsymbol{u},p,\boldsymbol{v},q) = L(\boldsymbol{v}) \quad \forall \quad (\boldsymbol{v},q) \text{ in } V\otimes Q. \tag{2.6}$$

where,

$$L(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}dx + \oint_{\partial\Omega_N}\boldsymbol{g}_N\cdot\boldsymbol{v}dS, \tag{2.7}$$

and

$$A(\boldsymbol{u},p,\boldsymbol{v},q) = a_n(\boldsymbol{u},\boldsymbol{v}) + a_q(\boldsymbol{u},q) + a_p(p,\boldsymbol{v}), \tag{2.8}$$

with,

$$a_n(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \nabla\boldsymbol{u}:\nabla\boldsymbol{v}dx, \tag{2.9}$$

$$a_q(\boldsymbol{u},q) = -\int_\Omega q\nabla\cdot\boldsymbol{u}dx \quad , \qquad a_p(\boldsymbol{v},p) = -\int_\Omega p\nabla\cdot\boldsymbol{v}dx. \tag{2.10}$$

In Eq. (2.10) it is now easy to see that the choice $q=-q$ made earlier, has made $a_q(\boldsymbol{u},q)$ and $a_p(\boldsymbol{v},p)$ symmetric with respect to exchange of $\boldsymbol{u}$ and $\boldsymbol{v}$. This property is important for the symmetry of the stiffness matrix, see Example 2.1, and is therefore of relevance for the efficiency in solving the linear system.

2.3 Step 2. Finite dimensional function spaces

The solution to the weak form resides in an infinite dimensional Sobolov space $V\otimes Q$ such that $\boldsymbol{u}\in V$ and $p\in Q$. Where V and Q are as defined in Eq. (2.1). To numerically solve this system, the idea is to reduce the problem to a finite dimensional function space such that the system can be solved as a set of algebraic equations. Following this strategy a finite N dimensional subspace is chosen such that, $V_h\otimes Q_h\subset V\otimes Q$. The subscript h is used to indicate the finite dimensional nature of the space. The approximate solution $(\boldsymbol{u}_h,p_h)$ defined over the finite space can then be expanded in its basis functions and the problem is reduced to determining the expansion coefficients $\{c_j\}_{j=1}^N$. Example 2.1 gives an idea of how the problem can be converted to a linear system by considering a simple 1D case scenario,

Example 2.1. ***Illustrative example, 1D Poisson eq. Linear system of the weak form.***
The weak form of the 1D Poisson equation corresponds to the 1D version of the bilinear form $a_n(u,v)$ in Eq. (2.9) *along with the 1D version of the linear form $L(v)$ in Eq.* (2.7)*. With $g_N = 0$ and by seeking the solution u_h from the N dimensional function space V_h, the weak form is given as: Find $u_h \in V_h$ such that,*

$$\int_0^1 \frac{du_h}{dx}\frac{dv_h}{dx}dx = \int_0^1 f(x)v_h dx \quad \forall v_h \in V_h. \tag{2.11}$$

By expanding $u_h \in V$ in the basis functions $\{\phi_i(x)\}_{i=1}^N$ of V and with $v = \phi_i(x)$ one gets,

$$\sum_{j=1}^N c_j \int_0^1 \frac{d\phi_j(x)}{dx}\frac{d\phi_i(x)}{dx}dx = \int_0^1 f(x)\phi_i dx,$$

where c_j are the expansion coefficients. This can be stated as a linear system,

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{b},$$

where $\boldsymbol{A}$ is a $N \times N$ matrix with the matrix elements,

$$A_{ij} = \int_0^1 \frac{d\phi_j(x)}{dx}\frac{d\phi_i(x)}{dx}dx,$$

and $\boldsymbol{b}$ is a N dimensional vector with the elements,

$$b_i = \int_0^1 f(x)\phi_i(x)dx.$$

The problem of solving the weak form stated in Eq. (2.11) *is therefore equivalent to solving a linear system. To obtain the approximate solution u_h it is therefore enough to solve for the coefficients $\{c_j\}_{j=1}^N$ in $\boldsymbol{c}$. The matrix $\boldsymbol{A}$ is typically referred to as the stiffness matrix. In order to do obtain this matrix and the vector $\boldsymbol{b}$, an explicit choice of basis functions is necessary. When this choice is made, the Dirichlet boundary conditions can be directly applied by setting the diagonal matrix elements $A_{k,k} = 1$ and the vector elements $b_k = g_D$ for all k such that $x_k \in \Gamma_D$. This procedure will make certain that $u_h = g_D$ on the Dirichlet boundary.*

2.4 Step 3. The finite element structure

Finite elements: For this work, Lagrange basis functions $l_j^k(\boldsymbol{x}_i)$ are adapted. These functions are polynomials defined with respect to a set of N discrete points $\{\boldsymbol{x}_i\}_{i=1}^N$, $\boldsymbol{x}_i \in \Omega$ and have the property that,

$$l_j^k(\boldsymbol{x}_i) = \begin{cases} 1 \text{ if } j = i \\ 0 \text{ if } j \neq i \end{cases} \tag{2.12}$$

where k indicates the polynomial degree. By introducing a partitioning of the computational domain Ω into a set of M elements $\{e_{k,i}\}_{i=1}^M$, it is possible to introduce piecewise continuous

Lagrange functions $l^k_{j,h}(\boldsymbol{x}_i)$ which are nonzero only within one element. It is exactly in this step that the notion of finite elements is introduced. The particular choice of finite elements corresponds a discretisation of the computational domain and the union of all elements resulting from the partitioning will then constitute the mesh $\mathcal{K}_{k,h} = \bigcup_{i=1}^{N} e_{k,i}$. This partitioning of the computational domain is the finite element structure that the finite element method has taken its name from. The advantage with this choice is that it greatly reduces the computational cost of solving the linear system. This follows because with piecewise basis functions the stiffness matrix becomes a sparse matrix with most of the entries zero. This feature can be understood from how the stiffness matrix $\boldsymbol{A}$ from Example 2.1 depends on the basis functions.

Endowed with Lagrange basis functions the elements $e_{k,i}$ are called Lagrange elements and will be denoted P_k. For this work the two dimensional Lagrange elements P_1 and P_2 will be used. These elements are illustrated in Fig. 2.1 where P_1 indicates that first order Lagrange polynomials are associated with the element. Similarly, P_2 indicates that second order Lagrange polynomials are associated with the element. For the explicit definition of these basis functions the reader is referred to [6].

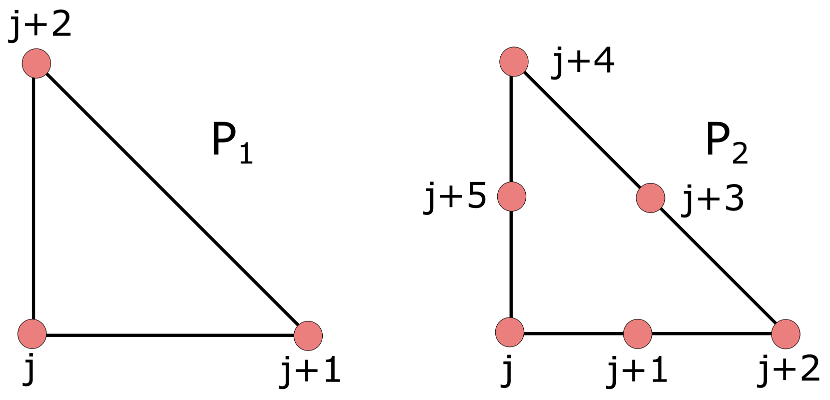

Figure 2.1: Illustration of Lagrange elements for 2D. The figure shows the first and second order Lagrange elements $P1$ and $P2$. The nodes are illustrated by pink circles and the node numbers are denoted by j.

Finite element function spaces: By choosing the finite element structure as described above, the approximate solution $(\boldsymbol{u}_h, p_h)$ can be sought from the finite dimensional space $\boldsymbol{V}_h \otimes \boldsymbol{Q}_h$ where,

$$\begin{aligned} \boldsymbol{V}_h &= [\mathcal{H}^1_h(\Omega, P_2) : \boldsymbol{u} = \boldsymbol{g}_D \text{ on } \partial\Omega_D] \\ Q_h &= [\mathcal{H}^0_h(\Omega, P_1)]. \end{aligned} \tag{2.13}$$

Here it is defined that the Dirichlet boundary conditions $\boldsymbol{g}_D$ is contained in the functions space. These boundary conditions will eventually be set directly on the matrix elements in the resulting linear system. The subscript h will now be taken to refer to the discretisation of the space, typically the average mesh size.

The mixed function space $\boldsymbol{V}_h \otimes Q_h$ is defined over the mixed P_2,P_1 element which is known as the Taylor-Hood element. This is a standard, stable element pair for the Stokes equations. In general the Taylor-Hood elements (P_{k+1}, P_k) for $k \geq 1$ are stable but the (P_2, P_1) element is computationally cheaper than elements with higher order polynomials and is therefore chosen for this work.

The weak form, defined over finite element function spaces: By seeking the solution to the weak form from the finite dimensional function spaces defined in Eq. (2.13), the weak form can be restated in the discrete case as: Find $(\boldsymbol{u}_h, p_h) \in V_h \otimes Q_h$ such that,

$$\boldsymbol{A}(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h, q_h) = L(\boldsymbol{v}_h) \quad \forall \quad (\boldsymbol{v}_h, q_h) \text{ in } V_h \otimes Q_h \tag{2.14}$$

where $A(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h, q_h)$ and $L(\boldsymbol{v}_h)$ are as defined in Eq. (2.7) through Eq. (2.10) with $(\boldsymbol{u}, p, \boldsymbol{v}, q)$ substituted for by $(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h, q_h)$. As motivated by Example 2.1, this form can be converted to a linear system with $(\boldsymbol{u}_h, p_h)$ and $(\boldsymbol{v}_h, q_h)$ expanded in their respective basis functions. The Stokes problem from Definition 1.1 has therefore been reduced to a problem of solving a linear system. The subscript h on $(\boldsymbol{u}_h, p_h)$ will for the remainder of the work be dropped except for situations where it is necessary to introduce for clarity.

CHAPTER 3

The Multimesh Finite Element Method

The multimesh finite element method (MMFEM) is a relatively new framework for solving partial differential equations and is one of the main pillars in this work. The method is tightly related to the finite element method, but allows solving PDEs on multiple independent meshes instead of a single mesh. This chapter aims to develop the necessary notation and concepts required to discuss the key aspects with the method. The chapter concludes by presenting the weak formulation of the Stokes problem from Definition 1.1, in a multimesh framework. The derivation presented here should be compared to the derivation in the finite element setting performed in Chapter 2. The discussion relies on several related works, in particular [23, 24, 32, 33], which again are related to the works on the similar cutFEM scheme in [17, 18]. The common denominator in these works is Nitsches method [37], which originally was a method designed to weakly enforce Dirichlet boundary conditions but for the multimesh scheme is adapted to handle the intersection of independent meshes. This chapter will explain how this method is applied on the multimesh formulation of the Stokes problem.

3.1 Fundamental idea

The fundamental idea behind the multimesh formulation is to partition the computational domain Ω into a background domain Ω_0 and M completely independent sub-domains Ω_i, $i = 1,2,\dots,M$, each with an associated mesh $\mathcal{K}_i$. This functionality can be advantageous in situations where the computational domain consists of naturally separate regions, see Definition 3.1, that move and deform independently of each other. Of particular relevance are situations where only a small subset of the computational domain is subject to change. In this context it can be natural to associate each of these regions with an independent sub-mesh and thereby reduce the volume that needs to be re-meshed. It is clear that this approach will be less computationally expensive than re-meshing the entire computational domain. A typical application for this scheme is for instance solid objects in a flow.

Definition 3.1. Naturally separate regions. Naturally separate regions are subsets of the computational domain Ω that move, deform or otherwise change independently of each other and are therefore natural to associate with independent meshes.

Remark 3.1. *Assumptions in this discussion*. *For simplicity, the discussion here will be restricted to $\mathbb{R}^2$ with M sub-domains, but the MMFEM scheme can easily be generalised to three dimensions. Furthermore, the sub-domains are not allowed to intersect each other or the boundary of the computational domain. The only intersection that will be allowed is that between the sub-domains and the background which will be partially overlapping eachother. In a more general formulation, also the sub-domains may overlap each other, see [24], but this is not in the scope of this work.*

3.2 Domain partition

This section will describe how the computational domain $\Omega \subset \mathbb{R}^2$ is partitioned in the multimesh scheme. The concepts and notation presented here will be important for the derivation of the multimesh formulation of the Stokes problem in Section 3.4.

3.2.1 Initial partitioning and handling of the intersection

When setting up the multimesh formulation of the weak form for a particular problem, the first step is to identify the naturally separate regions, see Definition 3.1, that are convenient to associate with independent sub-domains and sub-meshes. Under the assumption that these sub-domains cannot intersect, the division is simple and each sub-domain can be denoted Ω_j such that $\Omega_j \cap \Omega_i = \emptyset$ when $i \neq j$ and $i, j > 0$. The corresponding sub-meshes are denoted $\mathcal{K}_{j,h}$ where h indicates the mesh size.

For the background domain it is necessary with a bit more care since the background domain is overlapped by the sub-domains. To deal with this, the notion of a pre-background domain $\tilde{\Omega}_0$ and corresponding pre-background mesh $\tilde{\mathcal{K}}_{0,h}$ is introduced in Definition 3.2. The situation is illustrated in Fig. 3.1 where the blue area corresponds to the pre-background domain $\tilde{\Omega}_0$, which is taken to cover entire computational domain. When introducing the sub-domains Ω_j $j > 0$, these domains completely overlap the pre-background domain such that $\tilde{\Omega}_0 \cap \Omega_j = \Omega_j$. The reason for this setup is discussed next.

Definition 3.2. Pre-background domain and mesh. The pre-background domain is denoted $\tilde{\Omega}_0$ and corresponds to the entire computational domain Ω, such that $\tilde{\Omega}_0 = \Omega$. Likewise, the pre-background mesh is denoted $\tilde{\mathscr{K}}_{0,h}$ and covers the entire pre-background domain $\tilde{\Omega}$.

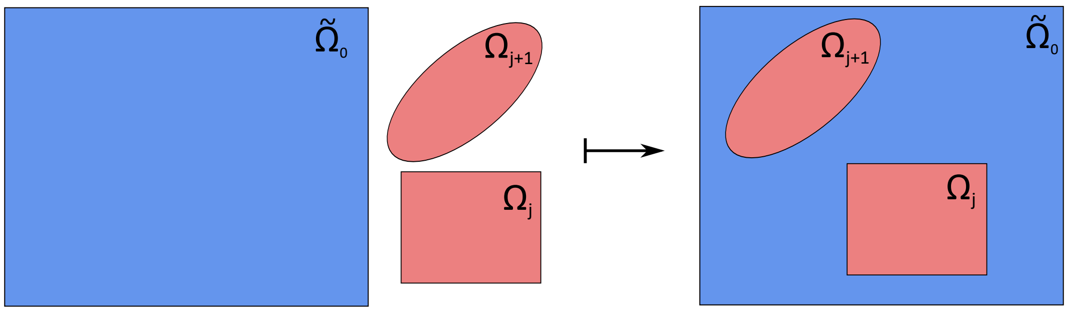

Figure 3.1: Illustration of the partitioning of the computational domain. The pre-background domain $\tilde{\Omega}_0$ corresponds to the blue area. The j-th and $j+1$-th sub-domains are represented by the pink circle and square. In general the sub-domains can have any shape.

Each of the separate sub-meshes $\mathscr{K}_{j,h}$ have the normal finite element structure as defined in Section 2.4. However, the overlap of the sub-meshes and the pre-background mesh gives rise to three different classes of finite element cells in $\tilde{\mathscr{K}}_{0,h}$. These are; cut, covered, and uncut elements as defined in Definition 3.3, 3.4 and 3.5 respectively. The situation is illustrated in Fig. 3.2. With this classification of the cells and with the pre-background mesh defined over the entire computational domain, the sub-domains can be moved and deformed without introducing the need to re-mesh the background. Instead, it is sufficient to update the cells contained in $\tilde{\mathscr{K}}_{0,h}$ to be either cut, uncut or covered. The integration domains can then be defined with respect to the classification of the cells rather than the actual mesh $\tilde{\mathscr{K}}_{0,h}$. The main idea is to set the covered cells inactive and not include them in the computation, while giving special treatment to the partially covered cells to handle the intersection of the domains. Section 3.2.2 will give more details on how the integration domains are defined with respect to the cut, uncut and covered cells. In Section 3.3 a method is presented that utilises this feature to efficiently represent changing holes in the computational domain.

Definition 3.3. Cut elements. The cut elements e_{0j}^{cut} is the set of all elements $e \in \tilde{\mathscr{K}}_{0,h}$ which are partially covered by the j-th sub-mesh $\mathscr{K}_{j,h}$. The set of all cut elements in $\tilde{\mathscr{K}}_{0,h}$ is defined as $e_0^{cut} = \{e_{0j}^{cut}\}_{j=1}^{M}$, where M is the number of sub-meshes.

Definition 3.4. Covered elements. The covered elements e_{0j}^{cov} is the set of all elements $e \in \tilde{\mathscr{K}}_{0,h}$ which are completely covered by the j-th sub-mesh $\mathscr{K}_{j,h}$. The set of all covered elements in $\tilde{\mathscr{K}}_{0,h}$ is defined as $e_0^{cov} = \{e_{0j}^{cov}\}_{j=1}^{M}$, where M is the number of sub-meshes.

Definition 3.5. Uncut elements. The uncut elements e_0^{uncut} of the pre-background mesh, is the set of all elements $e \in \tilde{\mathscr{K}}_{0,h}$ which are not intersected by any sub-mesh $\mathscr{K}_{j,h}$. Likewise, the uncut elements e_j^{uncut} of the j-th sub-mesh, is the set of all elements $e \in \mathcal{K}_{j,h}$ that are not intersected by any other sub-mesh $\mathcal{K}_{i,h}$ for $i \neq j$.

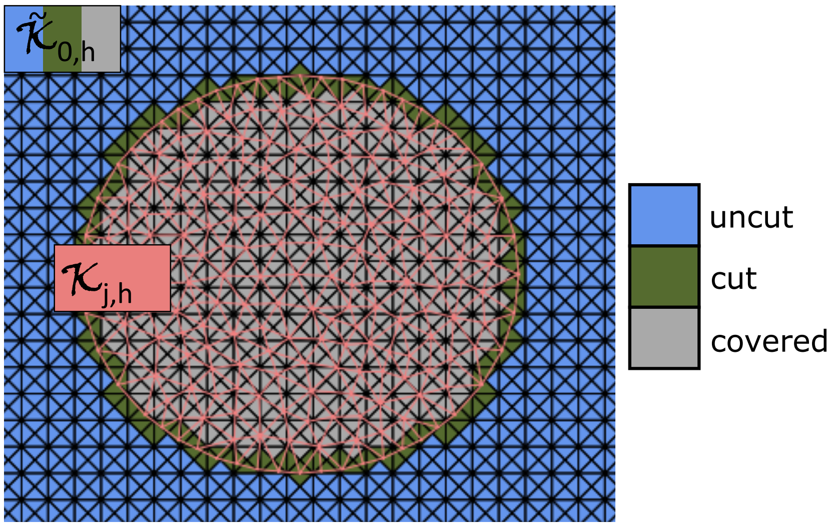

Figure 3.2: Illustration of pre-meshes with cut, uncut and covered cells. In the figure the blue cells are uncut elements. The green cells are cut elements and the grey cells are covered elements. The pre-background mesh is denoted $\tilde{\mathcal{K}}_{0,h}$ and contains all the cut, uncut and covered cells. The j-th sub-mesh is represented by the pink wireframe and is denoted $\mathcal{K}_{j,h}$. It contains only uncut cells.

Note that with the sub-domains Ω_j not allowed to intersect each other, both $e_{ij}^{cov} = \emptyset \; \forall i,j > 0$ and $e_{ij}^{cut} = \emptyset \; \forall i,j > 0$. The definitions of cut and covered elements are therefore only relevant for the background mesh. For this reason the sub-index 0 is used to label these elements.

3.2.2 Defining the integration domains

In the standard finite element formulation, the integration is performed over the computational domain Ω. With the partition of the domain in the multimesh scheme, follows three particular integration regions that are needed to handle the intersection between the overlapping domains. These are the overlap domains, the cut domains and the visible domains, as defined in Definition 3.6, 3.7 and 3.8. An overview of the situation is given in Fig. 3.3.

The integration over the cut domain $\{C_j\}_{j=1}^M$ from Definition 3.7 and the overlap domain $\{\mathcal{O}_j\}_{j=1}^M$ from Definition 3.6, is necessary to handle the intersection between the sub-meshes and the background mesh. What is special with the overlap domain is that it contains cells from both the background domain and the covering sub-domains. Therefore, when integrating over this domain, contributions from both $\boldsymbol{u}_0, p_0$ and $\boldsymbol{u}_j, p_j$, $j = 1, \dots, M$ will be counted. Here $\boldsymbol{u}_0, p_0$ are the trial functions associated with the background domain and $\boldsymbol{u}_j, p_j$ are the trial functions associated with the j-th sub-domain Ω_j. The last integration domains to be defined are the visible domains Ω_j, see Definition 3.8. These are disjoint sets and the union of all gives the

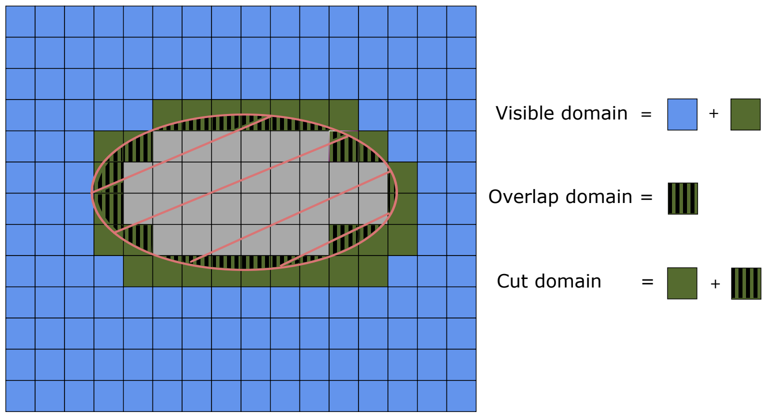

Figure 3.3: Illustration of the integration domains used in the multimesh scheme. The pink striped ellipse corresponds to the sub-domain that covers the background. The union of the blue and green regions corresponds to the visible integration domain. The overlap domain is the black and green striped region while the cut region is the union of the green region and the overlap domain. The grey region corresponds to the covered cells in the pre-background domain $\tilde{\mathcal{K}}_{0,h}$. Note that the overlap domain is defined for both the sub-domain and the background domain.

computational domain $\Omega = \bigcup_{j=0}^{M} \Omega_j$. The integration over all visible domains therefore corresponds to the integration over the entire computational domain.

What is clear from the definition of these three classes of integration domains, is that they dependent on the classification of the cells contained in the pre-background mesh $\tilde{\mathcal{K}}_{0,h}$. The advantage with this formulation is that a sub-domain Ω_j can be moved without having to re-mesh the pre-background mesh $\tilde{\mathcal{K}}_{0,h}$. When moving the j-th sub-domain it is sufficient to update the sets e_{0j}^{cov}, e_{0j}^{cut} and e_{0}^{uncov} and this automatically changes the integration domains. The computational expenses in this scheme is therefore reduced to the time it takes to move the j-th sub-mesh and to the time it takes to calculate the new sets of cut, uncut and covered cells and prepare the multimesh for integration. A more comprehensible discussion on this is given in [24]. In particular the functionality; Collision detection of cells, Intersection construction and Quadrature rule construction are relevant.

Definition 3.6. Overlap (integration) domains. The j-th overlap mesh is defined as,

$$\mathcal{O}_{j,h} = e_{0j}^{cut} \cap \mathcal{K}_{j,h}.$$

The j-th overlap domain is denoted $\mathcal{O}_j$ and corresponds to the region covered by the overlap mesh $\mathcal{O}_{j,h}$. This region is indicated by he black and green striped cells in Fig. 3.3.

Definition 3.7. Cut (integration) domains. The j-th cut mesh $C_{j,h}$, equals the set of cut elements e_{0j}^{cut} from Definition 3.3. This means that,

$$C_{j,h} = e_{0j}^{cut}$$

The j-th cut domain is denoted C_j and corresponds to the region covered by $C_{j,h}$. This region is indicated by the green cells and the black/green striped cells in Fig. 3.3.

Definition 3.8. Visible (integration) domains. The j-th visible integration domain is denoted Ω_j for $j = 0,\dots,M$ and corresponds to the part of the domain that is not covered by another domain. All the sub-domains Ω_j for $j = 1,2,\dots,M$ are visible. The visible integration domain of the background is defined as,

$$\Omega_0 = \tilde{\Omega}_0 \setminus \bigcup_{i=1}^{M} \Omega_i.$$

The visible domain of the background is illustrated by the blue and green regions in Fig. 3.3.

3.3 Mesh with hole method

In many situations it can be necessary to have holes in the computational domain Ω, over which the test and trial functions of the problem should not be defined. One such situation is where a solid object is introduced as an obstacle in a fluid flow. The multimesh method is very well suited to represent this type of objects when it is necessary to move or deform the objects during for instance an optimisation procedure.

A practical way to utilise the sub-domain structure was presented in [10]. The method involves making a sub-domain Ω_j formed as a doughnut. In other words a domain with a hole in the middle. The hole is also present in the associated sub-mesh $\mathscr{K}_{j,h}(\Omega_j)$. In the mesh with hole method, the cut e_{0j}^{cut} and uncut e_0^{uncut} elements in the pre-background mesh $\tilde{\mathscr{K}}_{0,h}$ that are inside or adjacent to the hole, are converted to covered elements e_{0j}^{cov} and thereby made inactive. The situation is illustrated in Fig. 3.4 before converting the relevant cells to covered. The same setup is illustrated in Fig. 3.5 after the conversion.

As seen from the figure, there are two boundaries associated with the hole. Depending on the boundary condition chosen on the inner boundary, the hole can be made to represent an object, an outflow region etc. The boundary Γ_j of Ω_j is partitioned in the following way,

$$\Gamma_j = \Gamma_j^{+/-} \cup \Gamma_j^{w}, \tag{3.1}$$

where $\Gamma_j^{+/-}$ is the interface between the sub-domain Ω_j and the visible background domain Ω_0 and Γ_j^w is the inner boundary of Ω_j. In this work the inner boundary will be used to represent the physical boundary of an object. The specific boundary conditions of the object can then be applied to the inner boundary, whereas $\Gamma_j^{+/-}$ and the sub-domain itself, offers the overlap $\mathscr{O}_j$ and interface necessary to perform the gluing of the sub-mesh $\mathscr{K}_{j,h}$ to the background mesh. This procedure will be covered in Section 3.4.

With this representation of the object. The object can easily be moved without changing the background mesh. The integration domains can then be updated as explained in Section 3.2.2, with limited computational effort.

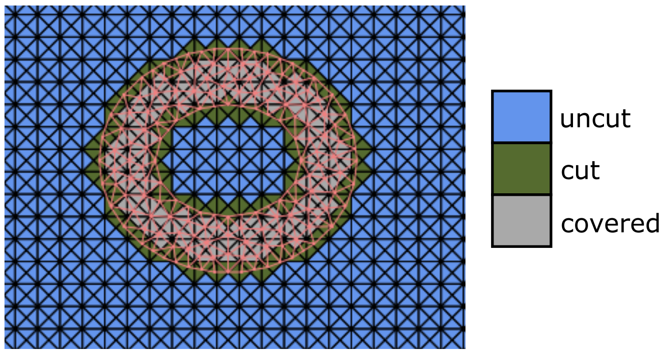

Figure 3.4: Figure showing the initial setup for the hole in domain method. The situation depicts the mesh before turning the mesh inside the hole to inactive. The union of the blue, green and grey mesh constitutes the pre-background mesh $\tilde{\mathscr{K}}_{0,h}$. The doughnut shaped pink wireframe on top of the background mesh, is the sub-mesh $\mathscr{K}_{j,h}$. In the background mesh the blue cells are uncut elements, the green cells are cut elements and the grey cells are covered elements.

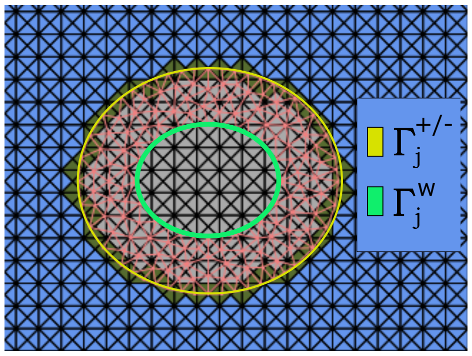

Figure 3.5: Figure showing the multimesh after the cut and uncut elements, adjacent to the hole, are changed to covered and thereby made inactive. See also Fig. 3.4. The physical boundary of the object, represented by the hole, is marked with the green inner circle. This boundary will be refereed to as Γ_j^w for the j-th sub-mesh. The interface between the sub-mesh and the background mesh is marked with a yellow circle and denoted $\Gamma_j^{+/-}$.

3.4 Multimesh formulation of the Stokes problem

This section will derive the multimesh formulation of the Stokes problem. As a starting point for this derivation it is necessary to reformulate the Stokes problem from Definition 1.1 to a multi-domain setting. The multi-domain formulation is given in Definition 3.9 below.

Definition 3.9. Multi-domain Stokes-problem. The multi-domain Stokes-problem corresponds to the strong form of the Stokes-problem, Definition 1.1, with the partitioning of the computational domain as described in Section 3.2. Let $\{\Omega_j\}_{j=0}^M$ be the visible domains that partition the computational domain $\Omega \subset \mathbb{R}^2$, as defined in Definition 3.8. The multi-domain Stokes-problem is then given as,

$$-\mu\nabla^2 \boldsymbol{u}_j + \nabla p_j = \boldsymbol{f} \quad \text{on } \Omega_j \quad j = 0,1,2,\dots,M, \tag{3.2}$$

$$\nabla\cdot\boldsymbol{u}_j = 0 \quad \text{on } \Omega_j \quad j = 0,1,2,\dots,M, \tag{3.3}$$

$$\boldsymbol{u}_j = 0 \quad \text{on } \Gamma_{j,D} \quad j = 0,1,2,\dots,M, \tag{3.4}$$

$$\nabla\boldsymbol{u}_j\cdot\boldsymbol{n}_j - p_j\boldsymbol{n}_j = \boldsymbol{g}_N \quad \text{on } \Gamma_{j,N} \quad j = 0,1,2,\dots,M. \tag{3.5}$$

with the additional interface conditions,

$$[\![\boldsymbol{u}]\!] = \boldsymbol{u}_j - \boldsymbol{u}_0 = 0 \quad \text{on} \quad \Gamma_j^{+/-} \quad j = 1,2,\dots,M, \tag{3.6}$$

$$[\![D_n \boldsymbol{u} - p]\!] = (D_n \boldsymbol{u}_j - D_n \boldsymbol{u}_0) - (p_j - p_0) = 0 \quad \text{on} \quad \Gamma_j^{+/-}. \quad j = 1,2,\dots,M, \tag{3.7}$$

The interface conditions are added to ensure continuity and proper behaviour of $\boldsymbol{u}, p$ and $\nabla \boldsymbol{u}$, across the interface $\Gamma_j^{+/-}$. Here $\Gamma_{j,D}$ and $\Gamma_{j,N}$ are respectively the Dirichlet and Neumann boundaries. For the j-th interface $\Gamma_j^{+/-}$, the unit normal $\boldsymbol{n}_j$ is chosen as the outwards pointing unit normal relative to the j-th sub-domain Ω_j. It follows therefore that the unit normal at $\Gamma_j^{+/-}$ with respect to Ω_0 is $-\boldsymbol{n}_j$. The notation $[\![z]\!] = z_j - z_0$ is denoting the jump across the j-th interface $\Gamma_j^{+/-}$ and $D_n \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ is the normal component of the Jacobian.

3.4.1 Multimesh function spaces

To convert the multi-domain Stokes-problem from Definition 3.9 to its weak form, it is first necessary to introduce the structure of the finite element spaces in a multimesh setting. The finite element spaces $\boldsymbol{V}_h$ and $\boldsymbol{Q}_h$ from Eq. (2.13) are still applicable but the function spaces must now be partitioned in accordance with the partitioning of the computational domain from Section 3.2. In this regard, one must pay particular care to define the function spaces associated with the background. The reason for this is that the computational domain of the background must be extended into the overlap region from Definition 3.6 in order to handle continuity of the gradient $\nabla \boldsymbol{u}$ across the interface. Therefore, these function spaces must be defined over both the visible domain Ω_0, and the set of overlap domains $\{\mathscr{O}_j\}_{j=1}^M$. For a compact definition of the function spaces the notion of an active domain $\Omega_{a,j}$ is introduced such that, $\Omega_{a,0} = \Omega_0 \bigcup_{j=1}^{M} \mathscr{O}_j$ for the background and simply $\Omega_{a,j} = \Omega_j$ $j = 1,\dots,M$ for the sub-domains. The function spaces for the j-th domain are then given as,

$$\boldsymbol{V}_{h,j} = [\mathscr{H}_h^1(\Omega_{a,j}, P_2) : u = g_D \text{ on } \partial\Omega_D] \quad \text{and} \quad \boldsymbol{Q}_{h,j} = [\mathscr{H}_h^0(\Omega_{a,j}, P_1)] \tag{3.8}$$

where $P_k = 1,2$ are Lagrange elements of respectively first and second order polynomials. For a definition of the Sobolov space $\mathscr{H}_h^s$ see Appendix B.1. The test and trial functions defined over these spaces are in general independent from each other and will have to be glued together when solving the problem. The mixed function space defined over the entire multimesh is then defined as,

$$\boldsymbol{W}_h^{mm} = \boldsymbol{V}_h^{mm} \otimes \boldsymbol{Q}_h^{mm} \tag{3.9}$$

where,

$$\boldsymbol{V}_h^{mm} = \bigotimes_{j=0}^{M} \boldsymbol{V}_{h,j} \quad \text{and} \quad \boldsymbol{Q}_h^{mm} = \bigotimes_{j=0}^{M} \boldsymbol{Q}_{h,j} \tag{3.10}$$

and where the mm index is added to stress the multimesh foundation.

3.4.2 The multimesh formation of the weak form

With the dynamic viscosity μ set to 1 for simplicity, the weak form of the multi-domain Stokes-problem from Definition 3.9 can be stated as follows: Find $(\boldsymbol{u},p) \in \boldsymbol{V}_h^{mm} \otimes \boldsymbol{Q}_h^{mm}$ such that,

$$A^{mm}(\boldsymbol{u},p,\boldsymbol{v},q) = L^{mm}(\boldsymbol{v}) \quad \forall \quad (v,q) \in \boldsymbol{V}_h^{mm} \otimes \boldsymbol{Q}_h^{mm} \tag{3.11}$$

where

$$\begin{aligned} A^{mm}(\boldsymbol{u},p,\boldsymbol{v},q) = {} & a_n(\boldsymbol{u},\boldsymbol{v}) + a_{q,n}(\boldsymbol{u},q) + a_{q,N}(\boldsymbol{u},q) + a_{p,n}(\boldsymbol{v},p) + a_{p,N}(\boldsymbol{v},p) \\ & + a_{IP,n}(\boldsymbol{u},\boldsymbol{v}) + a_{IP,N}(\boldsymbol{u},\boldsymbol{v}) + a_{\mathcal{O}}(\boldsymbol{u},\boldsymbol{v}) + d_h(\boldsymbol{u},p,\boldsymbol{v},q) \end{aligned} \tag{3.12}$$

and

$$L^{mm}(\boldsymbol{v}) = l_h(\boldsymbol{v}) + l_{h,s}(\boldsymbol{v}) \tag{3.13}$$

with,

$$\left.\begin{aligned} a_n(\boldsymbol{u},\boldsymbol{v}) &= \sum_{i=0}^{M} \int_{\Omega_i} \nabla \boldsymbol{u}_i : \nabla \boldsymbol{v}_i dx, \\ a_{q,n}(\boldsymbol{u},q) &= \sum_{i=0}^{M} \int_{\Omega_i} -q_i \nabla \cdot \boldsymbol{u}_i dx, \\ a_{p,n}(\boldsymbol{v},p) &= \sum_{i=0}^{M} \int_{\Omega_i} -p_i \nabla \cdot \boldsymbol{v}_i dx, \\ l_h(\boldsymbol{v}) &= \sum_{i=0}^{M} \int_{\Omega_i} \boldsymbol{f} \cdot \boldsymbol{v}_i dx + \int_{\Gamma_N} \boldsymbol{g}_N \cdot \boldsymbol{v}_i dS, \end{aligned}\right\} \begin{array}{l} \text{Standard weak form} \\ \text{on multiple domains} \end{array} \tag{3.14}$$

$$\left.\begin{aligned} a_{IP,n}(\boldsymbol{u},\boldsymbol{v}) &= -\sum_{i=1}^{M} \int_{\Gamma_i^{+/-}} \big([\![\boldsymbol{v}]\!] \langle\!\langle \nabla \boldsymbol{u} \cdot \boldsymbol{n} \rangle\!\rangle + [\![\boldsymbol{u}]\!] \langle\!\langle \nabla \boldsymbol{v} \cdot \boldsymbol{n} \rangle\!\rangle \big) dS, \\ a_{IP,N}(\boldsymbol{u},\boldsymbol{v}) &= \sum_{i=1}^{M} \frac{\beta_0}{\langle h \rangle} \int_{\Gamma_i^{+/-}} [\![\boldsymbol{u}]\!][\![\boldsymbol{v}]\!] dS, \\ a_{q,N}(\boldsymbol{u},q) &= \sum_{i=1}^{M} \int_{\Gamma_i^{+/-}} [\![\boldsymbol{n} \cdot \boldsymbol{u}]\!] \langle\!\langle q \rangle\!\rangle dS, \\ a_{p,N}(\boldsymbol{v},p) &= \sum_{i=1}^{M} \int_{\Gamma_i^{+/-}} [\![\boldsymbol{n} \cdot \boldsymbol{v}]\!] \langle\!\langle p \rangle\!\rangle dS. \end{aligned}\right\} \begin{array}{l} \text{Nitsche stabilisation} \\ \text{terms} \end{array} \tag{3.15}$$

$$\left.\begin{aligned} a_{\mathcal{O}}(\boldsymbol{u},\boldsymbol{v}) &= \sum_{i=1}^{M} \int_{\mathcal{O}_i} [\![\nabla \boldsymbol{u}]\!][\![\nabla \boldsymbol{v}]\!] dx, \\ d_h(\boldsymbol{u},p,\boldsymbol{v},q) &= \sum_{i=1}^{M} h^2 \int_{C_i} \big(\Delta \boldsymbol{u}_i - \nabla p_i\big)\big(\Delta \boldsymbol{v}_i + \nabla q_i\big) dx, \\ l_{h,s}(\boldsymbol{v},q) &= \sum_{i=1}^{M} h^2 \int_{C_i} \boldsymbol{f} \cdot \big(\Delta \boldsymbol{v}_i + \nabla q_i\big) dx. \end{aligned}\right\} \begin{array}{l} \text{Additional} \\ \text{Stabilisation terms} \end{array} \tag{3.16}$$

Here the parameter $\beta_0 > 0$ is the Nitsche parameter which must be sufficiently large to enforce continuity in $\boldsymbol{u}$ across the interface $\Gamma^{+/-}$ [22]. It is important to note that the subscripts i on the test and trial functions stress their belonging to the i-th sub-domain Ω_i for $i = 1,\dots,M$ and to the background domain for $i = 0$. These subscripts must therefore not be confused with vector components. The notation $[\![z_i]\!] = z_i - z_0$ is used to denote the jump in the function z across the interface $\Gamma_i^{+/-}$ and the notation $\langle\!\langle z_i\rangle\!\rangle = (z_i + z_0)/2$ corresponds to the average of the function across the interface. Similarly $\langle h\rangle$ is simply the average mesh size. The formulation stated here is taken from [22] where stability is shown for the simple case of one mesh intersecting the background mesh. The formulation with the additional stabilisation terms in Eq. (3.16) was first introduced in [32], also here the formulation was shown to be stable and convergent. The motivation for these terms will be given in the section below along with the motivation for the Nitsche stability terms in Eq. (3.15).

