NTNU
Norwegian University of
Science and Technology

Topology Optimization for Unsteady Flow with Applications in Biomedical Flows

Kari Elisabeth Skaar
Hasund

Master of Science in Physics and Mathematics
Submission date: February 2017
Supervisor: Anton Evgrafov, IMF
Co-supervisor: August Johansson, Simula Research Laboratory

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

In this thesis, we will apply a topology optimization method to unsteady fluid flow, using a density model and level set method, in order to optimize the shape of a coronary artery bypass anastomosis. The fluid movement is described by the unsteady, incompressible Navier-Stokes equations combined with Darcy's equation. These equations are discretized using a finite element approach in space and a backward Euler method in time, and solved using an incremental pressure correction scheme (IPCS). We will consider different objective functionals for the optimization problems. The topological derivative will be calculated based on an adjoint formulation of the Navier-Stokes equations combined with Darcy's equation, and used as a search direction in a gradient-based algorithm, in order to find the optimal channel shape.

Sammendrag

I denne masteroppgaven presenteres en topologioptimeringsmetode for fluidstrømning, der det brukes en tetthetsmodell og en levelsetmetode. Metoden vil anvendes til optimering av formen på en bypass-anastomose i en koronar arterie. Fluidstrømningen beskrives av de tidsavhengige inkompressible Navier-Stokes-ligningene kombinert med Darcys ligning. Ligningene diskretiseres ved bruk av en endelig elementmetode i rom, og en baklengs Eulermetode i tid, og løses ved bruk av et inkrementerende trykkorreksjonsskjema (IPCS). Vi vil se på forskjellige kostnadsfunksjonaler for optimeringsproblemet. Den topologisk deriverte vil bli regnet ut basert på en adjointformulering av Navier-Stokes-ligningene kombinert med Darcys ligning, og brukt som søkeretning i en gradientbasert algoritme for å finne den optimale kanalformen.

Preface

This thesis completes my master's degree in Industrial Mathematics at the Norwegian University of Science and Technology (NTNU). The work was carried out in collaboration with Simula Resarch Laboratory at Fornebu, Norway. I have since I was quite young been fascinated by biomedical research, with an inner wish to contribute to this field in some way. After a presentation at NTNU, where representatives from Simula talked about the modeling the heart, I was determined that I wanted to write my thesis at Simula.

I had earlier written a smaller project thesis on topology optimization in fluid flow at NTNU, under the supervision of Anton Evgrafov. Wanting to exploit the knowledge I had obtained during this work, I looked for a way to combine this theory with a biomedical problem setting. I contacted a few supervisors at Simula, in particular August Johansson, who is part of OptCutCell optimization group at Simula. This seemed like a good match, and we started to discover ways to combine fluid flow topology optimization with biomedical flow. We arrived at optimizing the shape of a bypass, where the initial geometry would be unknown, hence taking advantage of the nature of topology optimization.

With this goal setting in mind, the work started with what eventually would become this thesis. Since I started, I have learned a great deal. Not only about topology optimization, adjoints and the nature of the Navier-Stokes equations, but also about how to spend your time well, think like a researcher and take advantage of the knowledge of others. I was so lucky to be stationed at Simula Research Laboratory, where I was surrounded by some of the most clever minds I have ever met, and was inspired every day to work harder.

I would like to direct a thank you to my supervisor at NTNU, Anton, who holds a seemingly limit-less knowledge on the field of topology optimization, and has introduced me to the field. I would also like to thank the members of OptCutCell; Simon, August and Jørgen, who have spent numerous hours on guidance, discussions and debugging, in addition to weekly meetings. A special thanks to August, who has been my main supervisor through this work. None of my questions have been too trivial, and I have always felt welcome in your office. A big thank you also goes to Jakob and Carl Martin, for proofreading my thesis. I would like to thank my family for supporting me throughout writing this thesis, and I want to thank my boyfriend, Kristoffer. You have a unique ability to cheer me up, and take my mind off all of my worries when needed. Lastly I want to thank Simula Research Laboratory for making me feel welcome and letting me be part of their research society. For that I am grateful.

Fornebu, February 4, 2017

Kari Elisabeth Skaar Hasund

Table of Contents

Chapter 1

Introduction

"The art of structure is where to put the holes"
Robert Le Ricolais
French-American engineer and philosopher
(1894-1977)

1.1 Problem Description

In this thesis, we will apply a topology optimization method to time-dependent fluid flow, using the density method and level set method. This method will be applied to optimize the shape of a coronary artery bypass anastomosis, which is a vessel going around an occluded artery in the heart. The fluid movement is described by the unsteady, incompressible Navier-Stokes equations combined with Darcy's equation. These equations are discretized using a finite element approach in space and a backward Euler method in time. We will consider different objective functionals for the optimization problems. The topological derivative will be calculated based on an adjoint formulation of the Navier-Stokes equations. Using the adjoint solution, we are able to calculate the topological derivative with significantly lower computational costs. The topological derivative will be used as a search direction in a gradient-based algorithm in order to find the (locally) optimal channel shape.

1.2 Medical Background

The motivation behind the topology optimization method is to design a shape of a coronary artery bypass anastomosis. A bypass surgery is a surgical procedure to restore normal

blood flow to an occluded coronary artery. Figure 1.1 shows an illustration of a bypass.

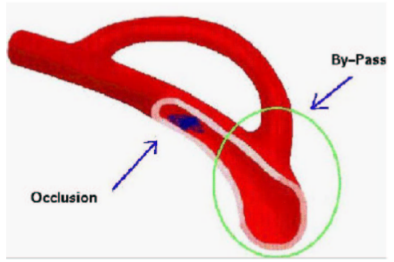
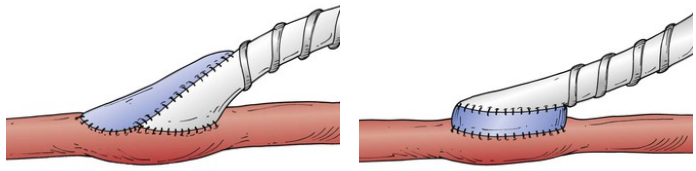


Figure 1.1: Simplified model of a bypass, collected from Quarteroni and Rozza [1]. The zone of the incoming branch of the bypass into the coronary, marked with a circle, is known as the toe.

Bypass surgery can provide relief of angina, which is chest pain related to lack of blood flow, but bypass surgery does not prevent future heart attacks. An understanding of coronaric diseases is very important to reduce surgical and post-surgical failures, as successful grafts typically last 8-15 years. Approximately 8 % of patients risk bypass failure every year, and 80 % of bypasses must be replaced within 10 years [1]. One of the principal causes is anastomotic intimal hyperplasia (Cole et al. [2]), which is a physiologic healing response to injury to the blood vessel wall. From clinical observations (Leuprecht et al. [3]), we know that intimal hyperplasia mostly occurs at outflow anastomoses of bypass grafts. Different shapes of coronary artery bypass anastomoses are available, such as Taylor Patch [4] and Miller Cuff [5], so different surgery procedures are practiced, see Figure 1.2 for an illustration.



(a) Taylor patch. **(b)** Miller cuff.

Figure 1.2: Illustration of different bypass anastomosis shapes, collected from [6].

The left and right coronary arteries are the blood vessels that run on the heart surface, and they regulate the blood flow levels to match the needs of the heart muscle, see Figure 1.3. These vessels are relatively narrow and can become blocked, causing angina or heart attack. The coronary arteries are the only source of blood supply to the heart muscle, so blockage of these vessels is critical, generally resulting in death of the heart tissue due to lack of blood supply. An anastomosis is an area of vessels that are interconnected, allowing dual blood supply to a region. This normally ensures blood flow even if there is a partial blockage in another branch. Since the anastomoses in the heart are very small,

blockage in a coronary artery often results in lack of blood supply and death of the heart cells affected.

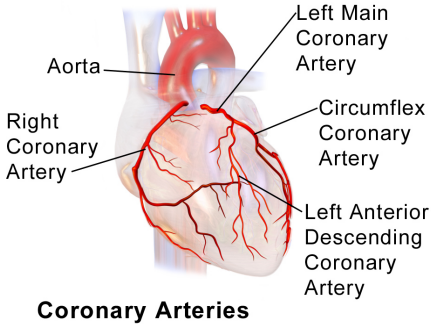


Figure 1.3: Coronary arteries on the heart, collected from [7]

Shape optimization of a coronary artery bypass anastomosis based on adjoint formulation has been proposed by Quarteroni and Rozza [1] in 2003. In this work, the zone of the incoming branch of the bypass into the coronary, also known as the toe, was optimized, and the cost functionals for vorticity and wall shear stress were minimized. The flow was modeled using steady Stokes flow, and resulted in an optimally shaped cuffed bypass, with a shape that resembled the Taylor arterial patch, described in Cole et al. [4]. In 2005 Rozza [8] extended this work by solving the unsteady Navier-Stokes equations in the original and final configuration of the shape proposed in [1], and a reduction of 40 % in vorticity was observed. In 2006 Agoshkov, Rozza and Quarteroni [9, 10] presented a new approach based on small perturbation theory and adjoint formulation in the study of aorto-coronary bypass anastomoses. One approach with steady Stokes flow [9], and another with unsteady Stokes equations [10], both showed reduction in vorticity of about 30 %. All previous optimization problems were modelled in a two-dimensional geometry and used a chosen velocity profile as inlet velocity. For future development it is suggested to apply fully unsteady incompressible Navier-Stokes equations and extend the geometry to three dimensions, in order to provide a more realistic design.

We will apply a topology optimization method to a coronary artery bypass anastomosis. As cost functionals, we will consider vorticity, pressure drop and energy dissipation, and the flow will be modeled in two dimensions using the unsteady incompressible Navier-Stokes system combined with Darcy's equation. We will use a flow rate waveform shape for a left anterior descending coronary artery from Thomas et al. [11] as inlet velocity.

1.3 Topology Optimization in Fluid Flow

The goal of topology optimization is to modify the shape and connectedness of a domain, such that the desired objective function is minimized. While shape optimization is limited

to determining an existing boundary, topology optimization can be used to design features within the domain, allowing new boundaries to be introduced into the design. This makes topology optimization more robust than for instance shape optimization, since less information about the goal shape is required.

Figure 1.4: An example of three steps of a fluid flow topology optimization procedure, collected from [12].

The underlying idea of topology optimization is to describe a geometry via its material distribution. By varying the material distribution, subdomains can be created or merged. See Figure 1.4 for an illustration of fluid flow topology optimization, with material distribution of fluid and solid. The density type method and level set method are two methods used in the implementation of topology optimization. The density method was first used for the design of stiffness and compliance mechanics and has been extended to many other physical fields. See Campeao et al. [13] and its references for examples.

For several optimization problems, the level set method has been successfully applied to compute optimal geometries without a priori knowledge about the connectedness of the domain. The level set method is a numerical technique for tracking interfaces and shapes in a domain using level sets. One of the advantages of the level set method is that you do not have to parametrize the shapes. This makes it well suited for topology optimization, for instance when a shape splits in two or a hole is developed. See for example Sethian and Wiegmann [14].

Topology optimization applied to fluid mechanics was first introduced in 2003 by Borrvall and Petersson [15] on stationary Stokes flow. Before this, only shape optimization had been applied to fluid mechanics. In [15] a relaxed Stokes flow formulation was proposed, where the impermeability works as a penalty factor on the flow velocity, inspired by Darcy's equation. Guest and Prévost [16] extended this work by applying a stabilized finite element formulation for the Stokes and Darcy equations, treating the solid phase as a porous medium.

Burger, Hackl and Ring [17] investigated the use of topological derivatives in combination with the level set method for optimization problems. Amstutz and Andrä [18] proposed a new algorithm for topology optimization using a level set method based only on the topological gradient. The level set method in fluid flow optimization was proposed by Challis and Guest [19]. This was suggested as an alternative to density-based approaches using the topological sensitivity of the Stokes flow.

Topology optimization for fluid mechanics was first extended to the Navier-Stokes equations by Gersborg-Hansen et al. [20] in 2005. They propose an analytical sensitivity

analysis using the adjoint method. Topological sensitivity was introduced by Sokolowski and Zochowski in 1999 [21]. The adjoint equations for channel flows were proposed by Othmer et al. [22], where a continuous adjoint formulation for the incompressible Navier-Stokes equations with Darcy porosity term was derived. Sá et al. [23] proposed a steepest descent algorithm based on the topological derivative formulation for the Stokes and Navier-Stokes equations and their adjoint formulations, obtaining an optimal design in few iterations.

Kreissl et al. [12] and Deng et al. [24, 25] applied topology optimization to the unsteady incompressible Navier-Stokes for low to moderate Reynolds numbers. In this thesis, our goal is to extend their work by combining the algorithm described in [23] with topological derivatives applied to unsteady fluid flow.

1.4 Organization of thesis

This thesis is organized as follows. In Chapter 2 we will present the tools of topology optimization. The topological derivative will be defined, and some basic examples related to fluid flow topology optimization are presented. In addition, the representation of fluid and solid will be be described through a level set function and impermeability constant. Chapter 3 describes the modelling of fluid flow in topology optimization. The optimization problem with respect to fluid flow problems will be presented, and a solution strategy for the optimization procedure, including algorithms, will be proposed. In Chapter 4, the adjoint method for both stationary and non-stationary partial differential equations is presented, and the adjoint equations for the steady and unsteady Navier-Stokes-Darcy systems are derived. The adjoint equations are incorporated in the derivation of the topological derivative with respect to the dissipation energy functional at the end of the chapter. The mathematical theory presented in these chapters are dimension-independent.

In Chapter 5, the Navier-Stokes-Darcy equations are discretized in space using the finite element method, and in time using a backwards scheme. Two different projection methods, Chorin's scheme and the Incremental Pressure Correction scheme, will be presented, and the time-dependent adjoint equations from Chapter 4 are discretized in time. At the end of the chapter, numerical software used for discretization is presented. In Chapter 6 we will verify the discretization proposed in Chapter 5 in 2D, using the numerical software presented. In addition, we will investigate the effect of Darcy's equation combined with the Navier-Stokes equations. We will also verify the numerically computed gradient, with respect to the topological derivative proposed in Chapter 4. In Chapter 7, we will present numerical experiments based on examples from the literature, also in 2D. In addition, the model and result of the optimal bypass anastomosis will be presented and discussed.

Chapter 2

Topology Optimization

Introducing the field of topology optimization, we will consider a very general case. Assume we have an open and bounded domain $\Omega \subset \mathbb{R}^d$, with $d = 2$ being the dimension of the space in our case, although the presentation is independent of the dimension. The optimization problem can be stated as

$$\begin{aligned} &\text{Minimize} && J, \\ &\text{subject to} && F = 0 \text{ on } \Omega, \end{aligned} \tag{2.1}$$

where J is the goal functional and F is a set of state equations.