As for the standard terms in (3.14), it is clear that the terms are the same as those for the standard weak form presented in Eq. (2.7) through Eq. (2.10). The only difference is that the integration over Ω is now partitioned into integration over the overlap domains $\{\mathcal{O}_j\}_{j=1}^M$, the cut domains $\{C_j\}_{j=1}^M$ and the visible domains $\{\Omega_j\}_{j=0}^M$. See Definition 3.6, 3.7 and 3.8 for a definition of these integration domains.

3.4.3 Nitsche stabilisation terms

The Nitsche stabilisation terms defined in Eq. (3.15) are motivated by Nitsche's method, first presented in [37]. Enforcement of the interface condition $[\![\boldsymbol{u}_i]\!] = 0$ is done using the Interior penalty term $a_{IP,N}(\boldsymbol{u},\boldsymbol{v})$. This term offer a weak enforcement of the interface condition in Eq. (3.6) in the sense that the condition is applied as a penalty term and not directly introduced in the integral terms.

The enforcement of the interface condition $[\![D_n\boldsymbol{u} - p\boldsymbol{n}]\!] = 0$, from Eq. (3.7), is done with the terms $a_{IP,n}, a_{p,N}$ and $a_{q,N}$. To see how this is done it is necessary to step back to the boundary term in Eq. (2.4) where the Neuman condition was included in the normal finite element derivation. With a multimesh formulation it is clear that the integral over $\partial\Omega$ must be split into an integral over Γ_N, Γ_D and an integral over the interface terms $\bigcup_{i=1}^M \Gamma_i^{+/-}$. The integral over Γ_D is dropped since the Dirichlet conditions will be directly applied on the final system matrix. The integral over $\partial\Omega$ contains one part with the pressure p and one part with the gradient $\nabla\boldsymbol{u}\cdot\boldsymbol{n}$. Looking at the gradient integral first gives,

$$-\int_{\partial\Omega} \nabla\boldsymbol{u}\cdot\boldsymbol{v}\cdot\boldsymbol{n}dS = -\int_{\Gamma_N} \nabla\boldsymbol{u}\cdot\boldsymbol{v}\cdot\boldsymbol{n}dS - \sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} \big(\nabla\boldsymbol{u}_i\cdot\boldsymbol{v}_i\cdot\boldsymbol{n}_i + \nabla\boldsymbol{u}_0\cdot\boldsymbol{v}_0\cdot\boldsymbol{n}_0\big)dS. \tag{3.17}$$

The first term is related to the standard Neuman condition and is of less interest. What is interesting is the last integral in this equation. Here $\boldsymbol{u}_0$ comes from the fact that $\Gamma_i^{+/-}$ is a shared interface between Ω_0 and Ω_i where both $\boldsymbol{u}_0$ and $\boldsymbol{u}_i$ are defined. Making use of the jump notation

$[\![z]\!] = z_i - z_0$ and with $\boldsymbol{n}_0 = -\boldsymbol{n}_i$ one gets,

$$
\begin{aligned}
-\sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} \big(\nabla\boldsymbol{u}_i\cdot\boldsymbol{v}_i\cdot\boldsymbol{n}_i + \nabla\boldsymbol{u}_0\cdot\boldsymbol{v}_0\cdot\boldsymbol{n}_0\big)dS &= -\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} [\![\nabla\boldsymbol{u}\cdot\boldsymbol{v}\boldsymbol{n}]\!]dS \\
&= -\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \big([\![\nabla\boldsymbol{u}\cdot\boldsymbol{n}]\!]\langle\!\langle\boldsymbol{v}\rangle\!\rangle + [\![\boldsymbol{v}]\!]\langle\!\langle\nabla\boldsymbol{u}\cdot\boldsymbol{n}\rangle\!\rangle\big)dS.
\end{aligned}
\tag{3.18}
$$

Here the identity $[\![\boldsymbol{x}\boldsymbol{y}]\!] = \langle\!\langle\boldsymbol{x}\rangle\!\rangle[\![\boldsymbol{y}]\!] + \langle\!\langle\boldsymbol{y}\rangle\!\rangle[\![\boldsymbol{x}]\!]$ has been used in the last step, see Identity A.6. The pressure term is treated in a similar manner. The integration over $\partial\Omega$ in the pressure term in Eq. (2.4) is changed to an integral over Γ_N, Γ_D and an integral over the interface terms $\bigcup_{i=1}^{M}\Gamma_i^{+/-}$. The integral over Γ_D is again dropped and one gets,

$$
\int_{\partial\Omega} p\boldsymbol{v}\cdot\boldsymbol{n}dS = \int_{\Gamma_N} p\boldsymbol{v}\cdot\boldsymbol{n}dS + \sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} \big(p_i\boldsymbol{v}_i\cdot\boldsymbol{n}_i + p_0\boldsymbol{v}_0\cdot\boldsymbol{n}_0\big)dS.
\tag{3.19}
$$

The first term here will combine with the Γ_N integral in Eq. (3.17) and cancel due to the Neumann boundary condition. Looking at the last term in Eq. (3.19) and again introducing the jump notation and identity A.6 gives,

$$
\begin{aligned}
\sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} \big(p_i\boldsymbol{v}_i\cdot\boldsymbol{n}_i + p_0\boldsymbol{v}_0\cdot\boldsymbol{n}_0\big)dS &= \sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} [\![p\boldsymbol{v}\cdot\boldsymbol{n}]\!]dS \\
&= \sum_{i=1}^{M}\int_{\Gamma_i^{+/-}} \big(\langle\!\langle p\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{v}]\!] + \langle\!\langle\boldsymbol{v}\rangle\!\rangle[\![\boldsymbol{n}p]\!]\big)dS.
\end{aligned}
\tag{3.20}
$$

In the last term the surface normal $\boldsymbol{n}$ is associated with the jump in p for reasons that soon will be seen. Combining the last terms in Eq. (3.18) and Eq. (3.20) gives,

$$
\begin{aligned}
&-\int_{\partial\Omega}\nabla\boldsymbol{u}\cdot\boldsymbol{v}\cdot\boldsymbol{n}dS + \int_{\partial\Omega} p\boldsymbol{v}\cdot\boldsymbol{n}dS = \\
&-\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \big([\![\nabla\boldsymbol{u}\cdot\boldsymbol{n} - p\boldsymbol{n}]\!]\langle\!\langle\boldsymbol{v}\rangle\!\rangle + [\![\boldsymbol{v}]\!]\langle\!\langle\nabla\boldsymbol{u}\cdot\boldsymbol{n}\rangle\!\rangle - \langle\!\langle p\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{v}]\!]\big)dS \\
&= -\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \big([\![\boldsymbol{v}]\!]\langle\!\langle\nabla\boldsymbol{u}\cdot\boldsymbol{n}\rangle\!\rangle - \langle\!\langle p\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{v}]\!]\big)dS
\end{aligned}
\tag{3.21}
$$

The motivation for these transformations can now readily be seen since the interface condition $[\![D_n\boldsymbol{u} - p\boldsymbol{n}]\!] = 0$ can be introduced into Eq.(3.21) by setting the first term to zero. To maintain symmetry the terms $[\![\boldsymbol{u}]\!]\langle\!\langle\nabla\boldsymbol{v}\cdot\boldsymbol{n}\rangle\!\rangle$ and $\langle\!\langle q\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{u}]\!]$ are added. This can be done since $[\![\boldsymbol{u}]\!] = 0$ on the interface. The result is therefore,

$$
\begin{aligned}
&-\int_{\partial\Omega}\nabla\boldsymbol{u}\cdot\boldsymbol{v}\cdot\boldsymbol{n}dS + \int_{\partial\Omega} p\boldsymbol{v}\cdot\boldsymbol{n}dS = -\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \big([\![\boldsymbol{v}]\!]\langle\!\langle\nabla\boldsymbol{u}\cdot\boldsymbol{n}\rangle\!\rangle + [\![\boldsymbol{u}]\!]\langle\!\langle\nabla\boldsymbol{v}\cdot\boldsymbol{n}\rangle\!\rangle\big)dS \\
&+\sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \langle\!\langle p\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{v}]\!]dS + \sum_{i=0}^{M}\int_{\Gamma_i^{+/-}} \langle\!\langle q\rangle\!\rangle[\![\boldsymbol{n}\cdot\boldsymbol{u}]\!]dS = a_{IP,n}(\boldsymbol{u},\boldsymbol{v}) + a_{p,N}(\boldsymbol{v},p) + a_{q,N}(\boldsymbol{u},q)
\end{aligned}
\tag{3.22}
$$

where the terms from (3.15) can be identified. This concludes the derivation and all the Nitsche stabilisation terms are therefore accounted for.

3.4.4 Additional stabilisation terms

In order to ensure stabilisation of the gradient jump $[\![\nabla \boldsymbol{u}]\!]$ across the interface $\Gamma_i^{+/-}$ a penalty term integrating over the overlap region $\mathcal{O}_i$ is added,

$$a_{\mathcal{O}}(\boldsymbol{u},\boldsymbol{v}) = \sum_{i=1}^{M} \int_{\mathcal{O}_i} [\![\nabla \boldsymbol{u}]\!][\![\nabla \boldsymbol{v}]\!] dx. \tag{3.23}$$

From the definition of $\mathcal{O}_i$ in Definition 3.6 it follows that the integration runs over the background mesh and the covering sub-mesh. These double contributions helps to obtain stability of the gradients across the interfaces Γ_i, see [32] for a more thorough discussion. The two last terms in Eq. (3.16), namely $d_h(\boldsymbol{u},p,\boldsymbol{v},q)$ and $l_{h,s}(\boldsymbol{v},q)$, are stabilisation terms defined over the cut domain, see Definition 3.7. A more thorough discussion on these terms are given in [32].

Part II

Optimization Theory

CHAPTER 4

Gradient Based Optimisation

The optimisation scheme employed in this work will be rooted on the principles of line search methods where the search direction will be estimated using a gradient descent scheme. This chapter will give a short introduction to the relevant aspects of gradient based line search. For a more thorough discussion the reader is referred to [38].

4.1 A typical optimisation problem and the reduced functional

The discussion in this chapter will use the Stokes-drag problem from Definition 1.2 as a working example. A typical strategy when solving optimisation problems of this form is to seek a parameterisation that allows the reduced functional to be introduced. In this context, let $\boldsymbol{m}$ be a vector containing all the design parameters that parameterise the computational domain Ω. The constraint equation $\boldsymbol{g}(\boldsymbol{u}, p, \Omega) = 0$, from Definition 1.2, implicitly maps any choice of $\Omega(\boldsymbol{m})$ to a unique solution $(\boldsymbol{u}, p)$. It therefore follows that the state variable $\boldsymbol{u}$ can be expressed as an implicit function of the design parameters such that $\boldsymbol{u} := \boldsymbol{u}(\boldsymbol{m})$. Substituting for this in the objective functional $J(\boldsymbol{u}, \Omega)$ gives the reduced functional $J(\boldsymbol{u}(\boldsymbol{m}), \boldsymbol{m})$. The reduced optimisation problem can then be stated as in Definition 4.1.

Definition 4.1. The reduced optimisation problem. The reduced form of the Stokes-drag problem in Definition 1.2 is given as,

$$
\begin{aligned}
\min_{\boldsymbol{m}} \quad & J(\boldsymbol{u}(\boldsymbol{m}), \boldsymbol{m}) \\
\text{subject to } & \boldsymbol{b}_l \le \boldsymbol{m} \le \boldsymbol{b}_u \\
& C(\boldsymbol{m}) = 0,
\end{aligned}
$$

where $\boldsymbol{m}$ is the design parameters that parameterise the computational domain Ω. Here $\boldsymbol{b}_u$ and $\boldsymbol{b}_l$ are the upper and lower bounds on the design parameters and $C(\boldsymbol{u})$ defines possible additional restrictions.

4.2 Line search methods

This section will cover the basic principles of gradient based line search methods.

4.2.1 Fundamental principle

With the reduced problem from Definition 4.1 as a starting point, the fundamental principles of line search methods can be explained. The essential procedure in each iteration of the algorithm, is to choose a search direction $\boldsymbol{p}$ along with a step length ϵ_k. The step length will then decide how far in this direction the design parameters $\boldsymbol{m}$ should be updated. The algorithm can be stated as follows: For each iteration k update $\boldsymbol{m}$ according to,

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \epsilon_k \boldsymbol{p}_k(\boldsymbol{m}), \tag{4.1}$$

where $\boldsymbol{m}_k$ is the design parameters at iteration number k.

There are several ways to choose the search direction. The particular way in which this is done will have impact on the the convergence and efficiency of the algorithm. For gradient descent methods the common property is that the search direction should be a descent direction, in other words $\boldsymbol{p}_k = -\boldsymbol{B}_k \nabla J_k$, where J_k is the objective function evaluated at iteration k and $\boldsymbol{B}$ is a positive definite and symmetric matrix. With $B = I$ the identity matrix, the search direction is the steepest descent, but in general this is not necessarily the best choice.

4.2.2 Convergence of gradient-based line search methods

It can be shown that, under certain conditions on the choice of step length ϵ_k, a line-search is globally convergent towards a stationary point as long as the angle θ_k between the search direction $\boldsymbol{p}_k$ and the gradient ∇J_k obeys Eq. (4.2) for some positive constant δ [38]. In other words, that θ_k is bounded away from 90^o. What this means is that it is generally not necessary to choose the steepest-descent direction, as any direction that is a descent direction will do.

$$\cos\theta_k \geq \delta \geq 0 \tag{4.2}$$

This convergence property of line search methods is important the be aware of when the performance of the gradient is evaluated in Chapter 9. In particular it gives the lead to investigate how the gradient expression perform around extrama points with regards to sign change. These are the places where it is most crucial with a precise gradient evaluation, since the convergence property implies that it is not necessary to know the exact gradient as long as the descent direction is known at all times.

4.3 Backtracking-Armijo line search

Relevant for this work will be a backtracking-Armijo line search scheme and this section aims to give a very brief cover of this.

4.3.1 Armijo condition

In order to decide on a step length ϵ_k, that will give an acceptable progress of the algorithm, a sufficient decrease condition can be imposed using the Armijo condition stated in Eq. (4.3),

$$J(\boldsymbol{m}_k + \epsilon_k \boldsymbol{p}_k) \leq J(\boldsymbol{m}_k) + \delta \epsilon_k \nabla J^T \boldsymbol{p}_k(\boldsymbol{m}_k). \tag{4.3}$$

Here J is the objective function and $\delta \in (0, \frac{1}{2})$ is some constant. This condition guarantees that only step lengths ϵ_k that results in a decrease of the objective function J are chosen. The problem with this condition, is that it does not provide any lower bounds on ϵ_k. In principle therefore, the step length can be chosen arbitrarily small which makes for a very inefficient algorithm. However, by choosing the ϵ_k in a cleaver way, using a backtracking scheme, this problem can be solved. This point is covered next.

4.3.2 Backtracking

In an Armijo-backtracking scheme the step length is chosen from a decreasing sequence of values, typically $\epsilon_k = \alpha^k$ with $\alpha \in (\frac{1}{2}, 1)$, and the trail point $\boldsymbol{m}_k + \epsilon_k \boldsymbol{p}_k$ must satisfy the Armijo condition Eq. (4.3) in order to be accepted. This approach chooses the first step length ϵ_k that satisfies a sufficient decrease in the objective function J and therefore guarantee reduction in J. At the same time this scheme prevents ϵ_k from being to small which would otherwise give a slow convergence.

The Ipopt algorithm [51] used for this work is based on this scheme but with several additional functionalities. For more details on the implementation of the algorithm see [52]. A simple introduction to the general Armijo-backtracking scheme can be found in [38].

CHAPTER 5

Shape Optimisation

The goal of this chapter is to derive the shape derivative of the Stokes-drag problem, see Definition 1.2, with basis in a multi-domain setting corresponding to the multimesh formulation from Chapter 3. This derivation will correspond to the analytical aspects of Objective 2, as stated in the introduction. To the best of the authors knowledge this is the first time this derivation has been done for this particular problem in a multi-domain setting. To get a better understanding of the optimisation strategy used for this work this chapter will start with an overview in Section 5.1. The necessary concepts of shape calculus and general shape optimisation are then presented in Section 5.2. The main concepts are the Hadamard formulation and the adjoint approach. The Hadamard form of the shape derivative is a known concept from literature see for instance [46]. Section 5.2.3 explains how the adjoint approach can be combined with the Hadamard formulation to derive the shape derivative for a general optimisation problem. Section 5.3 derives the shape derivative for the Stokes-drag problem in single-domain setting. Section 5.4 then extends this to the multi-domain case.

5.1 The continuous approach to the shape derivative

The shape optimisation strategy that is adapted for this work is the so called "continuous approach" which corresponds to a "differentiate-then-discretise" scheme. The problem that is studied in this work is the Stokes-drag problem from Definition 1.2. Figure 5.1 illustrates how the "continuous approach" will be applied to this problem.

The idea in this scheme is to derive the shape derivative of the objective functional before introducing the finite element structure, i.e. the discretisation, but after the computational domain has been partitioned according to a multi domain scheme. The derivation will therefore start with the strong form of the constraint equations, but replace the constraint equation from the initial problem definition with the multi-domain Stokes-problem from Definition 3.9. Note that these equations are still on the strong form, but are now compatible with a multi-domain formulation. The adjoint approach and corollaries from the Hadamard theorem [46] will then be used to attain an analytical expression for the shape derivative, where the final expression will be on Hadamard form. The discretisation necessary to solve the system with multimesh finite element software, is then introduced after this derivation is done.

The introduction of a multi-domain partitioning before deriving the shape derivative is the novel step in this work. This step introduce additional interface conditions that must be taken into consideration when taking the shape derivative. In the spirit of the "continuous approach" these interface constraints will be considered in their strong form during the derivation. It is of interest to see if the additional interface conditions will change the final expression for the shape derivative when the strong form of the constraints are assumed.

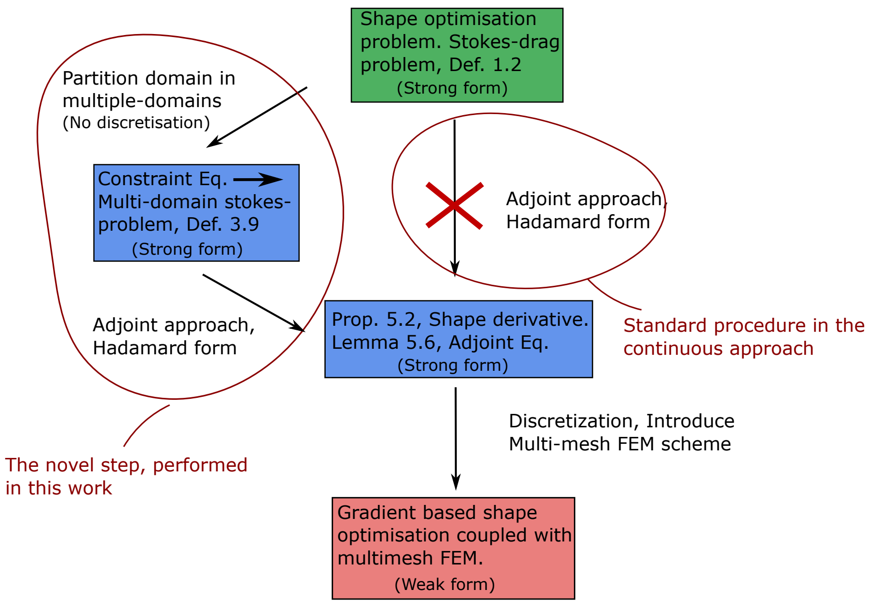
Shape optimisation problem. Stokes-drag problem, Def. 1.2 (Strong form)
Partition domain in multiple-domains (No discretisation)
Constraint Eq. Multi-domain stokes-problem, Def. 3.9 (Strong form)
Adjoint approach Hadamard form
Adjoint approach, Hadamard form
Prop. 5.2, Shape derivative. Lemma 5.6, Adjoint Eq. (Strong form)
Standard procedure in the continuous approach
The novel step, performed in this work
Discretization, Introduce Multi-mesh FEM scheme
Gradient based shape optimisation coupled with multimesh FEM. (Weak form)

Figure 5.1: Illustration of the "continuous approach" for deriving the shape derivative for the multi-domain Stokes-drag problem. The first step is to introduce the multi-domain structure by partitioning the computational domain. When doing this, the initial constraint equations from the Stokes-problem in Definition 1.1 are replaced by the multi-domain Stokes problem from Definition 3.9, this is the novel step introduced in this work. Note that for the derivation of the shape derivative in this work, it is the strong form of the constraint equations that is used. In this regard, when introducing the multi-domain structure, the new interface conditions will also be assumed in their strong form. After partitioning the domain, the standard "differentiate-then-discretise" scheme is applied. By appyling shape calculus and the adjoint approach the Hadamard form of the shape derivative is obtained. The discretisation is then introduced by applying a mutli-mesh structure over the multi-domain. The final system can be solved using finite element software that supports the multimesh scheme.

A note should also be made on the assumptions made when using the strong form of the interface conditions and the consequences of this choice. From a physical point of view the introduction of an artificial interface $\Gamma^{+/-}$ should not affect the physics of the problem. The objective function of interest, denoted J, measures the energy dissipation into heat and is as such a physical quantity. From this perspective, the final analytical expression for the shape derivative of J should be the same in the single-domain and multi-domain cases when the strong form of the interface conditions are assumed. A comparison of the shape derivative from a single-domain and a multi-domain treatment, will therefore be made to see if this statement can be confirmed. A limitation with this approach is that some of the interface conditions are only weakly enforced in the variational formulation of the problem, see the discussion in Section 3.4.3. As a consequence, by assuming strong enforcement in the derivation, the final expression does not account for these weakly enforced terms which would otherwise result in correction terms to the standard expression. Nevertheless, it is assumed that the weak enforcement is sufficient to make these correction terms insignificant from a performance perspective. The numerical analysis performed in Chapter 9 will shed light on this particular issue.

5.2 Introduction to shape calculus and shape optimisation

This section will introduce the concepts of shape calculus, the Hadamard form and the adjoint approach, in a general setting. As a starting point for this formal discussion, a general PDE constrained shape optimisation problem is defined in Definition 5.2. This problem is similar to the Stokes-drag problem from Definition 1.2, but to simplify the discussion only one state variable $\boldsymbol{z}$ will be used as opposed to the state variables $(\boldsymbol{u}, p)$ from the Stokes-drag problem. This discussion will be concerned with a general class of objective functions, known as shape functionals, as defined in Definition 5.1. The objective function will be denoted H to highlight the generality. Note that the drag functional J from Lemma 1.1 is also a shape functional.

Definition 5.1. Shape functional. A shape functional is a function $H(\Omega)$ such that $H(\Omega) : \Omega \longrightarrow \mathbb{R}$. Where Ω is some domain such that $\Omega \subset \mathbb{R}^n$. This work will consider shape functionals H and G on the form,

$$H(\Omega) = \int_\Omega h(\boldsymbol{x}) dx \quad \text{and} \quad G(\Omega) = \int_\Gamma g(\boldsymbol{x}) dS,$$

for the two dimensional case $n = 2$. Here h and g are respectively volume and surface objective functionals such that $h : \Omega \longrightarrow \mathbb{R}$ and $g : \Gamma \longrightarrow \mathbb{R}$.

Definition 5.2. A general PDE constrained shape optimisation problem. Let the objective function $H(\boldsymbol{z}, \Omega)$ be a shape functional as in Definition 5.1 and let $\boldsymbol{z} : \Omega \longrightarrow \mathbb{R}^2$ be the solution to a partial differential equation (PDE) defined by $\boldsymbol{g}(\boldsymbol{z}, \Omega) = 0$. A shape optimisation problem in this context can then be stated as,

$$\begin{aligned} &\min_{\boldsymbol{z},\Omega} \quad H(\boldsymbol{z},\Omega) && \text{on } \Omega, \\ &\text{subject to } \boldsymbol{g}(\boldsymbol{z},\Omega) = 0 && \text{on } \Omega. \end{aligned}$$

For a simplified notation the boundary conditions will be included in the constraint function $\boldsymbol{g}(\boldsymbol{z},\Omega)$.

A common way to approach optimisation problems, is to utilise the gradient of the objective function. This is a so called gradient based optimisation scheme as introduced in Chapter 4. In order to apply this kind of strategy to the shape optimisation problem from Definition 5.2, it is necessary to derive an expression for the shape derivative. This means to find the derivative of the objective functional H with respect to a change in the domain Ω, when $\boldsymbol{z}$ is subject to the constraints imposed by $\boldsymbol{g}$. The next sections will introduce the necessary concepts to calculate this derivative.

5.2.1 Definition of the shape derivative

To obtain a definition of the derivative with respect to the shape of Ω, it is convenient to consider the change in Ω as a perturbation $T_\epsilon(\boldsymbol{x})$ such that,

$$\Omega(\epsilon) = \{\boldsymbol{x}(\epsilon) = T_\epsilon(\boldsymbol{x}) \,|\, \forall\, \boldsymbol{x} \in \Omega\} \tag{5.1}$$

where $\Omega(\epsilon)$ is the perturbed domain and T_ϵ the perturbation defined as,

$$T_\epsilon(\boldsymbol{x}) = \boldsymbol{x} + \epsilon \boldsymbol{s}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{5.2}$$

The perturbation is characterised by the deformation parameter $\epsilon \geq 0$ and the displacement vector field $\boldsymbol{s}(\boldsymbol{x})$. Figure 5.2 illustrates how a domain Ω is moved and deformed under a perturbation $T_\epsilon(\boldsymbol{x})$ as defined in Eq. (5.1)-(5.2).

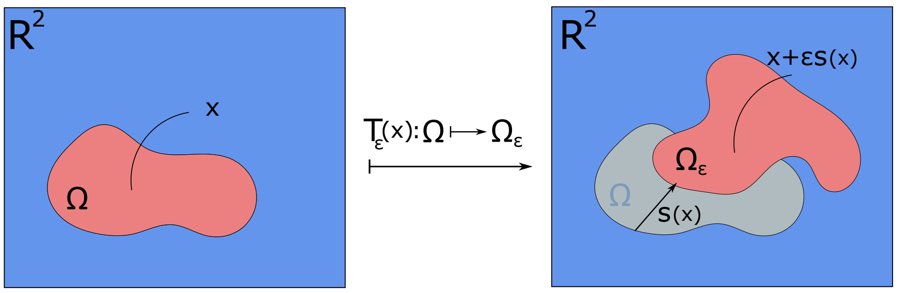

Figure 5.2: Illustration of a domain deformation under a perturbation $T_\epsilon(\boldsymbol{x}) : \Omega \longrightarrow \Omega_\epsilon$. In this particular case the perturbation vector field $\boldsymbol{s}(\boldsymbol{x})$ depends on $x \in \Omega$ in such a way that the domain is both deformed and the center of gravity is moved.

With this definition of the change in Ω, the volume and surface objective functionals h and g from Definition 5.1 should be written with an explicit dependency on the deformation parameter ϵ, to indicate their dependency on the current shape $\Omega(\epsilon)$ of the domain. This means that $h_\epsilon : \Omega(\epsilon) \longrightarrow \mathbb{R}$ where $h_\epsilon(\boldsymbol{x}) := h(\boldsymbol{x}, \epsilon)$. The functional $h(x,\epsilon)$ can for instance be the solution $\boldsymbol{z}_\epsilon$ of a PDE, with $\boldsymbol{z}_\epsilon := \boldsymbol{z}(\boldsymbol{x},\epsilon)$. The functional will then depend on its evaluation point $\boldsymbol{x}$ along with the shape of the domain $\Omega(\epsilon)$ over which the PDE is defined. To capture the dependency on ϵ in detail, the notion of the "material derivative" and the "local derivative" is typically introduced. The definitions of these derivatives are adapted from [42] and restated in Definition 5.3 and Definition 5.4 respectively.

Definition 5.3. Material derivative. Let $h(\boldsymbol{x}(\epsilon),\epsilon)$ be a shape sensitive functional evaluated in the perturbed domain $\Omega(\epsilon)$ and let $h(\boldsymbol{x},0)$ be the same functional evaluated in the unperturbed domain Ω. The material derivative in direction $\boldsymbol{s}$ can be defined as the total derivative,

$$\dot{h}(\boldsymbol{x})[\boldsymbol{s}] := \frac{d}{d\epsilon}\Big|_{\epsilon=0} h(\boldsymbol{x}(\epsilon),\epsilon) = \lim_{\epsilon\longrightarrow 0} \frac{\boldsymbol{z}(\boldsymbol{x}+\epsilon\boldsymbol{s}(\boldsymbol{x}),\epsilon) - \boldsymbol{z}(\boldsymbol{x},0)}{\epsilon}, \qquad \boldsymbol{x}\in\Omega$$

where $\boldsymbol{s}$ is the perturbation field as defined in Eq. (5.2).

Definition 5.4. Local shape derivative. Let $h(\boldsymbol{x},\epsilon)$ be a shape sensitive functional evaluated in the perturbed domain $\Omega(\epsilon)$ and let $h(\boldsymbol{x},0)$ be the same functional evaluated in the unperturbed domain Ω. The local shape derivative in direction $\boldsymbol{s}$ can be defined as the partial derivative,

$$h'[\boldsymbol{s}] := \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} h(\epsilon,\boldsymbol{x}) = \lim_{\epsilon\longrightarrow 0} \frac{\boldsymbol{z}(\boldsymbol{x},\epsilon) - \boldsymbol{z}(\boldsymbol{x},0)}{\epsilon}, \qquad \boldsymbol{x}\in\Omega$$

Note that in this definition, the $\boldsymbol{x}$ dependency in $\boldsymbol{h}'[\boldsymbol{s}](\boldsymbol{x})$ is suppressed to simplify the notation.

The material derivative should be thought of as the total derivative since it considers both the shift in the evaluation point $\boldsymbol{x}(\epsilon)$ and the explicit dependency on the domain geometry as parameterised by ϵ. The local shape derivative keeps the evaluation point $\boldsymbol{x}$ fixed and only evaluates the direct dependency on the deformation. The relation between the two derivatives is given by the chain rule as,

$$\dot{h}(\boldsymbol{x})[\boldsymbol{s}] = \frac{d}{d\epsilon}\Big|_{\epsilon=0} h(\boldsymbol{x},\epsilon) = \nabla h(\boldsymbol{x},0)\frac{d}{d\epsilon}\Big|_{\epsilon=0} \boldsymbol{x}(\epsilon) + \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} h(\boldsymbol{x}(0),\epsilon) = (\nabla h, s) + h'[\boldsymbol{s}]. \tag{5.3}$$

Here it was used that $\boldsymbol{s} = \frac{d}{d\epsilon}\Big|_{\epsilon=0} \boldsymbol{x}(\epsilon)$. For simplicity the notation $h(\boldsymbol{x}) := h(\boldsymbol{x},0)$ will from now on be adapted for all functions. By following the strategy used in [41] the shape derivative of an objective function $H(\boldsymbol{z},\Omega)$ can be defined as follows:

Definition 5.5. Shape derivative (Adapted from [41]) Let $\Omega \subset \mathbb{R}^2$ be measurable and let $\boldsymbol{s}$ be a vector field defined over Ω. Let $\Omega(\epsilon)$ be the perturbed domain as defined in Eq. (5.1). Then a shape sensitive function $H(\Omega(\epsilon))$ is shape differentiable at Ω if the derivative,

$$dH(\Omega)[\boldsymbol{s}] := \frac{d}{d\epsilon}\Big|_{\epsilon=0} H(\Omega(\epsilon)) = \lim_{\epsilon\longrightarrow 0} \frac{H(\Omega(\epsilon)) - H(\Omega)}{\epsilon},$$

exists for perturbations in all directions $\boldsymbol{s}$ and the mapping $\boldsymbol{s} \longrightarrow dH(\Omega)[\boldsymbol{s}]$ is linear and continuous. $dH(\Omega)[\boldsymbol{s}]$ will be referred to as the shape derivative of $H(\Omega)$ along $\boldsymbol{s}$.

5.2.2 Hadamard formulation of the shape derivative

For a shape functional $H(\Omega)$ as defined in Definition 5.1 and which is differentiable in the sense of Definition 5.5, it is possible to express the shape derivative $dH(\Omega)[\boldsymbol{s}]$ as an analytical expression that is known as the Hadamard formulation of the shape derivative. This follows from the Hadamard theorem which can be found with a proof in [46]. This is also true for a surface objective $G(\Omega)$ as Defined in Definition 5.1, but this discussion will first focus on the volume objective H. To prevent the discussion from becoming too technical the Hadamard theorem will not be restated here. Sufficient for this work will be Corollary 5.1,

Corollary 5.1. *(**Hadamard form**)(Corollary of the Hadamard theorem [46]) Let $H(\Omega) : \Omega \longrightarrow \mathbb{R}$ be shape differentiable as in Definition 5.5. Under the assumption that Ω has a piece-wise smooth boundary Γ, it follows from the Hadamard theorem [46] that the shape derivative $dH(\Omega)[\boldsymbol{s}]$ can be expressed as,*

$$dH(\Omega)[\boldsymbol{s}] = \int_{\Gamma} (\boldsymbol{s}, \boldsymbol{n})\tilde{k}(\boldsymbol{x})dS.$$

Here $\boldsymbol{s}(\boldsymbol{x}) : \Omega \longrightarrow \mathbb{R}$ is the displacement vector field in Eq. (5.2), $\boldsymbol{n}$ is the surface normal on Γ and $\tilde{k}$ is a scalar function, called the shape gradient, that must be differentiable on the boundary Γ.

***Proof:** Follows from the Hadamard theoreme and that $\tilde{k}$ is an integratable function on Γ. See [46], page. 59 – 60.*

The function $\tilde{k}$ known as the shape gradient, must be derived for each particular functional H. The essence in the Hadamard form from Corollary 5.1 is that the shape derivative only depends on the normal component of the perturbation field $\boldsymbol{s}$, multiplied by a boundary objective scalar function $\tilde{k}$. The independence from the interior nodes in the mesh, that was discussed in Section 1.1 can now readily be seen. Since $\tilde{k}$ is only evaluated at the boundary it does not depend on the interior nodes in the mesh. Since the shape derivative is a boundary integral, it follows therefore that also $dH(\Omega)[\boldsymbol{s}]$ is independent of these nodes. The advantages and disadvantages with this property was discussed in Section 1.1.

The "continuous" strategy for solving shape optimisation problems is to derive an analytical expression for $\tilde{k}$ for the particular objective function H of interest. A gradient based optimisation will then be performed by discretising $\tilde{k}$ and the state equations using a finite element scheme. In this work, this will be the multimesh formulation from Chapter 3.

The relevant shape derivatives for different domain sensitive functionals are listed bellow. These results are adapted from [9] and [43] and the reader is referred there for the proofs.

Lemma 5.1. *(**Shape derivative of a volume objective function**).*
Let Ω have a piece-wise smooth boundary Γ and let $\Omega(\epsilon)$ be the perturbed domain as defined in Eq. (5.1). For a shape functional H on the form,

$$H(\Omega(\epsilon)) = \int_{\Omega(\epsilon)} h(\boldsymbol{z}(\epsilon), \epsilon)dx, \tag{5.4}$$

where h is a general volume objective function $h : \Omega \longrightarrow \mathbb{R}$ and ϵ is the deformation paramter from Eq. (5.2)*, the shape derivative is given by,*

$$dH(\Omega)[\boldsymbol{s}] = \int_{\Gamma} (\boldsymbol{s}, \boldsymbol{n}) h(\boldsymbol{z}) dS + \int_{\Omega} h'[\boldsymbol{s}] dx.$$

Here Γ is the part of the surface of Ω that is changing under the perturbation field $\boldsymbol{s}$ and $h'[\boldsymbol{s}]$ is the local shape derivative as defined in Definition 5.4.

Proof: *The result follows by taking the shape derivative from Definition 5.5 of the shape functional H. The integral over the perturbed domain $\Omega(\epsilon)$ is then brought back to its initial domain Ω by a change of variable. The integral and the derivative can then swap and the material derivative from Definition 5.3 can be taken of the volume objective function $h(\boldsymbol{z}(\epsilon),\epsilon)$. The local shape derivative $h'[\boldsymbol{s}]$ is introduced by the explicit dependency on the deformation parameter ϵ and the relation between the material derivative and the local shape derivative given in Eq.* (5.3)*. For a complete proof the reader is referred to [9].*

Remark 5.1. ***(The shape gradient in Lemma 5.1)****. The shape gradient $\tilde{k}$ for a shape functional on the form defined in Lemma 5.1 is given as,*

$$\tilde{k} = h(\boldsymbol{z}).$$

This follows by comparing Lemma 5.1 with Corollary 5.1. To get the expression for the shape derivative $dH(\Omega)[\boldsymbol{s}]$ on Hadamard form, it is therefore necessary to eliminate the "local shape derivative term". This will be the purpose of the adjoint approach that is presented in the next section.

The Hadamard form can also be derived for surface objective functionals. The relevant forms are given below. Note that for these forms the shape gradient $\tilde{k}$ will attain more complex expressions.