We will apply optimization techniques based on the topological derivative and level sets to find the optimal topology. For a better understanding of the topological derivative, we will first explain the basic concept, before we illustrate it with an example using material distribution. At the end of this chapter, we will give a brief introduction to the level set method in topology optimization.

2.1 The Topological Derivative

The topological derivative is a measure of the change in a topology with respect to some functional. A topological change in Ω can be described by the removal of a circular ball $B_\varepsilon(\hat{x})$ with radius ε centered at an arbitrary point $\hat{x}$ in the interior of Ω, $\hat{x} \in \text{int}(\Omega)$. The perturbed domain is then given by $\Omega_\varepsilon(\hat{x}) = \Omega \backslash \overline{B_\varepsilon(\hat{x})}$. See Figure 2.1 for an illustration of the perturbation of a domain.

Now, let us introduce a functional J associated with the unperturbed domain, $J = J(\Omega)$, and find its topological derivative. The topological derivative is defined by the first order correction of the asymptotic expansion of the functional with respect to a small perturbation in the domain. For the perturbed domain, Ω_ε, we assume the associated functional

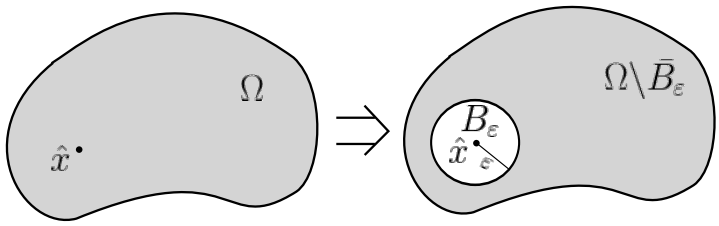


Figure 2.1: An illustration of a topological perturbation of a domain Ω by the removal of a circular ball B_ε with radius ε.

$J(\Omega_\varepsilon)$ admitting the following topological asymptotic expansion [26]

$$J(\Omega_\varepsilon(\hat{x})) = J(\Omega) + f(\varepsilon) D_T J(\hat{x}) + o(f(\varepsilon)). \tag{2.2}$$

The term $f(\varepsilon) D_T J(\hat{x})$ can be seen as the first order correction of $J(\Omega)$ to approximate $J(\Omega_\varepsilon(\hat{x}))$, and the function $f : \mathbb{R}^+ \to \mathbb{R}^+$ must have the property $f(\varepsilon) \to 0$ when $\varepsilon \to 0^+$. $D_T J(\hat{x})$ describes the change in J with respect to a perturbation at $\hat{x}$, and is called the topological derivative of J. From the expansion of J in equation (2.2) we obtain the general definition for the topological derivative

$$D_T J(\hat{x}) := \lim_{\varepsilon \to 0} \frac{J(\Omega_\varepsilon(\hat{x})) - J(\Omega)}{f(\varepsilon)}. \tag{2.3}$$

A basic illustration of the topological derivative is to consider the functional associated with the volume of a domain,

$$J(\Omega) = |\Omega| = \int_\Omega 1. \tag{2.4}$$

The corresponding perturbed volume functional, where a ball of radius ε has been removed, is given as

$$J(\Omega_\varepsilon) = \int_\Omega 1 - \int_{B_\varepsilon} 1 = J(\Omega) - \pi\varepsilon^2. \tag{2.5}$$

Comparing the topological expansion in (2.5) to equation (2.2), it is trivial to identify $f(\varepsilon) = \pi\varepsilon^2$ and $D_T J = -1$ as the topological derivative. In this particular example, the order term in (2.2) is equal to zero, and the topological derivative $D_T J$ is independent of $\hat{x}$. The negative sign appears because we remove a part of the domain.

2.2 Topological Derivative with respect to a Fluid Subdomain

Restricting ourselves to fluid flow topology optimization, let us investigate the distribution of fluid and solid material in a domain Ω. Denote a fluid subdomain by ω and the remaining

solid subdomain as $\Omega \backslash \bar{\omega}$. Consider now the topological derivative with respect to the fluid domain, i.e. let the functional J be $J = |\omega|$. If $\hat{x}$ is in the fluid subdomain ω, the perturbation would be to remove a ball B_ε from ω, and we would obtain the same topological derivative as in the basic volume example, namely $D_T J = -1$. If, on the other hand, $\hat{x}$ is stationed in the solid subdomain $\Omega \backslash \bar{\omega}$, the perturbation would be to add a ball B_ε to the fluid domain, and the sign would be positive. Hence, the topological derivative of $|\omega|$ is described by

$$D_T |\omega| = k(\hat{x}), \tag{2.6}$$

with

$$k(\hat{x}) = \begin{cases} +1 & \text{if } \hat{x} \in \Omega \backslash \omega \\ -1 & \text{if } \hat{x} \in \omega. \end{cases} \tag{2.7}$$

We will make use of this definition of k when we derive the topological derivative with respect to a certain functional in Section 4.7.

2.3 Level Set Representation of Fluid and Solid

The level set method is a numerical technique for tracking interfaces and shapes in a domain using level sets [27]. A level set can be expressed as the set where some function f is bounded by a constant c, $\{x \,|f(x) < c\}$. One of the advantages of the level set model is thatit is not needed to parametrize the shapes since the shape is implicitly defined. This makes it well suited for topology optimization, for instance when a shape is divided into two pieces or a hole is developed [14].

To model the distribution of fluid and solid in the domain Ω, we introduce a fictitious time t, such that the evolvement of the domain can be viewed as a family $(\Omega(t))_{t \geq 0}$. This evolution is represented by a level set function $\psi(x, t)$ such that

$$\begin{cases} \psi(t, x) \leq 0 & \text{for } x \in \omega(t) \\ \psi(t, x) > 0 & \text{for } x \in \Omega \backslash \bar{\omega}(t), \end{cases} \tag{2.8}$$

where $\omega(t)$ is the time-dependent fluid domain. Note that $k(x) = \text{sign}(\psi(t, x))$ from (2.7). Hence, the level set function ψ can be used as a design variable for the fluid-solid distribution.

A common approach is to model the evolution of the level set function by the Hamilton-Jacobi equation [17, 19], also known as the level set equation,

$$\frac{\partial \psi}{\partial t} = -v |\nabla \psi|, \tag{2.9}$$

where v is the speed of the moving domain boundary. In [17] Burger et al. investigate the use of topological derivatives in combination with the level set function (2.9) in a new

level set method. Another approach, first proposed in [18], is to use an evolution equation in combination with the level set function, based only on the concept of the topological derivative $D_T J$,

$$\frac{\partial \psi}{\partial t} = P(D_T J), \tag{2.10}$$

where $P(\cdot)$ is some function. We will use this formulation and discuss it, in the light of optimality conditions, in Section 3.3.

2.4 Material Distribution

We will distribute solid or fluid material at each point x of the domain Ω by letting an impermeability parameter $\alpha(x)$ vary between two non-negative constants,

$$\alpha(x) = \begin{cases} \alpha_L & \text{if } x \in \omega, \\ \alpha_U & \text{if } x \in \Omega \backslash \bar{\omega} \end{cases} \tag{2.11}$$

with $\alpha_U \gg \alpha_L \geq 0$. As in the previous section, ω denotes the fluid subdomain. We omit explicitly writing the time-dependency to simplify notation.

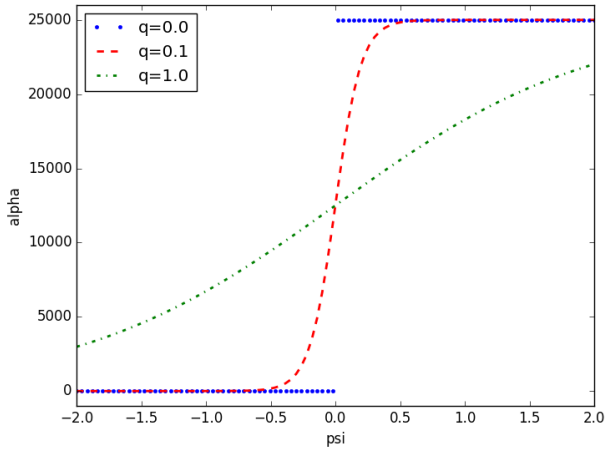


Figure 2.2: Interpolation of $\alpha(\psi)$ for different values of penalty q.

In some cases, we need the material distribution to be continuously differentiable on the domain, for instance when computing topological derivatives numerically. The impermeability α can in this case be represented by an interpolation function

$$\alpha(\psi) = \alpha_L + (\alpha_U - \alpha_L)\frac{1}{1 + e^{-\psi + q}} \tag{2.12}$$

where ψ is the level set function defined in (2.8) and q is a penalty parameter that restricts the elements with intermediate impermeability. An illustration of the interpolation can be seen in Figure 2.2, with $\alpha_L = 2.5 \cdot 10^{-4}$ and $\alpha_U = 2.5 \cdot 10^{4}$. Letting the penalty parameter $q = 0$ corresponds to the case of no interpolation.

2.5 Optimality Conditions

In order to carry out topology optimization, we need some basic optimality conditions. A locally necessary optimality condition is on the topological derivative defined in (2.3), is

$$D_T J(x) \geq 0 \quad \forall x \in \Omega. \tag{2.13}$$

The topological gradient with respect to the optimization problem, hereby denoted g, is a function that measures the sensitivity of J with respect to a change in the topology, i.e. the permeability. Hence, if the impermeability α changes from α_L to α_U, then the fluid volume $|\omega|$ is increased, and the gradient must be positive, as opposed to when it changes from α_U to α_L. Then $|\omega|$ is decreased, and the gradient must be negative, see [28] for more details. The gradient g needs to be a descent direction, so we define it as

$$g(x) = \begin{cases} -D_T J(x) & \text{for } x \in \omega \\ +D_T J(x) & \text{for } x \in \Omega \backslash \bar{\omega} \end{cases} \tag{2.14}$$

In this way, the local optimality condition (2.13) can be stated as

$$\begin{cases} g(x) \leq 0 & \text{for } x \in \omega \\ g(x) > 0 & \text{for } x \in \Omega \backslash \bar{\omega}. \end{cases} \tag{2.15}$$

This optimality condition on the gradient will be used in the optimization algorithm in Chapter 3.

Chapter 3

Fluid Flow Optimization

The optimization problem for the flow of a fluid through a channel is the same as in the general case 2.1,

$$\begin{aligned} &\text{Minimize} && J(u,\alpha) \\ &\text{subject to} && F(u,\alpha) = 0 \quad \text{on } \Omega, \end{aligned} \tag{3.1}$$

where $J(u,\alpha)$ is a functional and $F(u,\alpha)$ is a set of equations. The parameter u describes the movement of the fluid in the channel, and α is a control for fluid-solid distribution. Hence, in order to carry out the optimization, we need to calculate the movement of the fluid. Fluid motion can be described by the Navier-Stokes equations, which is an example of equations on the form $F(u,\alpha) = 0$. In order to incorporate the control α, we will use the concept of porous media flow, which can be modelled by Darcy's equation. In the following section we will combine these two systems of equations. At the end of this chapter, we will present a specific optimization problem, and a solution strategy for the problem.

3.1 Combining Navier-Stokes' and Darcy's Equations

To describe the fluid flow, let us consider the unsteady Navier-Stokes equations for an incompressible fluid.

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \mathbf{f} \qquad \text{in } \Omega, \tag{3.2}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega, \tag{3.3}$$

where $t \in (0,T)$ is the time, $\mathbf{u}$ is the velocity vector, p is the pressure, ν is the kinematic viscosity, ρ is the density, $\mathbf{f}$ is the body force and Ω is the domain on which the fluid flows. In addition we have suitable initial and boundary conditions, and these are discussed

below. Because we want to model a fluid flow on subdomains with different materials, as proposed in Section 2.4, we need a way to describe this. The approach used here is to model certain subdomains as a porous medium using Darcy's equation,

$$\alpha \mathbf{u} + \nabla p = \mathbf{0}, \tag{3.4}$$

where α is the impermeability, or the inverse permeability, see [29]. The term α is also called the Brinkman term after Brinkman [30]. The impermeability is a measure of the ability of a porous material to allow fluids to pass through it. Low impermeability gives rapidly moving fluid while high impermeability gives the opposite. In [15, 20, 25] it is proposed to add an artificial friction force into the body force term in (3.2)

$$\mathbf{f} = -\alpha \mathbf{u}. \tag{3.5}$$

The coupling of Darcy's law with the unsteady Navier-Stokes equation has been proved to have a unique solution by Cesmelioglu et al.[31]. The inclusion of the Darcy term $\alpha\mathbf{u}$ in the Navier-Stokes system allows for punishing counterproductive cells – the central component of topology optimization [22]. In this way we can model a domain with fluid and solid subdomains simply by letting the impermeability α vary, as described in Section 2.4.

3.2 The Complete Navier-Stokes-Darcy (NSD) system

Including Dirichlet and Neumann boundary conditions on $\partial\Omega_D$ and $\partial\Omega_N$, respectively, the complete Navier-Stokes-Darcy system on strong form yields: find $(\mathbf{u}, p)$ such that

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \alpha\mathbf{u} + \frac{1}{\rho}\nabla p = \mathbf{0} \qquad \text{in } \Omega, \tag{3.6}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega, \tag{3.7}$$

$$\mathbf{u} = \mathbf{u}_D \qquad \text{on } \partial\Omega_D, \tag{3.8}$$

$$\nu(\nabla\mathbf{u})\cdot\mathbf{n} - \frac{1}{\rho}p\mathbf{n} = \mathbf{g}_N \qquad \text{on } \partial\Omega_N, \tag{3.9}$$

$$\mathbf{u}(t_0) = \mathbf{u}_0 \qquad \text{in } \Omega, \tag{3.10}$$

where $\mathbf{n}$ is the outward unit normal, $\mathbf{g}_N$ is the traction and $\mathbf{u}_0$ is the initial velocity at time t_0. We will from here on refer to (3.6) - (3.9) as the Navier-Stokes-Darcy system or the NSD system.

In fluid flow modelling, the inflow velocity is commonly known, hence a Dirichlet boundary condition is applied to the inflow boundary. On the wall, a no-slip Dirichlet boundary condition with $\mathbf{u}_D = \mathbf{0}$ is applied. For the outflow, we either know the outflow velocity, like for instance in [19, 15, 23], or if the outflow profile cannot be determined a priori, one can use a traction-free boundary condition, $\mathbf{g}_N = \mathbf{0}$, at the outlets [16, 22, 12, 24]. This is known as a do-nothing boundary condition [32], as it naturally appears when doing

integration by parts. In particular, this open boundary condition will be used on the outlets in the numerical experiments presented in Chapter 7.

When solving partial differential equations using a finite element method, as we want here, it is necessary to set up the weak formulation and solve that instead. Then the equations are not required to hold absolutely, but has weak solutions with respect to certain test functions. For the weak form, let

$$\mathbf{U} = \{\mathbf{u} \in H^1(\Omega;\mathbb{R}) : \mathbf{u}|_{\partial\Omega_D} = \mathbf{u}_D,\ \mathbf{u}(t_0) = \mathbf{u}_0\} \tag{3.11}$$

$$\mathbf{V} = \{\mathbf{u} \in H^1(\Omega;\mathbb{R}) : \mathbf{u}|_{\partial\Omega_D} = \mathbf{0}\} \tag{3.12}$$

$$Q = L^2(\Omega), \tag{3.13}$$

denote the velocity spaces and pressure space, respectively, and let the test functions be defined as $\mathbf{v} \in \mathbf{V}$ and $q \in Q$. The NSD problem in weak form with initial and boundary conditions then yields: find $\mathbf{u} \in \mathbf{U}$ and $p \in Q$ such that

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} + \nu \int_\Omega \nabla\mathbf{u} \cdot \nabla\mathbf{v} + \int_\Omega (\mathbf{u}\cdot\nabla)\mathbf{u} \cdot \mathbf{v} + \int_\Omega \alpha\mathbf{u}\cdot\mathbf{v} - \int_\Omega \frac{1}{\rho}(\nabla\cdot\mathbf{v})p + \int_{\partial\Omega_N} p\mathbf{n}\cdot\mathbf{v} - \int_{\partial\Omega_N} \nu\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{v} = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}, \tag{3.14}$$

$$\int_\Omega (\nabla\cdot\mathbf{u})q = 0 \quad \forall\, q \in Q. \tag{3.15}$$

Note that we have included the Neumann boundary conditions (3.9) in the integral (3.14). We will later on use the weak form in the derivation of the topological derivative and in the numerical discretization.

3.3 The Optimization Problem

The optimization problem is to minimize a specific functional J in a fluid flow channel by optimizing the channel shape. Given inflow and outflow conditions on the boundary of a domain Ω, the aim is to distribute fluid and solid material to obtain the optimal shape with respect to J. In addition, a constraint is introduced to put a bound on the total volume of fluid.

If J is the cost functional to be minimized, the optimization problem can be stated in the same way as in (2.1):

$$\begin{aligned} &\text{Minimize} && J(\mathbf{u}, p, \alpha) \\ &\text{subject to} && \mathbf{F}(\mathbf{u}, p, \alpha) = \mathbf{0} \quad \text{on } \Omega. \end{aligned} \tag{3.16}$$

where $\mathbf{u}$ and p stands for velocity and pressure, respectively, and the impermeability parameter α serves the purpose of the design variable. $\mathbf{F}$ is the system of state equations, more specifically the incompressible Navier-Stokes-Darcy system,

$$\mathbf{F}(\mathbf{u}, p, \alpha) = \begin{pmatrix} \dfrac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \alpha\mathbf{u} + \nabla p \\ \nabla\cdot\mathbf{u} \end{pmatrix}. \tag{3.17}$$

We have left out the boundary and initial conditions for readability. These will be treated explicitly in each case. Now that we have formulated the equations needed to model our problem, we can proceed to the solution strategy for the optimization problem.

In order to solve the optimization problem, we will use the topological derivative of the objective functional J incorporated in a gradient-based algorithm. The topological derivative describes the change in J when the topology changes, and can be used as a steepest descent direction in order to minimize the objective function. The topological derivative is dependent on the movement of the fluid through $(\mathbf{u}, p)$, but also their adjoint states $(\mathbf{u}^a, p^a)$. Thus, we must solve both the Navier-Stokes-Darcy system and its adjoint system in each iteration. This will be described thoroughly in Chapter 4 and 5.

The level set function, ψ, defines the state of the material distribution. It will, as defined in (2.8), be non-positive in the fluid subdomain and positive in the solid subdomain. Hence, the impermeability parameter α, describing the material constant, is directly dependent on ψ. From the optimality condition (2.15), we see that it is fulfilled if the gradient g and ψ coincide, i.e. $g = \tau\psi$ for some $\tau > 0$. Hence, in the fluid domain ω we wish to increase ψ where $g > 0$, and in the solid domain $\Omega\backslash\bar{\omega}$ we wish to decrease ψ where $g \leq 0$.

Defining the level-set function ψ as proposed in [18], we have the equation

$$\frac{\partial \psi}{\partial t} = g - \langle g, \psi \rangle \psi \tag{3.18}$$

where the right hand side is the complement of the orthogonal projection of g onto ψ. Here, and in further calculations, we refer to the $L^2(\Omega)$ inner product when writing $\langle \cdot, \cdot \rangle$ and let $\| \cdot \|$ denote the corresponding norm. If ψ tends to a stationary point, then (3.18) is zero, and the level-set function ψ and g must coincide. Hence we have a local optimum.

Assume now that we can apply an Euler forward method in time to (3.18),

$$\psi_{i+1} = \psi_i + \Delta t P_i, \tag{3.19}$$

where P_i is the projection

$$P_i = g_i - \langle g_i, \psi_i \rangle \psi_i. \tag{3.20}$$

Here we have denoted $\psi(t_i) = \psi_i$, and the same applies to g_i. We want the new level set function to be normalized, $\|\psi_{i+1}\| = 1$, so we can represent it by a trigonometric relation. Let

$$\theta_i = \arccos\left(\frac{\langle g_i, \psi_i \rangle}{\|g_i\| \|\psi_i\|} \right) \tag{3.21}$$

be the non-oriented angle between ψ_i and g_i. For this value of θ_i there exists a step-size $\kappa_i \in [0, 1]$ such that the new normalized level-set function can be written as

$$\psi_{i+1} = \cos(\kappa_i \theta_i)\psi_i + \sin(\kappa_i \theta_i) \frac{P_i}{\|P_i\|}, \tag{3.22}$$

$$\|P_i\| = \|g_i - \langle g_i, \psi_i \rangle \psi_i\| = \sin\theta_i \|g_i\|, \tag{3.23}$$

where P_i is defined in (3.19). Note here that we get rid of the explicit time dependency. Inserting the expression for P_i into (3.22) we obtain

$$\psi_{i+1} = \left(\cos(\kappa_i\theta_i) - \frac{\cos\theta_i \sin(\kappa_i\theta_i)}{\sin\theta_i}\right)\psi_i + \frac{\sin(\kappa_i\theta_i)}{\sin\theta_i}\frac{g_i}{\|g_i\|} \tag{3.24}$$

$$= \frac{1}{\sin\theta_i}\left(\sin\left((1-\kappa_i)\theta_i\right)\psi_i + \sin(\kappa_i\theta_i)\frac{g_i}{\|g_i\|}\right). \tag{3.25}$$

We now have a way of updating the level-set function based on the topological gradient g_i, the angle θ_i between g_i and the previous level-set function ψ_i, and a step size κ_i.

Since the optimization problem is non-convex [23], the solution is dependent on the initial state, and a global minimum is not guaranteed. When θ reaches below some numerical tolerance ε_θ, the solution is considered optimal and the algorithm terminates.

3.4 Fluid Volume Penalization

The solution strategy presented above has no constraints on the size of the fluid volume. In fluid flow channel design optimization, it is reasonable to introduce a linear penalty parameter β on the fluid domain volume, such that the new functional to be minimized will be

$$J(\mathbf{u}, p, \alpha) + \beta|\omega|, \tag{3.26}$$

with topological derivative

$$D_T J(\mathbf{u}, p, \alpha) + k(x)\beta, \tag{3.27}$$

were the last term is collected from (2.6). The topological derivative of a specific functional will be derived in Chapter 4. One drawback with this approach is the introduction of an additional parameter β specific for the optimization problem, which must be adjusted in every case.

3.5 Backtracking Line Search

The step size κ_i has to be determined for each iteration. A backtracking line search with the Armijo condition [33] has been investigated, with the topological gradient g as search direction. When using the topological derivative, this shows poor results, as the inner product $\langle g, g\rangle$ takes on a very large value, hence it is hard to find a descent direction fulfilling the Armijo condition. Instead we implement a line search where κ_i is decreased until the objective function decreases, $J(\mathbf{u}, p, \alpha(\psi_{i+1})) < J(\mathbf{u}, p, \alpha(\psi_i))$. If the step size κ_i reaches below some numerical tolerance ε_κ, the solution at a local minimum, and the algorithm will terminate.

One drawback of a backtracking line search is that it requires us to solve the Navier-Stokes-Darcy system for every update of κ_i, which can be very costly. Therefore, an approach of letting $\kappa_i = 1.0$ until a threshold $\theta < \theta_\kappa$ is obtained, can be considered, since according to [23], the step size parameter generally starts to diminish only at the end of the iterative process, that is, when θ_i is small. It will be stated in each experiment in Chapter 7 whether this approach is applied or not. The line search is inspired by [28], and is performed as follows:

Algorithm 1: Backtracking Line Search

Choose initial step $\kappa_0 = 1.0$
At iteration $i \geq 1$, update $\kappa_i = \min(1.0,\ 1.5\kappa_{i-1})$
while $J(\psi_i) > J(\psi_{i-1})$ **do**
  $\kappa_i \leftarrow \kappa_i/2$
  if $\kappa_i < \varepsilon_\kappa$ **then**
    $J(\psi_i)$ local minimum, terminate
  end if
  update ψ_i according to (3.25)
  solve the Navier-Stokes-Darcy system to obtain $\mathbf{u}$, p
end while

3.6 Gradient-Based Algorithm

The optimization algorithm is a gradient-based method proposed in Sá et al.[23]. In the numerical algorithm, the gradient g defined by (2.14) will for the Navier-Stokes-Darcy system be

$$g(x) = \text{sign}(\psi)(D_T J(\mathbf{u}, p, \alpha) + k(x)\beta) \tag{3.28}$$
$$= k(x) D_T J(\mathbf{u}, p, \alpha) + \beta, \tag{3.29}$$

since $\text{sign}(\psi) = k(x)$. $D_T J(\mathbf{u}, p, \alpha)$ will also be dependent on the adjoint states of the velocity and pressure, $\mathbf{u}^a$ and p^a, respectively, so both the primal and adjoint system must be solved in each iteration. To solve these systems, knowledge of α which depends on ψ is necessary. The adjoint equations for the Navier-Stokes-Darcy system will be derived in Chapter 4. Combining the solution strategy presented above with the fluid volume penalization and backtracking line search, the gradient-based algorithm can be stated as follows:

The implementation of the gradient-based algorithm can be found in Appendix A.3.

Algorithm 2: Gradient-Based Algorithm

Initialize ψ_0 s.t. $\|\psi_0\| = 1.0$, $\kappa_0 = 1.0$
while $\theta > \varepsilon_\theta$ **do**
  $\alpha_i \leftarrow \alpha(\psi_i)$
  solve Navier-Stokes-Darcy system to obtain $\mathbf{u}, p$
  solve adjoint system to obtain $\mathbf{u}^a, p^a$
  $g_i \leftarrow k(x) D_T J(\mathbf{u}, p, \mathbf{u}^a, p^a, \alpha) + \beta$
  $\theta \leftarrow \arccos\left(\frac{\langle g_i, \psi_i \rangle}{\|g_i\| \|\psi\|}\right)$
  $\kappa_i \leftarrow$ backtracking line search
  $\psi_{i+1} \leftarrow \frac{1}{\sin\theta_i}\left(\sin\left((1-\kappa_i)\theta_i\right)\psi_i + \sin(\kappa_i\theta_i)\frac{g_i}{\|g_i\|}\right)$
end while

Chapter 4

The Adjoint Method and Topological Derivative

In order to carry out fluid flow optimization, we need to find the direction in which some functional of interest J decreases. This direction can be calculated using topological derivatives. Let us consider the optimization problem (3.1) where $F(u, \alpha) = 0$ is a system of partial differential equations and $J(u, \alpha)$ is the functional of interest. Since α is our control parameter, we want to evaluate the change in $J(u, \alpha)$ with respect to α by calculating the derivative

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\partial J}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}\alpha} + \frac{\partial J}{\partial \alpha}. \tag{4.1}$$

In this thesis we will apply a gradient-based optimization algorithm to find an optimal solution. Because the number of parameters in fluid flow optimization is generally quite large, this motivates the use of so-called adjoints. A nice property for the adjoints is that the number of computations is independent of the number of parameters for which you want the gradient. Incorporating the adjoint solution in the topological gradient, the computational costs will decrease significantly.

4.1 The Concept of Adjoints

Let us start with the basic definition of adjoint operators. The adjoint of an operator A, also called the Hermitian adjoint, is denoted A^*. Suppose H is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_H$ and $A : H \to H$. Then the adjoint operator A^* is satisfying

$$\langle Au, v \rangle_H = \langle u, A^* v \rangle_H. \tag{4.2}$$

For more details, see for instance [34] Chapter 6. Suppose now that A is the time-differential operator $\partial/\partial t$ with initial conditions at $t = 0$ integrated over a time interval,

$$\langle Au, v\rangle = \int_0^T \langle \frac{\partial}{\partial t} u, v\rangle \mathrm{d}t + \langle u, v\rangle|_{t=0}. \tag{4.3}$$

By evaluating the inner product, integration by parts shows that

$$\int_0^T \langle \frac{\partial}{\partial t} u, v\rangle \mathrm{d}t + \langle u, v\rangle|_{t=0} = -\int_0^T \langle u, \frac{\partial}{\partial t} v\rangle \mathrm{d}t + \langle u, v\rangle|_{t=T}. \tag{4.4}$$

Hence, the adjoint of the operator is

$$\langle u, A^* v\rangle = -\int_0^T \langle u, \frac{\partial}{\partial t} v\rangle \mathrm{d}t + \langle u, v\rangle|_{t=T}. \tag{4.5}$$

We note the negative sign in (4.4), and this happens for all odd derivatives. We will make use of this when deriving the adjoint equations for the non-stationary Navier-Stokes-Darcy system in Section 4.6.

Another approach for understanding adjoints is the adjoint method in optimization. The adjoint method formulates the gradient of a functional by evaluating the dual form of the optimization problem. It can be used to calculate the gradient efficiently. Let us investigate the adjoint of the optimization problem (2.1) by looking at the derivative of the functional J in equation (4.1). The following is inspired by the work of Funke and Farrell [35].