Lemma 5.2. ***(Shape derivative of a surface objective function)****.*
Let Ω have a piece-wise smooth boundary Γ and let $\Omega(\epsilon)$ be the perturbed domain as defined in Eq. (5.1)*. For a general surface objective shape functional G defined as,*

$$G(\Omega(\epsilon)) = \int_{\Gamma(\epsilon)} g(\boldsymbol{z}(\epsilon),\epsilon) dS, \tag{5.5}$$

where g is a surface objective function $g : \Gamma \longrightarrow \mathbb{R}$ with $\frac{\partial g}{\partial n}$ existing and ϵ is the deformation parameter defined in Eq. (5.2)*, the shape derivative is given by,*

$$dG(\Omega)[\boldsymbol{s}] = \int_{\Gamma} (\boldsymbol{s}, \boldsymbol{n}) \Big[\frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{n}} + \kappa g(\boldsymbol{z}) \Big] dS + \int_{\Gamma} g'[\boldsymbol{s}] dS. \tag{5.6}$$

Here Γ is the part of the surface of Ω that is changing under the partubation field $\boldsymbol{s}$ and $\kappa = div_{\Gamma}\boldsymbol{n}$ is the divergence of the normal in tangential direction (also known as curvature). $g'[\boldsymbol{s}]$ in the last

term is the "local shape derivative" from Definition 5.4.
Proof: *The result follows from a similar procedure as described in Lemma 5.1. For a complete proof the reader is referred to [9].*

Lemma 5.3. ***(Shape derivative of a surface objective function depending on the normal)****(Extension of Lemma* 3.24 *in [43]).*
Let Ω have a piece-wise smooth boundary Γ and let $\Omega(\epsilon)$ be the perturbed domain as defined in Eq. (5.1)*. For a surface objective shape function G defined as,*

$$G(\Omega(\epsilon)) = \int_{\Gamma(\epsilon)} (\boldsymbol{g}(\boldsymbol{z}(\epsilon),\epsilon)), \boldsymbol{n}(\epsilon)) dS, \tag{5.7}$$

where $\boldsymbol{g}$ is a vector valued function such that $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{n}}$ exists and where ϵ is the deformation parameter as defined in Eq. (5.2)*, the shape derivative is given by,*

$$dG(\Omega)[\boldsymbol{s}] = \int_\Gamma (\boldsymbol{s},\boldsymbol{n}) \Big[\frac{\partial \boldsymbol{g}(\boldsymbol{z})}{\partial \boldsymbol{n}} \boldsymbol{n} + div_\Gamma \boldsymbol{g}(\boldsymbol{z}) \Big] dS + \int_\Gamma (\boldsymbol{g}'[\boldsymbol{s}], \boldsymbol{n}) dx. \tag{5.8}$$

Here div_Γ is the tangential divergence as defined in Definition A.5 in the appendix and $g'[\boldsymbol{s}]$ is the local shape derivative from Definition 5.4.
Proof: *Extension of Lemma* 3.24 *in [43]. Follows from the proof of Lemma* 3.24 *in [43] which relies on the same principles that were described in Lemma 5.1. The extension in this result is the "local shape derivative term" which follows by using Lemma 5.2 (In this thesis) Instead of Lemma* 3.15 *(In [43].*

Remark 5.2. ***(The shape gradients in Lemma 5.2 and Lemma 5.3*** *). The shape gradient $\tilde{k}$ for a shape functional on the form defined in Lemma 5.2 is given as,*

$$\tilde{k} = \Big[\frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{n}} + \kappa g(\boldsymbol{z}) \Big].$$

This follows by comparing Lemma 5.2 with Corollary 5.1. Likewise, the shape gradient $\tilde{k}$ for a shape functional on the form defined in Lemma 5.3 is given as,

$$\tilde{k} = \Big[\frac{\partial \boldsymbol{g}(\boldsymbol{z})}{\partial \boldsymbol{n}} \boldsymbol{n} + div_\Gamma \boldsymbol{g}(\boldsymbol{z}) \Big],$$

which follows by comparing Lemma 5.3 with Corollary 5.1.

Similarly as for the volume objective function H, it follows from Remark 5.2 that to get the expressions for the shape derivative $dG(\Omega)[\boldsymbol{s}]$ from Lemma 5.2 and Lemma 5.3 on Hadamard form, it is necessary to eliminate the "local shape derivative terms". This will be the purpose of the adjoint approach that is explained in the next section. Corollary 5.2 summarises the results from this section by giving a general expression for the shape derivative of the class of shape functionals that will be considered in this thesis.

Corollary 5.2. ***(General form: Shape derivative of PDE constrained shape functionals)*** *Let Υ be either Ω or Γ. Furthermore let H be a shape functional on the form stated in Lemma 5.1 and Lemma 5.2 such that,*

$$H(\boldsymbol{z}(\epsilon),\Omega(\epsilon)) = \int_{\Upsilon(\epsilon)} h(\boldsymbol{z}(\epsilon),\epsilon)d\Upsilon$$

for $h : \Upsilon \longrightarrow \mathbb{R}$ where $\boldsymbol{z}$ is subject to a PDE. The expression for the Shape derivative is then,

$$dH(\Omega)[\boldsymbol{s}] = \underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}(h)dS}_{\textit{Hadamard form}} + \underbrace{\int_\Upsilon h'[\boldsymbol{s}]dx}_{\textit{Local shape deriv. term}} \quad ,$$

where $\boldsymbol{s}$ is the perturbation field and $\boldsymbol{n}$ is the surface normal on Γ.

For a shape functional on the form defined in Lemma 5.3, where h is replaced by the innerproduct $(\boldsymbol{h}(\boldsymbol{z},\epsilon),\boldsymbol{n}(\epsilon))$, a similar relation holds with the only difference that the "local shape derivative term" should be replaced by $\int_\Gamma(\boldsymbol{h}'[\boldsymbol{s}],\boldsymbol{n})dS$.

In the expression, $\tilde{k}$ is the Shape gradient specific for the functional that is considered and the local shape derivative is as defined in Definition 5.4.

Proof: *The proof is straight forward and follows by considering the shape derivatives in Lemmas 5.1- 5.3 along with Remarks 5.1-5.2.*

5.2.3 Adjoint approach

The adjoint approach is a well known method for computing the derivatives of objective functions, where the objective is subject to a set of constraint equations. A general presentation of the method can be found in [20]. In this section, the method will be explained in the context of the shape optimisation problem from Definition 5.2, for the specific class of shape functionals stated in Corollary 5.2. This is the relevant context for this work. Note that also boundary conditions can be added to the constraint part in Definition 5.2, but this is dropped for brevity of the discussion. Here the Lagrange based view of the adjoint approach is adapted. The first step in this regard is to form the Lagrangian,

$$\mathscr{L}(\boldsymbol{z},\Omega,\boldsymbol{\lambda}) = H(\boldsymbol{z},\Omega) + G(\boldsymbol{z},\Omega,\boldsymbol{\lambda}) = \int_\Omega h(\boldsymbol{z})dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{z},\Omega)dx \tag{5.9}$$

where $\boldsymbol{\lambda}(\Omega)$ is the Lagrange multiplier. Here $G(\boldsymbol{z},\Omega,\boldsymbol{\lambda})$ is introduced for convenience as,

$$G(\boldsymbol{z},\Omega,\boldsymbol{\lambda}) = \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{z},\Omega)dx \tag{5.10}$$

To relate the objective function and the Lagrangian the following lemma must be considered.

Lemma 5.4. *Let $\boldsymbol{z} := \boldsymbol{z}(\Omega)$ be the solution of the constraint equation, in a general optimisation problem as defined in Definition 5.2. Making this substitution for $\boldsymbol{z}$ in Eq. 5.9 gives the reduced functional and the relation between the objective function and the Lagrangian,*

$$H(\Omega) = H(\boldsymbol{z}(\Omega),\Omega) = \mathscr{L}(\boldsymbol{z}(\Omega),\Omega,\boldsymbol{\lambda}) \quad \textit{for arbitrary } \boldsymbol{\lambda} \tag{5.11}$$

Proof: *Since $\boldsymbol{z}(\Omega)$ is the solution to the constraint equation $\boldsymbol{g}(\boldsymbol{z},\Omega)$ from Definition 5.2, it follows that $g(\boldsymbol{z}(\Omega),\Omega) = 0$ by construction. Since the lagrange multiplier $\boldsymbol{\lambda}$ therefore is multiplied by zero, it follows that the above relation holds for any $\boldsymbol{\lambda}$.*

The shape derivative of the objective function H can be found by relating H to the Lagrangian using Lemma 5.4 and then taking the shape derivative of the Lagrangian. From Corollary 5.2 it then follows that the shape derivative of $H(\Omega)$ becomes,

$$\begin{aligned} dH(\Omega)[\boldsymbol{s}] &= d\mathscr{L}(\boldsymbol{z}(\Omega),\Omega,\lambda)[\boldsymbol{s}] \\ &= \underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} + \underbrace{\int_\Omega \frac{\partial h}{\partial \boldsymbol{z}}\boldsymbol{z}'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{\lambda}\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\boldsymbol{z}'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{g}\cdot\boldsymbol{\lambda}'[\boldsymbol{s}]dx}_{\text{Local shape derivative terms}}. \end{aligned} \tag{5.12}$$

For the "local shape derivative terms" in Eq. (5.12), the chain rule is used since $h = h(\boldsymbol{z}(\boldsymbol{x}))$ and $g = g(\boldsymbol{z}(\boldsymbol{x}))$. The essence of the adjoint method is to avoid calculating the local shape derivatives $\boldsymbol{z}'[\boldsymbol{s}]$ and $\boldsymbol{\lambda}'[\boldsymbol{s}]$. Since $\boldsymbol{g}(\boldsymbol{z}(\Omega),\Omega) = 0$ by construction, the term involving $\boldsymbol{\lambda}'[\boldsymbol{s}]$ is eliminated. The adjoint method now utilises that Eq. (5.11) holds for any $\boldsymbol{\lambda}$. The set of multipliers can therefore be chosen freely. With the particular choice $\boldsymbol{\lambda}$ such that,

$$\int_\Omega \Big(\frac{\partial h}{\partial \boldsymbol{z}}\boldsymbol{z}'[\boldsymbol{s}] + \int_\Omega \boldsymbol{\lambda}\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\boldsymbol{z}'[\boldsymbol{s}]\Big)dx = 0 \quad \forall z'[\boldsymbol{s}] \quad (\text{ Adjoint Eq.}), \tag{5.13}$$

all terms involving local shape derivatives are eliminated. Eq. (5.13) defines the adjoint equation. The remaining term in Eq. (5.12) is,

$$dH(\Omega)[\boldsymbol{s}] = \int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS. \tag{5.14}$$

Therefore, the shape derivative $dH(\Omega)[\boldsymbol{s}]$ has successfully been converted to its Hadamard form as defined in Corollary 5.1. For a specific problem, with an objective function on the form defined in Corollary 5.2, the Lemmas 5.1-5.3 can be used to identify the shape gradient $\tilde{k}$.

5.3 Shape derivative of the single-domain Stokes-drag problem

The Stokes-drag problem was stated in Definition 1.2. Before deriving the shape derivative of the Stokes-drag problem in a multi-domain setting, the shape derivative is first derived for a single-domain. Several results from this single-domain analysis can be used for the derivation of the full, multi-domain Stokes-drag problem in Section 5.4. The drag functional is restated here to make the discussion as compact as possible,

$$J(\boldsymbol{u},\Omega) = \int_\Omega j(\boldsymbol{u})dx = \int_\Omega \mu \sum_{i,j=1}^{2}\Big(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_j}\Big)^2 dx = \int_\Omega \mu\nabla\boldsymbol{u} : \nabla\boldsymbol{u}dx, \tag{5.15}$$

where $j(\boldsymbol{u})$ is introduced for convenience and $\boldsymbol{u}$ is the velocity field in the single-domain Stokes-problem. The notation in the last term in equation (5.15) will be adapted when it is convenient with a more compact expression.

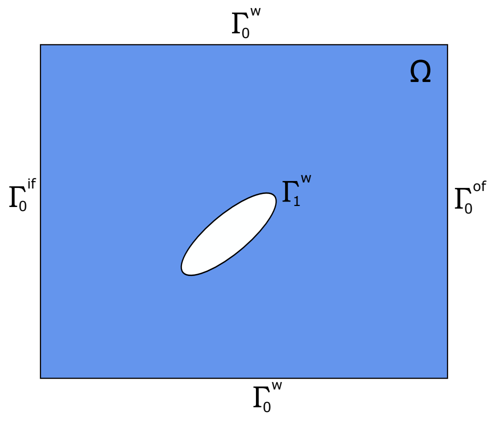

Figure 5.3: Illustration of the single-domain setup used in this derivation. The background domain Ω has three classes of boundaries an inflow boundary Γ_0^{if}, an outflow boundary Γ_0^{if} and two types of wall boundaries Γ_0^w and Γ_1^w. The wall boundary Γ_1^w corresponds to the boundary of a solid object in the domain that is represented by a hole in the computational domain. This object is indicated by the white hole. When taking the shape derivative, the white hole and the corresponding boundary is the only part of the domain Ω that is assumed to be perturbed.

The problem setup that will be used in this derivation is shown in Fig. 5.3. The shape derivative will be derived with respect to a perturbation of the white hole and the corresponding boundary Γ_1^w. The rest of the computational domain will be assumed fixed. The white hole is imagined to be a solid object in the Stokes flow and the Dirichlet boundary condition is therefore set to $\boldsymbol{u} = 0$ on the boundary of this region. The other boundaries corresponds to a wall boundary Γ_0^w, an inflow region Γ_0^{if} and an outflow region Γ_0^{of}.

5.3.1 Forming the Lagrangian of the problem

Let $\boldsymbol{g}$ be a vector containing all the constraints of the problem from Definition 1.1 and let $\boldsymbol{\lambda}$ be a vector containing the corresponding Lagrange multipliers. The Lagrangian with the drag functional $J(\boldsymbol{u},\Omega)$ and the constraints $\boldsymbol{g}$ can then be formed as,

$$\mathscr{L}(\boldsymbol{u},p,\Omega,\boldsymbol{\lambda}) = J(\boldsymbol{u},\Omega)+G(\boldsymbol{u},\Omega) = \int_\Omega j(\boldsymbol{u},\Omega)dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{u},p,\Omega)dx \tag{5.16}$$

where $G(\boldsymbol{u},\Omega) = \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{u},p,\Omega)$. The constraints contained in $\boldsymbol{g}$ are,

$$\boldsymbol{g}^s = -\mu\nabla^2\boldsymbol{u} + \nabla p = 0 \qquad \text{on } \Omega, \tag{5.17}$$

$$g^c = \nabla\cdot\boldsymbol{u} = 0 \qquad \text{on } \Omega, \tag{5.18}$$

$$\boldsymbol{g}^{if} = \boldsymbol{u} - \boldsymbol{c}^{if} = 0 \qquad \text{on } \Gamma_0^{if}, \tag{5.19}$$

$$\boldsymbol{g}^{of} = \mu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n} = 0 \qquad \text{on } \Gamma_0^{of}, \tag{5.20}$$

$$\boldsymbol{g}^{w0} = \boldsymbol{u} = 0 \qquad \text{on } \Gamma_0^w \tag{5.21}$$

$$\boldsymbol{g}^{w1} = \boldsymbol{u} = 0 \qquad \text{on } \Gamma_1^w. \tag{5.22}$$

Inserting the explicit expressions for the constraints $\boldsymbol{g}$ and the objective function J into Eq. (5.16) yields,

$$
\begin{aligned}
\mathscr{L}(\boldsymbol{u},p,\Omega,\boldsymbol{\lambda}) = & \underbrace{\int_\Omega \mu \sum_{ij} \Big(\frac{\partial u_i}{\partial x_j}\Big)^2 dx}_{\text{I}} \\
& + \underbrace{\int_\Omega \boldsymbol{\lambda}^s \cdot \Big(-\mu\nabla^2\boldsymbol{u} + \nabla p\Big) dx}_{\text{II}} + \underbrace{\int_\Omega \lambda^c \nabla\cdot\boldsymbol{u}^- dx}_{\text{III}} \\
& + \underbrace{\int_{\Gamma_0^{if}} \boldsymbol{\lambda}^{if}\cdot\Big(\boldsymbol{u}-\boldsymbol{c}^{if}\Big)dS}_{\text{IV}} + \underbrace{\int_{\Gamma_0^{of}} \boldsymbol{\lambda}^{of}\cdot\Big(\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n}\Big)dS}_{\text{V}} + \underbrace{\int_{\Gamma_0^{w}} \boldsymbol{\lambda}^{w0}\cdot\boldsymbol{u}dS}_{\text{VI}} \\
& + \underbrace{\int_{\Gamma_1^w} \boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}dS}_{\text{VII}}
\end{aligned} \tag{5.23}
$$

For simplification, the limits are dropped on all sums and the indices i,j are implicitly assumed to sum over all dimensions $i,j = 1,2$. Let $\boldsymbol{u} \coloneqq \boldsymbol{u}(\Omega)$ be the solution to the Stokes problem from Definition 1.1. The objective function can then be related to the Lagrangian by Lemma 5.4 such that, $J(\Omega) = \mathscr{L}(\boldsymbol{u}(\Omega),p,\Omega,\boldsymbol{\lambda})$. Taking the shape derivative of the Lagrangian in Eq. (5.23) and using Corollary 5.2 then gives,

$$
dJ(\Omega)[\boldsymbol{s}] = d\mathscr{L}(\Omega)[\boldsymbol{s}] = \underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} + \underbrace{\int j'[\boldsymbol{s}]dx + \int \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]d + \int \boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dS}_{\text{Local shape derivative terms}} \tag{5.24}
$$

Before stating the explicit form of the "Hadamard terms" in Eq. (5.24) the following simplification is made. It is assumed that the perturbation is only applied to the part of the boundary defined by $\Gamma_{\text{pert}} = \Gamma_1^w$. To implement this choice, $\boldsymbol{s}$ is chosen in such a way that $\big[(\boldsymbol{s}(\boldsymbol{x}) = 0 : \forall x \notin \Gamma_{\text{pert}}\big]$. With this assumption of the perturbation field the terms involving Γ_0^{if}, Γ_0^{of} and Γ_0^W are eliminated from the "Hadamard terms". Lemma 5.1 is used to identify the shape gradient $\tilde{k}$ in the terms (I), (II) and (III). This gives the terms (Sd), (S), (T) and (C) in Eq. (5.25). Lemma 5.2 does the same for (VII) which gives the (W) term in Eq. (5.25). The result is,

$$
\begin{aligned}
\underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} = & \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\mu\Big(\nabla\boldsymbol{u}:\nabla\boldsymbol{u}\Big)dS}_{\text{(Sd)}} \\
& + \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\big(-\mu\boldsymbol{\lambda}^s\cdot\nabla^2\boldsymbol{u}\big)dS}_{\text{S}} + \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\lambda^s\cdot\nabla p dS}_{\text{T}} \\
& + \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\lambda^c(\nabla\cdot\boldsymbol{u})dS}_{\text{C}} + \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial}{\partial\boldsymbol{n}}\big(\boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}\big) + \kappa\boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}\Big]dS}_{\text{W}}
\end{aligned} \tag{5.25}
$$

The "local shape derivative terms" in Eq. (5.24) are also written out on their explicit form. In order to facilitate for integration by parts and to easier derive the adjoint equation, the vector expressions are written on index notation. Using the chain rule as explained in Section 5.2.3 then gives,

$$
\begin{aligned}
&\underbrace{\int_\Omega j'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dx}_{\text{Local shape derivative terms}} = \underbrace{\int\limits_\Omega 2\mu\sum_{ij}\frac{\partial u_i}{\partial x_j}\frac{\partial}{\partial x_j}\left(u_i'[\boldsymbol{s}]\right)dx}_{\text{I}} \\
&+\underbrace{\int\limits_\Omega \left[-\mu\sum_{ij}\boldsymbol{\lambda}_i^s\frac{\partial^2}{\partial x_j^2}\left(u_i'[\boldsymbol{s}]\right)\right.}_{\text{II(a)}} + \underbrace{\left.\sum_i \lambda_i^s\frac{\partial}{\partial x_i}\left(p'[\boldsymbol{s}]\right)\right]dx}_{\text{II(b)}} + \underbrace{\int\limits_\Omega \lambda^c\sum_i\frac{\partial}{\partial x_i}\left(u_i'[\boldsymbol{s}]\right)dx}_{\text{III}} \\
(5.26)\qquad &+\underbrace{\int\limits_{\Gamma_0^{if}}\sum_i \lambda_i^{if}u_i'[\boldsymbol{s}]dS}_{\text{IV}} + \underbrace{\int\limits_{\Gamma_0^{of}}\sum_{ij}\left[\lambda_i^{of}\mu\frac{\partial}{x_j}\left(u_i^{-\prime}[\boldsymbol{s}]\right)n_j - \lambda_i^{of}n_ip'[\boldsymbol{s}]\right]dS}_{\text{V}} + \underbrace{\int\limits_{\Gamma_0^w}\sum_i \lambda_i^{w0}u_i'[\boldsymbol{s}]dS}_{\text{VI}} \\
&+\underbrace{\int\limits_{\Gamma_1^w}\sum_i \lambda_i^{w1}u_i'[\boldsymbol{s}]dS}_{\text{VII}} + \int\limits_\Omega \boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dx
\end{aligned}
$$

where the terms are labeled with the corresponding labels from Eq. (5.23). The terms involving $\boldsymbol{\lambda}'[\boldsymbol{s}]$ will from now on be skipped since these terms will only reproduce the constraints equations listed in Eq. (5.17)-(5.22).

5.3.2 Removing the local derivative terms: Adjoint equation on single-domain

Following the strategy of the adjoint approach, see Section 5.2.3, the terms involving $\boldsymbol{u}'[\boldsymbol{s}]$ and $p'[\boldsymbol{s}]$ or derivatives of these are removed by choosing $\boldsymbol{\lambda}$ in such a way that each of the linearly independent terms disappears. This process can be summarised in three steps,

1. Integrate by parts the terms involving $\int_\Omega$, these are marked (I) to (III)

2. Gather all linearly dependent terms, meaning all terms with the same integration domain and local derivative

3. For each of the domains choose the set of multipliers $\boldsymbol{\lambda}$ in such a way that the "local shape derivative terms" disappear.

Performing the first step, all volume objective integrals in Eq. (5.26) are integrated by parts to remove the derivatives on $\boldsymbol{u}'[\boldsymbol{s}]$ and $p'[\boldsymbol{s}]$ in the volume terms. Defining $\partial\Omega = \Gamma_0^{of}\cup\Gamma_0^{if}\cup\Gamma_0^{w}\cup\Gamma_1^{w}$

one has,

$$
\begin{aligned}
\text{I} \quad &= \int\limits_{\partial\Omega} 2\mu \sum_{ij} n_j \frac{\partial u_i}{\partial x_j} u'_i[\boldsymbol{s}] dS - \int\limits_{\Omega} 2\mu \sum_{ij} \frac{\partial^2 u_i}{\partial^2 x_j} u'_i[\boldsymbol{s}] dx, \\
\text{II(a)} \quad &= \int\limits_{\partial\Omega} -\mu \sum_{ij} \lambda^s_i \frac{\partial}{\partial x_j}\Big(u'_i[\boldsymbol{s}] n_j\Big) dS + \int\limits_{\partial\Omega} \sum_{ij} \mu n_j \frac{\partial \lambda^s_i}{\partial x_j}\Big(u'_i[\boldsymbol{s}]\Big) dS \\
&\quad - \int_{\Omega} \mu \sum_{ij} \frac{\partial^2 \lambda^s_i}{\partial x_j^2}\Big(u'_i[\boldsymbol{s}]\Big) dx \qquad , \\
\text{II(b)} \quad &= \int\limits_{\partial\Omega} \sum_i \lambda^s_i n_i \Big(p'[\boldsymbol{s}]\Big) dS - \int_{\Omega} \sum_i \frac{\partial \lambda^s_i}{\partial x_j}\Big(p'[\boldsymbol{s}]\Big) dx, \\
\text{III} \quad &= \int\limits_{\partial\Omega} \sum_i \lambda^c \Big(u'_i[\boldsymbol{s}]\Big) n^-_i dS - \int_{\Omega} \sum_i \frac{\partial \lambda^c}{\partial x_i}\Big(u'_i[\boldsymbol{s}]\Big) dx.
\end{aligned}
\tag{5.27}
$$

The next step is to sort all terms that are linearly dependent in Eq. (5.26) and (5.27) into separate groups and set each of these groups to zero. This gives for the volume integrals,

$$
\begin{aligned}
&\int\limits_{\Omega} \sum_i \left[-2\mu \sum_j \frac{\partial^2 u_i}{\partial x_j^2} - \mu \sum_j \frac{\partial^2 \lambda^s_i}{\partial x_j^2} - \frac{\partial \lambda^c}{\partial x_i}\Big(u'_i[\boldsymbol{s}]\Big)\right] dx = 0, \\
&\int\limits_{\Omega} \sum_i \frac{\partial \lambda^s_i}{\partial x_i}\Big(p'[\boldsymbol{s}]\Big) dx = 0.
\end{aligned}
\tag{5.28}
$$

The terms in Eq. (5.28) will result in the volume part of the adjoint equation. Conducting the same procedure for the boundary integrals will give the corresponding adjoint boundary conditions. For terms involving integration over Γ^w_1 one has,

$$
\begin{aligned}
&\int\limits_{\Gamma^w_1} \sum_i \left[\left(\lambda^{w+}_i + 2\mu \sum_j n_j \frac{\partial u^+_i}{\partial x_j} + \sum_j \mu n_j \frac{\partial \lambda^{s+}_i}{\partial x_j} + \lambda^{c+} n_i\right)\Big(u^{+\prime}_i[\boldsymbol{s}]\Big)\right] dS = 0, \\
&\int\limits_{\Gamma^w_1} -\mu \sum_{ij} \lambda^{s+}_i \frac{\partial}{\partial x_j}\Big(u^{+\prime}_i[\boldsymbol{s}]\Big) n_j dS = 0, \\
&\int\limits_{\Gamma^w_1} \sum_i \lambda^{s+}_i n_i \Big(p^{+\prime}[\boldsymbol{s}]\Big) dS = 0.
\end{aligned}
\tag{5.29}
$$

Similar equations are obtained for Γ^{of}_0, Γ^{if}_0 and Γ^w_0 with,

$$
\begin{aligned}
&\int\limits_{\Gamma^{of}_0} \sum_i \left[\left(\sum_j \mu n_j \frac{\partial \lambda^{s-}_i}{\partial x_j} + 2\mu \sum_j n_j \frac{\partial u^-_i}{\partial x_j} + \lambda^c n_i\right)\Big(u^{-\prime}[\boldsymbol{s}]\Big)\right] dS = 0, \\
&\int\limits_{\Gamma^{of}_0} -\mu \sum_{ij} \Big(\lambda^{s-}_i - \lambda^{of}_i\Big) \frac{\partial}{\partial x_j}\Big(u^{-\prime}_i[\boldsymbol{s}]\Big) n_j dS = 0, \\
&\int\limits_{\Gamma^{of}_0} \sum_i \Big(\lambda^{s-}_i n_i - \lambda^{of}_i n_i\Big)\Big(p^{-\prime}[\boldsymbol{s}]\Big) dS = 0,
\end{aligned}
\tag{5.30}
$$

$$\begin{aligned}
&\int_{\Gamma_0^w} \sum_i \left[\left(\lambda_i^{w0} + \sum_j \mu n_j \frac{\partial \lambda_i^{s-}}{\partial x_j} + 2\mu \sum_j n_j \frac{\partial u_i^-}{\partial x_j} + \lambda^c n_i\right)\left(u^{-\prime}[\boldsymbol{s}]\right)\right] dS = 0,\\
(5.31)\qquad &\int_{\Gamma_0^w} -\mu \sum_{ij} \lambda_i^{s-} \frac{\partial}{\partial x_j}\left(u_i^{-\prime}[\boldsymbol{s}]\right) n_j dS = 0,\\
&\int_{\Gamma_0^w} \sum_i \lambda_i^{s-} n_i \left(p^{-\prime}[\boldsymbol{s}]\right) dS = 0.
\end{aligned}$$

The terms involving Γ_0^{if} will be the same as those for Γ_0^w in Eq. (5.31). The multipliers $\boldsymbol{\lambda}$ are now chosen in such a way that Eq. (5.28)-(5.31) are satisfied. The resulting relations that the multipliers must satisfy on the boundary are,

$$(5.32)\qquad \Gamma_1^w : \begin{cases} \lambda_i^{w1} + 2\mu \dfrac{\partial u_i}{\partial \boldsymbol{n}} + \mu \dfrac{\partial \lambda_i^s}{\partial \boldsymbol{n}} + \lambda^c n_i = 0 \\ \lambda_i^s = 0 \\ \lambda_i^s n_i = 0 \end{cases}$$

$$(5.33)\qquad \Gamma_0^{of} : \begin{cases} \mu \dfrac{\partial \lambda_i^s}{\partial \boldsymbol{n}} + \lambda^c n_i = -2\mu \dfrac{\partial u_i}{\partial \boldsymbol{n}} \\ \lambda_i^{of} - \lambda_i^s = 0 \end{cases}, \quad \Gamma_0^{if} : \begin{cases} \lambda_i^{if} + \mu \dfrac{\partial \lambda_i^s}{\partial \boldsymbol{n}} + \lambda^c n_i + 2\mu \dfrac{\partial u_i}{\partial \boldsymbol{n}} = 0 \\ \lambda_i^s = 0 \\ \lambda_i^s n_i = 0 \end{cases}.$$

$$(5.34)\qquad \Gamma_0^w : \begin{cases} \lambda_i^{w0} + \mu \dfrac{\partial \lambda_i^s}{\partial \boldsymbol{n}} + \lambda^c n_i + 2\mu \dfrac{\partial u_i}{\partial \boldsymbol{n}} = 0 \\ \lambda_i^s = 0 \\ \lambda_i^s n_i = 0 \end{cases}$$

These relations must hold for each vector component, i.e for $i = 1,2$. By combining these relations and simplifying, the adjoint equation for the single-domain setup, can be summarised in the following Lemma:

Lemma 5.5. ***(Adjoint equation for the single-domain setup)****.*
The adjoint equation resulting from the Lagrangian formed in Eq. (5.16) *is,*

$$\left.\begin{aligned} -\mu\nabla^2 \boldsymbol{\lambda}^s - \nabla \lambda^c &= 2\mu \nabla^2 \boldsymbol{u} \\ \nabla \cdot \boldsymbol{\lambda}^s &= 0 \end{aligned}\right\} \quad on\ \Omega$$

with the boundary conditions,

$$\boldsymbol{\lambda}^s = 0 \quad on\ \Gamma_1^w, \qquad \boldsymbol{\lambda}^s = 0 \quad on\ \Gamma_0^{if}, \qquad \boldsymbol{\lambda}^s = 0 \quad on\ \Gamma_0^w$$

and

$$\mu \frac{\partial \boldsymbol{\lambda}^s}{\partial \boldsymbol{n}} + \lambda^c \boldsymbol{n} = -2\mu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \quad on\ \Gamma_0^{of}.$$

Proof: *The derivation is given above.*

In addition to the adjoint variables $\boldsymbol{\lambda}^s$ and λ^c one has the Lagrange multipliers $\boldsymbol{\lambda}^{if}$,$\boldsymbol{\lambda}^{of}$,$\boldsymbol{\lambda}^{w0}$ and $\boldsymbol{\lambda}^{w1}$. However, due to the way the perturbation field $\boldsymbol{s}$ was chosen, only $\boldsymbol{\lambda}^{w1}$ will contribute to the remaining analysis. This multiplier can be related to the adjoint variables from Eq. (5.32) as,

$$\Gamma_1^w : \quad \left\{ \boldsymbol{\lambda}^{w1} = -2\mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - \mu\frac{\partial \boldsymbol{\lambda}^s}{\partial \boldsymbol{n}} - \lambda^c \boldsymbol{n} \right. \quad \Bigg] \tag{5.35}$$

The similarities between the adjoint equation in Lemma 5.5 and the forward problem from Definition 1.1 can readily be seen. The difference is a sign change and the additional source terms $2\mu\nabla^2\boldsymbol{u}$ and $-2\mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ on the volume equation and the outflow boundary respectively. These source terms comes from the local shape derivative of the drag functional.

In a general shape optimisation problem it is necessary to numerically solve for the adjoint variables. For this particular system however, it is possible to attain an analytical solution by simple inspection.

Corollary 5.3. ***Solution to the adjoint equation on a single-domain***
The solution to the single-domain adjoint equation from Lemma 5.5 is

$$(\boldsymbol{\lambda}^s, \lambda^c) = (0, -2p)$$

where $(\boldsymbol{\lambda}^s, \lambda^c)$ are the adjoint variables and p is the solution to the Stokes-problem from Definition 1.1.

Proof: *This follows from direct substitution and utilising the relation between $\boldsymbol{u}$ and p given by the Stokes equation. For Ω one then has,*

$$\left.\begin{aligned} -\nabla(-2p) &= 2\mu\nabla^2\boldsymbol{u} \\ \nabla\cdot\boldsymbol{0} &= 0 \end{aligned}\right\} \quad on\ \Omega$$

which is satisfied due to the $\boldsymbol{g}^s$ constraint from Eq. (5.17). *The constraints on Γ_0^{if},Γ_0^{w0},Γ_1^{w1} are obviously satisfied. For Γ_0^{of} one has $\frac{\partial \boldsymbol{\lambda}^s}{\partial \boldsymbol{n}} = 0$ since $\lambda^s = 0$ everywhere and therefore,*

$$(-2p)\boldsymbol{n} = -2\mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \quad on\ \Gamma_0^{of},$$

which is satisfied by the $\boldsymbol{g}^{of}$ constraint from Eq. (5.20).

Corollary 5.3 has been verified experimentally and the result is presented in Section 9.1.1.

5.3.3 Hadamard terms single-domain

With the adjoint equation from Lemma 5.5, several simplifications can be made to the "Hadamard terms" in Eq. (5.25). First the W term is simplified. Using Relation (A.4) on has,

$$\begin{aligned}(\mathrm{W}) &= \int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial}{\partial \boldsymbol{n}}\big(\boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}\big)+\kappa\boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}\Big]dS \\ &= \int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial\boldsymbol{\lambda}^{w1}}{\partial \boldsymbol{n}}\cdot\boldsymbol{u}+\frac{\partial\boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\boldsymbol{\lambda}^{w1}+\kappa\boldsymbol{\lambda}^{w1}\cdot\boldsymbol{u}\Big]dS = \int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial\boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\boldsymbol{\lambda}^{w1}\Big]dS\end{aligned} \tag{5.36}$$

The last step in Eq. (5.36), used that the velocity constraint is $\boldsymbol{u}=0$ on Γ_1^w. From Corollary 5.3 one has that $\boldsymbol{\lambda}^s=0$. Since this is true in the entire domain Ω it follows that also $\frac{\partial\boldsymbol{\lambda}^s}{\partial\boldsymbol{n}}=0$. This means that the terms labeled S and T in Eq. (5.25) are eliminated. With this simplification along with the modification of W from Eq. (5.36), the "Hadamard terms" can be written as,

$$\begin{aligned}\underbrace{\int_\Gamma(\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} &= \underbrace{\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\mu\Big(\nabla\boldsymbol{u}:\nabla\boldsymbol{u}\Big)dS}_{\text{Shape deriv. J, (Sd)}} \\ &+\underbrace{\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\lambda^c\big(\nabla\cdot\boldsymbol{u}\big)dS}_{\text{C}}+\underbrace{\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\Big(-2\mu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}-\mu\frac{\partial\boldsymbol{\lambda}^s}{\partial\boldsymbol{n}}-\lambda^c\boldsymbol{n}\Big)\Big]dS}_{\text{W}}\end{aligned} \tag{5.37}$$

where $\boldsymbol{\lambda}^{w1}$ has been substituted for by its relation to the adjoint variables from Eq. (5.35). The following consideration is now made. Since $\boldsymbol{u}$ must satisfy the boundary condition $\boldsymbol{u}=0$ on Γ_1^w it follows that the tangential derivative is zero, $\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{\tau}}=0$. From Definition A.4 and by using Identity A.7, the double dot product of the gradient can be simplified to:

$$\nabla\boldsymbol{u}:\nabla\boldsymbol{u}=\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\boldsymbol{n}^T:\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\boldsymbol{n}^T=\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} \tag{5.38}$$

The same simplification can be made to the divergence of $\boldsymbol{u}$. This follows from Definition A.5 such that $\nabla\cdot\boldsymbol{u}=\mathrm{div}_\Gamma\boldsymbol{u}+\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\boldsymbol{n}=\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\boldsymbol{n}$. Using these relations Eq. (5.37) reduces to,

$$\begin{aligned}\underbrace{\int_\Gamma(\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} &= \mu\underbrace{\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}dS}_{\text{Shape deriv. J, (Sd)}}\underbrace{-2\mu\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}dS}_{\text{from W}} \\ &+\underbrace{\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\lambda^c\big(\nabla\cdot\boldsymbol{u}}_{\text{C}}\underbrace{-\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\boldsymbol{n}}_{\text{From W}}\big)dS\underbrace{-\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\Big[\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\Big(-\mu\frac{\partial\boldsymbol{\lambda}^s}{\partial\boldsymbol{n}}\Big)\Big]dS}_{\text{from W}} \\ &= -\mu\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n})\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}\cdot\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}dS.\end{aligned} \tag{5.39}$$

The "Hadamard terms" have now been simplified as much as possible. In addition, the "local shape derivative terms" were eliminated by choosing the Lagrange multipliers to satisfy the adjoint equation from Lemma 5.5 and the remaining conditions in Eq. (5.32)-(5.34). This result therefore concludes the derivation of the shape derivative. The result is summarised in Proposition 5.1.

Proposition 5.1. ***(Hadamard form of the shape derivative of the Stokes-drag problem).*** *Let the objective functional $J(\boldsymbol{u},\Omega)$ be the drag functional as stated in Lemma 1.1. Let the objective function be constrained by the Stokes-equations from Definition 1.1 of which $\boldsymbol{u}(\Omega)$ is a solution. Under a perturbation of the domain Ω on the form defined in Eq.* (5.1)-(5.2) *the shape derivative of J is given as,*

$$dJ(\Omega)[\boldsymbol{s}] = -\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n})\mu\Big(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}},\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big)dS,$$

where $\boldsymbol{s}$ is the perturbation vector field defined over Ω, such that $[\boldsymbol{s}(\boldsymbol{x}) = 0 : \forall x \notin \Gamma_1^w]$ and Γ_1^w is the perturbed boundary.

Proof: *The derivation is given in Section 5.3 with Eq.* (5.16) *through Eq.* (5.39).

5.4 Shape derivative of the multi-domain Stokes-drag problem

In this section the Hadamard form and the adjoint approach are used to derive the shape derivative of the drag functional $J(\boldsymbol{u},\Omega)$ from Lemma 1.1 in a multi-domain setting. This derivation corresponds to Objective 2 in this work as stated in Section 1.2. The derivation will be performed with the multi-domain Stokes-problem from Definition 3.9 as constraint equations. The drag functional was restated in Eq. (5.15) for the single-domain analysis. The notation defined there will also be adapted for this multi-domain analysis.

The multi-domain setting that is adapted for this derivation was presented in Chapter 3. In particular the "domain with hole" scheme [10] presented in Section 3.3 will be used. In this setting the computational domain Ω is partitioned, such that $\Omega = \Omega_0 \bigcup_{j=1}^{M} \Omega_j$ where Ω_0 is the background domain and M is the number of sub-domains Ω_j. These domains corresponds to the visible domains from Definition 3.8. The sub-domains are not allowed to intersect and therefore the derivation can be reduced to consider the partition $\Omega = \Omega_0 \bigcup \Omega_1$. The generalisation to M domains will then be made at the end of the derivation. Fig. 5.4 illustrates the setup where the background domain Ω_0 is colored blue and the sub-domain Ω_1 is colored pink. This figure should be compared to Fig. 5.3 for the single-domain derivation. The only difference from the single-domain analysis is the partitioning of the domain and the introduction of an interface between the sub-domain Ω_1 and the background domain Ω_0 denoted $\Gamma^{+/-}$. Henceforth all variables associated with the sub-domain will be denoted with a superscript $(+)$ and all variables associated with the background will be denoted with a superscript $(-)$.

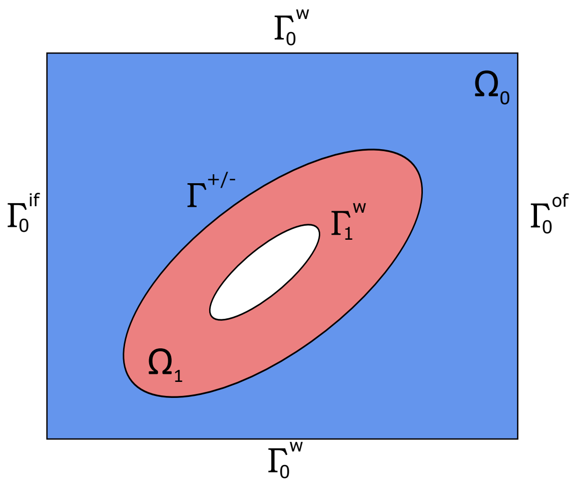

Figure 5.4: Schematic of the problem formulation. The setup only uses two domains. The background domain Ω_0 is represented by the blue region while the pink region corresponds to the sub-domain Ω_1. The background domain has four types of boundaries where Γ_0^{if} and Γ_0^{of} are the inflow and outflow boundaries and Γ_0^w corresponds to a wall. The last boundary is the interface $\Gamma^{+/-}$ which is shared with the sub-domain Ω_1. The sub-domain Ω_1 also has an inner boundary Γ_1^w. The perturbation of the domain will be restricted to the sub-domain Ω_1 and the white hole. This figure should be compared to Fig. 5.3 from the single-domain analysis. The only difference is the interface $\Gamma^{+/-}$ and the partitioning of the domain.

With this geometry and partition of the domain, the constraint equations can be explicitly stated in the following way. On the interface $\Gamma^{+/-}$ one has,

$$\boldsymbol{g}^d = [\![\boldsymbol{u}]\!] = \boldsymbol{u}^- - \boldsymbol{u}^+ = 0 \qquad \text{on } \Gamma^{+/-}, \tag{5.40}$$

$$\boldsymbol{g}^n = [[\mu D_n \boldsymbol{u} - p\boldsymbol{n}]] = \mu(D_n \boldsymbol{u}^+ - D_n \boldsymbol{u}^-) - (p^+ \boldsymbol{n} - p^- \boldsymbol{n}) = 0 \qquad \text{on } \Gamma^{+/-}, \tag{5.41}$$

where $[\![z]\!]$ denotes the jump in z across the interface. In fact $\boldsymbol{g}^n$ can also be written $\boldsymbol{g}^n = [[D_n \boldsymbol{u}]] - [[p\boldsymbol{n}]]$ which will be frequently used in the text. It is also important to remember the definition $D_n \boldsymbol{u} = (\nabla \boldsymbol{u}^T)^T \cdot \boldsymbol{n} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ where D is the Jacobian and D_n is the normal component of the Jacobian. For the background domain one has,

$$\boldsymbol{g}^{s-} = -\mu \nabla^2 \boldsymbol{u}^- + \nabla p^- = 0 \qquad \text{on } \Omega_0, \tag{5.42}$$

$$g^{c-} = \nabla \cdot \boldsymbol{u}^- = 0 \qquad \text{on } \Omega_0, \tag{5.43}$$

$$\boldsymbol{g}^{if} = \boldsymbol{u}^- - \boldsymbol{c}^{if} = 0 \qquad \text{on } \Gamma_0^{if}, \tag{5.44}$$

$$\boldsymbol{g}^{of} = \mu \frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}} - p\boldsymbol{n} = 0 \qquad \text{on } \Gamma_0^{of}, \tag{5.45}$$

$$\boldsymbol{g}^{w-} = \boldsymbol{u}^- = 0 \qquad \text{on } \Gamma_0^w. \tag{5.46}$$

For the sub-domain one has a similar set of constraints, namely,

$$\boldsymbol{g}^{s+} = -\mu\nabla^2\boldsymbol{u}^+ + \nabla p^+ = 0 \qquad \text{on } \Omega_1, \tag{5.47}$$

$$g^{c+} = \nabla\cdot\boldsymbol{u}^+ = 0 \qquad \text{on } \Omega_1, \tag{5.48}$$

$$\boldsymbol{g}^{w+} = \boldsymbol{u}^+ = 0 \qquad \text{on } \Gamma_1^w. \tag{5.49}$$

5.4.1 Forming the Lagrangian of the multi-domain problem

Let $\boldsymbol{g}$ be a vector containing all the constraints of the problem listed in Eq. (5.40)-(5.49) and let $\boldsymbol{\lambda}$ be a vector containing the corresponding Lagrange multipliers. The Lagrangian with the drag functional $J(\boldsymbol{u},\Omega)$ and the constraints contained in $\boldsymbol{g}$ can then be formed as,

$$\mathscr{L}(\boldsymbol{u},p,\Omega,\boldsymbol{\lambda}) = J(\boldsymbol{u},\Omega) + G(\boldsymbol{u},\Omega,\boldsymbol{\lambda}) = \int_\Omega j(\boldsymbol{u},\Omega)dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{u},p,\Omega)dx \tag{5.50}$$

where $G(\boldsymbol{u},\Omega,\boldsymbol{\lambda}) = \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{u},p,\Omega)$. By the argument of Lemma 5.4 the objective function $J(\Omega,\boldsymbol{u})$ can be related to the Lagrangian such that,

$$J(\Omega) = J(\Omega,\boldsymbol{u}(\Omega)) = \mathscr{L}(\boldsymbol{u}(\Omega),p,\Omega,\boldsymbol{\lambda}) \quad \text{for arbitrary } \boldsymbol{\lambda}, \tag{5.51}$$

where $\boldsymbol{u} := \boldsymbol{u}(\Omega)$ is the solution to the boundary value problem defined by the constraint equations in $\boldsymbol{g}$. Following the same procedure as before, the constraints in Eq. (5.40)-(5.49) are inserted into Eq. (5.50). As before the limits will be dropped on all sums so that the indices i,j are implicitly assumed to sum over all dimensions $i,j = 1,2$. Expanding Eq. (5.50) yields,

$$\begin{aligned}
\mathscr{L}(\boldsymbol{u},p,\Omega,\boldsymbol{\lambda}) = &\underbrace{\int_{\Omega_0}\mu\sum_{ij}\Big(\frac{\partial u_i^-}{\partial x_j}\Big)^2 dx}_{\text{I}^-} + \underbrace{\int_{\Omega_1}\mu\sum_{ij}\Big(\frac{\partial u_i^+}{\partial x_j}\Big)^2 dx}_{\text{I}^+} \\
&+\underbrace{\int_{\Omega_0}\boldsymbol{\lambda}^{s-}\cdot\Big(-\mu\nabla^2\boldsymbol{u}^- + \nabla p^-\Big)dx}_{\text{II}^-} + \underbrace{\int_{\Omega_0}\lambda^{c-}\nabla\cdot\boldsymbol{u}^- dx}_{\text{III}^-} \\
&+\underbrace{\int_{\Omega_1}\boldsymbol{\lambda}^{s+}\cdot\Big(-\mu\nabla^2\boldsymbol{u}^+ + \nabla p^+\Big)dx}_{\text{II}^+} + \underbrace{\int_{\Omega_1}\lambda^{c+}\nabla\cdot\boldsymbol{u}^+ dx}_{\text{III}^+} \\
&+\underbrace{\int_{\Gamma_0^{if}}\boldsymbol{\lambda}^{if}\cdot\Big(\boldsymbol{u}^- - \boldsymbol{c}^{if}\Big)dS}_{\text{IV}} + \underbrace{\int_{\Gamma_0^{of}}\boldsymbol{\lambda}^{of}\cdot\Big(\frac{\partial\boldsymbol{u}^-}{\partial\boldsymbol{n}} - p^-\boldsymbol{n}^-\Big)dS}_{\text{V}} + \underbrace{\int_{\Gamma_0^w}\boldsymbol{\lambda}^{w-}\cdot\boldsymbol{u}^- dS}_{\text{VI}} \\
&+\underbrace{\int_{\Gamma_1^w}\boldsymbol{\lambda}^{w+}\cdot\boldsymbol{u}^+ dS}_{\text{VII}} + \underbrace{\int_{\Gamma^{+/-}}\boldsymbol{\lambda}^d\cdot[\![\boldsymbol{u}]\!]dS}_{\text{VIII}} + \underbrace{\int_{\Gamma^{+/-}}\mu\boldsymbol{\lambda}^n\cdot[\![D_n\boldsymbol{u}]\!]dS}_{\text{IX}} - \underbrace{\int_{\Gamma^{+/-}}\boldsymbol{\lambda}^n\cdot\boldsymbol{n}[\![p]\!]dS}_{\text{X}}
\end{aligned} \tag{5.52}$$

The Lagrangian uses the same multipliers as in the single-domains analysis, in addition to λ^d and λ^n to enforce the interface conditions. Taking the shape derivative of the Lagrangian in Eq.

(5.52) and using Corollary 5.2 then gives as before,

$$dJ(\Omega)[\boldsymbol{s}] = d\mathscr{L}(\Omega)[\boldsymbol{s}] = \underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} + \underbrace{\int j'[\boldsymbol{s}]dx + \int \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]d + \int \boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dS}_{\text{Local shape derivative terms}} \tag{5.53}$$

Before stating the explicit form of Eq. (5.53) the following simplification is made. It is assumed that the perturbation is only applied to the top mesh, that is the part of the boundary defined by $\Gamma_{\text{pert}} = \Gamma_1^w \bigcup \Gamma^{+/-}$. To implement this choice, $\boldsymbol{s}$ is chosen in such a way that $\left[(\boldsymbol{s}(\boldsymbol{x}) = 0 : \forall x \notin \Gamma_{\text{pert}}\right]$. With this assumption of the perturbation field the terms involving Γ_0^{if}, Γ_0^{of} and Γ_0^W are eliminated from the "Hadamard terms". In order to easier utilise the results from the single-domain analysis in Section 5.3, the "Hadamard terms" are split in two parts, $A1$ and $A2$ such that,

$$\underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} = A1 + A2. \tag{5.54}$$

The $A1$ part corresponds to terms that have the same structure as those found in the single-domain analysis. Note that some of the $A1$ terms also contain integration over the interface $\Gamma^{+/-}$. The $A2$ terms corresponds to the interface conditions from Eq. (5.40)-(5.41) and are the only terms that are conceptually new. Lemma 5.1 is now used to identify the shape gradient $\tilde{k}$ in the terms labeled (I), (II) and (III) in Eq. (5.52), both for (+) and (−). This gives the (Sd), (S), (T) and (C) terms in (5.55). Likewise, Lemma 5.2 and the (VII) term in Eq. (5.52) gives the last term in Eq. (5.55), namely (W1). From this procedure the following expression is obtained for A1,

$$\begin{aligned} A1 = & \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)\mu\left(\nabla\boldsymbol{u}^- : \nabla\boldsymbol{u}^-\right)dS}_{\text{(Sd0)}} + \underbrace{\int_{\Gamma_1^w \bigcup \Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\mu\left(\nabla\boldsymbol{u}^+ : \nabla\boldsymbol{u}^+\right)dS}_{\text{(Sd1), (Sd2)}} \\ & + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)(-\mu\boldsymbol{\lambda}^{s-}\cdot\nabla^2\boldsymbol{u}^-)dS}_{\text{S0}} + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)\boldsymbol{\lambda}^{s-}\cdot\nabla p^- dS}_{\text{T0}} \\ & + \underbrace{\int_{\Gamma^{w+}\bigcup\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)(-\mu\boldsymbol{\lambda}^{s+}\cdot\nabla^2\boldsymbol{u}^+)dS}_{\text{S1 and S2}} + \underbrace{\int_{\Gamma^{w+}\bigcup\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\lambda^{s+}\cdot\nabla p^+ dS}_{\text{T1 and T2}} \\ & + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)\lambda^{c-}(\nabla\cdot\boldsymbol{u}^-)dS}_{\text{C0}} + \underbrace{\int_{\Gamma^{w+}\bigcup\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\lambda^{c+}(\nabla\cdot\boldsymbol{u}^+)dS}_{\text{C1 and C2}} \\ & + \underbrace{\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n}^+)\left[\frac{\partial}{\partial\boldsymbol{n}^+}(\boldsymbol{\lambda}^{w+}\cdot\boldsymbol{u}^+) + \kappa\boldsymbol{\lambda}^{w+}\cdot\boldsymbol{u}^+\right]dS}_{\text{W1}}. \end{aligned} \tag{5.55}$$

A similar treatment is performed for the $A2$ terms. From Lemma 5.2 the shape gradient $\tilde{k}$ can be identified for the (VIII) term. This gives the (U1) term in Eq. (5.56). Similarly, Lemma 5.3 is used for the (IX) and (X) terms to give (U2) and (P1) in Eq. (5.56). The expression for A2 becomes,

(5.56)
$$A2 = \underbrace{\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial}{\partial \boldsymbol{n}^+}(\boldsymbol{\lambda}^d\cdot[\![\boldsymbol{u}]\!]) + \kappa\boldsymbol{\lambda}^d\cdot[\![\boldsymbol{u}]\!]\Big]dS}_{\text{U1}} - \underbrace{\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial}{\partial \boldsymbol{n}}(\boldsymbol{\lambda}^n[\![p]\!])\boldsymbol{n} + \mathrm{div}_\Gamma(\boldsymbol{\lambda}^n[\![p]\!])\Big]dS}_{\text{P1}}$$
$$+\underbrace{\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial}{\partial \boldsymbol{n}^+}\Big(\boldsymbol{\lambda^n}\cdot[\![D\boldsymbol{u}]\!]\Big)\boldsymbol{n}^+ + \mathrm{div}_\Gamma\Big(\boldsymbol{\lambda^n}\cdot[\![D\boldsymbol{u}]\!]\Big)\Big]dS}_{\text{U2}}.$$

In the interface terms in $A2$ the convention for the surface normal $\boldsymbol{n}$ is chosen as $\boldsymbol{n}^+$. For the terms in $A1$ the convention is $\boldsymbol{n}^-$ on all terms derived from Ω_0 and $\boldsymbol{n}^+$ for terms derived from Ω_1.

The "local shape derivative terms" in Eq. (5.53) are also split into two groups such that,

$$\int_\Omega j'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]dx = \Big(\int_\Omega j'[\boldsymbol{s}]dx + \int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]dx\Big)_{B1} + \Big(\int_\Omega \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]dx\Big)_{B2} = B1 + B2. \tag{5.57}$$

The first group labeled $B1$ corresponds to the single-domain local shape derivatives. The only difference is that the integration over the domain is now split into integration over the background domain Ω_0 and the sub-domain Ω_1. These terms will lead to the same adjoint system as stated in Lemma 5.5, with the difference that the adjoint equations are split into a system for the background and a similar system for the sub-domain. The second group is labeled $B2$ and corresponds to the interface conditions on $\Gamma^{+/-}$. These terms will result in the interface conditions for the adjoint equation. To facilitate for integration by parts the vector expressions are also here written on index notation. Using the chain rule, as explained in Section 5.2.3, gives the following expressions,

$$\begin{aligned}
B1 =& \underbrace{\int\limits_{\Omega_0} 2\mu\sum_{ij}\frac{\partial u_i^-}{\partial x_j}\frac{\partial}{\partial x_j}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big)dx}_{\text{I}^-} + \underbrace{\int\limits_{\Omega_1} 2\mu\sum_{ij}\frac{\partial u_i^+}{\partial x_j}\frac{\partial}{\partial x_j}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big)dx}_{\text{I}^+}\\
&+\int\limits_{\Omega_0}\Big[\underbrace{-\mu\sum_{ij}\boldsymbol{\lambda}_i^{s-}\frac{\partial^2}{\partial x_j^2}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big)}_{\text{II}^-\text{(a)}} + \underbrace{\sum_i\lambda_i^{s-}\frac{\partial}{\partial x_i}\Big(p^{-\prime}[\boldsymbol{s}]\Big)\Big]dx}_{\text{II}^-\text{(b)}} + \underbrace{\int\limits_{\Omega_0}\lambda^{c-}\sum_i\frac{\partial}{\partial x_i}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big)dx}_{\text{III}^-}\\
&+\int\limits_{\Omega_1}\Big[\underbrace{-\mu\sum_{ij}\lambda_i^{s+}\frac{\partial^2}{\partial x_j^2}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big)}_{\text{II}^+\text{(a)}} + \underbrace{\sum_i\boldsymbol{\lambda}_i^{s+}\frac{\partial}{\partial x_i}\Big(p^{+\prime}[\boldsymbol{s}]\Big)\Big]dx}_{\text{II}^+\text{(b)}} + \underbrace{\int\limits_{\Omega_1}\lambda^{c+}\sum_i\frac{\partial}{\partial x_i}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big)dx}_{\text{III}^+}\\
&+\underbrace{\int\limits_{\Gamma_0^{if}}\sum_i\lambda_i^{if}u_i^{-\prime}[\boldsymbol{s}]dS}_{\text{IV}} + \underbrace{\int\limits_{\Gamma_0^{of}}\sum_{ij}\Big[\lambda_i^{of}\mu\frac{\partial}{x_j}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big)n_j - \lambda_i^{of}n_ip^{-\prime}[\boldsymbol{s}]\Big]dS}_{\text{V}} + \underbrace{\int\limits_{\Gamma_0^w}\sum_i\lambda_i^w u_i^{-\prime}[\boldsymbol{s}]dS}_{\text{VI}}\\
&+\underbrace{\int\limits_{\Gamma_1^w}\sum_i\lambda_i^{w+}u_i^{+\prime}[\boldsymbol{s}]dS}_{\text{VII}} + \int\limits_\Omega\boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dx
\end{aligned} \tag{5.58}$$

and

(5.59)
$$
\begin{aligned}
B2 = &\underbrace{\int_{\Gamma^{+/-}} \sum_i \lambda_i^d \Big(u_i^{+\prime}[\boldsymbol{s}] - u_i^{-\prime}[\boldsymbol{s}]\Big) dS}_{\text{VIII}} \\
&+ \underbrace{\int_{\Gamma^{+/-}} \mu \sum_{ij} \lambda_i^n \Big[\frac{\partial}{\partial x_j}\Big(u_i^{+\prime}[\boldsymbol{s}] n_j\Big) - \frac{\partial}{\partial x_j}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big) n_j \Big] dS}_{\text{IX}} - \underbrace{\int_{\Gamma^{+/-}} \sum_i \lambda_i^n n_i \Big(p^{+\prime}[\boldsymbol{s}] - p^{-\prime}[\boldsymbol{s}]\Big) dS}_{\text{X}}
\end{aligned}
$$

where the terms are labeled with the corresponding labels from Eq. (5.52). The terms involving $\boldsymbol{\lambda}'[\boldsymbol{s}]$ are again skipped since these terms will only reproduce the constraints listed in Eq. (5.40)-(5.49).

5.4.2 Removing the local shape derivative terms: Adjoint equation for the multi-domain setup

To remove the "local shape derivative terms" and derive the adjoint equations, it is sufficient to consider the terms that involve integration over the interfaces $\Gamma^{+/-}$ since the contribution from the domain integrals and the boundaries $\Gamma_1^w, \Gamma_0^{if}, \Gamma_0^{of}$ and Γ_0^w were already derived for the single-domain case. In this regard one must be careful since the domain integrals will give contributions also on the interfaces. To get these contributions, all volume objective integrals in Eq. (5.58) are integrated by parts to remove the derivatives on $\boldsymbol{u}'[\boldsymbol{s}]$ and $p'[\boldsymbol{s}]$. Starting with the Ω_0 integrals and defining $\partial\Omega_0 = \Gamma_0^{of} \cup \Gamma_0^{if} \cup \Gamma_0^w \cup \Gamma^{+/-}$ one has,

(5.60)
$$
\begin{aligned}
\text{I}^- &= \int_{\partial\Omega_0} 2\mu \sum_{ij} n_j^- \frac{\partial u_i^-}{\partial x_j} u_i^{-\prime}[\boldsymbol{s}] dS - \int_{\Omega_0} 2\mu \sum_{ij} \frac{\partial^2 u_i^-}{\partial^2 x_j} u_i^{-\prime}[\boldsymbol{s}] dx, \\
\text{II}^-\text{(a)} &= \int_{\partial\Omega_0} -\mu \sum_{ij} \lambda_i^{s-} \frac{\partial}{\partial x_j}\Big(u_i^{-\prime}[\boldsymbol{s}] n_j^-\Big) dS + \int_{\partial\Omega_0} \sum_{ij} \mu n_j^- \frac{\partial \lambda_i^{s-}}{\partial x_j}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big) dS - \int_{\Omega_0} \mu \sum_{ij} \frac{\partial^2 \lambda_i^{s-}}{\partial x_j^2}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big) dx, \\
\text{II}^-\text{(b)} &= \int_{\partial\Omega_0} \sum_i \lambda_i^{s-} n_i^- \Big(p^{-\prime}[\boldsymbol{s}]\Big) dS - \int_{\Omega_0} \sum_i \frac{\partial \lambda_i^{s-}}{\partial x_j}\Big(p^{-\prime}[\boldsymbol{s}]\Big) dx, \\
\text{III}^- &= \int_{\partial\Omega_0} \sum_i \lambda^{c-} \Big(u_i^{-\prime}[\boldsymbol{s}]\Big) n_i^- dS - \int_{\Omega_0} \sum_i \frac{\partial \lambda^{c-}}{\partial x_i}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big) dx.
\end{aligned}
$$

Doing the same for terms involving integration with respect to the turbine domain Ω_1 gives the same equations with $(-)$ replaced with $(+)$. The only difference is that the integration now runs over Γ_1^w and $\Gamma^{+/-}$. The resulting equations are listed bellow,

(5.61)

$$\begin{aligned}
\mathrm{I}^+ &= \int_{\Gamma_1^w \cup \Gamma^{+/-}} 2\mu \sum_{ij} n_j^+ \frac{\partial u_i^+}{\partial x_j} u_i^{+\prime}[\boldsymbol{s}] dS - \int_{\Omega_1} 2\mu \sum_{ij} \frac{\partial^2 u_i^+}{\partial^2 x_j} u_i^{+\prime}[\boldsymbol{s}] dx,\\
\mathrm{II}^+(\mathrm{a}) &= \int_{\Gamma_1^w \cup \Gamma^{+/-}} -\mu \sum_{ij} \lambda_i^{s+} \frac{\partial}{\partial x_j}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) n_j^+ dS + \int_{\Gamma^{+/-}} \sum_{ij} \mu n_j^+ \frac{\partial \lambda_i^{s+}}{\partial x_j}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) dS - \int_{\Omega_1} \mu \sum_{ij} \frac{\partial^2 \lambda_i^{s+}}{\partial x_j^2}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) dx,\\
\mathrm{II}^+(\mathrm{b}) &= \int_{\Gamma_1^w \cup \Gamma^{+/-}} \sum_i \lambda_i^{s+} n_i^+ \Big(p^{+\prime}[\boldsymbol{s}]\Big) dS - \int_{\Omega_1} \sum_i \frac{\partial \lambda_i^{s+}}{\partial x_j}\Big(p^{+\prime}[\boldsymbol{s}]\Big) dx,\\
\mathrm{III}^+ &= \int_{\Gamma_1^w \cup \Gamma^{+/-}} \sum_i \lambda^{c+}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) n_i^+ dS - \int_{\Omega_1} \sum_i \frac{\partial \lambda^{c+}}{\partial x_i}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) dx.
\end{aligned}$$

The next step is to group all linearly dependent terms with integration over the interface $\Gamma^{+/-}$. Considering the terms in Eq. (5.60) and (5.61) along with Eq. (5.59) one gets,

$$\begin{aligned}
&\int_{\Gamma^{+/-}} \sum_i \Bigg[\Bigg(-\lambda_i^d + \sum_j \mu(-n_j)\frac{\partial \lambda_i^{s-}}{\partial x_j} + 2\mu \sum_j (-n_j)\frac{\partial u_i^-}{\partial x_j} + \lambda^c(-n_i)\Bigg)\Big(u^{-\prime}[\boldsymbol{s}]\Big)\Bigg] dS = 0,\\
&\int_{\Gamma^{+/-}} \Bigg[-\mu \sum_{ij} \lambda_i^{s-} \frac{\partial}{\partial x_j}\Big(u_i^{-\prime}[\boldsymbol{s}]\Big)(-n_j) + \mu \sum_{ij} \lambda_i^n \Bigg(-\frac{\partial}{\partial x_j}\Big(u^{-\prime}[\boldsymbol{s}]\Big) n_j\Bigg)\Bigg] dS = 0, \qquad (5.62)\\
&\int_{\Gamma^{+/-}} \Bigg[\Bigg(\sum_i \lambda_i^{s-}(-n_i) + \lambda_i^n n_i\Bigg)\Big(p^{-\prime}[\boldsymbol{s}]\Big)\Bigg] dS = 0,
\end{aligned}$$

and

$$\begin{aligned}
&\int_{\Gamma^{+/-}} \sum_i \Bigg[\Bigg(\lambda_i^d + \sum_j \mu n_j \frac{\partial \lambda_i^{s+}}{\partial x_j} + 2\mu \sum_j n_j \frac{\partial u_i^+}{\partial x_j} + \lambda^c n_i\Bigg)\Big(u^{+\prime}[\boldsymbol{s}]\Big)\Bigg] dS = 0,\\
&\int_{\Gamma^{+/-}} \Bigg[-\mu \sum_{ij} \lambda_i^{s+} \frac{\partial}{\partial x_j}\Big(u_i^{+\prime}[\boldsymbol{s}]\Big) n_j + \mu \sum_{ij} \lambda_i^n \frac{\partial}{\partial x_j}\Big(u^{+\prime}[\boldsymbol{s}]\Big) n_j\Bigg] dS = 0, \qquad (5.63)\\
&\int_{\Gamma^{+/-}} \Bigg[\Bigg(\sum_i \lambda_i^{s+} n_i - \lambda_i^n n_i\Bigg)\Big(p^{+\prime}[\boldsymbol{s}]\Big)\Bigg] dS = 0.
\end{aligned}$$

The resulting relations that the multipliers must satisfy at the interface are listed below,

$$\Gamma^{+/-}: \quad \left\{\begin{aligned}
&-\lambda_i^d - \mu \frac{\partial \lambda_i^{s-}}{\partial \boldsymbol{n}} - 2\mu \frac{\partial u_i^-}{\partial \boldsymbol{n}} - \lambda^{c-} n_i = 0\\
&\lambda_i^d + \mu \frac{\partial \lambda_i^{s+}}{\partial \boldsymbol{n}} + 2\mu \frac{\partial u_i^+}{\partial \boldsymbol{n}} + \lambda^{c+} n_i = 0\\
&-\lambda_i^{s-} + \lambda_i^n = 0\\
&\lambda_i^{s+} - \lambda_i^n = 0
\end{aligned}\right] . \qquad (5.64)$$

These relations must hold for each vector component, i.e for $i = 1,2$. By simplifying these relations the interface constraints on the adjoint variables λ^{s+}, λ^{s-}, λ^{c+} and λ^{c-} are obtained. In fact, by adding the two equations containing $\boldsymbol{\lambda}^d$ one gets that $\left[\left[\frac{\partial \boldsymbol{\lambda}^s}{\partial \boldsymbol{n}} + \lambda^c \boldsymbol{n}\right]\right] = -2\mu\left[\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\right]\right]$ on $\Gamma^{+/-}$. This has the same structure as the interface condition on $D_n \boldsymbol{u}$ and p with an additional source term. From the adjoint equations in Lemma 5.5 the boundary conditions on Γ_0^{of}, Γ_0^{if} and Γ_0^w can readily be adapted. Similarly for the boundary condition on Γ_1^w. The adjoint equations on Ω from Lemma 5.5 are now split in two systems, one system for the adjoint variables λ^{s-}, λ^{c-} defined over Ω_0 and a similar system for λ^{s+}, λ^{c+} defined over Ω_1. From these considerations the adjoint equation for the multi-domain setup is summarised in Lemma 5.6,

Lemma 5.6. ***(Adjoint equation for the multi-domain setup)****.*
The adjoint equation resulting from the Lagrangian formed in Eq. (5.50) *is,*

$$\left.\begin{aligned} -\mu\nabla^2\boldsymbol{\lambda}^{s-} - \nabla\lambda^{c-} &= 2\mu\nabla^2\boldsymbol{u}^- \\ \nabla\cdot\boldsymbol{\lambda}^{s-} &= 0 \end{aligned}\right\} : \quad \Omega_0$$

$$\left.\begin{aligned} -\mu\nabla^2\boldsymbol{\lambda}^{s+} - \nabla\lambda^{c+} &= 2\mu\nabla^2\boldsymbol{u}^+ \\ \nabla\cdot\boldsymbol{\lambda}^{s+} &= 0 \end{aligned}\right\} : \quad \Omega_1$$

with the interface conditions,

$$\left.\begin{aligned} \boldsymbol{\lambda}^{s+} - \boldsymbol{\lambda}^{s-} &= 0 \\ \left[\left[\frac{\partial \boldsymbol{\lambda}^s}{\partial \boldsymbol{n}} + \lambda^c \boldsymbol{n}\right]\right] &= -2\mu\left[\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\right]\right] \end{aligned}\right\} \quad \text{on } \Gamma^{+/-}$$

and boundary conditions,

$$\boldsymbol{\lambda}^{s+} = 0 \text{ on } \Gamma_1^w, \qquad \boldsymbol{\lambda}^{s-} = 0 \text{ on } \Gamma_0^{if}, \qquad \boldsymbol{\lambda}^{s-} = 0 \text{ on } \Gamma_0^w$$

and

$$\mu\frac{\partial \boldsymbol{\lambda}^{s-}}{\partial \boldsymbol{n}} + \lambda^{c-}\boldsymbol{n} = -2\mu\frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}} \text{ on } \Gamma_0^{of}.$$

Proof: *The derivation is given above.*

The remaining multipliers needed to simplify the "Hadamard terms" can be related to the adjoint variables by the following expressions,

$$(5.65) \quad \Gamma_1^w : \quad \left\{ \lambda_i^{w+} = -2\mu\frac{\partial u_i^+}{\partial \boldsymbol{n}} - \mu\frac{\partial \lambda_i^{s+}}{\partial \boldsymbol{n}} - \lambda^{c+} n_i \right] \quad \Gamma^{+/-} : \quad \left\{ \begin{aligned} \lambda_i^d &= -\mu\frac{\partial \lambda^{s+}}{\partial n} - 2\mu\frac{\partial u_i^+}{\partial n} - \lambda^{c-} n_i \\ \lambda_i^d &= -\mu\frac{\partial \lambda^{s-}}{\partial n} - 2\mu\frac{\partial u_i^-}{\partial n} - \lambda^{c+} n_i \\ \lambda_i^n &= \lambda_i^{s-} \\ \lambda_i^n &= \lambda_i^{s+} \end{aligned} \right]$$

The multi-domain adjoint equation in Lemma 5.6 should be compared to the single-domain adjoint equation in Lemma 5.5. The only difference between these equations is that the volume part is split in two in accordance with the domain partitioning. The multi-domain adjoint equation should also be compared to the corresponding forward problem in Definition 3.9. Of particular interest are the interface conditions, by inspection they obey the same jump conditions as the forward problem with the exception of a sign change and the introduction of a source term. The source term $-2\mu\left[\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\right]\right]$ on the interface condition comes from the local shape derivative of the drag functional just like the additional source terms on the volume equations.

Corollary 5.4. ***Solution to the multi-domain adjoint equation***
The solution to the multi-domain adjoint equation from Lemma 5.6 is

$$\begin{aligned}(\boldsymbol{\lambda}^{s-},\lambda^{c-}) &= (0,-2p^{-}) \quad \textit{on } \Omega_0\\ (\boldsymbol{\lambda}^{s+},\lambda^{c+}) &= (0,-2p^{+}) \quad \textit{on } \Omega_1\end{aligned}$$

where $(\boldsymbol{\lambda}^{s-},\lambda^{c-})$ and $(\boldsymbol{\lambda}^{s+},\lambda^{c+})$ are the adjoint variables on Ω_0 and Ω_1 respectively.
Proof: *This follows from direct substitution and utilising the relation between $\boldsymbol{u}$ and p given by the Stokes equation. For the equations defined over Ω_0, Ω_1 and the boundary conditions $\Gamma_0^{if}, \Gamma_0^{w0}, \Gamma_1^{w1}$ and Γ_0^{of} the result can be adapted from the proof in Corollary 5.3. What remains to prove is therefore that the solution also satisfies the interface condtions on $\Gamma^{+/-}$. By substitution one has,*

$$\left.\begin{aligned}\boldsymbol{\lambda}^{s+}-\boldsymbol{\lambda}^{s-} &= 0\\ [\![0+(-2p)\boldsymbol{n}]\!] &= -2\mu\left[\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\right]\right]\end{aligned}\right\} \quad \textit{on } \Gamma^{+/-}$$

Here it is also substituted for $\frac{\partial \boldsymbol{\lambda}^{s-}}{\partial \boldsymbol{n}} = 0$ and $\frac{\partial \boldsymbol{\lambda}^{s+}}{\partial \boldsymbol{n}} = 0$ which follows from the solution $\boldsymbol{\lambda}^{s+} = 0$ and $\boldsymbol{\lambda}^{s-} = 0$ everywhere. Clearly the last equality holds as a direct consequence of the interface condition in Eq. (5.41)*. To see this remember that $D_n\boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$.*

5.4.3 Hadamard terms multi-domain

With the adjoint equations from Lemma 5.6 several simplifications can be made to the "Hadamard terms" in Eq. (5.55) and (5.56). The first step in this regard will be to simplify the interface terms $U1$, $U2$, $P1$ in equation (5.56) using relations that are derived in Appendix A.2. For the U1 term the Identity (A.4) along with the jump condition $[\![\boldsymbol{u}]\!] = 0$ on the interface $\Gamma^{+/-}$ is used to obtain,

(5.66)

$$\begin{aligned}(\mathrm{U1}) &= \int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^{+})\left[\frac{\partial}{\partial \boldsymbol{n}^{+}}(\boldsymbol{\lambda}^{d}\cdot[\![\boldsymbol{u}]\!]) + \kappa\boldsymbol{\lambda}^{d}\cdot[\![\boldsymbol{u}]\!]dS\right]\\ &= \int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^{+})\left[\frac{\partial \boldsymbol{\lambda}^{\boldsymbol{d}}}{\partial \boldsymbol{n}^{+}}\cdot[\![\boldsymbol{u}]\!] + \left(\frac{\partial}{\partial \boldsymbol{n}^{+}}[\![\boldsymbol{u}]\!]\right)\cdot\boldsymbol{\lambda}^{d} + \kappa\boldsymbol{\lambda}^{d}\cdot[\![\boldsymbol{u}]\!]\right]dS = \int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^{+})\left[\left(\frac{\partial \boldsymbol{u}^{+}}{\partial \boldsymbol{n}^{+}} - \frac{\partial \boldsymbol{u}^{-}}{\partial \boldsymbol{n}^{+}}\right)\cdot\boldsymbol{\lambda}^{d}\right]dS.\end{aligned}$$

In the last step in Eq. (5.66) the term involving the normal derivative of the jump is written out more explicitly as $\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}} - \frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}$. For the $P1$ term a similar result is obtained. Using the product rule and Definition A.5 from the appendix to write out div_Γ, then $P1$ can be expanded to give,

(5.67)

$$\begin{aligned}
(P1) &= -\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial}{\partial \boldsymbol{n}}\big(\boldsymbol{\lambda}^n[\![p]\!]\big)\boldsymbol{n} + \mathrm{div}_\Gamma\big(\boldsymbol{\lambda}^n[\![p]\!]\big)\Big]dS \\
&= \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial \boldsymbol{\lambda}^n}{\partial \boldsymbol{n}}\boldsymbol{n}[\![p]\!]) + \Big(\frac{\partial}{\partial \boldsymbol{n}}[\![p]\!]\Big)\boldsymbol{n}\cdot\boldsymbol{\lambda}^n + \mathrm{div}\big(\boldsymbol{\lambda}^n[\![p]\!]\big) - \boldsymbol{n}^T\nabla\big(\boldsymbol{\lambda}^n p\big)\boldsymbol{n}\Big]dS \\
&= -\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial \boldsymbol{\lambda}^n}{\partial \boldsymbol{n}}\boldsymbol{n}[\![p]\!]) + \Big(\frac{\partial}{\partial \boldsymbol{n}}[\![p]\!]\Big)\boldsymbol{n}\cdot\boldsymbol{\lambda}^n + [\![p]\!]\nabla\cdot\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^n\cdot\nabla[\![p]\!] - \boldsymbol{n}^T D\boldsymbol{\lambda}^n\boldsymbol{n}[\![p]\!] - \Big(\boldsymbol{n}^T D[\![p]\!]\Big)\boldsymbol{n}\cdot\boldsymbol{\lambda}^n\Big]dS \\
&= -\int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[[\![p]\!]\nabla\cdot\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^n\cdot\nabla[\![p]\!]\Big]dS.
\end{aligned}$$

In the last step the definition of the normal derivative, Definition A.2 in the appendix, is used on the two last terms which makes them cancel the two first terms. For $U2$ one has,

$$\text{(U2)} = \int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\mu\left[\frac{\partial}{\partial \boldsymbol{n}}\Big(\boldsymbol{\lambda^n}\cdot[\![D\boldsymbol{u}]\!]\Big)\boldsymbol{n} + \mathrm{div}_\Gamma\Big(\boldsymbol{\lambda^n}\cdot[\![D\boldsymbol{u}]\!]\Big)\right]dS. \tag{5.68}$$

Using Lemma A.1 from the appendix, Eq. (5.68) can be simplified to,

(5.69)

$$\text{(U2)} = \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\mu\left[\frac{\partial}{\partial \boldsymbol{n}}\Big(\boldsymbol{\lambda^n}\cdot[[D\boldsymbol{u}]]\Big)\boldsymbol{n} + \mathrm{div}_\Gamma\Big(\boldsymbol{\lambda^n}\cdot[[D\boldsymbol{u}]]\Big)\right]dS = \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\mu\Big[\nabla\boldsymbol{\lambda}^n : \nabla[\![\boldsymbol{u}]\!] + \nabla^2[\![\boldsymbol{u}]\!]\cdot\boldsymbol{\lambda}^n\Big]dS.$$

A final simplification is identified by merging the $(P1)$ and $(U2)$ terms. The terms of interest are those containing derivative of $\boldsymbol{\lambda}^n$. Since $\boldsymbol{\lambda}^n = \boldsymbol{\lambda}^{s+} = 0$ on $\Gamma^{+/-}$, as follows from Eq. (5.65) and Corollary 5.4, it is clear that the derivative of $\boldsymbol{\lambda}^n$ can have no tangential component on $\Gamma^{+/-}$. It follows that $\nabla\cdot\boldsymbol{\lambda}^n = \frac{\partial \lambda}{\partial \boldsymbol{n}}\boldsymbol{n}$ and $\nabla\boldsymbol{\lambda}^n = \frac{\partial \boldsymbol{\lambda}^n}{\partial \boldsymbol{n}}$. Similarly, $[\![\boldsymbol{u}]\!] = 0$ gives $[\![\nabla\boldsymbol{u}]\!] = [[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}]]$. Using these relations one gets,

$$\begin{aligned}
(U2)+(P1) &= \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\mu\nabla\boldsymbol{\lambda}^n : \nabla[\![\boldsymbol{u}]\!] + \mu\nabla^2[\![\boldsymbol{u}]\!]\cdot\boldsymbol{\lambda}^n - [\![p]\!]\nabla\cdot\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^n\cdot\nabla[\![p]\!]\Big]dS \\
&= \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[\frac{\partial \boldsymbol{\lambda}^n}{\partial \boldsymbol{n}}\cdot\Big(\Big[\Big[\mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\Big] - [\![p]\!]\boldsymbol{n}\Big) + \mu\nabla^2[\![\boldsymbol{u}]\!]\cdot\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^n\cdot\nabla[\![p]\!]\Big]dS \\
&= \int\limits_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\frac{\partial \boldsymbol{\lambda}^n}{\partial \boldsymbol{n}}\cdot\Big([\![\mu D_n\boldsymbol{u} - p\boldsymbol{n}]\!]\Big)dS = 0.
\end{aligned} \tag{5.70}$$

In the second to last step it is used that $D_n\boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ and that $\boldsymbol{\lambda}^n = \boldsymbol{\lambda}^{s+} = 0$ on $\Gamma^{+/-}$. From the interface condition in Eq. (5.41) it then follows that the expression is zero.