Let us assume that the derivative of the PDE constraint with respect to the control parameter α,

$$\frac{\mathrm{d}F}{\mathrm{d}\alpha} = \frac{\partial F}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}\alpha} + \frac{\partial F}{\partial \alpha} = 0, \tag{4.6}$$

is solvable, i.e. $(\mathrm{d}F/\mathrm{d}u)^{-1}$ exists. We can then insert the expression

$$\frac{\mathrm{d}u}{\mathrm{d}\alpha} = -\left(\frac{\partial F}{\partial u}\right)^{-1}\frac{\partial F}{\partial \alpha} \tag{4.7}$$

into (4.1), to obtain a new expression for derivative of the functional J with respect to the control α,

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = -\frac{\partial J}{\partial u}\left(\frac{\partial F}{\partial u}\right)^{-1}\frac{\partial F}{\partial \alpha} + \frac{\partial J}{\partial \alpha}. \tag{4.8}$$

Let us now assign the variables differentiated with respect to u to a new variable λ such that

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = -\lambda^T\frac{\partial F}{\partial \alpha} + \frac{\partial J}{\partial \alpha}. \tag{4.9}$$

The new variable λ is called the adjoint variable, or Lagrangian multiplier, and is obtained by the adjoint equation

$$\frac{\partial F}{\partial u}^T \lambda = \frac{\partial J}{\partial u}^T. \tag{4.10}$$

By taking the transpose of $\partial F/\partial u$, we reverse the flow of information in the system. One can think of it as when a lower triangular matrix is transposed to an upper triangular matrix. Then the system has to be solved with a backwards method, so the information propagates the opposite way. $(\partial J/\partial u)^T$ is the source term of the adjoint equation, and makes the adjoint solution specific to a particular functional. The advantage of the adjoint method is that the equation (4.10) is linear in u, which makes it is possible to compute the gradient of J with respect to any parameter α, in (4.9), extremely efficiently when the number of parameters α is large.

The adjoint approach can be summarized in the following steps:

i) Choose an initial α.

ii) Solve $F(u, \alpha) = 0$ and equation (4.10) for u and λ, respectively.

iii) Calculate $\mathrm{d}J/\mathrm{d}\alpha$ from equation (4.9) and use as gradient in gradient-based optimization algorithm, which gives a new value of α.

iv) If $\mathrm{d}J/\mathrm{d}\alpha$ is close to zero, terminate. Otherwise, go to ii)

What we have done in this section, is equivalent to deriving the Lagrangian equations for the PDE constrained optimization problem. The first-order necessary optimality conditions for (2.1), also known as the KKT conditions [33], include the Lagrangian

$$\mathcal{L}(u, \alpha, \lambda) = J(u, \alpha) - \lambda^T F(u, \alpha). \tag{4.11}$$

where λ is the Lagrangian multiplier. This is the same as the adjoint variable from (4.10). The KKT conditions state that for all local minima $(\bar{u}, \bar{\alpha})$, there exists a $\bar{\lambda}$ such that

$$\frac{\partial \mathcal{L}}{\partial u}(\bar{u}, \bar{\alpha}) = \frac{\partial J}{\partial u}(\bar{u}, \bar{\alpha}) - \bar{\lambda}^T \frac{\partial F}{\partial u}(\bar{u}, \bar{\alpha}) = 0, \tag{4.12}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha}(\bar{u}, \bar{\alpha}) = \frac{\partial J}{\partial \alpha}(\bar{u}, \bar{\alpha}) - \bar{\lambda}^T \frac{\partial F}{\partial \alpha}(\bar{u}, \bar{\alpha}) = 0, \tag{4.13}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda}(\bar{u}, \bar{\alpha}) = F(\bar{u}, \bar{\alpha}) = 0. \tag{4.14}$$

The first equation (4.12) can be recognized as the transposed of the adjoint equation (4.10).

4.2 Stationary Adjoint Method

In this section we will present the adjoint method for the incompressible steady Navier-Stokes-Darcy system, (3.6) - (3.10), with respect to a general functional $J(\mathbf{u}, p, \alpha)$, using

the Lagrangian equation (4.11). Notice that the previous variable u now is represented by the vector $(\mathbf{u}, p)$ where $\mathbf{u}$ and p represent the velocity and pressure, respectively.

Let the Lagrangian multiplier be written as $\lambda = (\mathbf{u}^a, p^a)$ where $\mathbf{u}^a$ is the adjoint velocity and p^a is the adjoint pressure. Using the stationary Navier-Stokes-Darcy equations as residuals,

$$\mathbf{F}(\mathbf{u}, p, \alpha) = \begin{pmatrix} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \alpha\mathbf{u} + \nabla p \\ \nabla\cdot\mathbf{u} \end{pmatrix} = \mathbf{0}, \tag{4.15}$$

where $\mathbf{F}$ is a three-dimensional system corresponding to F in (4.11), we obtain the Lagrangian function

$$\begin{aligned} \mathcal{L}(\mathbf{u}, p, \alpha) &= J(\mathbf{u}, p, \alpha) - \int_\Omega (\mathbf{u}^a, p^a)^T \mathbf{F}(\mathbf{u}, p, \alpha) && (4.16)\\ &= J(\mathbf{u}, p, \alpha) - \int_\Omega \mathbf{u}^a \cdot (-\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \alpha\mathbf{u} + \nabla p) \\ &\qquad - \int_\Omega p^a(-\nabla\cdot\mathbf{u}). && (4.17) \end{aligned}$$

To calculate the gradient of $J(\mathbf{u}, p, \alpha)$ with respect to the control parameters α, we differentiate the Lagrangian (4.16) with respect to the variables $\mathbf{u}, p, \alpha$,

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha} &= \frac{\partial\mathcal{L}}{\partial\mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial\mathcal{L}}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha} + \frac{\partial\mathcal{L}}{\partial\alpha} && (4.18)\\ &= \frac{\partial J}{\partial\mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial J}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha} + \frac{\partial J}{\partial\alpha} \\ &\quad - \int_\Omega \frac{\mathrm{d}}{\mathrm{d}\alpha}(\mathbf{u}^a, p^a)^T \underbrace{\mathbf{F}(\mathbf{u}, p, \alpha)}_{=0} \\ &\quad - \int_\Omega (\mathbf{u}^a, p^a)^T \left(\frac{\partial\mathbf{F}}{\partial\mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial\mathbf{F}}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha} + \frac{\partial\mathbf{F}}{\partial\alpha}\right) = 0. && (4.19) \end{aligned}$$

Note that we have neglected the variation of the eddy viscosity, $\nu = \mu/\rho$, since we are in 2D. In the case of 3D, this is correct only for laminar flows. For turbulent flows, neglecting this variation is a common approximation, also known as frozen turbulence [22]. After rearranging (4.18) with respect to the implicit derivatives, we get

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha} &= \frac{\partial J}{\partial\alpha} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial\mathbf{F}}{\partial\alpha} \\ &\quad + \left(\frac{\partial J}{\partial\mathbf{u}} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial\mathbf{F}}{\partial\mathbf{u}}\right)\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} \\ &\quad + \left(\frac{\partial J}{\partial p} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial\mathbf{F}}{\partial p}\right)\frac{\mathrm{d}p}{\mathrm{d}\alpha}. \end{aligned} \tag{4.20}$$

We can choose the Lagrangian multipliers $(\mathbf{u}^a, p^a)$ such that the last two terms in (4.20) vanish simultaneously,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial\mathcal{L}}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha} = 0. \tag{4.21}$$

Hence, the sensitivity of the Lagrangian with respect to α, using that

$$\frac{\partial \mathbf{F}}{\partial \alpha} = \begin{pmatrix} \mathbf{u} \\ 0 \end{pmatrix}, \tag{4.22}$$

is simply

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha} = \frac{\partial J}{\partial \alpha} - \int_\Omega \mathbf{u}^{a\,T}\mathbf{u}. \tag{4.23}$$

4.3 Adjoint Equations for the Stationary NSD System

The condition in (4.21) gives rise to the adjoint equations. For the terms containing implicit derivatives to vanish from the Lagrangian sensitivity equation (4.20), it is trivial to see that we need them to vanish individually, giving the expressions

$$\frac{\partial J}{\partial \mathbf{u}} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial \mathbf{u}} = 0, \tag{4.24}$$

$$\frac{\partial J}{\partial p} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial p} = 0, \tag{4.25}$$

which are the adjoint equations on general form. Let the variables $\mathrm{d}\mathbf{u}$ and $\mathrm{d}p$ be arbitrary functions that represent small variations in $\mathbf{u}$ and p, respectively, such that $\mathbf{u} + \mathrm{d}\mathbf{u}$ and $p + \mathrm{d}p$ satisfy the Navier-Stokes-Darcy equations. Using the state equations in (4.15) as residuals $\mathbf{F}$, the derivatives with respect to $\mathbf{u}$ and p are

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}} = \begin{pmatrix} -\nu\Delta \mathrm{d}\mathbf{u} + (\nabla \mathrm{d}\mathbf{u})\mathbf{u} + (\nabla \mathbf{u})\mathrm{d}\mathbf{u} + \alpha \mathrm{d}\mathbf{u} + \nabla p \\ \nabla \cdot \mathrm{d}\mathbf{u} \end{pmatrix}, \tag{4.26}$$

$$\frac{\partial \mathbf{F}}{\partial p} = \begin{pmatrix} \nabla \mathrm{d}p \\ 0 \end{pmatrix}. \tag{4.27}$$

Let us decompose the functional J into interior and boundary parts, such that

$$J = \int_\Omega J_\Omega + \int_{\partial\Omega} J_{\partial\Omega}. \tag{4.28}$$

We then insert (4.26) and (4.27) into the general adjoint equations (4.24) and (4.25), respectively. After integration by parts such that the equations are decomposed into interior and boundary integrals, we obtain two new equations,

$$\begin{aligned} \int_\Omega \left(\frac{\partial J_\Omega}{\partial \mathbf{u}} - (-\nu\nabla^2 \mathbf{u}^a - (\mathbf{u}\cdot\nabla)\mathbf{u}^a - \nabla \mathbf{u}^a \cdot \mathbf{u} + \alpha \mathbf{u}^a + \nabla p^a) \right) \cdot \mathrm{d}\mathbf{u} \\ + \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial \mathbf{u}} - (\nu \mathbf{n}\cdot\nabla \mathbf{u}^a + \mathbf{u}^a(\mathbf{u}\cdot\mathbf{n}) + \mathbf{n}(\mathbf{u}^a\cdot\mathbf{u}) - p^a\mathbf{n}) \right) \cdot \mathrm{d}\mathbf{u} \\ + \int_{\partial\Omega} \nu(\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}\cdot\mathbf{u}^a = 0, \end{aligned} \tag{4.29}$$

$$\int_\Omega \left(\frac{\partial J_\Omega}{\partial p} - (-\nabla \cdot \mathbf{u}^a) \right) \mathrm{d}p + \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial p} - \mathbf{u}^a \cdot \mathbf{n} \right) \mathrm{d}p = 0. \tag{4.30}$$

The equations (4.29) and (4.30) should hold for any $\mathrm{d}\mathbf{u}$ and $\mathrm{d}p$, respectively, which satisfy the stationary Navier-Stokes-Darcy equations (4.15). This can only be accomplished if the integrals vanish individually, hence, the integrands must be zero. From the interior integrals we get out the adjoint equations for the stationary Navier-Stokes-Darcy system:

$$-\nu\Delta\mathbf{u}^a - \nabla\mathbf{u}^a \cdot \mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{u}^a + \alpha\mathbf{u}^a + \nabla p^a = \frac{\partial J_\Omega}{\partial \mathbf{u}}, \tag{4.31}$$

$$-\nabla \cdot \mathbf{u}^a = \frac{\partial J_\Omega}{\partial p}. \tag{4.32}$$

The disappearance of the boundary integrals in (4.29) and (4.30) give rise to the adjoint boundary conditions,

$$0 = \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial \mathbf{u}} - (\nu(\mathbf{n}\cdot\nabla)\mathbf{u}^a + \mathbf{u}^a(\mathbf{u}\cdot\mathbf{n}) + \mathbf{n}(\mathbf{u}^a\cdot\mathbf{u}) - p^a\mathbf{n}) \right) \cdot \mathrm{d}\mathbf{u} + \int_{\partial\Omega} \nu(\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}\cdot\mathbf{u}^a, \tag{4.33}$$

$$0 = \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial p} - \mathbf{u}^a \cdot \mathbf{n} \right) \mathrm{d}p. \tag{4.34}$$

Here, we have to distinguish between Dirichlet boundary conditions (3.8) and Neumann boundary conditions (3.9) for the primary values $\mathbf{u}$ and p.

4.4 Boundary Conditions for the Adjoint Equations

The Dirichlet boundary conditions have a fixed primal velocity $\mathbf{u} = \mathbf{u}_0$. For instance, the no-slip value $\mathbf{u}_0 = 0$, and at the inlet $\mathbf{u}_0$ takes some prescribed value. Consequently we have $\mathrm{d}\mathbf{u} = \mathbf{0}$ on the Dirichlet boundary, and the first integral in equation (4.33) vanishes. For the second part of (4.33), we decompose the adjoint velocity into a normal and tangential part

$$\mathbf{u}^a = \mathbf{u}^a_n + \mathbf{u}^a_t. \tag{4.35}$$

Using that

$$\nabla \cdot \mathrm{d}\mathbf{u} = (\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}_n + \nabla_t \cdot \mathrm{d}\mathbf{u}_t = 0, \tag{4.36}$$

with ∇_t being the tangential component of ∇, and that $\mathrm{d}\mathbf{u}_t = \mathbf{0}$ along the Dirichlet boundary. It follows that $(\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}_n = \mathbf{0}$, and $\nabla\cdot\mathrm{d}\mathbf{u} = (\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}_t$, and the boundary integrals of (4.33) and (4.34) reduce to

$$\int_{\partial\Omega} \nu(\mathbf{n}\cdot\nabla)\mathrm{d}\mathbf{u}_t \cdot \mathbf{u}^a_t = 0, \tag{4.37}$$

$$\int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial p} - \mathbf{u}^a \cdot \mathbf{n} \right) \mathrm{d}p = 0, \tag{4.38}$$

respectively. Hence, we can conclude that the adjoint Dirichlet boundary conditions must be

$$\mathbf{u}_t^a = \mathbf{0}, \tag{4.39}$$

$$\mathbf{u}_n^a = \frac{\partial J_{\partial\Omega}}{\partial p}. \tag{4.40}$$

We do not get any conditions on the adjoint pressure p^a. This is because the Navier-Stokes-Darcy equations only impose zero gradient boundary condition of the primal pressure p. This condition is also applied to the adjoint pressure, so

$$\mathbf{n} \cdot \nabla p^a = 0. \tag{4.41}$$

For the Neumann boundaries, we often have the open boundary conditions

$$\nu(\mathbf{n} \cdot \nabla)\mathrm{d}\mathbf{u} - p\mathbf{n} = \mathbf{0}, \tag{4.42}$$

at the outlet of a ducted flow. The pressure p is usually set to zero at the outlet, which means that $\mathrm{d}p = 0$, so equation (4.34) is therefore zero. This implies that $\nu(\mathbf{n} \cdot \nabla)\mathrm{d}\mathbf{u} = 0$, so the second integral of equation (4.33) vanishes. The remaining integral vanishes if the integrand is zero,

$$\frac{\partial J_{\partial\Omega}}{\partial \mathbf{u}} - \nu(\mathbf{n} \cdot \nabla)\mathbf{u}^a - \mathbf{u}^a(\mathbf{u} \cdot \mathbf{n}) - \mathbf{n}(\mathbf{u}^a \cdot \mathbf{u}) + p^a\mathbf{n} = 0. \tag{4.43}$$

Equation (4.43) is the adjoint boundary condition for open Neumann boundaries.