Putting pieces back together: From Corollary 5.4 it follows that the terms in Eq. (5.55) that contain $\boldsymbol{\lambda}^{s+}$ or $\boldsymbol{\lambda}^{s-}$ are eliminated since the solution to the adjoint equation is $\boldsymbol{\lambda}^{s+} = \boldsymbol{\lambda}^{s-} = 0$. This corresponds to the terms labeled $(S0-S2)$ and $(T0-T2)$. The remaining terms that involve integration over Γ_1^w are identical to the set of terms that were used to derive the result in Proposition 5.1. The only exception is that now $\boldsymbol{u}$ is replaced by $\boldsymbol{u}^+$ and the boundary Γ_1^w is the boundary of Ω_1 not Ω. These terms, namely $(sd1)$,$(C1)$ and $(W1)$, can therefore be replaced by the result from Proposition 5.1. The terms involving $(U2)$ and $(P2)$, from Eq. (5.56), were shown to cancel in Eq. (5.70). The remaining terms in Eq. (5.56) are now added to the remaining terms in (5.55) to form the final expression for the shape derivative. The multiplier $\boldsymbol{\lambda}^d$ is substituted for by its relation to the adjoint variables in Eq. (5.65). Similarly λ^{c+} and λ^{c-} are substituted for by $-2p^+$ and $-2p^-$ as it follows from Corollary 5.4. The remaining terms are therefore,

(5.71)

$$\begin{aligned}
\underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}\,dS}_{\text{Hadamard terms}} =& \underbrace{-\mu\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n}^+)\Big(\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}},\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\Big)dS}_{\text{From (sd1), (C1) and (W1)}} + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)\mu\Big(\nabla \boldsymbol{u}^- : \nabla \boldsymbol{u}^-\Big)dS}_{\text{(Sd0)}} + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\mu\Big(\nabla \boldsymbol{u}^+ : \nabla \boldsymbol{u}^+\Big)dS}_{\text{(sd2)}} \\
&+ \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^-)\lambda^{c-}(\nabla\cdot \boldsymbol{u}^-)dS}_{\text{C0}} + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\lambda^{c+}(\nabla\cdot \boldsymbol{u}^+)dS}_{\text{C2}} \\
&+ \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n}^+)\Big[-2\mu\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}} - \frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}\lambda^{c+} + 2\mu\frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}} + \frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}\lambda^{c-}\Big]dS}_{\text{U1}}
\end{aligned}$$

Rearranging these terms and making the substitution $\boldsymbol{n}^+ = \boldsymbol{n}$ and $\boldsymbol{n}^- = -\boldsymbol{n}$ gives the expression,

(5.72)

$$\begin{aligned}
\underbrace{\int_\Gamma (\boldsymbol{s},\boldsymbol{n})\tilde{k}\,dS}_{\text{Hadamard terms}} =& \underbrace{-\mu\int_{\Gamma_1^w} (\boldsymbol{s},\boldsymbol{n}^+)\Big(\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}},\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\Big)dS}_{\text{From (sd1), (C1) and (W1)}} + \underbrace{\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\mu[\![\nabla \boldsymbol{u} : \nabla \boldsymbol{u}]\!]dS}_{\text{from (sd0) and (sd2)}} \\
&-2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big[\underbrace{p^+\nabla\cdot \boldsymbol{u}^+}_{\text{C2}} - \underbrace{p^-\nabla\cdot \boldsymbol{u}^-}_{\text{C0}}\Big]dS + 2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big[\underbrace{p^+\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}}_{\text{from U1}} - \underbrace{p^-\frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}}_{\text{from U1}}\Big]dS \\
&\underbrace{-2\mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big[\!\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\!\Big]dS}_{\text{from U1}}
\end{aligned}$$

It is possible to get this equation on a more compact form by utilising the identities derived in Appendix A.2. Using the definition of the tangential divergence, see Definition A.5, the $(C0)$,$(C2)$ and $(U1)$ terms combine to one term. This follows from the substitution $\nabla\cdot\boldsymbol{u}^+ - \frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\cdot\boldsymbol{n} = \mathrm{div}_\Gamma \boldsymbol{u}^+$

and $\nabla\cdot\boldsymbol{u}^- - \frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}\cdot\boldsymbol{n} = \mathrm{div}_\Gamma \boldsymbol{u}^-$ such that,

$$
\begin{aligned}
&2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big[\underbrace{p^+\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}}_{\text{from U1}} - \underbrace{p^+\nabla\cdot\boldsymbol{u}^+}_{\text{C2}} + \underbrace{p^-\nabla\cdot\boldsymbol{u}^-}_{\text{C0}} - \underbrace{p^-\frac{\partial \boldsymbol{u}^-}{\partial \boldsymbol{n}}\cdot\boldsymbol{n}}_{\text{from U1}}\Big]dS = -2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})[\![p\,\mathrm{div}_\Gamma\boldsymbol{u}]\!]dS \\
(5.73)\quad &= -2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\big([\![p]\!]\langle\!\langle \mathrm{div}_\Gamma\boldsymbol{u}\rangle\!\rangle + [\![\mathrm{div}_\Gamma\boldsymbol{u}]\!]\langle\!\langle p\rangle\!\rangle\big)dS = -2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})([\![p]\!]\langle\!\langle \mathrm{div}_\Gamma\boldsymbol{u}\rangle\!\rangle dS
\end{aligned}
$$

In the second to last step the Identity A.6 was used to expand the jump term. In the last step it was used that the jump in the tangential divergence of $\boldsymbol{u}$ is zero as follows from $[\![\boldsymbol{u}]\!] = 0$. The remaining part of $(U1)$ can be combined with the $(sd2)$ and $(sd0)$ terms. To do this it is first necessary to rewrite the jump term $[\![\nabla\boldsymbol{u} : \nabla\boldsymbol{u}]\!]$. From Lemma A.2 it follows that, the jump term can be written as $[\![\nabla\boldsymbol{u} : \nabla\boldsymbol{u}]\!] = \big[\!\big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\big]\!\big] + \big[\!\big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\big]\!\big]$. With this expression and using Identity A.6 the $(U1)$,$(sd0)$ and $(sd2)$ terms combine to give,

$$
\begin{aligned}
&\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\mu\underbrace{\Big(\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\Big] + \Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\Big]\!\Big]\Big)}_{\text{from (sd0) and (sd2)}}dS - 2\mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\underbrace{\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\Big]}_{\text{from U1}}dS \\
&= \mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big(\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\Big]\!\Big] - \Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\Big]\Big)dS \\
(5.74)\quad &= \mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big(2\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\Big]\!\Big]\Big\langle\!\Big\langle\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\Big\rangle\!\Big\rangle - 2\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\Big]\Big\langle\!\Big\langle\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big\rangle\!\Big\rangle\Big)dS \\
&= -2\mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})\Big[\!\Big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big]\!\Big]\Big\langle\!\Big\langle\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big\rangle\!\Big\rangle dS
\end{aligned}
$$

In the last step it is used that the jump $\big[\!\big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\big]\!\big] = 0$ which follows from the interface condition $[\![\boldsymbol{u}]\!] = 0$. It is now possible to combine the expressions in (5.73) and (5.74) by utilising the relation between the pressure and the normal derivative offered by the interface constraint $[\![\mu D_n\boldsymbol{u} - p\boldsymbol{n}]\!] = 0$. With $D_n\boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ from Definition A.2, the normal derivative can be substituted for by $p\boldsymbol{n}$. This gives,

$$
(5.75)\quad -2\mu\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})[\![p\boldsymbol{n}]\!]\cdot\Big\langle\!\Big\langle\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\Big\rangle\!\Big\rangle dS - 2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})[\![p]\!]\langle\!\langle \mathrm{div}_\Gamma\boldsymbol{u}\rangle\!\rangle dS = -2\int_{\Gamma^{+/-}} (\boldsymbol{s},\boldsymbol{n})[\![p]\!]\langle\!\langle \nabla\cdot\boldsymbol{u}\rangle\!\rangle dS.
$$

In the last step the definition of the tangential divergence with $\nabla\cdot\boldsymbol{u} = \mathrm{div}_\Gamma\boldsymbol{u} + \boldsymbol{n}\cdot\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$, see Definition A.5 in the appendix, has been used to combine the two terms in Eq. (5.75). In the strong enforcement of the problem the divergence term in Eq. (5.75) is zero. This follows from the argument that the volume divergence $\nabla\cdot\boldsymbol{u}$ is pointwise zero in both Ω_0 and Ω_1 as defined by the divergence constraints in Eq. (5.43) and (5.48). Since the interface $\Gamma^{+/-}$ is part of both domains and since the continuity of the velocity is assumed across the interface, i.e $[\![\boldsymbol{u}]\!] = 0$, the divergence must also be zero on the points contained in the boundary. From this result it is therefore possible to eliminate the four last integrals in Eq. (5.72) and one gets,

$$\underbrace{\int_{\Gamma}(\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} = \underbrace{-\mu\int_{\Gamma_1^w}(\boldsymbol{s},\boldsymbol{n}^+)\Big(\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}},\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\Big)dS}_{\text{From (sd1), (C1) and (W1)}}, \tag{5.76}$$

which concludes the simplification of the "Hadamard terms".

Remark 5.3. ***Units check:*** *As a final check on the result, the units of the final interface terms in Eq.* (5.75) *are compared with the final expression in Eq.* (5.76)*. Let* $[z]_{un}$ *denote the units of a function* z*. From the interface condition* $[[\mu D_n \boldsymbol{u} - p\boldsymbol{n}]] = 0$ *from Eq.* (5.41) *it is clear that* $[p]_{un} = \big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\mu\big]_{un}$*. It then follows that* $[p]_{un}[\nabla\cdot\boldsymbol{u}]_{un} = \big[\mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\big]_{un}$ *which is the same units as for the standard term in Eq.* (5.76)*. This holds since* $[\nabla\cdot\boldsymbol{u}]_{un} = \big[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\big]_{un}$.

5.4.4 Concluding the derivation

To summarise this discussion, the initial equation in this derivation, namely Eq. (5.53), is restated here,

$$dJ(\Omega)[\boldsymbol{s}] = d\mathscr{L}(\Omega)[\boldsymbol{s}] = \underbrace{\int_{\Gamma}(\boldsymbol{s},\boldsymbol{n})\tilde{k}dS}_{\text{Hadamard terms}} + \underbrace{\int j'[\boldsymbol{s}]dx + \int \boldsymbol{\lambda}\cdot\boldsymbol{g}'[\boldsymbol{s}]d + \int \boldsymbol{\lambda}'[\boldsymbol{s}]\cdot\boldsymbol{g}dS}_{\text{Local shape derivative terms}}. \tag{5.77}$$

In this derivation the adjoint approach has been used to eliminate the "local shape derivative terms" and the adjoint equations in Lemma 5.6 emerged as a consequence. The Lemmas 5.1-5.3 were used to obtain explicit expressions for the "Hadamard terms". Finally, the adjoint equations along with the strong form of the interface conditions were used to simplify the expressions which led to the result in Eq. (5.76).

It is now straight forward to generalise the remaining expression to a system of M sub-domains. The result is stated in Proposition 5.2, where the inner product is denoted as $(\cdot,\cdot)$ for compactness.

Proposition 5.2. ***(Hadamard form of the shape derivative of the multi-domain Stokes-drag problem)****.*
Let the objective functional $J(\boldsymbol{u},\Omega)$ *be the drag functional as stated in Lemma 1.1. Let the objective function be constrained by the multi-domain Stokes-problem from Definition 3.9 where the domain* Ω *is partitioned according to Definition 3.8, such that* $\Omega = \bigcup_{j=0}^{M}\Omega_j$*. Under a perturbation of the sub-domains* $\{\Omega_j\}_{j=1}^{M}$ *on the form defined in* (5.1)-(5.2)*, the shape derivative of* J *is given as,*

$$dJ(\Omega)[\boldsymbol{s}] = -\mu\int_{\Gamma^w}(\boldsymbol{s},\boldsymbol{n})\Big(\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}},\frac{\partial \boldsymbol{u}^+}{\partial \boldsymbol{n}}\Big)dS,$$

where $\boldsymbol{s}$ *is the perturbation vector field defined over* Ω*, such that* $[\boldsymbol{s}(\boldsymbol{x}) = 0 : \forall x \notin \Gamma_{pert}]$*. Here* $\Gamma_{pert} = \Gamma^{+/-}\bigcup\Gamma^w$*, with* $\Gamma^w = \bigcup_{j=1}^{M}\Gamma_j^w$ *and* $\Gamma^{+/-} = \bigcup_{j=1}^{M}\Gamma_j^{+/-}$*. This is the union of all boundaries of perturbed sub-domains* Ω_j*.*

Proof: *The derivation is given in Section 5.4 with Eq.* (5.15) *through* (5.76)*. In the derivation, Lemmas 5.1-5.3 are used to get the explicit expressions for the "Hadamard terms". The adjoint equation from Lemma 5.6 along with the Lagrange multipliers in Eq.* (5.65) *are used to eliminate the "local shape derivative terms" and to simplify the "Hadamard terms". The generalisation to M sub-domains $\{\Omega_j\}_{j=1}^{M}$ follows from the assumption that the domains are non-intersecting. Under this assumption a similar derivation can be made for each of the M sub-domains independently and with the same result. By the linearity of the integral operator and by defining $\Gamma^w = \bigcup_{j=1}^{M} \Gamma_j^w$ as the integration domain the generalisation holds.*

Remark 5.4. ***Comments on the result and limitations with the final expression:*** *This remark follows up on the discussion made in Section 1.1. The final expression for the shape derivative of the multi-domain Stokes-drag problem is presented in Proposition 5.2. This result is derived from the strong formulation of the problem, as defined in Definition 3.9, before any discretisation scheme is introduced. This starting point has enabled the use of corollaries from the Hadamard theorem and the final result is independent of the mesh structure in the domain. If the system were solved in its strong form the expression in Proposition 5.2 would be the correct expression. However, since the numerical implementation relies on a discretisation scheme, additional mesh related correction terms will arise as shown in [4]. By deriving the result from the strong form, these correction terms are lost which means that the result is not discretely consistent.*

In addition to this limitation, the derivation has also assumed strong enforcement of the interface constraints on $\boldsymbol{u}$, $D_n\boldsymbol{u}$ and p, such that $[\![\boldsymbol{u}]\!] = 0$ and $[\![D_n\boldsymbol{u} - p\boldsymbol{n}]\!] = 0$ is assumed to hold. However, in the weak form of the problem these interface conditions are only enforced weakly as discussed in Section 3.4.3. Therefore, one should expect several correction terms to the expression in Proposition 5.2. As long as the jump in $[\![\boldsymbol{u}]\!]$ and the Neumann condition $[\![D_n\boldsymbol{u} - p\boldsymbol{n}]\!]$ is sufficiently small, by the means of the weak enforcement, these correction terms should be very small and not effect the result in any significant way. This is the conjecture that this derivation relies on. The numerical analysis performed in Chapter 9 aims to shed some light on how this choice effects the precision in the gradient evaluations. For future work it would also be interesting with a full implementation of all the correction terms to better see the order of magnitude in the error.

Remark 5.5. ***Independence from the adjoint variables:*** *By using the adjoint approach, the Hadamard formulation of the shape derivative takes for the Stokes-drag problem in Definition 1.2, a particularly favourable form which does not depend on the adjoint variables. This result is a consequence of the self adjoint nature of the Stokes problem, giving the analytical solution from Corollary 5.4, which has allowed for several simplifications.*

Due to this result, gradient evaluations are expected to be very inexpensive in terms of time consumption since they only require the evaluation of a boundary integral. The benefit of using the adjoint approach and the Hadamard representation in the context of the Stokes-drag problem will be further investigated with focus on this issue in Chapter 9.

Remark 5.6. ***Physical interpretation of the result:*** *As commented in the overview section in this chapter, the introduction of the interface $\Gamma^{+/-}$, when the strong interface conditions are assumed, should be purely artificial and not give any change to the physics. In this regard one should expect to see the same shape derivative in Proposition 5.2 as that derived for the single-domain case in Proposition 5.1. This derivation has shown that this is indeed the case for the strong form defined in Definition 3.9. Therefore, this result gives evidence to the consistency between the mathematical formulation and the physics of the problem.*

It is also of interest to see, that this result can be obtained without directly enforcing continuity in the normal derivative $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{u}}$ and the pressure p. Indeed, the derivation has shown that the weaker Neumann constraint $[\![D_n \boldsymbol{u} - p\boldsymbol{n}]\!] = 0$ is sufficient.

Part III

Numerical Results and Conclusion

Chapter 6

Test Problem for the Numerical Analysis

The intention of this chapter is to formulate a test setup that can be used to verify the correctness and analyse the performance of the multimesh variational form of the Stokes problem from Chapter 3 along with the Hadamard formulation of the shape derivative derived in Proposition 5.2. The test setup is imagined to be tidal turbines in Stokes flow. Section 6.1 motivates this choice while Section 6.2 defines the geometrical setup. Section 6.3 then concludes the chapter by stating an optimisation problem that is based on the combined multimesh shape-optimisation scheme developed in the previous chapters.

6.1 Motivation for the test setup

The test setup that will be used as a working example for the remainder of the work is tidal turbines in Stokes flow. The model developed here will be restricted to two dimensions, but this setup along with the overall theory developed in this work can easily be generalised to three dimensions. It is clear that this is not the most realistic model from a physical point of view, since Stokes flow does not describe the typical environment of tidal turbines. However, as a simple test setup to check the correctness and analyse the performance of the theory developed in Chapter 3 and Chapter 5, this model is considered sufficient. The model that is developed is designed to bear resemblance to tidal turbine setups employed in previous work [15, 25]. This way a benchmark can be made to test the optimisation scheme developed in this thesis. It is important to note that the results obtained for the tidal turbine setup can easily be extended to a wider range of optimisation problems. Section 1.3.1 gives examples of several other relevant problems.

6.2 Geometrical setup

This section will develop the tidal turbine setup that is used for the remainder of this work. Section 6.2.1 explains how the multimesh formalism is used to represent the turbines. Section 6.2.2 then defines the design parameters of the problem. Section 6.2.3 concludes by defining the computational domain and the constraints on the design parameters.

6.2.1 Representing the turbines using the multimesh formalism

To represent the turbines, the mesh with a hole method derived in Section 3.3 will be utilized. This method requires a set of sub-domains $\{\Omega_j\}_{j=1}^{M}$ to be defined over the computational domain Ω, with a hole that represents the shape of the turbine. For M turbines one needs a set of M independent sub-domains Ω_j. On the inner boundary Γ_j^w of the j-th turbine, the no-slip boundary condition $\boldsymbol{u}_j = 0$ is defined to represent the solid surface of the turbine. The situation is illustrated in Fig. 6.1. With this representation, the turbines can be moved without having to re-mesh the background mesh as explained in Section 3.3.

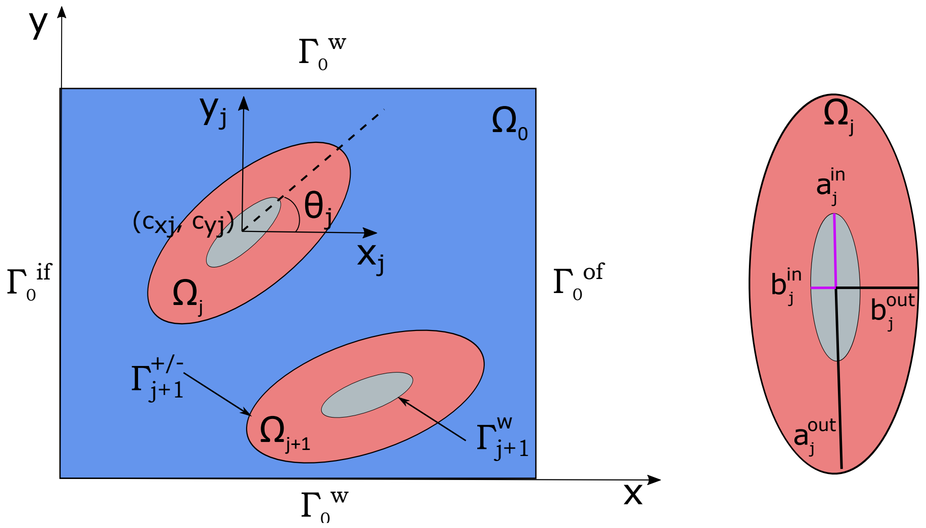

Figure 6.1: Illustration of the geometrical setup of a turbine system. The background domain Ω_0 is colored blue and the physical part of the turbines are the grey ellipses. The background domain has an inflow Γ_0^{if}, an outflow Γ_0^{of} and two wall Γ_0^w -boundaries. In addition, the background share an interface boundary $\Gamma_j^{+/-}$ with each of the M sub-domains. The pink regions corresponds to the sub-domains Ω_j used to represent the turbines. The pink regions works as overlaps between the background and the turbine boundaries Γ_j^w. This region is used to stabilise velocity and pressure over the interfaces $\Gamma_j^{+/-}$. The parameterisation of the turbines is illustrated for the j-th turbine with the center position $(c_{x,j}, c_{y,j})$ and the angular orientation θ_j. The coordinate system (x_j, y_j) is the frame of reference of the j-th turbine.

The shape of the turbines will be that of an ellipse, with outer semi-axes (a_j^{out}, b_j^{out}) and inner semi-axes (a_j^{in}, b_j^{in}). This is illustrated in Fig. 6.1. This shape has the advantage of a simple implementation and that it can be thought to represent the cross section of a turbine blade.

6.2.2 Design parameters

The test setup will be used to optimise the positions of the turbines and not the shape itself. To uniquely describe a turbine it is therefore sufficient to parameterize the center position and the rotational orientation. The design parameters for the j-th turbine are indicated in Fig. 6.1 which shows the center positions $(c_{x,j}, c_{y,j})$ along with the angular orientation θ_j. The angle θ_j is measured relative to the x-axis as shown in Fig. 6.1. With M turbines the design parameters will be stored in the vectors,

$$\begin{aligned} \boldsymbol{m} &= \begin{bmatrix} c_{x,1}, & c_{y,1}, & \dots, & c_{x,M}, & c_{y,M} \end{bmatrix}, \\ \boldsymbol{m}_\theta &= \begin{bmatrix} \theta_1, & \theta_2, & \dots, & \theta_M \end{bmatrix}. \end{aligned} \tag{6.1}$$

Since only the positions of the turbines will be optimised for, the angular orientations $\boldsymbol{\theta}_j$ will be specified for each system configuration and then kept fixed during the optimisation.

6.2.3 Computational domain and constraints on the design parameters

The computational domain used for this work will be defined as a 2D unit square $\Omega = [0,1]^2$. The boundary $\partial\Omega$ will be divided into an inflow region Γ_0^{if}, an outflow region Γ_0^{of} and a wall region Γ_0^{w}. These boundaries are indicated on Fig. 6.1. The regions are formally defined as follows,

$$\begin{aligned} \Gamma_0^{if} &= \{(x,y) : x = 0 \text{ and } 0 \le y \le 1\} \\ \Gamma_0^{of} &= \{(x,y) : x = 1 \text{ and } 0 \le y \le 1\} \\ \Gamma_0^{w} &= \{(x,y) : 0 \le x \le 1 \text{ and } y = 0 \vee 1\} \end{aligned} \tag{6.2}$$

In order to avoid the turbines to move out of the domain during the optimisation algorithm, the allowed domain of movement is restricted to $\Omega_r = [0.2, 0.8]^2$. To enforce these constraints the turbine center positions $\boldsymbol{c}_j = (c_{x,j}, c_{y,j})$ are restricted by the constraint equations,

$$\boldsymbol{x}_l \le \boldsymbol{c}_j \le \boldsymbol{x}_u \text{ for } j = 1, 2, \dots, M \tag{6.3}$$

Where $\boldsymbol{x}_l = (0.2, 0.2)$ and $\boldsymbol{x}_u = (0.8, 0.8)$. The margin between the boundaries $\partial\Omega$ and $\partial\Omega_r$ is chosen so that the turbines can have any angular orientation θ_j.

The numerical values chosen for the dimensions of the turbines are,

$$\begin{aligned} (a_j^{out}, b_j^{out}) &= (0.8, 0.55) \\ (a_j^{in}, b_j^{in}) &= (0.6, 0.35), \end{aligned} \tag{6.4}$$

unless otherwise stated in the text. Similarly, the boundary conditions that are used are,

$$
\begin{aligned}
&\boldsymbol{g}_N = 0 && \text{on } \Gamma_0^{of} \\
&\boldsymbol{u} = 0 && \text{on } \Gamma_0^{w} \\
&\boldsymbol{u} = 5\sin\pi y \quad \vee \quad \boldsymbol{u} = 5y^2\sin\pi y && \text{on } \Gamma_0^{if} \\
&\boldsymbol{u} = 0 && \text{on } \Gamma_1^{w}
\end{aligned}
\tag{6.5}
$$

unless otherwise stated. Two different inflow profiles will be used. The symmetric $\sin\pi y$ and the skewed $y^2\sin\pi y$. Both profiles are chosen such that $\boldsymbol{u} = 0$ at the intersection with Γ_0^w. The skewed inflow profile is the one that will be used for the actual optimisation problem. The other is chosen for its symmetry properties, to check the behaviour of the gradient around the maxima in y that is present at $y = 0.5$ due to the symmetry of the computational domain.

In order to prevent the turbines to overlap, which is considered nonphysical, a constraint on the minimum distance r_{ij}^{min} between two turbine centers $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ for $i \neq j$ is defined as,

$$
g(\boldsymbol{c}_i, \boldsymbol{c}_j) = \left(r_{ij}^{min}\right)^2 - \left\|\boldsymbol{c}_j - \boldsymbol{c}_i\right\|^2 \leq 0 \quad \forall \quad i,j : 1 \leq i \leq j \leq M
\tag{6.6}
$$

The minimum allowed distance between the turbines will be $r_{ij}^{min} = \max(a_i^{out}, b_i^{out}) + \max(a_j^{out}, b_j^{out})$ where a^{out} and b^{out} are the outer semi-axis in the ellipses of turbine i and j.

6.3 The optimisation problem

With the geometrical setup and parameterization in place, one can finally define the Stokes-drag optimisation problem from Definition 1.2 in a multimesh setting. This formulation corresponds to the discretization step which is the final step in the "continuous approach to the shape derivative" presented in Fig. 5.1.

The turbines, i.e the sub-domains Ω_j, are parameterized using the set of design parameters $\boldsymbol{m}$. With $\Omega_j(\boldsymbol{m})$ for $j = 1, 2, \ldots, M$ it follows that $\boldsymbol{u}(\Omega) = \boldsymbol{u}(\boldsymbol{m})$. It is therefore convenient to express the drag J as a function of $\boldsymbol{m}$, which gives the reduced optimisation problem,

$$
\min_{\boldsymbol{m}} \quad (-J(\boldsymbol{m})) \tag{6.7}
$$

$$
\text{subject to } F(\boldsymbol{u}, p, \boldsymbol{v}, q) = 0 \quad \forall \quad (\boldsymbol{v}, q) \in V_h^{mm} \oplus Q_h^{mm} \tag{6.8}
$$

$$
\boldsymbol{x}_l \leq \boldsymbol{m} \leq \boldsymbol{x}_u \tag{6.9}
$$

$$
g(\boldsymbol{m}) \leq 0. \tag{6.10}
$$

Here $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are the upper and lower constraints on the turbine centers given in Section 6.2.3 and $g(\boldsymbol{m})$ is the constraint defined in Eq. (6.6) that prevents the turbines to intersect. The state equation $F(\boldsymbol{u}, p, \boldsymbol{v}, q) = 0$ corresponds to the residual $A^{mm}(\boldsymbol{u}, p, \boldsymbol{v}, q) - L^{mm}(\boldsymbol{v}) = 0$ from the multimesh variational formulation of the Stokes problem stated in Section 3.4. The two test functions $\boldsymbol{u}$ and p are the velocity and the pressure in the Stokes flow and the corresponding trail functions are $\boldsymbol{v}$ and q. Finally, $V_h^{mm} \oplus Q_h^{mm}$ is the mixed multimesh function space as defined in Section 3.4.1.

CHAPTER 7

Implementation and Verification of the Multimesh Stokes Solver

To solve the optimisation problem defined in Section 6.3, it is first necessary to implement a solver for the multimesh variational form of the Stokes problem from Section 3.4. The Multimesh finite element solver (MMFEM solver) for the Stokes equations has been implemented using the FEniCS framework. Section 7.1 gives a short introduction to the FEniCS project and explains how it has been utilised in this work. In Section 7.2 a benchmark problem is made in order to verify the convergence of the implemented solver.

7.1 Software and implementation in Python

7.1.1 The FEniCS project

The FEniCS project [2, 30] is a collaboration between several research institutes and universities with the goal of developing tools for scientific computing. The project consist of several software libraries that are made to operate together. Of special interest for this work is the DOLFIN library [29, 31] which employs a framework for easily generating PDE solvers using the finite element method. The FEniCS version 2018.1.0, that is used in this work, includes framework for solving PDEs using the multimesh formulation described in Chapter 3. For mesh generation the mshr library [2] in the FEniCS project is used. This is a *C*++ library with a Python interface that provide an easy to use mesh generator for meshes of intermediate complexity.

7.1.2 Implementation

The code developed in this work can be found in the GitHub repository [40]. The Dolfin library offers a problem solving environment and framework in which one can state variational problems for PDEs in a notation very close to the mathematical formulation. To do this, Dolfin uses the Unified form language (UFL) [1] which offers easy to use notation for expressing integrals, inner products and other mathematical operations needed to specify a finite element variational form.

To get a better understanding of how this works, some selected pieces of the multimesh solver, implemented for this work, are discussed. In Listing 7.1 the MMFEM function spaces for the test and trial functions $\boldsymbol{u}$ and $\boldsymbol{v}$ are defined. The code extract corresponds to the implementation of the function spaces defined in Section 3.4.1. As can be seen, the code is quite similar to the mathematical formulation.

```python
import dolfin*

# Define vector spaces for test and trial functions. Specify element type
# and polynomial degree.
multimesh = final_turbSystem.get_multimesh()
P2_ele = VectorElement("Lagrange", multimesh.ufl_cell(), 2)
P1_ele = FiniteElement("Lagrange", multimesh.ufl_cell(), 1)

MizedElement_taylorHood = MixedElement([P2_ele, P1_ele])
W = MultiMeshFunctionSpace(multimesh, MizedElement_taylorHood)

V = MultiMeshSubSpace(W, 0)
Q = MultiMeshSubSpace(W, 1)

# Define trial and test functions
(u, p) = TrialFunctions(W)
(v, q) = TestFunctions(W)
```

Listing 7.1: Code extract that shows the implementation of the Sobolov spaces for the test and trial functions. The extract illustrates how Dolfin is used to implement the mathematical definitions in a notation similar to the mathematical formulation. The Taylor-Hood element $P2, P1$, with the Lagrange basis functions can easily be specified and the test and trial functions can be defined over the resulting multimesh function spaces as seen in the last line.

With the test and trial functions defined over the chosen function spaces, the multimesh variational form from Section 3.4 can be implemented. As an example the implementation of the standard bi-linear form $a_n(\boldsymbol{u},\boldsymbol{v})$ in Eq. (3.14) is presented in Listing 7.2. Here it can be seen that to represent the integral it is sufficient to specify the measure. There are several built in measures like the volume integration measure dX used here. For a more specific integration, like integration over the physical boundary of the turbines, it is possible to make user defined measures.

```python
def a_n(self, u, v):
    return inner(grad(u), grad(v))*dX
```

Listing 7.2: Implementation bi-linear form $a_n(\boldsymbol{u},\boldsymbol{v})$ from Eq. 3.14. The implementation illustrates how the UFL language can be used to define the mathematical expressions in a variational form.

From the variational form the linear system can be assembled. The Dolfin framework also offers functionality for applying the Dirichlet boundary conditions directly on to the system matrix. To solve the resulting linear system the standard linear solver "MUMPS" has been used. This is a direct solver for sparse linear equations on the form $\boldsymbol{Ax} = \boldsymbol{b}$, i.e systems where the stiffness matrix $\boldsymbol{A}$ has a large amount of zeros. This solver is offered through the Dolfin framework and is sufficient for the purpose of this work. For an idea on how the stiffness matrix relates to the variational form the reader is referred to Example 2.1.

7.2 Verification of the multimesh Stokes solver

This section presents the benchmark problem used to test the convergence of the multimesh Stokes solver. The convergence of the solution is then checked in standard error norms.

7.2.1 Benchmark model

To check the convergence of the solution a simple setup with a known analytical solution is made. For this purpose the geometrical setup defined in Section 6.2 is used with only one elliptic sub-domain, labeled Ω_1. For simplicity, the hole in domain method from Section 3.3 is not used in the benchmark. The inner semi-axes a^{in} and b^{in} from Section 6.2 are therefore removed. For the the boundary conditions the simple choice of $\boldsymbol{u} = 0$ and $p = 0$ over the entire boundary $\partial\Omega$ is used.

With the source term f in Eq. 1.1 given as,

$$\boldsymbol{f} = 2\pi \begin{bmatrix} \sin(2\pi x_2)(\cos 2\pi x_1 - 2\pi^2 \cos(2\pi x_1) + \pi^2) \\ \sin(2\pi x_1)(\cos(2\pi x_2) + 2\pi^2 \cos(2\pi x_2) - \pi^2) \end{bmatrix} \tag{7.1}$$

and the boundary conditions as stated above, an exact solution is,

$$\boldsymbol{u}(x_1,x_2) = 2\pi \sin(\pi x_1)\cos(\pi x_2) \begin{bmatrix} \cos(\pi x_2)\sin(\pi x_1) \\ -\cos(\pi x_1)\sin(\pi x_2) \end{bmatrix}, \tag{7.2}$$

$$p = \sin(2\pi x_1)\sin(2\pi x_2). \tag{7.3}$$

This will be the setup that is used for the convergence tests in the next section.

7.2.2 Convergence of the solution

The convergence rates of the solution $(\boldsymbol{u}_h, p_h)$ to the implemented multimesh variational form of the Stokes problem from Section 3.4.2 is measured in the $\|\cdot\|_{L^2}$ and $\|\cdot\|_{H_0^1}$ norms. These norms where defined in Definition B.2 and Definition B.5 in the appendix. Let E_h represent the respective norms, the relation between the mesh size h and E_h is then expected to obey the relation,

$$\ln(E_h) = C + r\ln(h) \tag{7.4}$$

where r is the convergence rate in the respective norm and C is some constant of proportionality. The convergence rate r is related to the polynomial degree of the basis functions used to represent the finite element functions $(\boldsymbol{u}_h, p_h)$. With $(\boldsymbol{u}_h, p_h) \in V_h \otimes Q_h$ as defined in Section 3.4.1, the expected convergence rate is $r = 2$ for $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H_0^1}$ and $\|p - p_h\|_{L^2}$ while it is $r = 3$ for $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{L^2}$. This follows from the Taylor-Hood element chosen for the function space with the (P_2, P_1) pair.

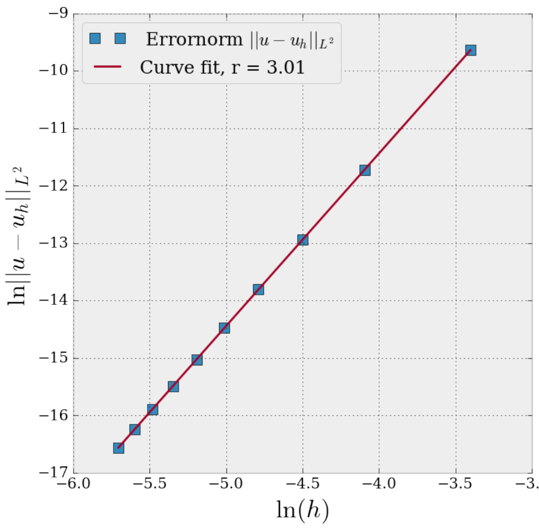

Figure 7.1: The figure shows the convergence rate of the velocity $\boldsymbol{u}$ in the $\|\cdot\|_{L^2}$ norm. With P_2 elements it is expected that this rate should be $r = 3$. The blue squares corresponds to the logarithm of the error norm as a function the logarithm of the mesh size h. The purple line corresponds to the regression curve that gives a convergence rate estimate of $r = 3.01$.

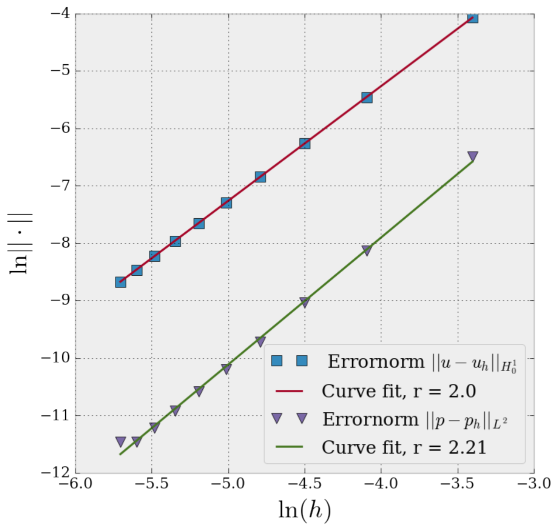

Figure 7.2: The figure shows the convergence rates for $\boldsymbol{u}$ and p in the error norms $\|\cdot\|_{H_0^1}$ and $\|\cdot\|_{L^2}$. The blue squares corresponds to the logarithm of the error norm $\|\cdot\|_{H_0^1}$ as a function of the logarithm of the mesh size h. The purple triangles is likewise the logarithm of the error norm $\|\cdot\|_{L^2}$. The purple and the green lines corresponds to the regression curves.

A linear regression of the logarithm of $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{L^2}$ plotted against the logarithm of the mesh size h, is presented in Fig. 7.1. From the linear regression the convergence rate r from Eq. (7.4) is found to be $r = 3.01$ as expected. Similar curve fits where made for $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H_0^1}$ and $\|p - p_h\|_{L^2}$

	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{H_0^1}$	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$	$\|p-p_h\|_{L^2}$
r	2.01	3.01	2.21

Table 7.1: Convergence rate obtained from linear regression of the log-log plot of the error norms, $\|\boldsymbol{u}-\boldsymbol{u}_h\|_{H_0^1}$, $\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$ and $\|p-p_h\|_{L^2}$, against the mesh size h.

h	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{H_0^1}$	$\|p-p_h\|_{L^2}$
3.3e-2	1.7e-2	1.5e-3
8.0e-3	1.1e-3	6.1e-5
3.7e-3	1.7e-4	1.0e-5

Table 7.2: Table showing the residual errors in $\boldsymbol{u}$ and p in the $\|\cdot\|_{H_0^1}$ and $\|\cdot\|_{L^2}$ norms as a function of the cell size h.

which gave convergence rates close to the expected value of $r = 2$. These plots are shown in Fig. 7.2. The convergence rates found from the linear regressions are summarised in Table 7.1.