4.5 Time-dependent Adjoint Method

Assume now that the objective functional and the system of PDEs are time-dependent, such that the objective functional $\tilde{J}$ is defined as

$$\tilde{J} = \int_0^T J(\mathbf{u}(t), p(t), \alpha)\, \mathrm{d}t, \tag{4.44}$$

and the residuals are the unsteady incompressible Navier-Stokes-Darcy equations,

$$\mathbf{F}(\mathbf{u}, p, \alpha) = \begin{pmatrix} \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \alpha\mathbf{u} + \nabla p \\ \nabla \cdot \mathbf{u} \end{pmatrix} = \mathbf{0}. \tag{4.45}$$

For the time-dependent adjoint method [36], the Lagrangian can be written as

$$\mathcal{L}(\mathbf{u}, p, \alpha) = \int_0^T \left(J(\mathbf{u}(t), p(t), \alpha) - \int_\Omega (\mathbf{u}^a(t), p^a(t))^T \mathbf{F}(\mathbf{u}(t), p(t), \alpha) \right) \mathrm{d}t. \tag{4.46}$$

Now, we want to differentiate (4.46) with respect to α. To calculate the gradient of $J(\mathbf{u}(t), p(t), \alpha)$ with respect to the control parameters α, just like in the stationary case,

we differentiate with respect to the different variables. The difference is that the derivatives depend on time.

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha}(\mathbf{u},p,\alpha) = &\int_0^T \frac{\partial J}{\partial \alpha} + \frac{\partial J}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial J}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha}\mathrm{d}t \\ &- \int_0^T \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial \alpha} + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} + \frac{\partial \mathbf{F}}{\partial p}\frac{\mathrm{d}p}{\mathrm{d}\alpha}\mathrm{d}t. \end{aligned} \tag{4.47}$$

Rearranging (4.47) with respect to implicit derivatives, we get

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha}(\mathbf{u},p,\alpha) = &\int_0^T \frac{\partial J}{\partial \alpha} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial \alpha}\mathrm{d}t \\ &+ \int_0^T \left(\frac{\partial J}{\partial \mathbf{u}} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\right)\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha}\mathrm{d}t \\ &+ \int_0^T \left(\frac{\partial J}{\partial p} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial p}\right)\frac{\mathrm{d}p}{\mathrm{d}\alpha}\mathrm{d}t. \end{aligned} \tag{4.48}$$

We want to choose the time-dependent adjoints $\mathbf{u}^a, p^a$ so that the last two terms in (4.48) vanish.

4.6 Adjoint Equations for the Non-Stationary NSD System

Following the same procedure as for the stationary Navier-Stokes-Darcy system, we get the unstationary adjoint equations on the general form

$$\int_0^T \left(\frac{\partial J}{\partial \mathbf{u}} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\right)\mathrm{d}t = \mathbf{0}, \tag{4.49}$$

$$\int_0^T \left(\frac{\partial J}{\partial p} - \int_\Omega (\mathbf{u}^a, p^a)^T \frac{\partial \mathbf{F}}{\partial p}\right)\mathrm{d}t = 0. \tag{4.50}$$

Let the variables $\mathrm{d}\mathbf{u}$ and $\mathrm{d}p$ be arbitrary functions such that $\mathbf{u}+\mathrm{d}\mathbf{u}$ and $p+\mathrm{d}p$ satisfy the unsteady Navier-Stokes-Darcy equations. The derivatives with respect to $\mathbf{u}$ and p of the residuals $\mathbf{F}$ in (4.45) are

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial(\mathrm{d}\mathbf{u})}{\partial t} - \nu\Delta\mathrm{d}\mathbf{u} + (\nabla \mathrm{d}\mathbf{u})\mathbf{u} + (\nabla\mathbf{u})\mathrm{d}\mathbf{u} + \alpha\mathrm{d}\mathbf{u} + \nabla p \\ \nabla\cdot\mathrm{d}\mathbf{u} \end{pmatrix}, \tag{4.51}$$

$$\frac{\partial \mathbf{F}}{\partial p} = \begin{pmatrix} \nabla \mathrm{d}p \\ 0 \end{pmatrix}. \tag{4.52}$$

We insert (4.26) and (4.27) into the general unsteady adjoint equations (4.49) and use the decomposition of J as in (4.28) to find the weak form of the adjoint equations. Let

us in particular investigate what happens when we use integration by parts on the time-differentiated term in (4.51),

$$\int_0^T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{u}^a = \int_\Omega \int_0^T \left(\frac{\partial (\mathbf{u} \cdot \mathbf{u}^a)}{\partial t} - \mathbf{u} \cdot \frac{\partial \mathbf{u}^a}{\partial t} \right) \mathrm{d}t \tag{4.53}$$

$$= \int_\Omega (\mathbf{u} \cdot \mathbf{u}^a)|_{t=T} - (\mathbf{u}_0 \cdot \mathbf{u}^a)|_{t=0} - \int_0^T \int_\Omega \frac{\partial \mathbf{u}^a}{\partial t} \cdot \mathbf{u} \, \mathrm{d}t. \tag{4.54}$$

By moving the time-differentiation operator from the primal velocity $\mathbf{u}$ to the adjoint velocity $\mathbf{u}^a$, this can be interpreted as taking the adjoint of the primal inner product with initial condition at $t = 0$, analogously to the example in equation (4.4). We get a negative sign on the time-differentiated adjoint velocity, and an 'initial' condition at end time $t = T$, as the adjoint is computed backwards in time. Using integration by parts on the other terms, analogously to the stationary case, we get the equations

$$\begin{aligned} &- \int_0^T \int_\Omega \left(-\frac{\partial \mathbf{u}^a}{\partial t} - \nu \Delta \mathbf{u}^a - (\mathbf{u} \cdot \nabla)\mathbf{u}^a - \nabla \mathbf{u}^a \cdot \mathbf{u} + \alpha \mathbf{u}^a + \nabla p^a \right) \cdot \mathrm{d}\mathbf{u} \, \mathrm{d}t \\ &\qquad + \int_0^T \int_\Omega \frac{\partial J_\Omega}{\partial \mathbf{u}} \cdot \mathrm{d}\mathbf{u} \, \mathrm{d}t + \int_\Omega (\mathbf{u}^a \cdot \mathrm{d}\mathbf{u})|_{t=T} + \int_{\partial\Omega} \nu (\mathbf{n} \cdot \nabla) \mathrm{d}\mathbf{u} \cdot \mathbf{u}^a \, \mathrm{d}t \\ &+ \int_0^T \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial \mathbf{u}} - (\nu \mathbf{n} \cdot \nabla \mathbf{u}^a + \mathbf{u}^a (\mathbf{u} \cdot \mathbf{n}) + \mathbf{n}(\mathbf{u}^a \cdot \mathbf{u}) - p^a \mathbf{n}) \right) \cdot \mathrm{d}\mathbf{u} = \mathbf{0}, \end{aligned} \tag{4.55}$$

$$\int_0^T \int_\Omega \left(\frac{\partial J_\Omega}{\partial p} - (-\nabla \cdot \mathbf{u}^a) \right) \mathrm{d}p \, \mathrm{d}t + \int_0^T \int_{\partial\Omega} \left(\frac{\partial J_{\partial\Omega}}{\partial p} - \mathbf{u}^a \cdot \mathbf{n} \right) \mathrm{d}p \, \mathrm{d}t = 0. \tag{4.56}$$

The equations (4.55) and (4.56) hold for any $\mathrm{d}\mathbf{u}$ and $\mathrm{d}p$, respectively, that satisfy the non-stationary NSD equations (4.45). This can only be accomplished if the integrals vanish individually. Hence, the integrands must be zero, and from the interior integrals we get the adjoint equations for the non-stationary NSD system,

$$-\frac{\partial \mathbf{u}^a}{\partial t} - \nu \Delta \mathbf{u}^a - \nabla \mathbf{u}^a \cdot \mathbf{u} - (\mathbf{u} \cdot \nabla)\mathbf{u}^a + \alpha \mathbf{u}^a + \nabla p^a = \frac{\partial J_\Omega}{\partial \mathbf{u}}, \tag{4.57}$$

$$-\nabla \cdot \mathbf{u}^a = \frac{\partial J_\Omega}{\partial p}. \tag{4.58}$$

Notice that in equation (4.57), the first term has a negative sign. The condition on the adjoint velocity at $t = T$,

$$\mathbf{u}^a(T) = \mathbf{0} \quad \text{in } \Omega. \tag{4.59}$$

appears since the information is propagating backwards in time.

The boundary conditions are the same as in the stationary case, with noslip on the Dirichlet boundaries,

$$\mathbf{u}_t^a = 0, \tag{4.60}$$

$$\mathbf{u}_n^a = \frac{\partial J_{\partial\Omega}}{\partial p}, \tag{4.61}$$

and open Neumann boundaries,

$$\frac{\partial J_{\partial\Omega}}{\partial \mathbf{u}} - \nu(\mathbf{n}\cdot\nabla)\mathbf{u}^a - \mathbf{u}^a(\mathbf{u}\cdot\mathbf{n}) - \mathbf{n}(\mathbf{u}^a\cdot\mathbf{u}) + p^a\mathbf{n} = \mathbf{0}. \tag{4.62}$$

4.7 Exact Topological Derivative of the Dissipation Energy Functional

We will in this section make use of the adjoint formulation of the Navier-Stokes-Darcy system to derive the topological derivative for a given functional. The relation between the primal and adjoint solution will be presented in equation (4.82), and the topological derivative in equation (4.88).

The dissipation energy functional over a time interval can be expressed as

$$\tilde{J}(\mathbf{u},\alpha) = \int_0^T J(\mathbf{u},\alpha)\mathrm{d}t, \tag{4.63}$$

where $J(\mathbf{u},\alpha)$ can be expressed as

$$J(\mathbf{u},\alpha) = \int_\Omega \nu|\nabla\mathbf{u}|^2 + \int_\Omega \alpha|\mathbf{u}|^2. \tag{4.64}$$

Here $|\cdot|^2$ denotes the standard Euclidean l^2 inner product. Let us consider a small perturbation of the domain Ω, where we remove a ball of radius ε at an arbitrary point $\hat{x} \in \Omega$. The perturbed domain is defined as $\Omega_\varepsilon = \Omega\backslash B_\varepsilon(\hat{x})$. To look at the variation in (4.64), we need the dissipation energy functional associated with the perturbed domain,

$$J_\varepsilon(\mathbf{u}_\varepsilon,\alpha) = \int_{\Omega_\varepsilon} \nu|\nabla\mathbf{u}_\varepsilon|^2 + \int_{\Omega_\varepsilon} \alpha_\varepsilon|\mathbf{u}_\varepsilon|^2. \tag{4.65}$$

Here, $\alpha_\varepsilon = \gamma_\varepsilon\alpha$ is the topologically perturbed impermeability. The actual topological perturbation can be defined as

$$\gamma_\varepsilon = \begin{cases} 1 & \text{if } x \in \Omega\backslash\bar{B}_\varepsilon \\ \gamma & \text{if } x \in B_\varepsilon, \end{cases} \tag{4.66}$$

where γ is a contrast parameter that permits the impermeability to switch value. In this way, the permeability $\alpha(x)$ changes its value if $x \in B_\varepsilon(\hat{x})$ and remains unchanged otherwise. With the definition of α in (2.11), we can define the contrast parameter as

$$\gamma = \begin{cases} \alpha_L/\alpha_U & \text{if } x \in \Omega\backslash\bar{\omega} \\ \alpha_U/\alpha_L & \text{if } x \in \omega, \end{cases} \tag{4.67}$$

such that if x is located in the perturbed area, α will change from α_L to α_U or α_U to α_L, depending on whether the perturbed area is changed to solid or fluid domain.

Using the definition of the topological derivative, defined in (2.3), we need to calculate the the difference between the perturbed and unperturbed functionals,

$$J_\varepsilon(\mathbf{u}_\varepsilon) - J(\mathbf{u}) = \int_\Omega \nu \left(|\nabla \mathbf{u}_\varepsilon|^2 - |\nabla \mathbf{u}|^2\right) + \int_\Omega \left(\alpha_\varepsilon |\mathbf{u}_\varepsilon|^2 - \alpha |\mathbf{u}|^2\right). \tag{4.68}$$

To simplify the notation, we will denote $|\mathbf{v}|^2$ by $\mathbf{v}^2$. Let us expand the viscosity terms,

$$|\nabla \mathbf{u}_\varepsilon|^2 - |\nabla \mathbf{u}|^2 = 2\nabla \mathbf{u} \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) - 2\nabla \mathbf{u} \cdot \nabla \mathbf{u}_\varepsilon + (\nabla \mathbf{u})^2 + (\nabla \mathbf{u}_\varepsilon)^2 \tag{4.69}$$

$$= 2\nabla \mathbf{u} \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) + (\nabla \mathbf{u}_\varepsilon - \nabla \mathbf{u})^2, \tag{4.70}$$

and the Darcy terms,

$$\alpha_\varepsilon |\mathbf{u}_\varepsilon|^2 - \alpha |\mathbf{u}|^2 = \alpha_\varepsilon(\mathbf{u}_\varepsilon^2 - 2\mathbf{u}_\varepsilon \mathbf{u} + \mathbf{u}^2) + 2\alpha_\varepsilon \mathbf{u}_\varepsilon \mathbf{u} - \alpha_\varepsilon \mathbf{u}^2 - \alpha \mathbf{u}^2 \tag{4.71}$$

$$= \alpha_\varepsilon(\mathbf{u}_\varepsilon - \mathbf{u})^2 + 2\alpha\gamma_\varepsilon \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) + (\gamma_\varepsilon - 1)\alpha \mathbf{u}^2. \tag{4.72}$$

Using the definition of the contrast parameter γ in (4.66), the Darcy expansion (4.72) with integrals yields

$$\int_\Omega \alpha\gamma_\varepsilon \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) = \int_{\Omega \setminus \bar{B}_\varepsilon} \alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) + \int_{B_\varepsilon} \gamma\alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) \tag{4.73}$$

$$= \int_\Omega \alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) - \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}), \tag{4.74}$$

and combined with the viscosity terms, we get a new expression for (4.68)

$$\begin{aligned} J_\varepsilon(\mathbf{u}_\varepsilon) - J(\mathbf{u}) = {} & 2\nu \int_\Omega \nabla \mathbf{u} \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) + \nu \int_\Omega (\nabla \mathbf{u}_\varepsilon - \nabla \mathbf{u})^2 \\ & + \int_\Omega \alpha_\varepsilon |\mathbf{u}_\varepsilon - \mathbf{u}|^2 + 2 \int_\Omega \alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) \\ & - 2 \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) - \int_{B_\varepsilon} (1-\gamma)\alpha |\mathbf{u}|^2. \end{aligned} \tag{4.75}$$

We use the weak form of the NSD equations and corresponding adjoint weak form to derive the relation between the primal and adjoint velocities $\mathbf{u}$ and $\mathbf{u}^a$. Using this relation, which will be presented in equation (4.82), we can substitute some of the terms in (4.75), in order to derive the topological derivative from the definition (2.3).

The weak form of the NSD equations with divergence-free velocity spaces yields: find $\mathbf{u} \in \tilde{\mathbf{U}}$ such that

$$\nu \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} + \int_\Omega (\mathbf{u} \cdot \nabla)\mathbf{u} \cdot \mathbf{v} + \int_\Omega \alpha \mathbf{u} \cdot \mathbf{v} = 0 \quad \forall \mathbf{v} \in \tilde{\mathbf{V}}, \tag{4.76}$$

where $\tilde{\mathbf{U}} = \{\mathbf{v} \in \mathbf{U} : \nabla \cdot \mathbf{v} = 0\}$ and $\tilde{\mathbf{V}} = \{\mathbf{v} \in \mathbf{V} : \nabla \cdot \mathbf{v} = 0\}$. $\mathbf{U}$ and $\mathbf{V}$ are defined in (3.11). The pressure term disappears because the test-function $\mathbf{v}$ is divergence free. The weak form of the perturbed divergence-free system yields: find $\mathbf{u}_\varepsilon \in \tilde{\mathbf{U}}$ such that

$$\nu \int_\Omega \nabla \mathbf{u}_\varepsilon \cdot \nabla \mathbf{v} + \int_\Omega (\mathbf{u}_\varepsilon \cdot \nabla)\mathbf{u}_\varepsilon \cdot \mathbf{v} + \int_\Omega \alpha_\varepsilon \mathbf{u}_\varepsilon \cdot \mathbf{v} = 0 \quad \forall \mathbf{v} \in \tilde{\mathbf{V}}. \tag{4.77}$$

Let us now subtract (4.76) from (4.77) and use integration by parts on the non-linear term,

$$\begin{aligned}
&\nu \int_\Omega \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \nabla \mathbf{v} + \int_\Omega ((\mathbf{u}_\varepsilon \cdot \nabla)\mathbf{u}_\varepsilon - (\mathbf{u} \cdot \nabla)\mathbf{u}) \cdot \mathbf{v} + \int_\Omega (\alpha_\varepsilon \mathbf{u}_\varepsilon - \alpha \mathbf{u}) \cdot \mathbf{v} \\
&= \nu \int_\Omega \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \nabla \mathbf{v} + \int_\Omega ((\mathbf{u} \cdot \nabla)\mathbf{v} + (\nabla \mathbf{v})\mathbf{u}) \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) \\
&+ \int_\Omega (\nabla(\mathbf{u}_\varepsilon - \mathbf{u}))(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{v} + \int_\Omega \alpha(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{v} - \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}_\varepsilon \cdot \mathbf{v} = 0. \quad (4.78)
\end{aligned}$$

Setting the adjoint velocity as test function, $\mathbf{v} = \mathbf{u}^a$ in (4.78), we get the equation

$$\begin{aligned}
\nu \int_\Omega \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \nabla \mathbf{u}^a + \int_\Omega ((\mathbf{u} \cdot \nabla)\mathbf{u}^a + (\nabla \mathbf{u}^a)\mathbf{u}) \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) \\
+ \int_\Omega \alpha(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{u}^a \\
= (1-\gamma) \int_{B_\varepsilon} \alpha \mathbf{u}_\varepsilon \cdot \mathbf{u}^a - \int_\Omega (\nabla(\mathbf{u}_\varepsilon - \mathbf{u}))(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{u}^a. \quad (4.79)
\end{aligned}$$

The weak form of the divergence-free adjoint equation associated with the energy dissipation functional is: find $\mathbf{u}^a \in \tilde{\mathbf{U}}$ such that

$$\begin{aligned}
\nu \int_\Omega \nabla \mathbf{u}^a \cdot \nabla \mathbf{v} + \int_\Omega ((\mathbf{u} \cdot \nabla)\mathbf{u}^a + (\nabla \mathbf{u}^a)\mathbf{u}) \cdot \mathbf{v} + \int_\Omega \alpha \mathbf{u}^a \cdot \mathbf{v} \\
= \int_\Omega 2\nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} + \int_\Omega 2\alpha \mathbf{u} \cdot \mathbf{v} \quad \forall \mathbf{v} \in \tilde{\mathbf{V}}, \quad (4.80)
\end{aligned}$$

where the right hand side is the differentiated dissipation energy functional $\partial J / \partial \mathbf{u}$. Setting $\mathbf{v} = (\mathbf{u}_\varepsilon - \mathbf{u})$ in (4.80), we get the equation

$$\begin{aligned}
\nu \int_\Omega \nabla \mathbf{u}^a \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) + \int_\Omega ((\mathbf{u} \cdot \nabla)\mathbf{u}^a + (\nabla \mathbf{u}^a)\mathbf{u}) \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) \\
+ \int_\Omega \alpha \mathbf{u}^a \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) = 2\nu \int_\Omega \nabla \mathbf{u} \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) + 2 \int_\Omega \alpha \mathbf{u} \cdot (\mathbf{u}_\varepsilon - \mathbf{u}). \quad (4.81)
\end{aligned}$$

Comparing equation (4.79) to (4.81), we get the important relation

$$\begin{aligned}
2\nu \int_\Omega \nabla \mathbf{u} \cdot \nabla(\mathbf{u}_\varepsilon - \mathbf{u}) + 2 \int_\Omega \alpha \mathbf{u} \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) \\
= (1-\gamma) \int_{B_\varepsilon} \alpha \mathbf{u}_\varepsilon \cdot \mathbf{u}^a - \int_\Omega (\nabla(\mathbf{u}_\varepsilon - \mathbf{u}))(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{u}^a, \quad (4.82)
\end{aligned}$$

which can be substituted into equation (4.75) to obtain a new expression for the difference between the perturbed and unperturbed functionals,

$$\begin{aligned}
J_\varepsilon(\mathbf{u}_\varepsilon) - J(\mathbf{u}) = \ & (1-\gamma) \int_{B_\varepsilon} \alpha \mathbf{u}_\varepsilon \cdot \mathbf{u}^a - \int_{B_\varepsilon} (1-\gamma)\alpha |\mathbf{u}|^2 \\
& - \int_\Omega (\nabla(\mathbf{u}_\varepsilon - \mathbf{u}))(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{u}^a + \nu \int_\Omega (\nabla \mathbf{u}_\varepsilon - \nabla \mathbf{u})^2 \\
& + \int_\Omega \alpha_\varepsilon |\mathbf{u}_\varepsilon - \mathbf{u}|^2 - 2 \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}). \quad (4.83)
\end{aligned}$$

We collect all the terms containing $(\mathbf{u}_\varepsilon - \mathbf{u})$ in the expression

$$\begin{aligned}\mathcal{E}(\varepsilon) = & \int_\Omega |\nabla \mathbf{u}_\varepsilon - \nabla \mathbf{u}|^2 + \int_\Omega \alpha_\varepsilon |\mathbf{u}_\varepsilon - \mathbf{u}|^2 - 2\int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}(\mathbf{u}_\varepsilon - \mathbf{u}) \\ & + \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u}^a \cdot (\mathbf{u}_\varepsilon - \mathbf{u}) - \int_\Omega (\nabla(\mathbf{u}_\varepsilon - \mathbf{u}))(\mathbf{u}_\varepsilon - \mathbf{u}) \cdot \mathbf{u}^a, \end{aligned} \tag{4.84}$$

and it is possible to show that $\mathcal{E}(\varepsilon) = \mathcal{O}(\varepsilon^2)$, such that it vanishes when $\varepsilon \to 0$, i.e. the perturbation is small. The perturbed energy shape functional can then be expressed as

$$\begin{aligned} J_\varepsilon(\mathbf{u}_\varepsilon) &= J(\mathbf{u}) - \int_{B_\varepsilon} (1-\gamma)\alpha \mathbf{u} \cdot (\mathbf{u} - \mathbf{u}^a) + \mathcal{E}(\varepsilon) && (4.85) \\ &= J(\mathbf{u}) - |B_\varepsilon|(1-\gamma)\alpha(\hat{x})\mathbf{u}(\hat{x})(\mathbf{u}(\hat{x}) - \mathbf{u}^a(\hat{x})) + \mathcal{E}(\varepsilon), && (4.86) \end{aligned}$$

where the last equality comes from the Lebesgue differentiation theorem [37]. Again, using the definition of the topological derivative (2.3), we recognize $|B_\varepsilon|$ as the term $f(\varepsilon)$ and using the definition of the contrast parameter γ in (4.66), we get the topological derivative with respect to a point x as,

$$D_T J(x) = -k(x)(\alpha_U - \alpha_L)\mathbf{u}(x)(\mathbf{u}(x) - \mathbf{u}^a(x)), \tag{4.87}$$

where $k(x)$ is given by (2.7). Summing up the contributions over time, the topological derivative of the functional $\tilde{J}(\mathbf{u}, \alpha)$ in equation (4.63) is given by

$$D_T \tilde{J}(\mathbf{u}, \alpha) = -k(x)(\alpha_U - \alpha_L) \int_0^T \mathbf{u}(x)(\mathbf{u}(x) - \mathbf{u}^a(x))\mathrm{d}t. \tag{4.88}$$

Chapter 5

Numerical Discretization

To solve the problems numerically, a finite element approximation of the velocity and pressure functions is applied. Although the mathematical presentation is dimension-independent, the numerical experiments assume a triangular grid in 2D. We assume that the domain Ω is discretized into finite elements in 2D with the parameter h denoting the characteristic size of the largest element.

5.1 Discretization in Space of the NSD Equations

Let the spaces $\mathbf{V} = [H^1(\Omega)]^d$ and $Q = L^2(\Omega)$ denote the velocity and pressure spaces, respectively. For the discretized velocities and pressures, $\mathbf{u}_h$ and p_h, we need spaces $\mathbf{V}_h \subset \mathbf{V}$ and $Q_h \subset Q$, which are families of finite-dimensional spaces that depend on a discretization parameter h, see [38] for more details. The Navier-Stokes-Darcy system discretized in space can be written as

$$\frac{\partial \mathbf{u}_h}{\partial t} - \nu \Delta \mathbf{u}_h + (\mathbf{u}_h \cdot \nabla)\mathbf{u}_h + \alpha \mathbf{u}_h + \nabla p_h = \mathbf{0}, \tag{5.1}$$

$$\nabla \cdot \mathbf{u}_h = 0. \tag{5.2}$$

To find the weak form, let us now define test functions for the velocity and pressure, $\mathbf{v}_h$ and p_h, respectively. The weak form is derived by multiplying the test functions with (3.6) and integrate over the domain Ω. The Galerkin approximation in space of the NSD

problem then yields: find $\mathbf{u}_h \in \mathbf{U}_h$ and $p_h \in Q_h$ such that

$$\int_\Omega \frac{\partial \mathbf{u}_h}{\partial t} \cdot \mathbf{v}_h + \nu \nabla \mathbf{u}_h \cdot \nabla \mathbf{v}_h + (\mathbf{u}_h \cdot \nabla)\mathbf{u}_h \cdot \mathbf{v}_h + \alpha \mathbf{u}_h \cdot \mathbf{v}_h + \nabla p_h \cdot \mathbf{v}_h - \int_{\partial\Omega} \nu \mathbf{n} \nabla \mathbf{u}_h \mathbf{v}_h = 0 \quad \forall\, \mathbf{v}_h \in \mathbf{V}_h, \tag{5.3}$$

$$\int_\Omega (\nabla \cdot \mathbf{u}_h) q_h = 0 \quad \forall\, q_h \in Q_h. \tag{5.4}$$

Time is here kept continuous, and will be discretized below.

5.2 Finite Elements and Inf-Sup Stability

As the Navier-Stokes-Darcy problem (3.6) is of order 1 in p and of order 2 in $\mathbf{u}$, it makes sense to use piecewise polynomials of degree

$$k \quad \text{for } \mathbf{V}_h, \tag{5.5}$$

$$k-1 \quad \text{for } Q_h, \tag{5.6}$$

for the basis functions. This is known as Taylor-Hood elements, in general written $\mathbb{P}_k - \mathbb{P}_{k-1}$, $k \geq 2$. One popular choice is quadratic piecewise polynomials for the velocity components and linear piecewise polynomials for the pressure [39], in other words $k = 2$. See Figure 5.1 for an illustration of $\mathbb{P}_2 - \mathbb{P}_1$ Taylor-Hood elements. The Ladyzenskaja-

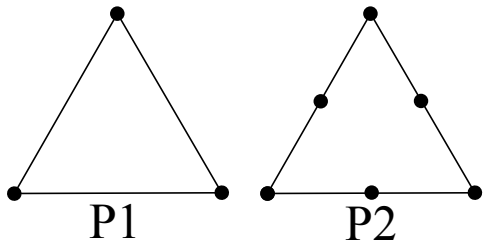


Figure 5.1: $\mathbb{P}_1$ and $\mathbb{P}_2$ Taylor-Hood elements.

Babuska-Brezzi (LBB) condition, also known as the inf-sup condition, states that there exists a constant χ independent of the grid size h such that

$$\inf_{q_h \in Q_h} \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{\langle \nabla \cdot \mathbf{v}_h, q_h \rangle}{\|\mathbf{v}_h\|_{\mathbf{V}_h} \|q_h\|_{Q_h}} \geq \chi > 0, \tag{5.7}$$

see see 16.3 in [32]. This condition guarantees solvability and stability of the Navier-Stokes problem. Hence we must choose the element spaces $\mathbf{V}_h, Q_h$ in such a way that the LBB condition (5.7) is satisfied. Elements that do not satisfy the LBB condition may not converge to the solution, and if they do, they may not converge as fast as the order of the

elements would suggest. The Taylor-Hood elements satisfy the LBB condition, and are a stable choice for the Navier-Stokes problem, according to [40].