The residual errors in velocity and pressure, measured in the $\|\cdot\|_{H_0^1}$ and $\|\cdot\|_{L^2}$ norms respectively, are presented in Table 7.2 for some selected mesh sizes h. The error decreases with finer meshes at the correct rates. From this analysis it is demonstrated that the implemented multimesh solver for the Stokes problem has optimal order convergence.

Chapter 8

Implementation and Verification of the Optimisation Algorithm

In order to test the performance of the shape derivative derived in Proposition 5.2, a gradient based optimisation algorithm has been implemented in python. For this purpose the software package Ipopt has been used. Section 8.1 will begin the chapter with an introduction to the Ipopt software and the optimisation algorithm. The chapter is concluded by Section 8.2 which explains how the gradient from Proposition 5.2 is parameterised.

8.1 Software and verification of the optimisation algorithm

This section will give a short introduction to the Ipopt software and explain the most crucial parts of the optimisation algorithm that is implemented in this package. In Section 8.1.3, a simple benchmark problem, that is not involving the shape derivative, is designed to test the performance of the combined multimesh Ipopt optimisation algorithm.

8.1.1 Interior point line search filter method (Ipopt)

Ipopt is a software package for large scale nonlinear optimisation. The software is open source and extensive documentation is available. In [52] a thorough description of the algorithm is presented while a more light-weight introduction is provided by [49]. Examples of previous work that utilise this algorithm can be found on [7]. More mathematical details on the algorithm can be found in several papers including [39, 50, 51].

The Ipopt software implements an interior point line search filter method that seeks to minimise the objective function, the details of the implementation can be found in [52]. It is important to note that the algorithm only seeks to find a local minimizer and the solution will therefore rely on the initial conditions. The python module pyipopt [53], developed by Eric Xu, has been used to get access to the Ipopt package through python. This functionality is available through the DOLFIN-adjoint project [11] and in particular the optimisation framework [14]. The Ipopt package is designed to solve optimisation problems similar to the problem defined in Section 6.3. It is therefore natural to utilise this software when implementing an optimisation algorithm for the turbine configuration.

8.1.2 The Ipopt algorithm

The Ipopt algorithm is based on an Armijo-backtracking scheme similar to that described in Chapter 4 and relies on a gradient descent method to chose appropriate search directions. A more detailed description of this particular implementation can be found in [52]. Around this simple core there are several additional features with the algorithm that handles domain and parameter constraints and helps increase the efficiency of the algorithm. The most important aspects will be presented below.

Interior point (Barrier): To enforce domain constraints similar to those defined on the design parameters $\boldsymbol{m}$ in Eq. 6.9, the Ipopt solver uses an Interior point (or barrier) method. The idea is to introduce a barrier term to the objective function J similar to,

$$B(\boldsymbol{m}) = J(\boldsymbol{m}) - \sum_{m \in \boldsymbol{m}} \left[\mu_l \ln(m_c - x_l) - \mu_u \ln(x_u - m_c)\right], \tag{8.1}$$

and then minimise the function B instead. Here μ represent the strength of the barrier. If any of the design parameters $\boldsymbol{m}$ gets close too its domain bounds x_l, x_u the barrier term blows up. Minimizers on the boundary of the allowed domain can therefore not be reached using this algorithm and thereof the name "Interior point". The parameter $\boldsymbol{\mu} = [\mu_l, \mu_u]$ determines the strength of the barrier. The method will start with a large value for this parameter and then reduce it in sequences when certain conditions are met. See [39] for more details. In general the solution of the barrier problem should converge to the standard problem when $\boldsymbol{\mu} \longrightarrow 0$, see [49]. It is important to be aware of this feature with the algorithm as it can, for an intermediate set of iterations, increase the objective function in the minimization process.

Filter method: The no-overlap condition on the turbines was defined in Eq. (6.10). The strategy for handling this constraint problem is to use a Filter method. This concept was first introduced by [12] and its implementation into the Ipopt software can be found in [51, 52]. The main idea in this method is to convert the problem into a dual-objective optimisation problem where both the objective function and the constraint violation are minimized. In addition the method contains a filter $\mathscr{F}_k$ that contains information about the objective function and constraint violations at previous iterations. A trail point $\boldsymbol{m}_k$ in the line-search is therefore accepted if it sufficiently decrease either the constraint violation or the objective function while it is not sorted out as prohibited by the filter.

Restoration phase If the backtracking line-search cannot find an acceptable step size ϵ_k, i.e $\epsilon_k \leq \epsilon^{min}$, then the algorithm enters the feasibility restoration phase.This can typically happen if the constraint condition prevents steps that will be optimal for the objective function or if the linear system is ill-conditioned. In this phase the objective function is ignored and the algorithm tries to minimise the constraint violation according to some scheme, see [52] for more details. Similar to the barrier term, this phase could intermediately increase the objective function and is therefore important to be aware of.

8.1.3 Verification of the combined multimesh Ipopt algorithm

In order to confirm that the optimisation algorithm combined with the multimesh scheme is working correctly, a simple test is performed of the convergence. The test will also serve as a reference for the remaining work. The idea is to introduce a dummy function f with a known optimal solution. The dummy is defined as,

$$f(\boldsymbol{m}) = \sum_{i=1}^{M} (m_i - P_i)^2 \tag{8.2}$$

where $\{m_i\}_{i=1}^{M}$ is the design parameters of the turbine system defined in Section 6.2.2 and the set of variables $\{P_i\}_{i=1}^{M}$ gives the configuration of the system that minimises f. The gradient with respect to the turbine possitions is an array $\nabla f(c_y, c_y) = \left[\nabla f_i\right]$ with the i-th component given as,

$$\nabla f(c_y, c_y)_i = 2(m_i - P_i). \tag{8.3}$$

The results from the preliminary convergence test is presented in Table 8.1 were all turbines can be seen to have reached their optimal position. The algorithm terminated after 12 iterations and reduced the objective function from $f = 4725.7$ to $f = 2.3e-6$. A similar test was done where the optimal positions $\{P_i\}_{i=1}^{M}$ were chosen such as to not be feasible without violating the constraints. The initial configuration for this test is shown in Fig. 8.1. The optimal (infeasible) positions are then chosen as summarised in Table 8.2. The objective positions of Turbine 1 (red) and Turbine 2 (yellow) were deliberately chosen to the same point. This allows to test how the algorithm handles the non-intersection constraint stated in Eq. (6.6). The optimal position of Turbine 3 (orange) was chosen outside of the allowed domain, which allows to test how the algorithm handles the boundary constraint stated in Eq. (6.3). The algorithm terminated after 17 iterations with the default tolerance of the Ipopt solver. The objective function was reduced from $f = 4725.7$ to $f = 0.039$ and all the constraint conditions defined in Section 6.3 were respected.

The resulting configuration is shown in Fig. 8.2. From the figure and Table 8.2 it is clear that the turbines have all reached relatively close to their optimal positions, given the current constraints.

Table 8.1: Table summarising the results from a preliminary convergence test for the optimisation algorithm using a dummy function. In this test the optimal positions were chosen so as to not violate any constraints.

	Turb. 1	Turb. 2	Turb. 3
Final pos.	(0.7002, 0.7001)	(0.3004, 0.2999)	(0.2986, 0.6994)
Opt. pos.	(0.7, 0.7)	(0.3, 0.3)	(0.3, 0.7)
Final angl.	60.0	25.0	90.0
Opt. angl.	60.0	25.0	90.0

Table 8.2: Table summarising the results from a convergence test for the optimisation algorithm using a dummy function. The optimal positions shown in the third row were chosen such that the turbines could not reach the positions without violating the non-overlap condition.

	Turb. 1	Turb. 2	Turb. 3
Init. pos.	(0.2, 0.2)	(0.5,0.5)	(0.75, 0.2)
Final pos.	(0.59, 0.60)	(0.41, 0.39)	(0.23, 0.79)
Opt. pos. (infeasible)	(0.5, 0.5)	(0.5, 0.5)	(0.23, 0.82)
Init. angl.	30	10	30
Final angl.	60.0	25.0	90.0
Opt. angl.	60.0	25.0	90.0

A graphical representation of the progress during the optimisation is shown in Fig. 8.3. The figure shows the function value $\|f\|$ plotted against the number of function evaluations. The blue circles corresponds to each time f is evaluated. The purple crosses corresponds to the function value at a new optimisation iteration. In the new iteration the gradient is again evaluated and a new line search direction decided. The spikes in the blue curve in Fig. 8.3, is typical for the Ipopt solver and can be related to the barrier parameter or the restoration phase, as discussed in Section 8.1.2.

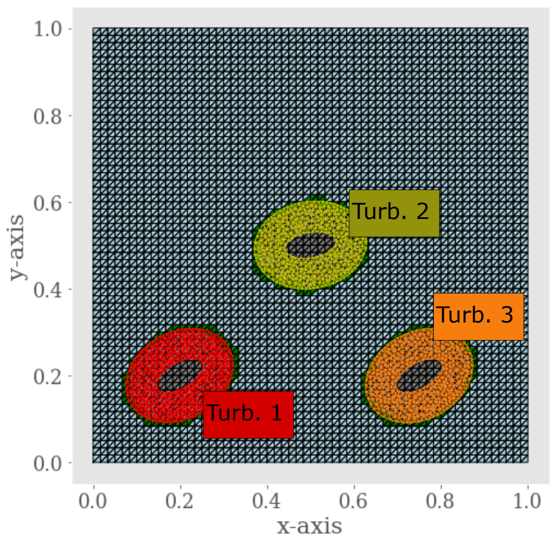

Figure 8.1: Initial configuration of turbine system. The three turbines are colored red, orange and yellow to distinguish them from each other.

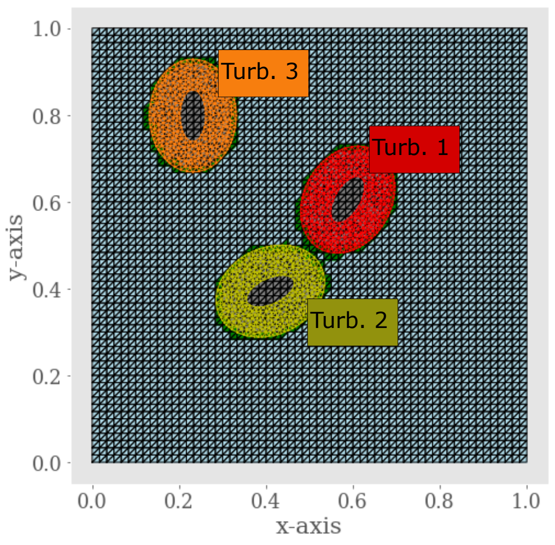

Figure 8.2: Final configuration of turbine system. The optimal configuration were here chosen such that both the red and the yellow turbine had the same optimal position. The final configuration shows how the optimisation algorithm solved this issue.

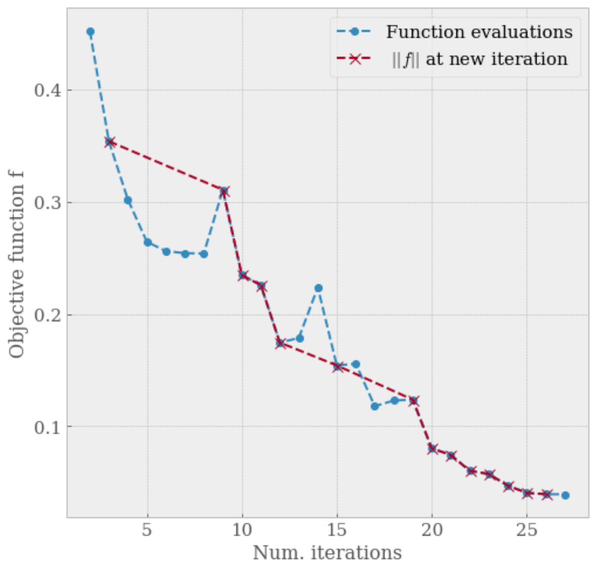

Figure 8.3: Progress plot for the objective function f. The first 2 iterations are omitted to resolve the final progress. The number of function evaluations are shown with a dotted blue line. The x-marks on the purple line corresponds to the function value at a new optimisation iteration. The spikes in the blue curve is typical for the Ipopt algorithm and follows from the restoration phase or an increase in the barrier parameter as discussed in Section 8.1.2

.

8.2 Implementation

This section will describe how the Hadamard form of the shape derivative from Proposition 5.2 is implemented. To implement a gradient representation that can be combined with the optimisation algorithm it is necessary to define a parameterisation for the perturbation field $\boldsymbol{s}$ from Eq. (5.2). In $\mathbb{R}^2$ and with movement restricted to translation of the turbines, it is sufficient to parameterise the perturbation with an x and y component. With this parameterisation the gradient for translational movement can be calculated. Section 8.2.1 explains how this has been done.

8.2.1 Calculating the gradient for translational movement

The parameterisation of the turbine domains Ω_j from Section 6.2.2 makes translation of the domain easy to implement. By parameterising the perturbation field with,

$$\boldsymbol{s}(\boldsymbol{x})_{x,j} = (1,0)\forall \quad \boldsymbol{x} \in \Omega_j. \tag{8.4}$$

$$\boldsymbol{s}(\boldsymbol{x})_{y,j} = (0,1) \quad \forall \boldsymbol{x} \in \Omega_j. \tag{8.5}$$

$$\boldsymbol{s}(\boldsymbol{x})_j = (0,0) \quad \text{otherwise}, \tag{8.6}$$

one can ensure movement only of the turbine of interest. Note that $\boldsymbol{x} = (x, y)$. In the two dimensional domain the gradient $\nabla J(\Omega)$ can then be represented as,

$$\nabla J_j = \left[dJ(\Omega)[\boldsymbol{s}_{x,j}] \quad dJ(\Omega)[\boldsymbol{s}_{y,j}] \right], \tag{8.7}$$

for perturbation of the j-th sub-domain (turbine). Here $dJ(\Omega)[\boldsymbol{s}]$ corresponds to the Hadamard formulation of the shape derivative from Proposition 5.2. In the expression for the shape derivative in Proposition 5.2, the dynamic viscosity μ is set to $\mu = 1$ to be consistent with the choice made for the multimesh variational form of the Stokes problem in Section 3.4.2.

CHAPTER 9

Numerical Results for Shape Optimisation on Multiple Meshes

This chapter is dedicated to the numerical analysis of the multimesh formulation from Chapter 3, the Hadamard formulation of the shape derivative from Chapter 5 and to the combined multimesh shape-optimisation scheme from Objective 3. These analyses corresponds to the numerical aspects of Objective 1,2 and 3 stated in the introduction. The chapter is divided in two parts. The first part, Section 9.1, presents preliminary tests for the shape derivative. The second part, Section 9.2, is then dedicated to the performance of the multimesh and the combined multimesh shape optimisation scheme.

9.1 Verification of the shape derivative

This section will present preliminary tests to verify the proficiency of the shape derivative from Proposition 5.2. The section starts with convergence tests performed on the single-domain adjoint solution from Corollary 5.3. It then concludes with a section considering the general behaviour of the shape derivative in several test scenarios.

9.1.1 Convergence tests for the adjoint equation

To check the analytical solution $(\boldsymbol{\lambda}^s, \lambda^c) = (0, -2p)$ found for the adjoint equations in Lemmas 5.5 and 5.6, a numerical implementation has been made. This implementation is for the single-domain case, using a unit square domain $\Omega = [0,1]^2$ and the inflow profile $5\sin\pi y$. The boundaries and boundary conditions are otherwise as given in Section 6.2. The results from this test are

Table 9.1: Table showing the residual error between the adjoint variables $(\boldsymbol{\lambda}^s, \lambda^c)$ and the analytical solution $(\boldsymbol{\lambda}^s, \lambda^c) = (0, -2p)$ for increasing number of cells. The convergence rate is shown in the far right columns. For $\boldsymbol{\lambda}^s$, a prediction for the convergence rate was not possible as the residual was already close to zero for very few cells, the error only oscillated around $\sim 10^{-12}$ for increasing mesh refinement as can be seen in the table. The expected convergence rate in λ^c of $r = 2$ is reached to an acceptable degree.

Cells	$\|\boldsymbol{\lambda}^s\|_{H^1}$	$\|\lambda^c - (-2p)\|_{L^2}$	r_{λ^c}
10,000	2.6e-13	4.3e-4	1.88
40,000	1.5e-12	1.1e-4	1.89
90,000	2.7e-12	4.9e-5	1.90
160,000	9.5e-12	2.9e-5	1.90
250,000	2.4e-12	1.9e-5	1.90
360,000	4.9e-12	1.5e-5	1.89

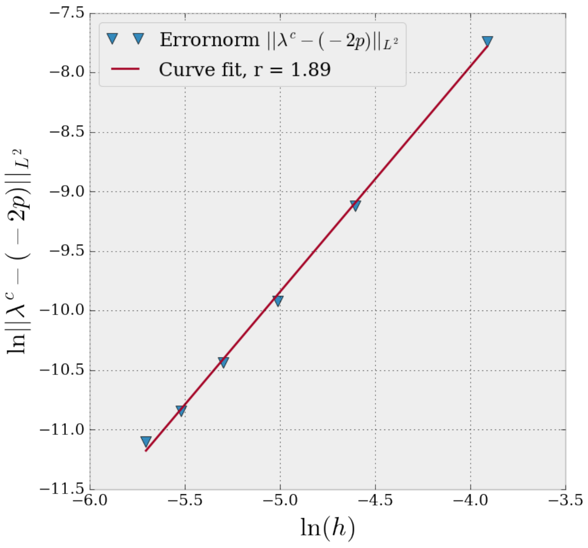

Figure 9.1: Convergence test for the single-domain adjoint equation from Lemma 5.5. An estimate for the convergence rate r is obtained by linear regression of the logarithm of $\|\lambda^c - (-2p)\|_{L^2}$ plotted against the logarithm of the mesh size h.

summarised in Table 9.1. An estimate for the convergence rate in $\|\lambda^c - (-2p)\|_{L^2}$ was obtained by linear regression of the logarithm of the error norm and the logarithm of the mesh size h. Fig 9.1 shows how this was done. The convergence rate was estimated to be $r_{\lambda^c} = 1.89$ which is acceptably close to the expected convergence rate of $r = 2.0$. A similar estimate for the convergence rate in the adjoint variable $\boldsymbol{\lambda}^s$ was not possible since the residual error $\|\boldsymbol{\lambda}^s\|_{H^1}$ was already very close to zero as seen from the second column in Table 9.1. As can be seen, the error only oscillates around $\sim 10^{-12}$ when increasing the number of cells. For λ^c on the other hand, the residual error $\|\lambda^c - (-2p)\|_{L^2}$ is decreasing with increasing mesh refinement. Based on these results the analytical solution of the single-domain adjoint equation found in Corollary 5.3 is confirmed.

For the multi-domain case, the analytical solution was also found to be $(\boldsymbol{\lambda}^s, \lambda^c) = (0, -2p)$. Since it is not obvious how to implement a multimesh formulation of the adjoint equations from Lemma 5.6, the analytical solution found in Corollary 5.4 will be trusted without a numerical verification. It is therefore left to future work to develop a multimesh formulation for the multi-domain adjoint equations. For future treatment of this problem, it should be made particular care to handle the additional source terms that distinguish the adjoint equations in Lemma 5.6 from the forward problem in Definition 3.9.

9.1.2 Verification of the shape derivative

In order to investigate the correctness of the shape derivative from Proposition 5.2, a set of Taylor tests have been conducted for increasing mesh refinement. In addition, particular care has been made to verify the gradient around two local extrema, when refining the mesh. The motivation for this was given in Section 4.2.2, where it was argued why it is important that the estimated gradient correctly predicts the descent direction. In this context it is essential that it has a sign change at the extrema points.

Taylor convergence tests: A simple Taylor expansion of the objective function, gives $J(\boldsymbol{z}) = J(\boldsymbol{z}_0) + \boldsymbol{\epsilon_1} \cdot \nabla J(\boldsymbol{z}_0) + C\|\boldsymbol{\epsilon_1}\|^r$ where $\boldsymbol{\epsilon_1} = \boldsymbol{z} - \boldsymbol{z}_0$ is the displacement and where ∇J is calculated according to Eq. (8.7). With a correct gradient, the convergence rate is expected to be $r = 2$. In order to verify this, two Taylor tests have been performed with the displacement directions $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$. Here $\boldsymbol{\epsilon}_1$ corresponds to displacement along the y-axis from the start position $(0.5, 0.5)$ while $\boldsymbol{\epsilon}_2$ corresponds to displacement along the x-axis with initial position $(0.5, 0.7)$. The setup consists of 1 turbine with angular orientation 30^o. The inflow profile is chosen as $5\sin\pi y$ and the inflow boundary and the general geometry of the domain is otherwise as defined in Section 6.2.

For the first Taylor test, with displacement along $\boldsymbol{\epsilon}_1$, a linear regression is performed between the log of the first order residual $\ln R(\boldsymbol{\epsilon}_1) = \ln\|J(\boldsymbol{z}) - J(\boldsymbol{z}_0) + \boldsymbol{\epsilon}_1 \cdot \nabla J(\boldsymbol{z}_0)\|$ and the log of the displacement length $\ln\|\boldsymbol{\epsilon}_1\|$. The regression is shown in Fig. 9.2 where it can bee seen that the expected convergence rate of $r = 2$ is reached by refining the mesh. The convergence rates are summarised in the second column in Table 9.2.

Table 9.2: The table shows the estimated convergence rate, in the first order Taylor expansion, as a function of number of cells in the total mesh. When the mesh is refined the expected convergence rate of $r = 2$ is reached. The Taylor tests were performed in two different directions $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$. The convergence rates found in the two directions are denoted r_1 and r_2 respectively.

Cells	r_1	r_2	$\langle h \rangle$
129,600	2.44	1.25	0.006
176,400	2.16	1.6	0.005
230,400	1.99	1.8	0.004

To visualise how the convergence rate depends on the displacement $\boldsymbol{\epsilon}_1$, the convergence rate $r(\boldsymbol{\epsilon}_1) = (\ln R(\boldsymbol{\epsilon}_1) - \ln C)/\ln\|\boldsymbol{\epsilon}_1\|$ is plotted against the displacement $\boldsymbol{\epsilon}_1$. The result is shown in Fig. 9.3. Here the constant C is obtained from the linear regression that was discussed above. It is interesting to note that for very small displacements the convergence rate diverges from the expected value. The author state that this behaviour is due to the discrete inconsistency of the shape derivative on Hadamard form, as discussed previously. Another explanation can be that the step length of 0.005 is close to the average cell size $\langle h \rangle$ of the two coarser meshes, see the last column in Table 9.2. This could introduce a numerical error for the smallest displacements.

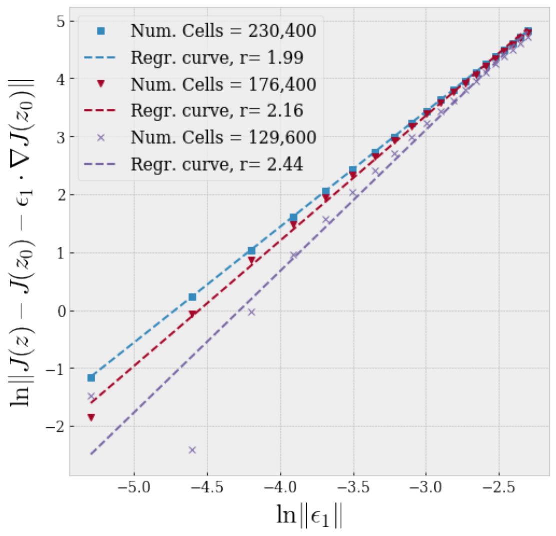

Figure 9.2: To obtain an estimate for the convergence rate from the first order Taylor test a linear regression is made between the logarithm of the first order Residual $\ln R(\boldsymbol{\epsilon}_1) = \ln\|J(\boldsymbol{z}) - J(\boldsymbol{z}_0) + \boldsymbol{\epsilon}_1 \cdot \nabla J(\boldsymbol{z}_0)\|$ and the logarithm of the displacement $\ln\|\boldsymbol{\epsilon}_1\|$. The procedure is performed for increasing mesh refinement.

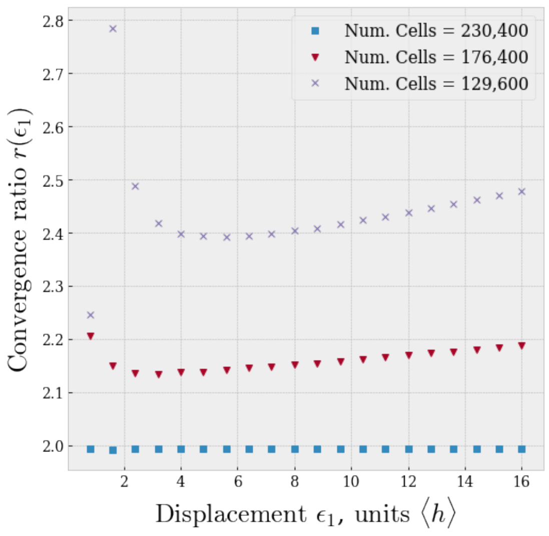

Figure 9.3: The convergence rate r in a first order Taylor expansion of the objective, $J(\boldsymbol{z}) = J(\boldsymbol{z}_0) + \boldsymbol{\epsilon_1} \cdot \nabla J(\boldsymbol{z}_0) + C\|\boldsymbol{\epsilon_1}\|^r$, is investigated. Here z_0 corresponds to the point where the gradient is evaluated. The displacement from this point is then given by $\boldsymbol{\epsilon_1} = \boldsymbol{z} - \boldsymbol{z}_0$ and measured in units of average mesh size h. The gradient ∇J is calculated according to Eq. (8.7).

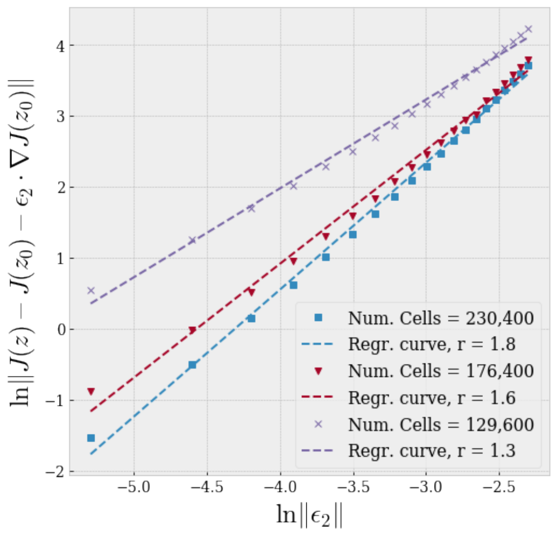

Figure 9.4: An estimate of the convergence rate is obtained in the same way as in Fig. 9.2 but for a different displacement direction $\boldsymbol{\epsilon_2}$.

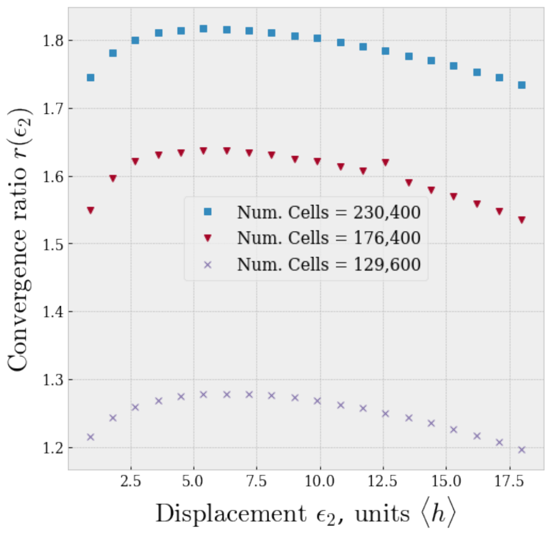

Figure 9.5: A Taylor test similar to that described in Fig. 9.3 is performed in a different displacement direction $\boldsymbol{\epsilon}_2$. The displacement direction is measured in units of average mesh size h.

A similar Taylor test was performed in a different direction $\boldsymbol{\epsilon}_2$ and the results are presented in Fig. 9.4 and Fig. 9.5. The convergence rates from this test are summarised in the third column in Table 9.2. Also here the expected convergence rate of $r = 2$ is reached when refining the mesh.

Shape derivative at extrema when refining the mesh: To check the precision in sign change for the calculated gradient, the local maximas in two different test setups were considered. Both test setups used 1 turbine with angular orientation 90^o along with the geometrical setup defined in Section 6.2. The only difference between the setups was the inflow profiles.

Table 9.3: Table showing how the gradient evaluated at the maxima y^* is improving when refining the mesh. The correct gradient should be 0 at the extrema point. The gradient is measured as the decrease in J per units of average mesh size h. The third column gives the gradient relative to the objective function J. The value of J is given in the fourth column. The scan was performed in the y direction with a step length of α. This step length is given relative to the average mesh size in the fifth column.

Num. Cells bg.	$(\nabla J)(y^*)$	$(\nabla J/J)(y^*)$	$(J)(y^*)$	α/h	h
129,600	-7.2e-3 h^{-1}	-2.3e-5 h^{-1}	306.2	0.91	0.0055
160,000	1.1e-2 h^{-1}	3.9e-5 h^{-1}	306.2	1.0	0.005
211,600	1.5e-2 h^{-1}	4.8e-5 h^{-1}	306.2	1.16	0.0043
230.400	-2.3e-4 h^{-1}	-7.5e-7 h^{-1}	306.2	1.19	0.0042
250,000	-4.5e-5 h^{-1}	-1.4e-7 h^{-1}	306.2	1.25	0.004

For the first setup, the symmetric inflow profile $5\sin\pi y$ from Eq. (6.5) was used. In addition, the mesh was made symmetric over the computational domain. A local maxima was then located at $y^* = 0.5$ by sampling roughly one time for every cell for the scan along the y-direction. The setup is illustrated in Fig. 9.7, where the initial position and scan direction is indicated. By keeping the x-coordinate fixed and scanning $J(y-y^*)$ and $\nabla J(y-y^*)$ over roughly $y = y^* \pm 16$ cells, the drag function and corresponding gradient values were mapped out. To see how the gradient depends on refining the mesh a series of meshes, with mesh sizes ranging from $129,600-250,000$ cells, were considered. The results are presented in Table 9.3. It is clear that there is a gain in precision from refining the mesh. The step length in these scans were $\alpha = 0.005$ and the step length relative to the cell sizes α/h are summarised in the second to last column in Table 9.3.

For the system with $230,400$ cells the drag profile is shown in Fig. 9.6. In the figure the horizontal axis corresponds to displacement from maxima y^* in units of average cell size h. The gradient is illustrated by the five near-tangential lines on the graph of J. Ideally these lines should be tangential to the graph but for the two lines further away from $y^* = 0$ one can visually see a small offset. At the maxima y^* the gradient is expected to be 0, but from the sampling it is found to be $\nabla J(y^*) = -4.5\text{e-}5\ \text{h}^{-1}$ which is slightly off. This value must be seen in relation to the function value. The normalised gradient is $\nabla J(y^*)/J(y^*) = -1.4\text{e-}7\ \text{h}^{-1}$, which is acceptably close to zero. For coarser meshes, $< 160,000$ cells, the gradient was observed to be more unpredictable.

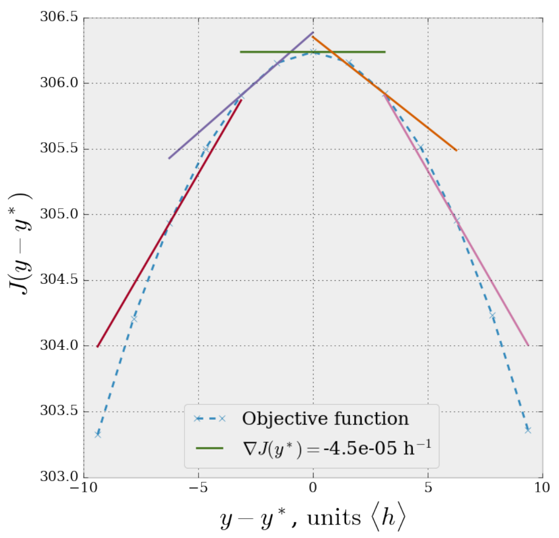

Figure 9.6: Figure illustrating the gradient sampled at different points along the objective function J. The gradient evaluated at a point y gives the decrease in J per unit cell displaced from this point. The displacement $y-y^*$ from the maxima is measured in average mesh size h. Of particular interest is the gradient at the maxima y^*, which corresponds to the point $y-y^*=0$ on the horizontal axis. Ideally the gradient should be zero at this point.

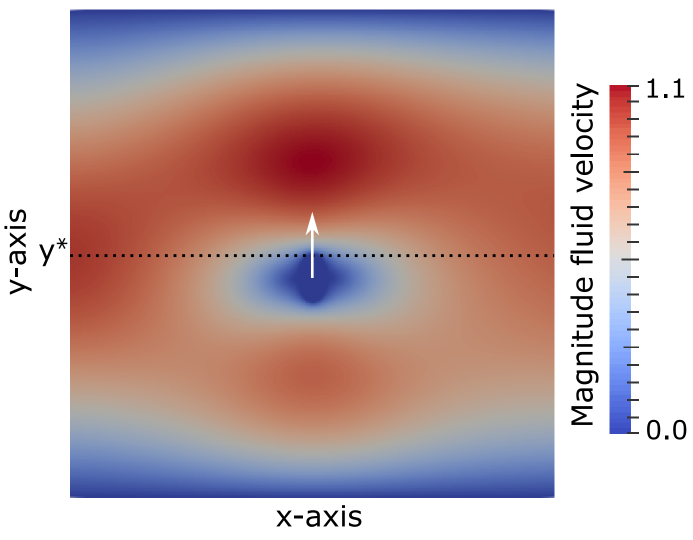

Figure 9.7: Figure illustrating how the turbine is moved along the y-axis to map out the maxima at y^* in the objective function J.

It still performed well around the maxima, but in some cases could be up to 30^o off from tangential to the graph of J.

An error source in these types of tests is the finite step length $\alpha = 0.005$. If the exact maximum is missed due to the resolution of the scans then it could lead to a gradient with an offset from zero at the assumed maxima. The motivation for choosing a symmetric setup can now readily be seen. With a symmetric geometry of the domain and background mesh along with a symmetric inflow profile $5\sin \pi y$, the maxima can be expected to be located at $y^* = 0.5$. This corresponds to the middle of the domain along the vertical axis. This edge is indicated on Fig. 9.7 by the dotted black line. This must be so, since if the maxima is to either side of the middle, then it will interfere with the symmetry of the setup. With this arrangement it was therefore made certain that the point $y = 0.5$ was sampled and this should reduce the error related to the finite step sizes. It must be pointed out that since the turbine mesh was not exactly symmetrical then the exact maxima could still be slightly off from $y = 0.5$.

To test the performance for a wider set of situations, a similar experiment was conducted for a stronger and more unison inflow profile. In addition, a different scan direction with displacement

Table 9.4: Table showing how the gradient evaluated at the maxima x^* is improving when refining the mesh. The correct gradient should be 0 at the extrema point. The gradient is measured as the decrease in J per units of average mesh size h. The third column gives the gradient relative to the objective function J. The value of J is given in the fourth column. The scan was performed along the x-axis with a step length of α. This step length is given relative to the average mesh size in the fifth column.

Num. Cells bg.	$(\nabla J)(x^*)$	$(\nabla J/J)(x^*)$	$(J)(x^*)$	α/h	h
129,600	-0.22 h^{-1}	-4.3e-5 h^{-1}	5,107	0.91	0.0055
160,000	-0.28 h^{-1}	-5.4e-5 h^{-1}	5,152	1.0	0.005
211,600	-0.13 h^{-1}	-2.4e-5 h^{-1}	5,213	1.16	0.0043
230.400	-0.02 h^{-1}	-3.8e-6 h^{-1}	5,231	1.19	0.0042
250,000	+0.07 h^{-1}	+1.3e-5 h^{-1}	5,248	1.25	0.004

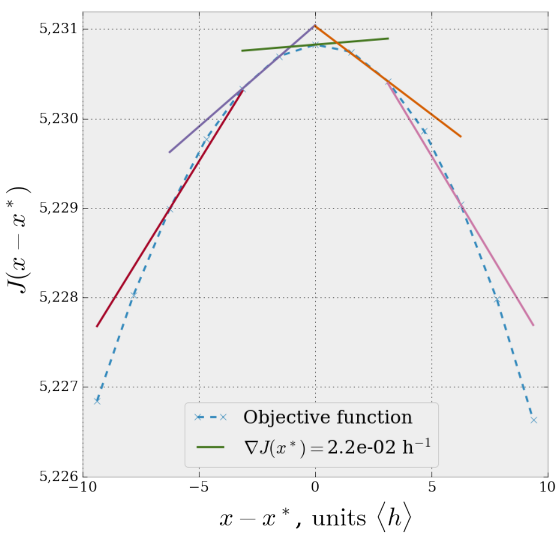

Figure 9.8: Figure illustrating the gradient sampled at different points along the objective function J. The gradient evaluated at point a x gives the decrease in J per unit cell displaced from this point. The displacement $x - x^*$ from the maxima is measured in average mesh size h. Of particular interest is the gradient at the maxima x^*, which corresponds to the point $x - x^* = 0$ on the horizontal axis. Ideally the gradient should be zero at this point.

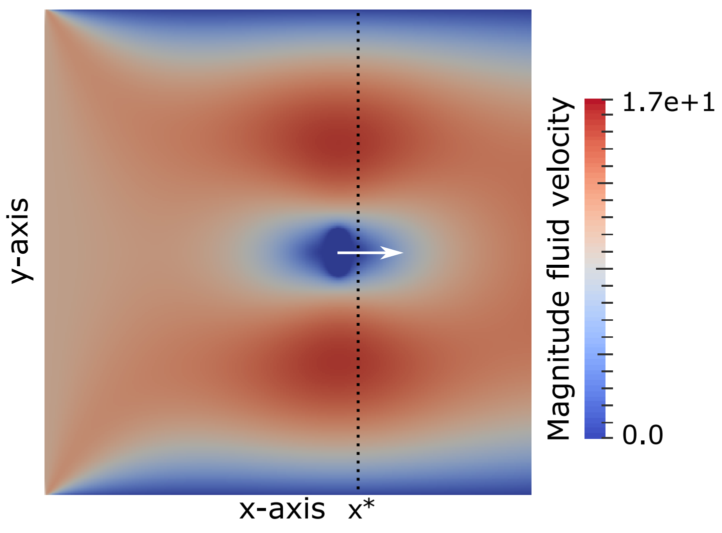

Figure 9.9: Figure illustrating how the turbine is moved along the x-axis to map out the maxima at x^* in the objective function J.

along the x-axis instead of the y-axis was used. With this setup a maxima along the x-axis was mapped out around $(x^*, y^*) = (0.63, 0.5)$ by sampling roughly one time for every cell when scanning along the x-direction. The step length α relative to the mesh size h is given in the fifth column in Table 9.4. The situation is illustrated in Fig. 9.9 and the result from the scan with $230,400$ cells in the background is shown in Fig. 9.8. The behaviour of the gradient is similar to that found in Fig. 9.6 for the y-scan, but with slightly larger offsets from 0 at the extrema point x^*. This scan did not benefit from symmetry properties like the scan along the y-axis. It is therefore possible that the larger offset are due to the finite step length α which could have prevented the exact maxima to be sampled. In general, the sampled gradients are also here not exactly tangential, as seen from Fig. 9.8, but they have the right sign and the precision in the sign change is also here, acceptable. The mesh dependence of ∇J, for the scan along the x-axis is presented in Table 9.4. Also here it is clear that the precision of the gradient is improving when refining the mesh.