The level set function ψ, defined in (2.8), will be discretized in space by $\mathbb{P}_1$ elements. As a consequence, one can not expect convergence of more than 1st order in space when including the Darcy term α, which is dependent on the level set function, see Section 2.4.

5.3 Discretization in Time of the NSD Equations

The incompressible Navier-Stokes-Darcy equations can be solved simultaneously with a coupled solver, using for instance a Newton solver. Coupled solvers are known to be robust, stable and accurate. The downside is that they demand a lot of memory, and may not be efficient. Since we have a time-dependent optimization problem, we want to use a method that is faster, but be can allow less accuracy. The most popular numerical solution strategies for the incompressible Navier-Stokes equations are based on operator splitting, also known as projection methods [39].

Consider the first part of the discretized Navier-Stokes-Darcy system (5.1). The equation must be discretized in time, so we introduce the first-order backwards differencing operator at time step $k+1$,

$$D_t\mathbf{u} = \frac{1}{\Delta t}(\mathbf{u}^{k+1} - \mathbf{u}^k), \tag{5.8}$$

where Δt is the time step size. We omit writing the h in $\mathbf{u}_h$ for better readability. The equation (5.1) with a backward difference scheme in time now becomes

$$\frac{1}{\Delta t}(\mathbf{u}^{k+1} - \mathbf{u}^k) + [(\mathbf{u}\cdot\nabla)\mathbf{u}]^{k+1} - \nu\Delta\mathbf{u}^{k+1} + \alpha\mathbf{u}^{k+1} + \nabla p^{k+1} = 0. \tag{5.9}$$

The weak form of this equation will be presented later in Section 5.5.

We need a way to discretize the second, non-linear term $[(\mathbf{u}\cdot\nabla)\mathbf{u}]^{k+1}$. One option that is suggested in the literature [39, 41, 42] is a simple linearization $(\mathbf{u}^k\cdot\nabla)\mathbf{u}^{k+1}$, which is a first order accurate form. Another faster, but not so accurate form is to just use the old value $(\mathbf{u}^k\cdot\nabla)\mathbf{u}^k$. If the latter option is used, the matrix on the left hand side of the equation is independent of time, and can be computed once. This saves a lot of computational time, and is a good choice in our case.

5.4 Time Discretization and Stability

Numerical experiments for the Navier–Stokes-Darcy equations (3.6) indicate that the first order discretization in time is stable under the Courant–Friedrichs–Lewy (CFL) conditions proposed in [43], where a diffusive and convective stability constraint are proposed. The

diffusive stability constraint states

$$\nu \frac{\Delta t}{\Delta x^2} \leq (\frac{1}{2})^d \tag{5.10}$$

where the dimension $d = 2$ in our case. The convective stability constraint, which is the usual CFL type,

$$\|u\|_{L^\infty} \frac{\Delta t}{\Delta x} \leq 1. \tag{5.11}$$

The stability is highly affected by the Reynolds number [44], $Re = UL/\nu$ where U is a characteristic velocity, L is a characteristic length and ν is the kinematic viscosity. If ν is small (large Reynolds number), implicit treatment of the viscous term does nothing to stabilize the convection term, so (5.11) is critical. For large ν (small Reynolds number), (5.10) dominates (5.11).

When ν is large, the time step has to be very small in order to keep the scheme stable. As an example, if the spatial discretization $\Delta x = 1/40$, $U = 1$ and the viscosity $\nu = 1.0$, then the diffusive stability constraint (5.10) requires a time step $\Delta t \leq 0.00015$ for the system to be stable while the convective stability constraint (5.11) requires only $\Delta t \leq 0.025$.

5.5 Projection Methods

Projection methods are based on the idea of splitting the system (3.6) into a series of simpler, familiar equations. The first step is neglecting the incompressibility condition and computing a tentative velocity. In the next step, the velocity is corrected by performing a projection onto the divergence free vector fields. The tentative velocity $\mathbf{u}^*$ is computed by solving the equation

$$\frac{1}{\Delta t}(\mathbf{u}^* - \mathbf{u}^k) + (\mathbf{u}^k \cdot \nabla)\mathbf{u}^k - \nu \Delta \mathbf{u}^* + \alpha \mathbf{u}^* + \xi \nabla p^k = 0, \tag{5.12}$$

where ξ is an adjustable factor for flexible control of the pressure information. Note here that it is actually the weak form of (5.12) that will be solved, but for readability, we use the strong form for the explanation of the projection methods. At the end of this section, we will present the weak form of the equations to be solved. If ξ is set to zero, we get the non-incremental pressure correction scheme, also known as Chorin's method [39]. Taking $\xi = 1$, we get the incremental pressure correction scheme, also known as IPCS [42]. The boundary conditions for $\mathbf{u}^*$ are the same as for the original problem. Since $\mathbf{u}^*$ does not fulfill the incompressibility equation (5.2), we need to correct it. This is done by evaluating the pressure and velocity in certain terms at time step $k + 1$

$$\frac{1}{\Delta t}(\mathbf{u}^{k+1} - \mathbf{u}^k) + (\mathbf{u}^k \cdot \nabla)\mathbf{u}^k - \nu \Delta \mathbf{u}^* + \alpha \mathbf{u}^{k+1} + \nabla p^{k+1} = 0. \tag{5.13}$$

Subtracting (5.12) from (5.13), we end up with

$$\frac{1}{\Delta t}(\mathbf{u}^{k+1} - \mathbf{u}^*) + \alpha(\mathbf{u}^{k+1} - \mathbf{u}^*) + \nabla(p^{k+1} - \xi p^k) = 0. \tag{5.14}$$

Now we take the divergence of the equation (5.14) to get rid of the unknown velocity, using that $\nabla \cdot \mathbf{u}^{k+1} = 0$, and get a Poisson equation for the pressure p^{k+1}

$$\delta(p^{k+1} - \xi p^k) = (\frac{1}{\Delta t} + \alpha)\nabla \cdot \mathbf{u}^*. \tag{5.15}$$

We observe from (5.14) that $\nabla(p^{k+1} - \xi p^k) \cdot \mathbf{n}|_{\partial\Omega_D} = 0$ since $\mathbf{u}^{k+1}$ and $\mathbf{u}^*$ have the same value on $\partial\Omega_D$. This implies that

$$\nabla p^{k+1} \cdot \mathbf{n}|_{\partial\Omega_D} = \xi \nabla p^k \cdot \mathbf{n}|_{\partial\Omega_D} = \ldots = \xi^{k+1} \nabla p^0 \cdot \mathbf{n}|_{\partial\Omega_D}. \tag{5.16}$$

These non-physical Neumann boundary conditions will introduce a numerical boundary layer [42]. This may result in large errors near the boundary, and often an error of $\mathcal{O}(1)$ is experienced in a boundary layer with width $\approx \sqrt{\nu \Delta t}$, according to [39].

Solving for the pressure p^{k+1}, there remains only one unknown in equation (5.14), and the velocity $\mathbf{u}^{k+1}$ can be computed. The updated velocity will be divergence free, but not satisfy the boundary conditions exactly. It is possible to enforce boundary conditions on the velocity update.

The pressure correction scheme in weak form can be summed up in three equations to be solved:

i) $$\int_\Omega \left(\frac{1}{\Delta t} + \alpha\right) \mathbf{u}^* \cdot \mathbf{v} + \nu \nabla \mathbf{u}^* \cdot \nabla \mathbf{v} = \int_\Omega \frac{1}{\Delta t} \mathbf{u}^k \cdot \mathbf{v} - (\mathbf{u}^k \cdot \nabla)\mathbf{u}^k \cdot \mathbf{v} - \xi \nabla p^k \cdot \mathbf{v},$$

ii) $$\int_\Omega \nabla p^{k+1} \cdot \nabla q = \int_\Omega \xi \nabla p^k \cdot \nabla q - (\frac{1}{\Delta t} + \alpha)(\nabla \cdot \mathbf{u}^*) \cdot q,$$

iii) $$\int_\Omega \left(\frac{1}{\Delta t} + \alpha\right) \mathbf{u}^{k+1} \cdot \mathbf{v} = \int_\Omega \left(\frac{1}{\Delta t} + \alpha\right) \mathbf{u}^* \cdot \mathbf{v} - \nabla(p^{k+1} - \xi p^k) \cdot \mathbf{v}.$$

These three steps can be described as i) tentative velocity, ii) pressure correction and iii) velocity update. It now becomes clear that the matrices on the left hand are independent of the solution at time k, and only needs to be assembled in the first iteration.

5.6 Discrete Time-Dependent Adjoint Equations

For the discrete time-dependent adjoint equations, let us divide the time interval $[0, T]$ into N_t equidistant time steps, such that we have $N_t + 1$ residual systems of equations. In this way, we can express the objective function as a sum

$$\tilde{J}(\mathbf{u}, p, \alpha) = \sum_0^{N_t} J(\mathbf{u}^{(i)}, p^{(i)}, \alpha) =: \sum_0^{N_t} J^{(i)}, \tag{5.17}$$

and each system of residuals can be expressed as a function of the state history,

$$\mathbf{R}^{(i)} := \mathbf{R}(\mathbf{u}^{(i)}, \mathbf{u}^{(i-1)}, \dots \mathbf{u}^{(0)}, p^{(i)}, p^{(i-1)}, \dots, p^{(0)}, \alpha), \tag{5.18}$$

where the superscript denotes the time step number. Based on this, the Lagrangian can be written as

$$\mathcal{L}(\mathbf{u}, p, \alpha) = \sum_{i=0}^{N_t} \left(J^{(i)} - \int_\Omega (\mathbf{u}^a, p^a)^{(i)^T} \mathbf{R}^{(i)} \mathrm{d}\Omega \right). \tag{5.19}$$

Now, we want to differentiate (5.19) with respect to α, and calculate the sensitivities. In order to calculate the gradient of $J(\mathbf{u}, p, \alpha)$ with respect to the control parameters α, just like the stationary method, we differentiate with respect to each variable. The difference now, is that the derivatives depend on state variables from different time steps, so the expression becomes quite comprehensive.

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha}(\mathbf{u}, p, \alpha) = &\sum_{i=0}^{N_t} \left(\frac{\partial J^{(i)}}{\partial \alpha} + \sum_{k=0}^{i} \frac{\partial J^{(i)}}{\partial \mathbf{u}^{(k)}} \frac{\mathrm{d}\mathbf{u}^{(k)}}{\mathrm{d}\alpha} + \sum_{k=0}^{i} \frac{\partial J^{(i)}}{\partial p^{(k)}} \frac{\mathrm{d}p^{(k)}}{\mathrm{d}\alpha} \right) \\
&- \sum_{i=0}^{N_t} \left(\int_\Omega (\mathbf{u}^a, p^a)^{(i)^T} \left(\frac{\partial \mathbf{R}^{(i)}}{\partial \alpha} + \sum_{k=0}^{i} \frac{\partial \mathbf{R}^{(i)}}{\partial \mathbf{u}^{(k)}} \frac{\mathrm{d}\mathbf{u}^{(k)}}{\mathrm{d}\alpha} \right.\right. \\
&\qquad \left.\left. + \sum_{k=0}^{i} \frac{\partial \mathbf{R}^{(i)}}{\partial p^{(k)}} \frac{\mathrm{d}p^{(k)}}{\mathrm{d}\alpha} \right) \mathrm{d}\Omega \right).
\end{aligned} \tag{5.20}$$

Rearranging (5.20) with respect to implicit derivatives, we get

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\alpha}(\mathbf{u}, p, \alpha) = &\sum_{i=0}^{N_t} \frac{\partial J^{(i)}}{\partial \alpha} - \sum_{i=0}^{N_t} \left(\int_\Omega (\mathbf{u}^a, p^a)^{(i)^T} \frac{\partial \mathbf{R}^{(i)}}{\partial \alpha} \right) \\
&+ \left(\frac{\partial J^{(N_t)}}{\partial \mathbf{u}^{(N_t)}} - \int_\Omega (\mathbf{u}^a, p^a)^{(N_t)^T} \frac{\partial \mathbf{R}^{(N_t)}}{\partial \mathbf{u}^{(N_t)}} \right) \frac{\mathrm{d}\mathbf{u}^{(N_t)}}{\mathrm{d}\alpha} \\
&+ \sum_{i=0}^{N_t-1} \left(\sum_{k=i}^{N_t} \left(\frac{\partial J^{(k)}}{\partial \mathbf{u}^{(i)}} - \int_\Omega (\mathbf{u}^a, p^a)^{(k)^T} \frac{\partial \mathbf{R}^{(k)}}{\partial \mathbf{u}^{(i)}} \right) \right) \frac{\mathrm{d}\mathbf{u}^{(i)}}{\mathrm{d}\alpha} \\
&+ \left(\frac{\partial J^{(N_t)}}{\partial p^{(N_t)}} - \int_\Omega (\mathbf{u}^a, p^a)^{(N_t)^T} \frac{\partial \mathbf{R}^{(N_t)}}{\partial p^{(N_t)}} \right) \frac{\mathrm{d}p^{(N_t)}}{\mathrm{d}\alpha} \\
&+ \sum_{i=0}^{N_t-1} \left(\sum_{k=i}^{N_t} \left(\frac{\partial J^{(k)}}{\partial p^{(i)}} - \int_\Omega (\mathbf{u}^a, p^a)^{(k)^T} \frac{\partial \mathbf{R}^{(k)}}{\partial p^{(i)}} \right) \right) \frac{\mathrm{d}p^{(i)}}{\mathrm{d}\alpha}.
\end{aligned} \tag{5.21}$$

We seek the values of $\mathbf{u}^a, p^a$ such that the implicit terms vanish, which means we want the four last lines of (5.21) to disappear. This gives rise to the adjoint equations for the unstationary Navier-Stokes-Darcy system.