9.2 Performance of the multimesh shape optimisation scheme

This section will begin by evaluating the time consumption related to geometry changes when using the multimesh scheme. This analysis is presented in Section 9.2.1. The next section, Section 9.2.2, will then look at the time consumption related to gradient evaluations. Section 9.2.3 then concludes the discussion by considering a benchmark consisting of tidal turbines in Stokes flow. With this benchmark the goal is to test the performance of the combined multimesh shape-optimisation scheme with a realistic optimisation problem.

9.2.1 Performance of the multimesh formulation in terms of mesh update

Along with the implementation and convergence tests performed in Chapter 7, this section will cover the numerical analysis aspects of Objective 1. As previously mentioned, one of main motivations for the multimesh formulation is to reduce the computational cost related to re-meshing changing geometries. The multimesh formalism achieves this by reducing the size of the domain that needs to be re-meshed. Previous work [10] has compared the performance of the multimesh FEM formulation against the traditional FEM method, where a physical system of 3 multi-cables was optimised. The intention here has been to compare mesh-update/mesh-build times to assembly/solve times and to compare the model developed here with the results presented in [10]. Since the state equation used here is different from that in [10], it is not expected to see the same performance for operations involving the linear system. However, the mesh update and mesh build are only related to the multimesh formulation and these operations can therefore be compared.

Five numerical experiments have been performed for $2, 4, 6, 8$ and 10 turbines, where the average times of relevant operations were timed. The system configuration from Section 6.2.3

Table 9.5: Table summarising the computational times related to domain deformation in the multimesh scheme. The time was measured for an increasing number of turbines. The "mesh update" and "mesh build" times are specific for the multimesh formulation. "Mesh update" corresponds to the time to move the mesh coordinates of the perturbed sub-mesh to their new position at iteration $k+1$. "Mesh build" measures the time to calculate the cut, uncut and covered cells and prepare the multimesh for integration. The assembly time measures the time to assemble the linear system and the time to solve this system is given by the "Solve time". The average time of an optimisation iteration is given in the last column.

# Turbines	Cells	Assembly	Solve	Mesh build	Mesh update	Opt. iter.
2	20,156	2,933 ms	1,322 ms	47 ms	0.76 ms	8,933 ms
4	21,264	5,310 ms	1,341 ms	85 ms	1.51 ms	13,460 ms
6	22,380	7,702 ms	1,430 ms	128 ms	2.26 ms	20,845 ms
8	23,492	11,319 ms	1,510 ms	183 ms	3.06 ms	28,782 ms
10	24,602	12,892 ms	1,626 ms	216 ms	3.73 ms	36,897 ms

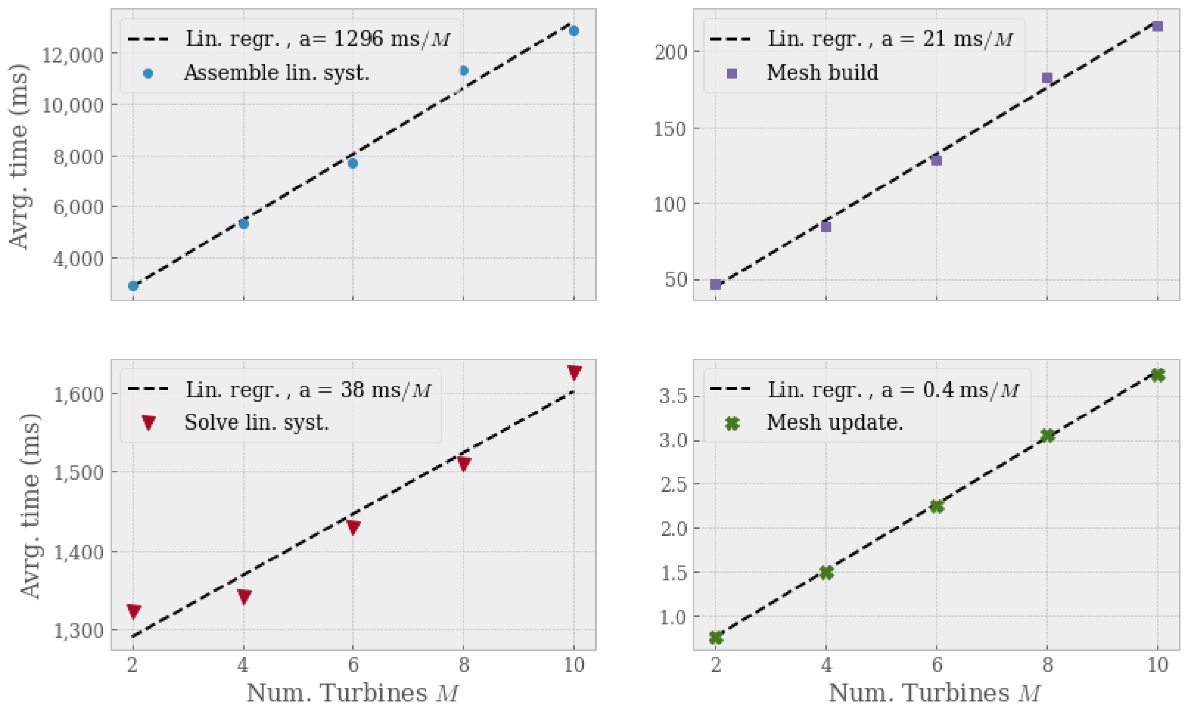

Figure 9.10: Comparison of the average time consumption of different operations related to a changing geometry. An explanation to the operations was given in Table 9.5.

was used in all runs with the symmetric inflow profile $5\sin\pi y$ from Eq. (6.5) and the angular orientation $\theta_j = 30$ for all turbines. Similar to [10] the linear systems were solved using direct LU decomposition. The timing results are presented in Table 9.5. An explanation to the different operations listed in the table is in order. The "Assembly time" corresponds to the time to assemble the linear system and the time to solve this system is denoted "Solve time". The "Mesh build" corresponds to the time it takes to calculate the cut, uncut and covered cells and prepare the multimesh for integration, see [24] for a more comprehensive discussion on this. In essence this means the time to "glue" the background mesh to the sub-meshes. The "Mesh update" is simply the time to move the mesh coordinates of the perturbed sub-meshes to their new position at iteration $k+1$. The combined time of these two operations corresponds to the re-mesh time of traditional FEM. The average time of an optimisation iteration is shown in the last column in Table 9.5. It should be noted that the number of cells, listed in the second column, are slightly increasing with the number of turbines since the mesh density in the turbines is slightly higher than for the background.

From Table 9.5 it can be seen that the mesh build and mesh update times are significantly lower than the assembly and solve times. This demonstrates the potential of the multimesh method, as the operations related to re-meshing are among the most computationally demanding in traditional FEM schemes. In [10] a system of 3 sub-meshes and $22,775$ cells were used and the average times found for mesh update and mesh build were 0.94 ms and 222 ms respectively. These times are very close to those found in this thesis. In [10] it was shown that the combined time for mesh update and mesh build was ~ 46 times faster than for the corresponding re-mesh operations in traditional FEM. A confirmation of these low times in this thesis therefore gives further evidence for the proficiency of the multimesh FEM method, in terms of cost saving with regards to re-meshing.

It is of interest to see how the times for the different operations, depends on the complexity of the mesh. In this regard the number of turbines M, that are moved in each iteration, will correspond to the mesh complexity. A linear dependence is found for all operations as shown in Fig. 9.10. The slopes for mesh build and mesh update are significantly lower than that of assembly with respectively $a = 0.4$ ms/M and $a = 21$ ms/M versus $a = 1296$ ms/M. It follows that both mesh build and mesh update becomes more and more negligible as the complexity of the mesh is increased, which confirms the proficiency of the multimesh method in terms of cost savings related to re-meshing. On the other hand, for the multimesh scheme, several stabilisation terms were introduced to ensure the "gluing" of the upper sub-meshes to the background domain, see Section 3.4.2. When the number of sub-meshes increases, these interface terms will make the size of the linear system increase faster than for traditional FEM and thereby increase the assembly and solve times. For future work, a comparison to traditional FEM could therefore be conducted with focus on this particular issue.

9.2.2 Time consumption of gradient evaluations

For the Stokes-drag problem from Definition 1.2, the Hadamard formulation takes a particularly favorable form when using the adjoint approach in the derivation. As commented in Remark 5.5 this is due to the self adjoint nature of the Stokes-drag problem, which makes the expression from Proposition 5.2 independent of the adjoint variables. Due to this property it is not necessary to solve the adjoint system for each gradient evaluation. The time consumption related to solving the adjoint equation is comparable to the assembly/solve times for the forward equation. The independence from the adjoint variables therefore offers a large computational saving compared to systems that are not self adjoint. What is more, the current gradient representation is extremely fast compared to for instance a finite difference approach where it is necessary to assemble and solve the linear system $N+1$ times for a system parameterised by N design parameters. The gradient evaluations using the expression derived in Proposition 5.2 are therefore expected to be very computationally cheap compared to alternative approaches that does not utilise the adjoint approach.

To investigate the time consumption related to gradient evaluations, two test setups with 2 and 10 turbines were considered. For the tests, the domain setup was chosen as described in Section 6.2 with the inflow profile $5\sin\pi y$. The results are presented in the left graph in Fig. 9.11 and for comparison the combined assembly/solve times are presented in the right graph in the figure. The assembly/solve operations were defined in Section 9.2.1.

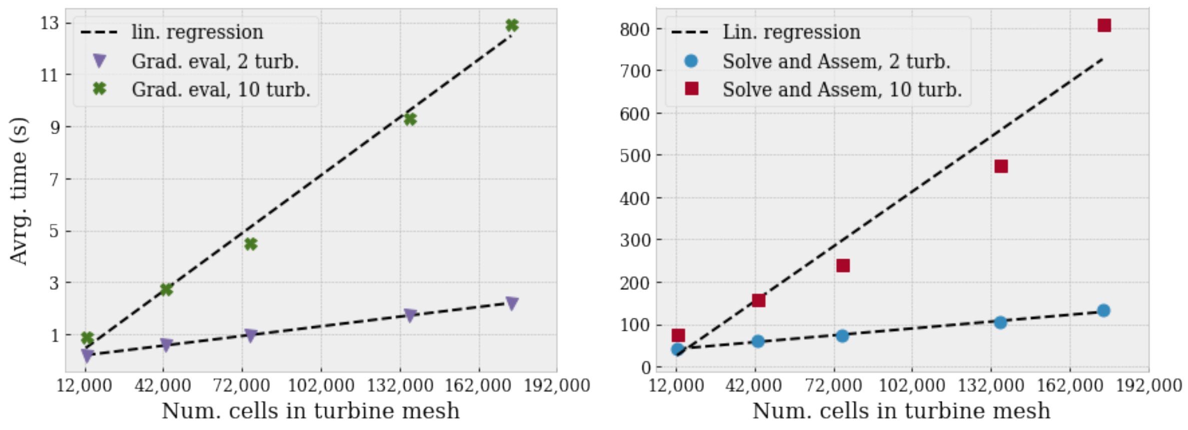

Figure 9.11: In the left graph the average time consumption of gradient evaluations is plotted against the numbers of cells in the turbine mesh. For comparison the average time required to assemble and solve the linear system is plotted in the right graph. The regression curves are shown with black dotted lines.

It is evident from Fig. 9.11 that the time spent on gradient evaluations becomes negligible when the mesh is refined. For the system with 10 turbines, the combined assembly/solve time is found to increase ~ 58 times faster when refining the mesh than the time spent on gradient evaluations. Based on these findings it is clear that the Hadamard formulation offers very time efficient gradient evaluations when doing shape optimisation involving the Stokes-drag problem. It must be noted that this result is limited to this particular case, where the final expression is independent of the adjoint variables. For systems that are not self adjoint, this must be expected to not hold true.

9.2.3 Benchmark for the combined multimesh optimisation scheme

The performance of the merged optimisation and multimesh scheme has been tested by optimising the positions of $2,4,6,8,10,12,14$ turbines in Stokes flow. The setup for this test is chosen as described in Section 6.2.3 with the inflow profile $y^2 \sin \pi y$ from Eq. (6.5). Note that the allowed region for the turbines is restricted according to Eq. (6.3), in order to prevent out of domain movement. The restriction also helps to create a more realistic model with a tidal turbine farm in a much larger ocean. The domain restriction on the center positions of the turbines is illustrated by the white dotted square in Fig. 9.14. In the experiment, each turbine had $75,000$ cells. For the background domain a setup with $230,400$ cells and an average mesh size of $h = 0.0041$ was chosen. The stopping criterion for termination was that the relative change in functional value should be less than 5e-4 over 10 consecutive optimisation iterations as stated in Eq 9.1,

$$\frac{\|J_k - J_{k-j}\|}{\|J_k\|} < 5\text{e-4} \quad \text{for } j = 1,\ldots,10, \tag{9.1}$$

where k is the iteration number. In order to maximise the drag functional J in a minimisation scheme the negative $(-J)$ of the functional was considered.

Table 9.6: Table summarising the results from the optimisation runs for systems of $2-14$ turbines. The second column shows the initial functional value. The final value, reached after termination of the optimisation algorithm, is shown in the third column. The number of function evaluations is shown in the fourth column. In the fifth column the number of optimisation iterations is presented, while the last column summarises the time spent before termination, measured in hours.

# Turbines	$(-J)$ initial	$(-J)$ final	Num. func eval.	Num. iter.	Time
2	-20.5	-134.6	38	23	0.5 hour
4	-24.1	-231.1	75	53	1.31 hour
6	-41.6	-318.4	70	44	1.57 hour
8	-56.3	-384.4	133	66	3.56 hour
10	-146.3	-454.3	47	34	1.42 hour
12	-146.8	-532.4	133	49	4.69 hour
14	-190.5	-582.4	37	35	1.52 hour

Progression plots: The progression plots for the 7 turbine systems are shown in Fig. 9.12 where $(-J)$ is plotted against the number of optimisation iterations. The results are summarised in Table 9.6. where also the number of function evaluations and total optimisation-time are presented. The larger amount of function evaluations compared to the optimisation iterations are due to the linesearch algorithm, which relies on several backtracking steps before an acceptable step length is chosen. From Fig. 9.12 it is apparent that the optimisation algorithm makes fast progress in the first $20-30$ iterations before flatting out. In fact, forcing the algorithm to stop after only 20 iterations gave very similar results for the final geometry. This behaviour was the motivation for the stopping criteria in Eq. (9.1). The algorithm successfully terminated by the standard tolerances, see [8], for systems up to 6 turbines. After this it was found that the algorithm had problems to terminate reliably and rather made incremental updates of the turbine positions without much gain. It is possible that this behaviour is due to an incorrect gradient close to the optimum. With an inaccurate gradient the algorithm may have problems finding an acceptable step length and therefore several backtracking steps are needed. The stopping criteria was therefore necessary for the larger turbine systems and was used also for the smaller systems for consistency.

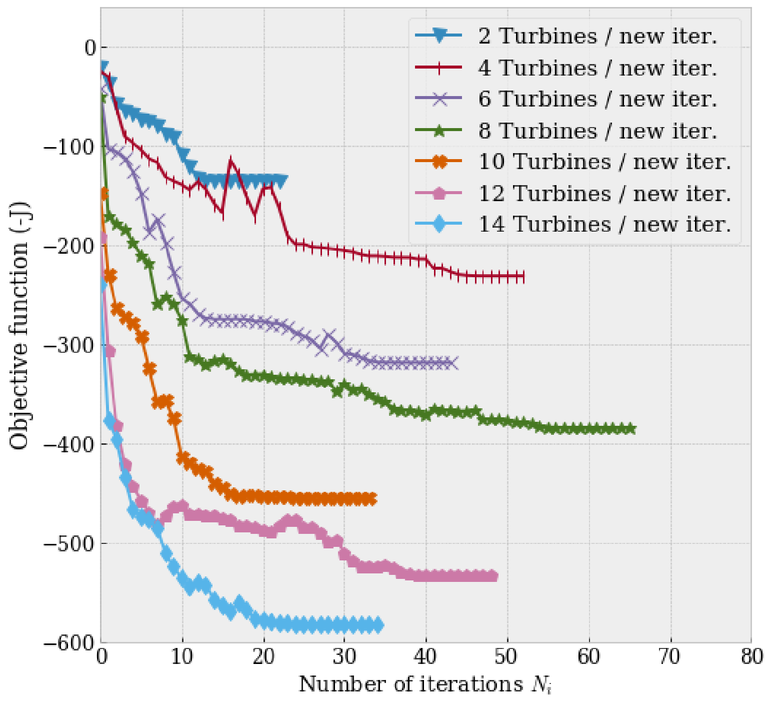

Figure 9.12: Figure showing progression plots for the optimisation of the drag functional $(-J)$, for systems of $2-14$ turbines. The decrease in the objective function is measured against the number of optimisation iterations N_i. The objective function is sampled at the beginning of a new iteration.

Optimal configuration of the tidal turbine systems: The final geometries for the turbine systems of 8, 10 and 14 turbines are shown in Fig. 9.14, 9.16 and 9.18 respectively. For comparison the initial configurations are shown in Fig. 9.13, 9.15 and 9.17. It is interesting to observe the barrier-zone structure that is building up as the number of turbines increase. The system of 10 turbines in Fig. 9.16 shows an upstream barrier aligned with the inflow profile and a wall of turbines at the outflow region. Between the two barriers an open blue region for free flow is present. The same barrier-zone structure was observed in [25] where a "turbine density field" was used to optimise the positions. Similar results were also reported in a related paper [15],

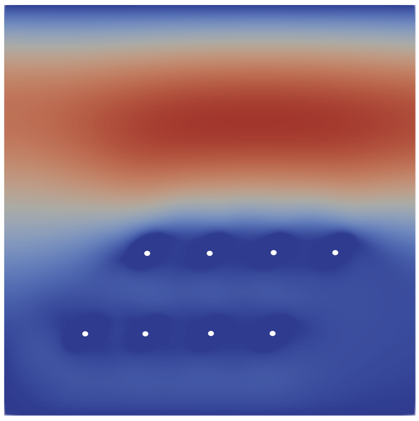

Figure 9.13: Figure showing the initial configuration of a system of 8 turbines. The dark-blue uniform ellipses with a white dot in the middle, corresponds to the turbines. The inflow region is the left side of the figure and the outflow region is the right side. The inflow profile that is used is the $y^2 \sin \pi y$ function from Eq. (6.5). The upper and lower edges of the figure are wall boundaries with a zero velocity conditions.

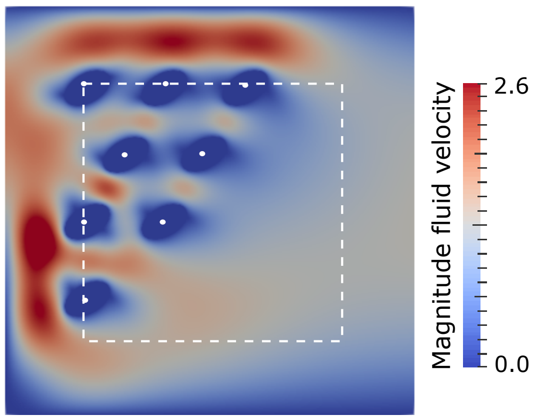

Figure 9.14: Final configuration of the turbine system with 8 turbines. See also Fig. 9.13 for the initial configuration. The turbines have arranged around the inflow region on the left side. It is evident that the turbines orient towards the inflow profile which is $y^2 \sin \pi y$ as defined in Eq. (6.5). The white dotted square illustrates the restricted area for the turbines.

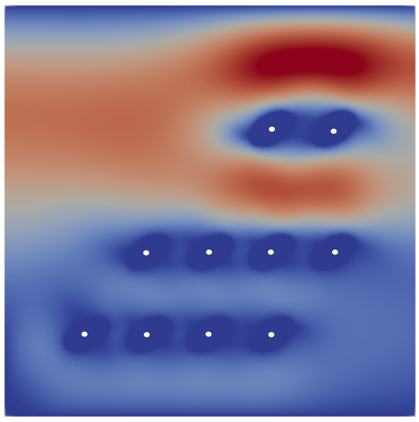

Figure 9.15: Figure showing the initial configuration of a system of 10 turbines. The dark-blue uniform ellipses with a white dot in the middle, corresponds to the turbines. The inflow region is the left side of the figure and the outflow region is the right side. The inflow profile that is used is the $y^2 \sin \pi y$ function from Eq. (6.5). The upper and lower edges of the figure are wall boundaries with a zero velocity condition.

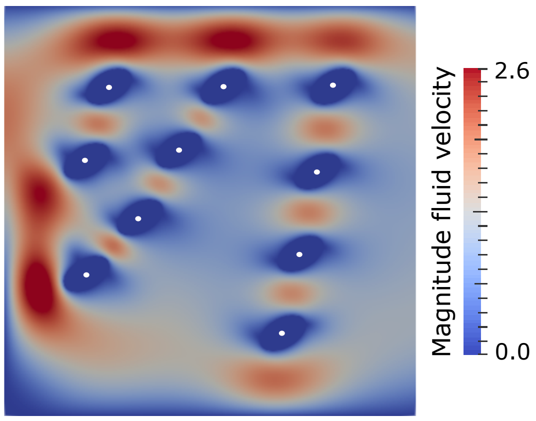

Figure 9.16: Final configuration of the turbine system with 10 turbines. See also Fig. 9.15 for the initial configuration. The turbines have arranged around the inflow region on the left side and with a barrier down stream on the right side to block the remaining flow.

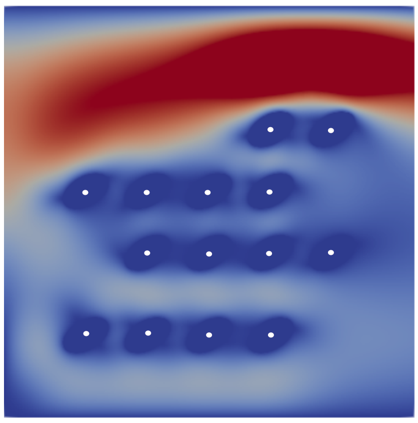

Figure 9.17: Figure showing the initial configuration of a system of 14 turbines. The dark-blue uniform ellipses with a white dot in the middle, corresponds to the turbines. The inflow region is on the left side of the figure and the outflow region is at the right side. The inflow profile that is used is the $y^2 \sin \pi y$ function from Eq. (6.5). The upper and lower edges of the figure are wall boundaries with a zero velocity condition.

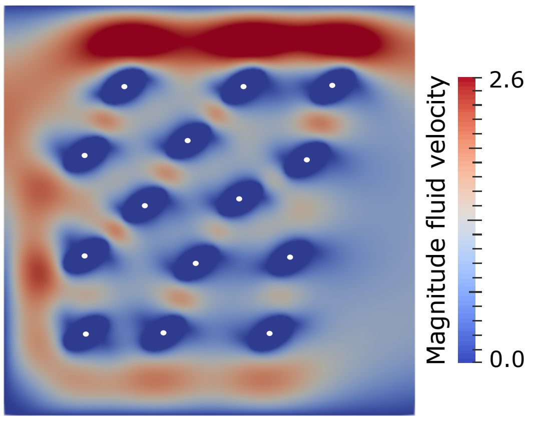

Figure 9.18: Final configuration of the turbine system with 14 turbines. See also Fig. 9.17 for the initial configuration. The turbines have also here oriented towards the inflow region on the left side. A separate downstream barrier, like that observed for 10 turbines, is not observed.

which relied on gradient-based optimisation and the adjoint approach. The system of 12 turbines demonstrated a similar structure as that for 10 turbines with one turbine added to each of the two barriers. For the system of 14 turbines, in Fig. 9.18, the upstream and downstream barriers combine to one since the available space starts to become limited. For ≤ 8 turbines the downstream barrier was not present, as shown in Fig. 9.14 for 8 turbines. It is believed that this is because the inflow barrier was still building up and that this barrier is prioritised before the downstream barrier at the outflow. These observations shows that the optimisation scheme employed here, successfully reproduce results from the literature which gives testimony to the proficiency of the combined shape optimisation multimesh scheme.

Dependence on design parameters: A goal when designing optimisation algorithms is to obtain independence from the number of design parameters with regards to the number of optimisation iterations needed to solve for an optimal configuration. In this context it is interesting to compare the number of iterations for each of the 7 turbine systems listed in Table 9.6. In Fig. 9.19 the number of iterations and function evaluations are plotted as a function of the number of turbines in the system. There is no clear indication of an increase in iterations with increasing number of parameters. In fact, the trend seem to be that the number of iterations and function evaluations are fluctuating around some mean. The medians when comparing the 7 series in Table 9.6 are 44 iterations for the optimisation iterations and 70 iterations for the function evaluations. The two notable exceptions from this behaviour are the systems of 8 and 12 turbines which required significantly more function evaluations to terminate.

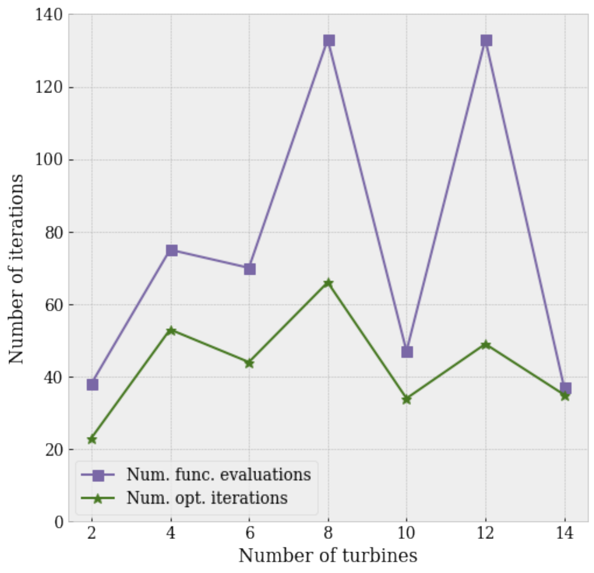

Figure 9.19: Figure showing how the performance of the optimisation algorithm depends on the number of design parameters. The green line with star markers shows the number of optimisation iterations and the purple line with square markers shows the number of function evaluations.

The most probable cause for this behaviour is an inaccurate gradient which, as mentioned earlier, could result in more backtracking steps. One reason that this effect is particularly prominent for the systems of 8 and 12 turbines, can be that an inaccurate gradient is combined with an unfortunate optimal configuration close to violating some domain or intersection constraints.

Based on these observations, it seems that the performance of the algorithm is fairly independent of the number of design parameters. At the same time the termination criteria is found to be rather sensitive to other effects, such as inaccuracies in the gradient evaluations.

CHAPTER 10

CONCLUSION AND OUTLINE FOR FUTURE WORK

This chapter will give a summary of the work conducted in this thesis and highlight the main results. The chapter concludes by pointing out yet unanswered questions and by giving a formal outline for future work.

10.1 Summary and conclusion

The central concern of this work has been to investigate the proficiency of the multimesh scheme and the Hadamard form, for use in shape optimisation. In this context the Hadamard form of the shape derivative has been derived in a multimesh setting. To test the resulting analytical expression of the shape derivative in combination with the multimesh formalism, a numerical model with layout optimisation of tidal turbines has been employed.

The Hadamard formulation of the shape derivative has been derived for the drag functional subject to the multi-domain Stokes problem from Definition 3.9. The derivation relied on established methodology utilising the "continuous approach", with corollaries from the Hadamard Theorem [46], along with the adjoint method [20]. The novelty in the derivation performed here, has been the derivation of the Hadamard representation of the shape derivative when also taking into account the interface conditions from the multi-domain partitioning. The strong form of the interface conditions has been used. The final expression from this treatment, Proposition 5.2, was found to be equivalent to that of a single-domain analysis. This result shows that by enforcing the the strong interface conditions from Definition 3.9, then the introduction of an interface between the independent meshes does not affect the physical situation, or at least not the physics that are contained in the drag functional. This observation gives evidence that the interface is purely artificial. The conclusion that can be drawn from this is therefore that the multi-domain Stokes problem from Definition 3.9 is up to this point consistent with the physics it aims to describe.

Using a tidal turbine layout problem as a test scenario, the correctness of the shape derivative derived in Proposition 5.2 has been tested. Taylor tests in multiple directions have been conducted and shown to give the expected convergence rate when the mesh is sufficiently refined. The gradient has been found to have an offset from the tangential direction and refining the mesh was shown to give improvements but not completely remove the offset. More importantly, the gradient has been shown to be close to the expected value of 0 at extrema points, with values around $10^{-5} - 10^{-7}h^{-1}$ relative to the objective function for mesh sizes $h \leq 0.0042$. The precision was found to be improved by refining the mesh. These findings, along with the results presented in the previous paragraph, answer the second objective made in the introduction.

In the simulation part of the work, the multimesh finite element method has been used to solve the Stokes problem in a multimesh setting. The implementation was based on the multimesh variational form presented in [22]. Using a benchmark model, with a known analytical solution, the implementation has been shown to give optimal convergence. The performance of the method was demonstrated in terms of translation of turbines in a tidal turbine farm. The average time consumption of several operations related to the mesh update were sampled. These operations can be divided into two categories, mesh update due to the changing geometry and solving the linear system. This work has shown that in the multimesh scheme, when the number of turbines M is increased, the operations related to mesh update becomes negligible compared to the time required to assemble and solve the linear system. With $\sim 22,000$ cells in the mesh, the linear increase was found to be 21.4 ms/M for the mesh related operations. At the same time the linear increase for assembly and solve was 1334 ms/M. The average times of the mesh related operations were found to be in the same order of magnitude as those found in a related survey [10]. These findings correspond to the first objective of this work, as stated in the introduction.

To conclude the analysis, several optimisation runs were conducted with turbine systems consisting of $2-14$ turbines. The optimisation problem was to maximise the energy dissipation into heat with respect to the relative positions of the turbines in Stokes flow. The final configurations were compared to related work [15, 25] and was found to show similar behavior. In particular, the turbines were found to arrange in barrier-zones at the inflow and outflow regions. The inflow barrier was found to orient towards the region with the strongest inflow, while the outflow barrier served the purpose of blocking as much as possible of the remaining flow. The barrier zones were therefore reasonable from a physical point of view.

With basis in the results presented above, it is clear that the discrete inconsistency of the Hadamard formulation and the error introduced by assuming strongly enforced interface conditions, gives a limited impact on the performance of the gradient. Therefore, the combination of the multimesh scheme and the Hadamard representation of the shape derivative offers a reliable framework for shape optimisation. As such, the cost efficiency demonstrated for the multimesh scheme and for the Hadamard formulation in general, makes the combined framework attractive when solving problems where only a subset of the computational domain require remeshing. These results answers the third and final objective stated in the introduction.

10.2 Outline for future work

The combination of shape optimisation with the multimesh finite element method, is a relatively new idea which has shown promising results. Future work should aim to investigate this combined scheme further. In the following is a brief summary of several interesting aspects that could be studied;

- Apply the developed multimesh shape optimisation scheme to relevant applications like those listed in Section 1.3.1.
- Develop and implement a multimesh variational form of the adjoint equations from Lemma 5.6.
- Develop a multimesh formulation of the Navier-Stokes equations.
- Derive the Hadamard formulation of the shape derivative for the drag functional constrained by the Navier-Stokes equations.
- As mentioned in Remark 5.4, several correction terms where eliminated from the final expression of the shape derivative in Proposition 5.2 by the assumption of strong interface conditions. Since the interface conditions are only weakly enforced in the actual variational form, future work should investigate if the inclusion of these terms will help remedy the termination problems mentioned in Section 9.2.3.

Part IV

Appendix

APPENDIX A

NOTATION AND DERIVATION OF NECESSARY VECTOR IDENTITIES

A.1 Notation

Some linear algebra notation will be used and it is therefore necessary to define the conventions used. In particular this will be necessary for the derivations in appendix A.2. For this thesis, all vectors $\boldsymbol{v} \in \mathbb{R}^2$ will be represented as a column vector and the transpose of a vector will be represented as a row vector. Namely,

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \qquad \boldsymbol{v}^T = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \tag{A.1}$$

For the inner product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$ the following conventions will be used,

$$\boldsymbol{v} \cdot \boldsymbol{u} = \boldsymbol{v}^T \boldsymbol{u} = (\boldsymbol{v}, \boldsymbol{u}) \tag{A.2}$$

Similarly, the outer product is defined as,

$$\boldsymbol{v}\boldsymbol{u}^T \tag{A.3}$$

A.1.1 Differential operators

For the gradient ∇ and Jacobian D one have,

$$\nabla = \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix}, \qquad \nabla \boldsymbol{v}^T = \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix} \tag{A.4}$$

and

$$D\boldsymbol{v} = (\nabla \boldsymbol{v}^T)^T \tag{A.5}$$

The transpose notation on the vector $\boldsymbol{v}$ in $\nabla \boldsymbol{v}^T$ will be dropped to simplify the notation. The Hessian is the jacobian of the gradient and will be denoted $D^2 \boldsymbol{f}$ where,

$$D^2\boldsymbol{v} = D\nabla\boldsymbol{v} = \nabla^T\nabla\boldsymbol{v} \tag{A.6}$$

For a vector function $v : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ the Hessian will be a third rank tensor and can be seen as a vector of two elements, where the elements are Hessians of the elements of $\boldsymbol{v}$ namely,

$$D^2\boldsymbol{v} = \begin{bmatrix} D^2 v_1 & D^2 v_2 \end{bmatrix}. \tag{A.7}$$

A.1.2 Index notation

When natural, index notation will be used. In this context, summation over repeated indices will be implied if a summation sign is not present. This means,

$$\sum_{i=1}^{2} v_i u_i = v_i u_i \tag{A.8}$$

In index notation the gradient will be represented as

$$\nabla\boldsymbol{v} = \sum_{i,j=1}^{n} \frac{\partial u_i}{\partial x_j} \tag{A.9}$$

A.2 Vector identities and necessary definitions

A.2.1 Definitions

All Definitions stated in this section are adapted from [41].

Definition A.1. Two dimensional vector function: Let $\Omega \subset \mathbb{R}^2$ be a sub-domain in two dimensions. A two dimensional vector function is a mapping such that $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$. For this work it will be assumed that the vector functions are all differentiable.

Definition A.2. Normal derivative: Let $\Omega \subset \mathbb{R}^2$ and let $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$ be a two dimensional vector function defined over the domain Ω. Let $\boldsymbol{n} : \Gamma \longrightarrow \mathbb{R}^2$ be the unit normal of this domain, where Γ is the boundary. Under the assumption that $\boldsymbol{u}$ is defined in a neighbourhood of Ω such that the normal derivative exists, the normal derivative of $\boldsymbol{u}$ can be expressed as,

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = D\boldsymbol{u} \cdot \boldsymbol{n} = D_n \boldsymbol{u}.$$

where D is the Jacobian matrix from Eq. (A.5).

Definition A.3. Tangential derivative: Let $\Omega \subset \mathbb{R}^2$ and let $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$ be a two dimensional vector function defined over the domain Ω. Let $\boldsymbol{\tau} : \Gamma \longrightarrow \mathbb{R}^2$ be the unit tangent of the boundary Γ. The tangential derivative of $\boldsymbol{u}$ can then be defined as,

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}} = D\boldsymbol{u} \cdot \boldsymbol{\tau} = D_\tau \boldsymbol{u}.$$

Definition A.4. Gradient expanded in an orthonormal basis: Let $\Omega \subset \mathbb{R}^2$ be a domain in two dimensions with boundary Γ. Let the set $\{\boldsymbol{n}, \boldsymbol{\tau}\}$ form an orthonormal basis in $\mathbb{R}^2$. The gradient $\nabla \boldsymbol{u}$ of a vector function $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$ can then be expressed in the basis as,

$$\nabla \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \boldsymbol{n}^T + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}} \boldsymbol{\tau}^T$$

Here $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ and $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}$ are the normal and tangential derivatives as defined in definition A.2 and A.3.

Definition A.5. Tangential divergence: Let $\Omega \subset \mathbb{R}^2$ be the domain over which the vector function $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$ is defined. The tangential divergence of $\boldsymbol{u}$ can then be defined as,

$$\mathrm{div}_\Gamma(\boldsymbol{u}) = \mathrm{div}(\boldsymbol{u}) - \boldsymbol{n}^T (D\boldsymbol{u})\boldsymbol{n} = \nabla \cdot \boldsymbol{u} - \boldsymbol{n} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}.$$

A.2.2 Identities

It is also necessary to derive the following vector identities,

Identity A.1. *For a vector $\boldsymbol{a}$ and a second order tensor B. The following identity holds,*

$$\nabla(\boldsymbol{a}^T B^T) = (D\boldsymbol{a})^T B^T + \boldsymbol{a}^T (DB)^T \tag{A.10}$$

Proof:

$$\begin{aligned}
\nabla(Ba)^T = \nabla(a^T B^T) &= \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \left(\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \right)^T \\
&= \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \begin{bmatrix} B_{11}a_1 + B_{12}a_2 & B_{21}a_1 + B_{22}a_2 \end{bmatrix} = \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \left(\begin{bmatrix} B_{11}a_1 & B_{21}a_1 \end{bmatrix} + \begin{bmatrix} B_{12}a_2 & B_{22}a_2 \end{bmatrix} \right) \\
&= \left(\begin{bmatrix} \partial_1 a_1 \\ \partial_2 a_1 \end{bmatrix} \begin{bmatrix} B_{11} & B_{21} \end{bmatrix} + a_1 \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \begin{bmatrix} B_{11} & B_{21} \end{bmatrix} \right) + \left(\begin{bmatrix} \partial_1 a_2 \\ \partial_2 a_2 \end{bmatrix} \begin{bmatrix} B_{12} & B_{22} \end{bmatrix} + a_2 \begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} \begin{bmatrix} B_{12} & B_{22} \end{bmatrix} \right) \\
&= \begin{bmatrix} \partial_1 a_1 & \partial_1 a_2 \\ \partial_2 a_1 & \partial_2 a_2 \end{bmatrix} \begin{bmatrix} B_{11} & B_{21} \\ B_{12} & B_{22} \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \partial_1 B_{11} & \partial_1 B_{21} \\ \partial_2 B_{11} & \partial_2 B_{21} \end{bmatrix} & \begin{bmatrix} \partial_1 B_{12} & \partial_1 B_{22} \\ \partial_2 B_{12} & \partial_2 B_{22} \end{bmatrix} \end{bmatrix}^T \\
&= \nabla a^T B^T + a^T \nabla B^T = (D\boldsymbol{a})^T B^T + \boldsymbol{a}^T (DB)^T.
\end{aligned}$$

Identity A.2. *For a vector $\boldsymbol{a}$ and a second order tensor B. The following identity holds,*

$$D(\boldsymbol{a}^T B^T) = BD\boldsymbol{a} + DB \cdot \boldsymbol{a}$$

Proof: *Follows from A.1 and $D(\boldsymbol{a}^T B^T) = (\nabla(\boldsymbol{a}^T B^T))^T$*

Identity A.3. *For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ the following identity holds,*

$$div\big(\boldsymbol{a}^T D\boldsymbol{b}\big) = \nabla\boldsymbol{a} : \nabla\boldsymbol{b} + div(\nabla\boldsymbol{b}) \cdot \boldsymbol{a}$$

Here the double-dot-product : is taken to mean summation over all indices such that $\boldsymbol{A} : \boldsymbol{B} = A_{ij}B_{ij}$, for second order tensors A and B.
Proof:

$$\begin{aligned}
div(\boldsymbol{a}^T D\boldsymbol{b}) &= div\left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \begin{bmatrix} \partial_1 b_1 & \partial_2 b_1 \\ \partial_1 b_2 & \partial_2 b_2 \end{bmatrix}\right) = div\left(\begin{bmatrix} a_1\partial_1 b_1 + a_2\partial_1 b_2 & a_1\partial_2 b_1 + a_2\partial_2 b_2 \end{bmatrix}\right) \\
&= \partial_1\Big(a_1\partial_1 b_1 + a_2\partial_1 b_2\Big) + \partial_2\Big(a_1\partial_2 b_2 + a_2\partial_2 b_2\Big) \\
&= (\partial_1 a_1)(\partial_1 b_1) + a_1\partial_1^2 b_1 + (\partial_1 a_2)(\partial_1 b_2) + a_2\partial_1^2 b_2 + (\partial_2 a_1)(\partial_2 b_1) + a_1\partial_2^2 b_1 + (\partial_2 a_2)(\partial_2 b_2) + a_2\partial_2^2 b_2 \\
&= (\partial_i b_j)(\partial_i b_j) + a_i\partial_j^2 b_i = \nabla\boldsymbol{a} : \nabla\boldsymbol{b} + div(\nabla\boldsymbol{b})
\end{aligned}$$

Identity A.4. *For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ the following relation holds,*

$$\nabla(\boldsymbol{a} \cdot \boldsymbol{b}) = \nabla\boldsymbol{a} \cdot \boldsymbol{b} + \nabla\boldsymbol{b} \cdot \boldsymbol{a}$$

Proof:

$$\nabla(\boldsymbol{a} \cdot \boldsymbol{b}) = \partial_k(a_i b_i) = (\partial_k a_i)b_i + (\partial_k b_i)a_i = \nabla\boldsymbol{a} \cdot \boldsymbol{b} + \nabla\boldsymbol{b} \cdot \boldsymbol{a}$$

Identity A.5. *For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ the following relation holds,*

$$\boldsymbol{n}^T \nabla\boldsymbol{a}(\nabla\boldsymbol{b})^T \boldsymbol{n} = \boldsymbol{n}^T \nabla\boldsymbol{b}(\nabla\boldsymbol{a})^T \boldsymbol{n}$$

Proof:

$$\begin{aligned}
\boldsymbol{n}^T \nabla\boldsymbol{a}(\nabla\boldsymbol{b})^T \boldsymbol{n} &= \begin{bmatrix} n_1 & n_2 \end{bmatrix} \begin{bmatrix} \partial_1 a_1 & \partial_1 a_2 \\ \partial_2 a_1 & \partial_2 a_2 \end{bmatrix} \begin{bmatrix} \partial_1 b_1 & \partial_2 b_1 \\ \partial_1 b_2 & \partial_2 b_2 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} n_1\partial_1 a_1 + n_2\partial_2 a_1 & n_1\partial_1 a_2 + n_2\partial_2 a_2 \end{bmatrix}}_{\boldsymbol{x}^T} \underbrace{\begin{bmatrix} n_1\partial_1 b_1 + n_2\partial_2 b_1 \\ n_1\partial_1 b_2 + n_2\partial_2 b_2 \end{bmatrix}}_{\boldsymbol{y}}
\end{aligned}$$

Similarly,

$$\boldsymbol{n}^T \nabla\boldsymbol{b}(\nabla\boldsymbol{a})^T \boldsymbol{n} = \underbrace{\begin{bmatrix} n_1\partial_1 b_1 + n_2\partial_2 b_1 & n_1\partial_1 b_2 + n_2\partial_2 b_2 \end{bmatrix}}_{\boldsymbol{y}^T} \underbrace{\begin{bmatrix} n_1\partial_1 a_1 + n_2\partial_2 a_1 \\ n_1\partial_1 a_2 + n_2\partial_2 a_2 \end{bmatrix}}_{\boldsymbol{x}}$$

The dot product forms an inner product on the euclidean space and is therefore commutative, $\boldsymbol{x}^T\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{x}$, it therefore follows that the above relation holds.

Identity A.6. *Let $\boldsymbol{a},\boldsymbol{b} \in \mathbb{R}^2$ be 2 dimensional vectors. With the notation $[\![\boldsymbol{z}]\!] = \boldsymbol{z} - \boldsymbol{z}_0$ and $\langle\!\langle \boldsymbol{z} \rangle\!\rangle = (\boldsymbol{z} + \boldsymbol{z}_0)/2$ the following identity holds,*

$$[\![\boldsymbol{ab}]\!] = [\![\boldsymbol{a}]\!]\langle\!\langle \boldsymbol{b} \rangle\!\rangle + \langle\!\langle \boldsymbol{b} \rangle\!\rangle[\![\boldsymbol{a}]\!]$$

Proof: *By expanding the expressions one gets,*

$$\begin{aligned}[\![\boldsymbol{b}]\!]\langle\!\langle \boldsymbol{a} \rangle\!\rangle + \langle\!\langle \boldsymbol{b} \rangle\!\rangle[\![\boldsymbol{a}]\!] &= \frac{1}{2}\big(\boldsymbol{ab} + \boldsymbol{a}_0\boldsymbol{b} - \boldsymbol{b}_0\boldsymbol{a} - \boldsymbol{b}_0\boldsymbol{a}_0 + \boldsymbol{ba} + \boldsymbol{b}_0\boldsymbol{a} - \boldsymbol{ba}_0 - \boldsymbol{b}_0\boldsymbol{a}_0\big) \\ &= \boldsymbol{ab} - \boldsymbol{a}_0\boldsymbol{b}_0 = [\![\boldsymbol{ab}]\!]\end{aligned}$$

Identity A.7. *Let $\boldsymbol{C} \in \mathbb{R}^2$ be a two dimensional matrix generated by taking the outer product of two vectors $\boldsymbol{a} \in \mathbb{R}^2$ and $\boldsymbol{b} \in \mathbb{R}^2$ such that $\boldsymbol{C} = \boldsymbol{ab}^T$. Let $\boldsymbol{b}$ be a unit vector such that $\|\boldsymbol{b}\| = b_1^2 + b_2^2 = 1$, where b_i are the components of $\boldsymbol{b}$. The double dot product of $\boldsymbol{C}$ with itself then gives,*

$$\boldsymbol{C} : \boldsymbol{C} = \boldsymbol{ba} \cdot \boldsymbol{a} = \boldsymbol{a}^T\boldsymbol{a}$$

Proof: *By the definition of the double dot product it follows that,*

$$\begin{aligned}\boldsymbol{C} : \boldsymbol{C} = \boldsymbol{ab}^T : \boldsymbol{ab}^T &= \sum_{ij=1}^{2} (ab)_{ij}^2 \\ &= (b_1^2 + b_2^2)a_1^2 + (b_1^2 + b_2^2)a_2^2 = a_1^2 + a_2^2 = \boldsymbol{a} \cdot \boldsymbol{a} = \boldsymbol{a}^T\boldsymbol{a}\end{aligned}$$

Identity A.8. *Let $\boldsymbol{A} \in \mathbb{R}^2$ be a two dimensional matrix generated by taking the outer product of two vectors $\boldsymbol{a} \in \mathbb{R}^2$ and $\boldsymbol{b} \in \mathbb{R}^2$ such that $\boldsymbol{A} = \boldsymbol{ab}^T$. Likewise let $C \in \mathbb{R}^2$ be a matrix generated by taking the outer product of the vectors $\boldsymbol{c} \in \mathbb{R}^2$ and $\boldsymbol{d} \in \mathbb{R}^2$ such that $\boldsymbol{C} = \boldsymbol{cd}^T$. Furthermore, let the set $\{\boldsymbol{b},\boldsymbol{d}\}$ be an orthonormal basis for $\mathbb{R}^2$ such that $\boldsymbol{b} \cdot \boldsymbol{d} = b_1d_1 + b_2d_2 = 0$, where b_i, d_i for $i = 1,2$ are the components of the basis vectors. The double dot product of $\boldsymbol{A}$ with $\boldsymbol{C}$ then gives,*

$$\boldsymbol{A} : \boldsymbol{C} = 0$$

Proof: *By the definition of the double dot product it follows that,*

$$\begin{aligned}\boldsymbol{A} : \boldsymbol{C} = \boldsymbol{ab}^T : \boldsymbol{cd}^T &= \sum_{ij=1}^{2} (ab)_{ij}(cd)_{ij} \\ &= a_1c_1(b_1d_1 + b_2d_2) + a_2c_2(b_1d_1 + b_2d_2) = 0\end{aligned}$$

Lemma A.1. *Let $\boldsymbol{u}, \boldsymbol{\lambda}^n \in \Omega \subset \mathbb{R}^2$ be two vector functions. Let $\boldsymbol{n}$ be the surface normal on the boundary Γ of Ω. The following relation then holds:*

$$\left[\frac{\partial}{\partial \boldsymbol{n}}\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot [[D\boldsymbol{u}]]\right)\boldsymbol{n} + div_\Gamma\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot [[D\boldsymbol{u}]]\right)\right] = \left[\nabla\boldsymbol{\lambda}^n : \nabla[\![\boldsymbol{u}]\!] + \nabla^2[\![\boldsymbol{u}]\!] \cdot \boldsymbol{\lambda}^n\right]$$

Proof: *To simplify the notation the definition $D\boldsymbol{u}^d := [[D\boldsymbol{u}]]$ is introduced. The first part of this expression can be written out using identity A.1. In this identity, since $Dx = \left(\nabla x^T\right)^T$ it will be natural to define $(B)^T = (\nabla \boldsymbol{u}^T)^T = D\boldsymbol{u}$ and similarly $a^T = \left(\boldsymbol{\lambda}^n\right)^T$. From now on transpose on λ will be implicit if multiplied from the left of a matrix. The expression then becomes,*

$$\frac{\partial}{\partial \boldsymbol{n}}\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot D\boldsymbol{u}^d\right) = \boldsymbol{n}^T\nabla\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right)\boldsymbol{n} = \boldsymbol{n}^T\left(D\boldsymbol{\lambda}^n\right)^T\left(D\boldsymbol{u}^{\boldsymbol{d}}\right)\boldsymbol{n} + \boldsymbol{n}^T\boldsymbol{\lambda}^n\left(D\nabla\boldsymbol{u}^d\right)^T\boldsymbol{n} \tag{A.11}$$

Using the definition of div_Γ from Eq. (A.5) *along with identity A.2 in the appendix, also the second term can be written out more explicitly. Again, let $a^T = \left(\lambda^n\right)^T$ and $B^T = D\boldsymbol{u}$ the resulting equation is,*

$$\begin{aligned} div_\Gamma\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right) &= div\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right) - \boldsymbol{n}^T D\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right)\boldsymbol{n} \\ &= \nabla\boldsymbol{\lambda}^n : \nabla\boldsymbol{u}^d + div\left(\nabla\boldsymbol{u}^d\right)\cdot\boldsymbol{\lambda}^n - \boldsymbol{n}^T\left(\nabla\boldsymbol{u}^d\right)\left(D\boldsymbol{\lambda}^n\right)\boldsymbol{n} - \boldsymbol{n}^T\left(D\nabla\boldsymbol{u}^d\right)\boldsymbol{\lambda}^n\boldsymbol{n} \end{aligned} \tag{A.12}$$

In the last line, Identity A.3 is used to expand $div\left(\lambda^n \cdot D\boldsymbol{u}^d\right)$ and the term involving $\boldsymbol{n}^T D\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right)\boldsymbol{n}$ is expanded using identity A.2. Setting all this together by combining equation (A.11) *and* (A.12) *gives,*

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{n}}\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot D\boldsymbol{u}^d\right) + div_\Gamma\left(\boldsymbol{\lambda}^n \cdot D\boldsymbol{u}^d\right) &= \boldsymbol{n}^T\left(D\boldsymbol{\lambda}^n\right)^T\left(D\boldsymbol{u}^{\boldsymbol{d}}\right)\boldsymbol{n} + \boldsymbol{n}^T\boldsymbol{\lambda}^n\left(D\nabla\boldsymbol{u}^d\right)^T\boldsymbol{n} \\ &+ \nabla\boldsymbol{\lambda}^n : \nabla\boldsymbol{u}^d + div\left(\nabla\boldsymbol{u}^d\right)\cdot\boldsymbol{\lambda}^n - \boldsymbol{n}^T\left(\nabla\boldsymbol{u}^d\right)\left(D\boldsymbol{\lambda}^n\right)\boldsymbol{n} - \boldsymbol{n}^T\left(D\nabla\boldsymbol{u}^d\right)\boldsymbol{\lambda}^n\boldsymbol{n}. \end{aligned}$$

It is now possible to make several simplifications. The terms involving $\boldsymbol{n}^T\left(D\boldsymbol{\lambda}^n\right)^T\left(D\boldsymbol{u}^{\boldsymbol{d}}\right)\boldsymbol{n}$ and $\boldsymbol{n}^T\left(\nabla\boldsymbol{u}^d\right)\left(D\boldsymbol{\lambda}^n\right)\boldsymbol{n}$ cancel which follows from identity A.5. There are also two terms containing $(D\nabla\boldsymbol{u}^d) = D^2\boldsymbol{u}^d$. This expression corresponds to the Hessian of $\boldsymbol{u}^d$. Due to the symmetry of the Hessian the two terms cancel and one is left with.

$$\begin{aligned} \left[\frac{\partial}{\partial \boldsymbol{n}}\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot [[D\boldsymbol{u}]]\right)\boldsymbol{n} + div_\Gamma\left(\boldsymbol{\lambda}^{\boldsymbol{n}} \cdot [[D\boldsymbol{u}]]\right)\right] &= \left[\nabla\boldsymbol{\lambda}^n : \nabla\boldsymbol{u}^d + div\left(\nabla\boldsymbol{u}^d\right)\cdot\boldsymbol{\lambda}^n\right] \\ &= \left[\nabla\boldsymbol{\lambda}^n : \nabla[\![\boldsymbol{u}]\!] + \nabla^2[\![\boldsymbol{u}]\!]\cdot\boldsymbol{\lambda}^n\right] \end{aligned}$$

which concludes the proof. In the last step it is used that $div\nabla = \nabla^2$ and $\boldsymbol{u}^d$ is changed back to $[\![\boldsymbol{u}]\!]$. Remember that $\boldsymbol{u}^d$ was only introduced for convenience to simplify the notation in the derivation.

Lemma A.2. *Let $\boldsymbol{u} \in \Omega \subset \mathbb{R}^2$ be a vector function. Let the surface normal $\boldsymbol{n}$ and the surface tangent $\boldsymbol{\tau}$ form an orthonormal basis in $\mathbb{R}^2$. The following relation then holds;*

$$[\![\nabla\boldsymbol{u} : \nabla\boldsymbol{u}]\!] = [[\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} \cdot \frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}}]] + [[\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{\tau}} \cdot \frac{\partial\boldsymbol{u}}{\partial\boldsymbol{\tau}}]].$$

Proof: *The first step is to rewrite the jump term to $[\![\nabla \boldsymbol{u} : \nabla \boldsymbol{u}]\!] = \nabla \boldsymbol{u}^{+} : \nabla \boldsymbol{u}^{+} - \nabla \boldsymbol{u}^{-} : \nabla \boldsymbol{u}^{-}$. With the surface normal $\boldsymbol{n}$ and the surface tangent $\boldsymbol{\tau}$ forming an orthonormal basis the gradient can be expanded according to Definition A.4. This gives for the double dot products,*

$$
\begin{aligned}
\nabla \boldsymbol{u} : \nabla \boldsymbol{u} &= \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\boldsymbol{n}^T + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\boldsymbol{\tau}^T\right) : \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\boldsymbol{n}^T + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\boldsymbol{\tau}^T\right) \\
&= \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\boldsymbol{n}^T : \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\boldsymbol{n}^T + 2\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}\boldsymbol{n}^T : \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\boldsymbol{\tau}^T + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\boldsymbol{\tau}^T : \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}\boldsymbol{\tau}^T \\
&= \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}
\end{aligned}
$$

In the last step, Identity A.8 with $b = \boldsymbol{n}$, $d = \boldsymbol{\tau}$, $a = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$ and $c = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}$ is used to remove the cross terms. Similarly, Identity A.7 was used to reduce the double dot product of the remaining terms to simple dot products. This result follows by setting $a = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}$, $b = \boldsymbol{n}$ in Identity A.7 and likewise for the tangential derivative. The jump terms can therefore be written as $[\![\nabla \boldsymbol{u} : \nabla \boldsymbol{u}]\!] = [[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}}]] + [[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\tau}}]]$ which concludes the proof.

APPENDIX B

FUNCTIONAL ANALYSIS

The main purpose of this chapter is to define the functional analysis background that is necessary for a discussion on the finite element method. In particular the definition of a Sobolov space is presented. A more thorough discussion on Sobolov spaces can be found in [5]. It is here assumed that basic concepts of functional analysis like vector spaces, inner products and norms are known. A thorough definition of these concepts can be found in [26].

B.1 Function spaces

Definition B.1. Hilbert space: A Hilbert space is a complete inner product space with a norm associated with the inner product. For more details see [26].

Definition B.2. L^2 norm: For a vector valued function $\boldsymbol{h}(\boldsymbol{x})$ defined over a domain $\Omega \subset \mathbb{R}^d$ such that $\boldsymbol{h} : \Omega \longrightarrow \mathbb{R}^n$. The L^2 norm is defined as,

$$\|\boldsymbol{h}\|_{L^2} = \sqrt{\int_\Omega \|\boldsymbol{h}\|^2_{\mathbb{R}^n}\, dx} \tag{B.1}$$

where $\|\cdot\|_{\mathbb{R}^n}$ is the standard Euclidean norm on $\mathbb{R}^n$. Note that an inner product $\|\boldsymbol{h}\|^2_{L^2}$ can be associated with the L^p norm.

Definition B.3. $\mathbf{L}^2(\Omega, \mathbb{R}^n)$ space: For a vector valued function $\boldsymbol{h}(\boldsymbol{x})$ defined over a domain $\Omega \subset \mathbb{R}^d$ such that $\boldsymbol{h} : \Omega \longrightarrow \mathbb{R}^n$. The $L^2(\Omega, \mathbb{R}^n)$ space is defined as,

$$L^2(\Omega, \mathbb{R}^n) = \{\boldsymbol{h} \in \mathbb{R}^n : \|\boldsymbol{h}\|_{L^2} < \infty\}. \tag{B.2}$$

Note that the $\mathbf{L}^2(\Omega, \mathbb{R}^n)$ space is a Hilbert space, since the L^p norm forms an inner product.

B.1.1 Sobolov spaces

Sobolov spaces are generalisations of L^p spaces in the sense that the derivatives are added to the standard L^p norm. For this work it is sufficient to consider the special case of $p = 2$ for which the Sobolov space is a Hilbert space and will be denoted H^s. In this case the H^s norm and the Sobolov space $\mathscr{H}^s(\Omega, \mathbb{R}^n)$ are defined below.

Definition B.4. Sobolov norm $\|\cdot\|_{H^s}$ **:** For a vector valued function $\boldsymbol{h}(\boldsymbol{x})$ defined over a domain $\Omega \subset \mathbb{R}^d$ such that $\boldsymbol{h} : \Omega \longrightarrow \mathbb{R}^n$. The H^s norm is defined as,

$$\|\boldsymbol{h}\|_{H^s} = \sqrt{\int_\Omega \sum_{k\leq s} \left\| \left(\frac{\partial}{\partial \boldsymbol{x}}\right)^k \boldsymbol{h} \right\|^2_{\mathbb{R}^n})}$$

Definition B.5. Sobolov semi-norm $\|\cdot\|_{H^s_0}$ **:** For a vector valued function $\boldsymbol{h}(\boldsymbol{x})$ defined over a domain $\Omega \subset \mathbb{R}^d$ such that $\boldsymbol{h} : \Omega \longrightarrow \mathbb{R}^n$. The H^s_0 norm is defined as,

$$\|\boldsymbol{h}\|_{H^s_0} = \sqrt{\int_\Omega \sum_{k=s} \left\| \left(\frac{\partial}{\partial \boldsymbol{x}}\right)^k \boldsymbol{h} \right\|^2_{\mathbb{R}^n})}$$

Definition B.6. Sobolov space $\mathscr{H}^s(\Omega, \mathbb{R}^n)$: For a vector valued function $\boldsymbol{h}(\boldsymbol{x})$ defined over a domain $\Omega \subset \mathbb{R}^d$ such that $\boldsymbol{h} : \Omega \longrightarrow \mathbb{R}^n$. The Sobolov space $\mathscr{H}^s(\Omega, \mathbb{R}^n)$ is defined as,

$$\mathscr{H}^s(\Omega, \mathbb{R}^n) = \left\{ \boldsymbol{h} \in \mathbb{R}^n : \|\boldsymbol{h}\|_{H^s} \leq \infty \right\}$$

The relevant Sobolov space for this work is given bellow.

Definition B.7. Sobolov space $\mathscr{H}^1(\Omega, \mathbb{R}^n)$ **:** See def. B.6 and def. B.4 with $s = 1$.

Definition B.8. Sobolov space $\mathscr{H}^0(\Omega, \mathbb{R}^n)$ **:** See def. B.6 and def. B.4 with $s = 0$. This corresponds to the $L^2(\Omega, \mathbb{R}^n)$ space in definition B.3.

B.2 General concepts

The notion of orthogonality from euclidean space can be extended to vector spaces in terms of the inner product.

Definition B.9. Orthogonality: Let $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{X}$, where $\boldsymbol{X}$ is an inner product space. Then they are orthogonal if,

$$(\boldsymbol{u}, \boldsymbol{v}) = 0 \tag{B.3}$$

APPENDIX C

MOTIVATION BEHIND THE WEAK FORMULATION

This chapter will give a motivation for forming the weak formulation of a boundary value problem. As a working example the $1D$ Poisson problem is used.

(C.1) $$\mathscr{D}\boldsymbol{u}(x) = -\nabla^2 \boldsymbol{u} = \boldsymbol{f}(x) \quad \text{on} \quad \Omega$$

(C.2) $$\boldsymbol{u}(x) = \boldsymbol{g}_D \quad \text{on} \quad \partial\Omega_D$$

(C.3) $$\nabla \boldsymbol{u}(x) = \boldsymbol{g}_N \quad \text{on} \quad \partial\Omega_N.$$

Here $\boldsymbol{u}(x)$ is a vector valued function over some domain $\Omega \subset \mathbb{R}$ and $\mathscr{D}$ is a linear differential operator. The source terms is given as $\boldsymbol{f}(x)$. The boundary conditions are given in Eq. (C.2) and (C.3), as Dirichlet and Neuman conditions respectively. This formulation of the boundary value problem will be referred to as the strong form.

C.1 Weak formulation

The recipe to transform a boundary value problem as stated in Eq. (C.1) to its weak form is simple. Multiplying the governing equation with a test function $\boldsymbol{v} \in V$ and integrate by parts gives,

(C.4) $$\int_0^1 \frac{d\boldsymbol{u}}{dx}\frac{d\boldsymbol{v}}{dx} dx - \left[\frac{d\boldsymbol{u}}{dx}\boldsymbol{v}\right]_0^1 = \int_0^1 \boldsymbol{f}(x)\boldsymbol{v}\, dx \quad \forall \boldsymbol{v} \in V.$$

The space $V = [\mathscr{H}_0^1 : \boldsymbol{v} = \boldsymbol{g}_D$ on $\partial\Omega_D]$ is a Sobolov space as defined in Definition B.6. The boundary term in Eq. (C.4) is substituted for by the Neuman condition (C.3) on Neuman boundaries $\partial\Omega_N$ and included in the right hand side of the equation. The enforcement of the Dirichlet condition will be actively set when solving the system. The weak form of the boundary value problem is then stated as. Find $\boldsymbol{u} \in V$ such that,

$$a(\boldsymbol{u},\boldsymbol{v}) = L(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in V \tag{C.5}$$

Here the bilinear form $a(\boldsymbol{u},\boldsymbol{v})$ is defined as,

$$a(\boldsymbol{u},\boldsymbol{v}) = \int_0^1 \frac{d\boldsymbol{u}}{dx}\frac{d\boldsymbol{v}}{dx}dx \tag{C.6}$$

and $L(\boldsymbol{v})$ is a linear form defined as,

$$L(\boldsymbol{v}) = \int_0^1 \boldsymbol{f}(x)\boldsymbol{v}dx - \boldsymbol{g}_N. \tag{C.7}$$

Note that the term involving $\boldsymbol{g}_N$ is 0 on $\partial\Omega_D$.

C.2 Discretisation

The solution $\boldsymbol{u}$ to the original boundary value problem stated in Eg. (C.1) lives in a infinite dimensional space of functions that satisfy $\mathbb{C}^2(\Omega)$ continuity. The weak form seeks an approximate solution $\boldsymbol{u}$ in the sense that $\boldsymbol{u}$ no longer need to be two times differentiable. In fact it is sufficient that $\boldsymbol{u} \in \mathscr{H}^1$ a Sobolev space, see Definition B.6. The finite element method aims to convert the weak formulation into a set of algebraic equations. In order to do this it is necessary to restrict the solution space to a finite N dimensional space $V_h = [\mathscr{H}^1_{h,g_D} : \boldsymbol{g} = \boldsymbol{g}_D$ on $\partial\Omega_D]$ where $\mathscr{H}_h^1 \subset \mathscr{H}^1$. Here h indicates the "discretization" of the space.

With this definition of the solution space, it is practical to expand the trail function $\boldsymbol{u}_h \in V_h$ in a set of basis functions $\{\boldsymbol{\phi}(x)_i\}_{i=1}^N$ such that,

$$\boldsymbol{u}_h(x) = \sum_{j=1}^N c_j \boldsymbol{\phi}_j(x). \tag{C.8}$$

The problem is therefore reduced to finding the N coefficients $\{c_i\}_{i=1}^N$ corresponding to the degrees of freedom in the approximation space (trail space) V_h.

C.3 Galerkin's method

There are several ways to define the test and trail functin and it is not necessry that they are defined over the same space. However, the Galerkins method offer a particular advantageous choice and this will be discussed in this section. Let $\boldsymbol{u}^e \in V$ be the exact solution of the strong form of the boundary value problem defined at the beginning of this chapter. Where $V = \mathscr{H}$ is a in-finte dimensional Hilber space. Let $\boldsymbol{u}_h \in V_h \in V$ be the approximate solution to this problem. As explained in section C.2 $\boldsymbol{u}_h$ can be expanded in the basis of V_h with the coefficients $\{c_j\}_{i=1}^N$. The goal is now to determine the coefficients in such a way that $\boldsymbol{u}_h$ gives a good aproximation to $\boldsymbol{u}^e$. A natural way is to chose $\{c_j\}_{i=1}^N$ such that it minimises the error $\boldsymbol{u}^e - \boldsymbol{u}_h$. However, since $\boldsymbol{u}^e$ is known only through Eq. (C.1) the best one can do is to minimise the residual error $\boldsymbol{e}_R = \mathscr{D}(\boldsymbol{u}_h) - \boldsymbol{f}(\boldsymbol{x})$.

The strategy in the Galerkin's method is to find $\{c_j\}_{i=1}^N$ and thereby $\boldsymbol{u}_h$ such that $\boldsymbol{e}_R \perp V_h$ which in the standard inner product $\langle k(x), h(x)\rangle = \int_\Omega k(x)h(x)dx$, results in the problem statement: Find $\boldsymbol{u} \in V_h$ such that,

$$\int_\Omega (\mathscr{D}\boldsymbol{u}(x) - f(x))\boldsymbol{v}_h dx = 0 \quad \forall \quad \boldsymbol{v}_h \in V_h \tag{C.9}$$

The motivation for this can be understood by a simple geometrical argument. The shortest distance, in the norm $\|h(x)\| = \sqrt{\langle h(x), h(x)\rangle}$, between the space V_h and the error $\boldsymbol{e}_R$ is the distance measured from $\boldsymbol{e}_R$ and perpendicularly down on V_h. Integrating Eq. (C.9) by parts now result in the weak formulation of the boundary value problem. This concludes the motivation for the weak form.

C.4 Galerkin orthogonality

The interesting result emerging from the galerkins aproach is the concept of Galerkin orthogonality. Let $\boldsymbol{u}$ be the exact solution of the weak form and let $\boldsymbol{u}_h$ be the approximate solution. Then they satisfy,

$$a(\boldsymbol{u}, \boldsymbol{v}) = L(\boldsymbol{v}) \quad \forall \quad \boldsymbol{v} \in V \tag{C.10}$$

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) = L(\boldsymbol{v}_h) \quad \forall \quad \boldsymbol{v}_h \in V_h \subset V, \tag{C.11}$$

where V is an infinite dimensional Sobolov space an V_h is a finite dimensional subspace. The concept of Galerking orthogonality can then be stated as follows,

Lemma C.1. ***(Galerkin orthogonality).***
The following statement holds. When the bilinear form $a(\boldsymbol{u}, \boldsymbol{v})$. satisfies the properties of an inner product, then the error $e = \boldsymbol{u} - \boldsymbol{u}_h$ is orthogonal to the space V_h in the inner-product defined by the bilinear form.

Proof: *Since $\boldsymbol{u}_h$ is the solution of the weak form* (C.11) *then,*

$$a(\boldsymbol{e}, \boldsymbol{v}_h) = a(\boldsymbol{u}, \boldsymbol{v}_h) - a(\boldsymbol{u}_h, \boldsymbol{v}_h) = L(\boldsymbol{v}_h) - L(\boldsymbol{v}_h) = 0 \tag{C.12}$$

Lemma C.2. ***(Minimised error in Energy norm).***
Let $\|\boldsymbol{v}\|_E = \sqrt{a(\boldsymbol{v}, \boldsymbol{v})}$ be the norm induced by the bilinear form $a(\boldsymbol{v}, \boldsymbol{v})$, where $a(\boldsymbol{u}_h, \boldsymbol{v}_h)$ satisfy Galerkins orthogonality. Then

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_E \leq \|\boldsymbol{u} - \boldsymbol{v}_h\|_E \quad \forall \boldsymbol{v}_h \in V_h \tag{C.13}$$

Proof: *Follows from galerkins orthogonality and the Cauchy-schwartz inequality.*

What this result means is that Galerkins method not only chooses the best minimizer of $\boldsymbol{e}_R$ as mentioned in section C.3. It also chooses the best approximation to $\boldsymbol{u}$ in the space V_h when the error is measured in the energy norm $\|\cdot\|_E$.

BIBLIOGRAPHY

[1] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, "Unified form language: A domain-specific language for weak formulations of partial differential equations," *ACM Transactions on Mathematical Software (TOMS)*, vol. 40, no. 2, p. 9, 2014.

[2] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, "The FEniCS project version 1.5," *Archive of Numerical Software*, vol. 3, no. 100, pp. 9–23, 2015.

[3] G. Bai, J. Li, P. Fan, and G. Li, "Numerical investigations of the effects of different arrays on power extractions of horizontal axis tidal current turbines," *Renewable Energy*, vol. 53, pp. 180–186, 2013.

[4] M. Berggren, "A unified discrete-continuous sensitivity analysis method for shape optimization," in *Applied and numerical partial differential equations*. Springer, 2010, pp. 25–39.

[5] H. Brezis, *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[6] M. A. Celia and W. G. Gray, *Numerical methods for differential equations: Fundamental concepts for scientific and engineering applications*. Pearson College Div, 1992.

[7] COIN-OR, "Papers/publications related to ipopt," https://projects.coin-or.org/Ipopt/wiki/IpoptPapers, 2018, (accessed April 23, 2019).

[8] ——, "Papers/publications related to ipopt," https://www.coin-or.org/Ipopt/documentation/node40.html, 2018, (accessed May 16, 2019).

[9] M. C. Delfour and J.-P. Zolâsio, *Shapes and geometries: metrics, analysis, differential calculus, and optimization*. Siam, 2011, vol. 22.

[10] J. S. Dokken, S. W. Funke, A. Johansson, and S. Schmidt, "Shape Optimization using the Finite Element Method on Multiple Meshes," *arXiv e-prints*, p. arXiv:1806.09821, Jun 2018.

[11] P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes, "Automated derivation of the adjoint of high-level transient finite element programs," *SIAM Journal on Scientific Computing*, vol. 35, no. 4, pp. C369–C393, 2013.

[12] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Mathematical programming*, vol. 91, no. 2, pp. 239–269, 2002.

[13] P. Fullsack, "An arbitrary lagrangian-eulerian formulation for creeping flows and its application in tectonic models," *Geophysical Journal International*, vol. 120, no. 1, pp. 1–23, 1995.

[14] S. W. Funke and P. E. Farrell, "A framework for automated PDE-constrained optimisation," *arXiv e-prints*, p. arXiv:1302.3894, Feb 2013.

[15] S. W. Funke, P. E. Farrell, and M. Piggott, "Tidal turbine array optimisation using the adjoint approach," *Renewable Energy*, vol. 63, pp. 658–673, 2014.

[16] R. Glowinski and J. He, "On shape optimization and related issues," in *Computational Methods for Optimal Design and Control*. Springer, 1998, pp. 151–179.

[17] A. Hansbo and P. Hansbo, "An unfitted finite element method, based on Nitsche's method, for elliptic interface problems," *Computer methods in applied mechanics and engineering*, vol. 191, no. 47-48, pp. 5537–5552, 2002.

[18] A. Hansbo, P. Hansbo, and M. G. Larson, "A finite element method on composite grids based on Nitsche's method," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 37, no. 3, pp. 495–514, 2003.

[19] H. Harbrecht and F. Loos, "Optimization of current carrying multicables," *Computational optimization and applications*, vol. 63, no. 1, pp. 237–271, 2016.

[20] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*. Springer Science & Business Media, 2008, vol. 23.

[21] R. Hoshiko and M. Kawahara, "An analysis of the flow around a wind propeller using the finite element method," *Journal of Algorithms & Computational Technology*, vol. 6, no. 2, pp. 261–279, 2012.

[22] A. Johansson, M. G. Larson, and A. Logg, "High order cut finite element methods for the Stokes problem," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 2, no. 1, p. 24, 2015.

[23] A. Johansson, M. G. Larson, and A. Logg, "MultiMesh Finite Element Methods II: Analysis," *arXiv e-prints*, p. arXiv:1804.06455, Apr 2018.

[24] A. Johansson, B. Kehlet, M. G. Larson, and A. Logg, "Multimesh finite element methods: Solving PDEs on multiple intersecting meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 343, pp. 672–689, 2019.

[25] S. C. Kramer, S. W. Funke, and M. D. Piggott, "A continuous approach for the optimisation of tidal turbine farms," in *Proceedings of the 11th European Wave and Tidal Energy Conference Series, Nantes, France*, 2015, pp. 6–11.

[26] E. Kreyszig, *Introductory functional analysis with applications*. wiley New York, 1978, vol. 1.

[27] N. Kroll, N. R. Gauger, J. Brezillon, R. Dwight, A. Fazzolari, D. Vollmer, K. Becker, H. Barnewitz, V. Schulz, and S. Hazra, "Flow simulation and shape optimization for aircraft design," *Journal of Computational and applied mathematics*, vol. 203, no. 2, pp. 397–411, 2007.

[28] E. Lauga, "Bacterial hydrodynamics," *Annual Review of Fluid Mechanics*, vol. 48, pp. 105–130, 2016.

[29] A. Logg and G. N. Wells, "Dolfin: Automated finite element computing," *ACM Transactions on Mathematical Software (TOMS)*, vol. 37, no. 2, p. 20, 2010.

[30] A. Logg, K.-A. Mardal, and G. Wells, *Automated solution of differential equations by the finite element method: The FEniCS book*. Springer Science & Business Media, 2012, vol. 84.

[31] A. Logg, G. N. Wells, and J. Hake, "Dolfin: A c++/python finite element library," in *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012, pp. 173–225.

[32] A. Massing, M. G. Larson, A. Logg, and M. E. Rognes, "A stabilized Nitsche overlapping mesh method for the Stokes problem," *Numerische Mathematik*, vol. 128, no. 1, pp. 73–101, 2014.

[33] A. Massing, M. G. Larson, and A. Logg, "Efficient implementation of finite element methods on nonmatching and overlapping meshes in three dimensions," *SIAM Journal on Scientific Computing*, vol. 35, no. 1, pp. C23–C47, 2013.

[34] B. Mohammadi and O. Pironneau, "Shape optimization in fluid mechanics," *Annu. Rev. Fluid Mech.*, vol. 36, pp. 255–279, 2004.

[35] ——, *Applied shape optimization for fluids*. Oxford university press, 2010.

[36] E. J. Nielsen and M. A. Park, "Using an adjoint approach to eliminate mesh sensitivities in computational design," *AIAA journal*, vol. 44, no. 5, pp. 948–953, 2006.

[37] J. Nitsche, "Über ein variationsprinzip zur lösung von Dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind," in *Abhandlungen Aus Dem Mathematischen Seminar Der Universitat Hamburg*, vol. 36, no. 1. Springer, 1971, pp. 9–15.

[38] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[39] J. Nocedal, A. Wächter, and R. A. Waltz, "Adaptive barrier update strategies for nonlinear interior methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1674–1693, 2009.

[40] A. Oslandsbotn, "Master-thesis-spring-2019, github/repository," https://github.com/andreasOslandsbotn/Master-thesis-spring-2019/tree/master/Local%20container%20files, 2019, (accessed June 07, 2019).

[41] S. Schmidt, "Efficient large scale aerodynamic design based on shape calculus," Ph.D. dissertation, University of Trier, 2010.

[42] ——, "Weak and strong form shape hessians and their automatic generation," *SIAM Journal on Scientific Computing*, vol. 40, no. 2, pp. C210–C233, 2018.

[43] S. Schmidt and V. Schulz, "Shape derivatives for general objective functions and the incompressible Navier-Stokes equations," *Control and Cybernetics*, vol. 39, no. 3, pp. 677–713, 2010.

[44] S. Silvestri and E. Schena, "Micromachined flow sensors in biomedical applications," *Micromachines*, vol. 3, no. 2, pp. 225–243, 2012.

[45] S. N. Skinner and H. Zare-Behtash, "State-of-the-art in aerodynamic shape optimisation methods," *Applied Soft Computing*, vol. 62, pp. 933–962, 2018.

[46] J. Sokolowski and J.-P. Zolesio, *Introduction to Shape Optimization*. Springer, 1992.

[47] E. Tekin and I. Sabuncuoglu, "Simulation optimization: A comprehensive review on theory and applications," *IIE transactions*, vol. 36, no. 11, pp. 1067–1081, 2004.

[48] C. Valdemar Hansen, A. Logg, and C. Lundholm, "Simulation of flow and view with applications in computational design of settlement layouts," *arXiv e-prints*, p. arXiv:1610.02277, Oct 2016.

[49] A. Wächter, "Short tutorial: Getting started with ipopt in 90 minutes," in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[50] ——, "An interior point algorithm for large-scale nonlinear optimization with applications in process engineering." 2003.

[51] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 1–31, 2005.

[52] ——, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[53] E. Xu, "A python connector to ipopt," https://github.com/xuy/pyipopt, 2018, (accessed May 31, 2019).

[54] L. Zhu, E. Lauga, and L. Brandt, "Low-reynolds-number swimming in a capillary tube," *Journal of Fluid Mechanics*, vol. 726, pp. 285–311, 2013.

[55] G. Łukaszewicz and P. Kalita, *Navier-Stokes Equations: An Introduction with Applications*. Springer, 2016.