For the four last terms in (5.21) to vanish, it is obvious that the solution of $\{(\mathbf{u}^a, p^a)^{(i)}\}$ is given by

$$\begin{aligned}
\int_\Omega (\mathbf{u}^a, p^a)^{(N_t)^T} \frac{\partial \mathbf{R}^{(N_t)}}{\partial \mathbf{u}^{(N_t)}} &= \frac{\partial J^{(N_t)}}{\partial \mathbf{u}^{(N_t)}}, \\
\int_\Omega (\mathbf{u}^a, p^a)^{(N_t)^T} \frac{\partial \mathbf{R}^{(N_t)}}{\partial p^{(N_t)}} &= \frac{\partial J^{(N_t)}}{\partial p^{(N_t)}}, \\
\int_\Omega (\mathbf{u}^a, p^a)^{(i)^T} \frac{\partial \mathbf{R}^{(i)}}{\partial \mathbf{u}^{(i)}} &= \frac{\partial J^{(i)}}{\partial \mathbf{u}^{(i)}} + \sum_{k=i+1}^{N_t} \left(\frac{\partial J^{(k)}}{\partial \mathbf{u}^{(i)}} + \int_\Omega (\mathbf{u}^a, p^a)^{(k)^T} \frac{\partial \mathbf{R}^{(k)}}{\partial \mathbf{u}^{(i)}} \right), \\
\int_\Omega (\mathbf{u}^a, p^a)^{(i)^T} \frac{\partial \mathbf{R}^{(i)}}{\partial p^{(i)}} &= \frac{\partial J^{(i)}}{\partial p^{(i)}} + \sum_{k=i+1}^{N_t} \left(\frac{\partial J^{(k)}}{\partial p^{(i)}} + \int_\Omega (\mathbf{u}^a, p^a)^{(k)^T} \frac{\partial \mathbf{R}^{(k)}}{\partial p^{(i)}} \right).
\end{aligned} \tag{5.22}$$

To compute the adjoints, we need access to the whole primal velocity and pressure $\{\mathbf{u}^{(i)}, p^{(i)}\}$. This can be done by storing the velocity and pressure at each time step in memory. If the solution is too big, a possibility is to use a checkpointing scheme [45], where the solution is stored at chosen intervals. The missing solutions can be reconstructed from the nearest available checkpoint when it is needed. Once the missing solution is available, the corresponding adjoint can be computed. For instance, a checkpoint interval can be of the size corresponding to the required memory capacity, in order to minimize the number of checkpoints.

5.7 Numerical Software

The solution $\mathbf{u}_h$ and p_h of the Navier-Stokes-Darcy system is obtained by using the FEniCS environment [46]. FEniCS is a collection of open source software components made to enable automated solution of differential equations. The differential equations can be specified in near-mathematical notation, such as finite element variational problems, and hence it is very useful tool for computational mathematics. The language used to express weak forms is called UFL (Unified Form Language). FEniCS can be programmed in both C++ and Python, using a library called DOLFIN which functions as the main user interface of FEniCS. A just-in-time compiler called FFC (FEniCS Form Compiler) generates efficient low-level C++ code from the high-level mathematical description (UFL) of the variational problem.

To demonstrate, a small part of a simple Python program that solves the Navier-Stokes-Darcy problem with $\mathbb{P}_2 - \mathbb{P}_1$ elements and Chorin's method is presented under.

```
# Function spaces for velocity and pressure:
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)

# Functions
u = TrialFunction(V)
p = TrialFunction(Q)
```

```
v = TestFunction(V)
q = TestFunction(Q)

u0 = Function(V)
u1 = Function(V)
p1 = Function(Q)

# Boundary conditions
bcu = DirichletBC(V, (0, 0), "on_boundary")

# Tentative velocity
F1 = (1./dt)*inner(u - u0, v)*dx + inner(u0*grad(u0), v)*dx \
    + nu*inner(grad(u), grad(v))*dx + alpha*inner(u, v)*dx
a1 = lhs(F1)
L1 = rhs(F1)

# Pressure update
a2 = inner(grad(p), grad(q))*dx
L2 = -(1./dt)*div(u1)*q*dx - alpha*div(u1)*q*dx

# Velocity update
a3 = (1./dt + alpha)*inner(u, v)*dx
L3 = (1./dt + alpha)*inner(u1, v)*dx - inner(grad(p1), v)*dx

while t < T:
    t += dt
    solve(a1 == L1, u1, bcu)
    solve(a2 == L2, p1)
    solve(a3 == L3, u1, bcu)
    u0.assign(u1)
```

dolfin-adjoint [45] exploits the high-level structure of the finite element method and builds on the FEniCS Project. As the forward model is executed through FEniCS, dolfin-adjoint records and analyzes the dependencies and structure of the equations. The resulting execution graph stores a symbolic mathematical representation of these equations and captures the entire information flow through the simulation. By reasoning about this graph, dolfin-adjoint can linearize the equations to derive a symbolic representation of the discrete tangent linear equations, and reverse the propagation of information through the graph to derive the corresponding adjoint equations. By invoking the FEniCS automatic code generator on these equations, dolfin-adjoint obtains solutions of the adjoint models.

In our case, dolfin-adjoint will be used to compute the topological derivative for different objective functionals, and used as gradients in the gradient-based optimization algorithm. The advantage of computing the gradient numerically is that one omits deriving the adjoint model and topological derivative, which can be quite comprehensive and time consuming. The disadvantage is that the gradient is not exact.

Chapter 6

Numerical Verification

The Navier-Stokes-Darcy problems is solved using a finite element method in space and a pressure correction scheme, either Chorin's scheme or IPCS, in time, see Chapter 5. It is expected to get a convergence of second order in space and first order in time for all cases. In further results, we ended up using IPCS, as it is more stable and gives more accurate solutions.

To verify the solutions, we consider various test problems, and use the method of manufactured solutions to calculate the source term. The Navier-Stokes-Darcy problem (3.6) can in 2D be written in component form as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \alpha u = f_x, \tag{6.1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \alpha v = f_y, \tag{6.2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{6.3}$$

where u and v are the x- and y-components of $\mathbf{u}$, respectively, and f_x and f_y are the x- and y-components of the source term $\mathbf{f}$, respectively. The density ρ is set to 1 in all cases.

Since mainly two values for α will be used, representing fluid and solid material, we will check convergence in both cases. The velocities within solid material and along solid-fluid interface (no-slip condition) need to vanish. Figure 5 in [12], section 3.1 suggests that a maximum impermeability of 10^4 is sufficient to enforce zero velocity. Inspired by the work of Borrvall and Peterson [15], the numerical values used for α in all experiments are

$$\alpha_L = \frac{2.5\nu}{100^2}, \tag{6.4}$$

$$\alpha_U = \frac{2.5\nu}{0.01^2}. \tag{6.5}$$

In Kreissl et. al [12], they investigate the maximum impermeability needed to enforce zero velocity. They find that for different values of Reynolds number and velocity norms, $\alpha_U > 10^4$ is sufficient.

The numerical solution $\mathbf{u}$ is computed on a $N \times N$ mesh at end time T, and is compared to the analytical solution $\mathbf{u}_e$. The error is calculated by taking the L^2-norm of the difference between the solutions, $\|\mathbf{u} - \mathbf{u}_e\|$, using the FEniCS function `errornorm`. This module computes the error by first interpolating both $\mathbf{u}$ and $\mathbf{u}_e$ to a common space with finer discretization, then subtracting the two fields and then evaluating the integral. A code snippet is included below to illustrate how the error is calculated using Python and the FEniCS environment.

```
def errornorm(u, u_e, Ve):
    u_= interpolate(u, Ve)
    u_e_ = interpolate(u_e, Ve)
    e_ = Function(Ve)
    e_.vector()[:] = u_.vector().array() - u_e_.vector().array()
    error = e_**2*dx
    return sqrt(assemble(error))
```

6.1 Taylor-Green Flow

First, we will verify the Navier-Stokes solver without the Darcy term, because in this case we have an analytical solution. Taylor-Green flow is an analytical solution to the non-stationary incompressible Navier-Stokes equations on a domain with periodic boundary conditions. The exact solution in 2 dimensions is given by

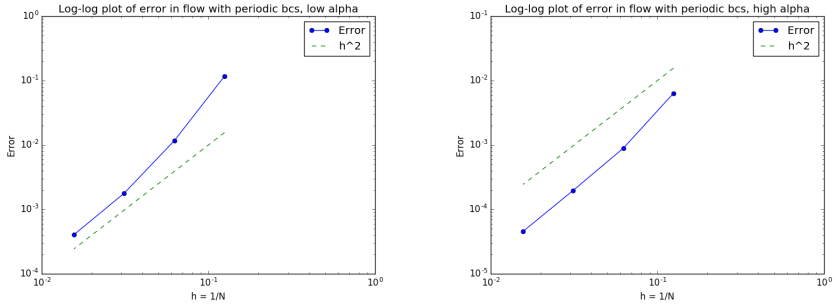
$$u = -\cos(\pi x)\sin(\pi y)e^{-2\pi^2\nu t}, \tag{6.6}$$

$$v = \cos(\pi y)\sin(\pi x)e^{-2\pi^2\nu t}, \tag{6.7}$$

$$p = \frac{1}{4}\left(\cos(2\pi x) + \cos(2\pi y)\right)e^{-4\pi^2\nu t}, \tag{6.8}$$

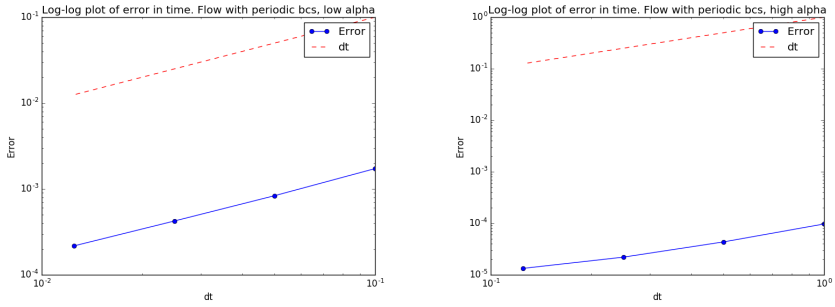
with the source term $\mathbf{f} = \mathbf{0}$. Because we have an additional Darcy-term in our problem, we have to compute the new source term, which is simply $\mathbf{f} = -\alpha\mathbf{u}$.

The Taylor-Green flow is computed using Chorin's method on a domain $\Omega = [0, 2] \times [0, 2]$ with periodic boundary conditions in both spatial directions. The kinematic viscosity is $\nu = 0.01$ and the errors are computed at end time $T = 1.0$. The convergence in space is presented in Figure 6.1, with mesh size $N = [8, 16, 32, 64]$ and time step $\Delta t = 0.02$. The convergence in time is presented in Figure 6.2 with time step size $\Delta t = [1.0, 0.5, 0.25, 0.125]$ and grid size $N = 200$. As expected, we have a convergence of second order in space and first order in time.



(a) Low α **(b)** High α

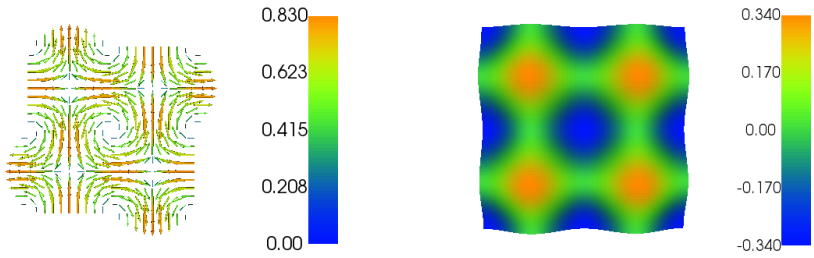
Figure 6.1: Convergence rates in space on velocity in L^2-norm for Taylor-Green flow at $T = 1.0$.



(a) Low α **(b)** High α

Figure 6.2: Convergence rates in time on velocity in L^2-norm for Taylor-Green flow at $T = 1.0$.

The computed solution at time $T = 1.0$ can be seen in Figure 6.3.



(a) Taylor Green velocity **(b)** Taylor Green pressure

Figure 6.3: Computed solution of Taylor-Green flow at $T = 1.0$.

6.2 Convergence of Chorin's Method vs IPCS

In this section, we will compare Chorin's method to IPCS, in order to see which method is better to proceed with. Because the optimization problems will have Dirichlet boundary conditions, it is reasonable to include the verification of such a model. We choose the analytical solution

$$u = -\cos(\pi x)\sin(\pi y)e^{-t}, \tag{6.9}$$
$$v = \cos(\pi y)\sin(\pi x)e^{-t}, \tag{6.10}$$
$$p = \sin(\pi x)\cos(\pi y), \tag{6.11}$$

and use the method of manufactured solutions to calculate the source term. Inserting (6.9) into (6.1), we get the source terms

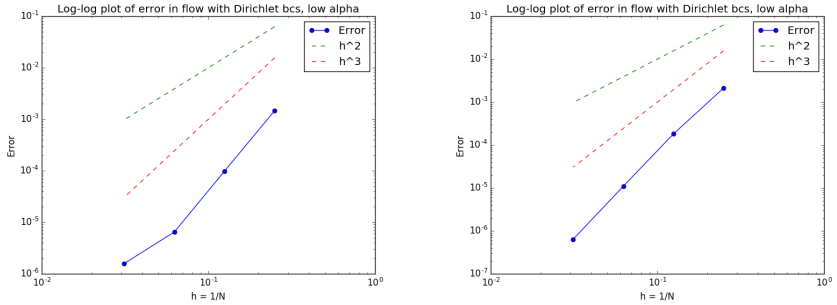
$$\begin{aligned} f_x =& -\alpha e^{-t}\sin(\pi y)\cos(\pi x) - 2\nu\pi^2 e^{-t}\sin(\pi y)\cos(\pi x) \\ &+ \pi\cos(\pi x)\cos(\pi y) - \pi e^{-2t}\sin(\pi x)\sin(\pi y)^2\cos(\pi x) \\ &- \pi e^{-2t}\sin(\pi x)\cos(\pi x)\cos(\pi y)^2 + e^{-t}\sin(\pi y)\cos(\pi x), \end{aligned} \tag{6.12}$$

$$\begin{aligned} f_y =& \alpha e^{-t}\sin(\pi x)\cos(\pi y) + 2\nu\pi^2 e^{-t}\sin(\pi x)\cos(\pi y) \\ &- \pi\sin(\pi x)\sin(\pi y) - \pi e^{-2t}\sin(\pi x)^2\sin(\pi y)\cos(\pi y) \\ &- \pi e^{-2t}\sin(\pi y)\cos(\pi x)^2\cos(\pi y) - e^{-t}\sin(\pi x)\cos(\pi y), \end{aligned} \tag{6.13}$$

and zero divergence. The source terms are computed using the python library SymPy [47]. The manufactured solution is computed using both Chorin's method and the IPCS method on a domain $\Omega = [0,1] \times [0,1]$, and the analytical solution (u, v) is applied as Dirichlet boundary conditions on the whole boundary. The pressure p^{k+1} is normalized in each iteration, i.e. the mean value of the pressure is subtracted from the pressure solution to keep it in the same range. This is done because p is only defined up to a constant by the Navier-Stokes-Darcy equations (3.6).

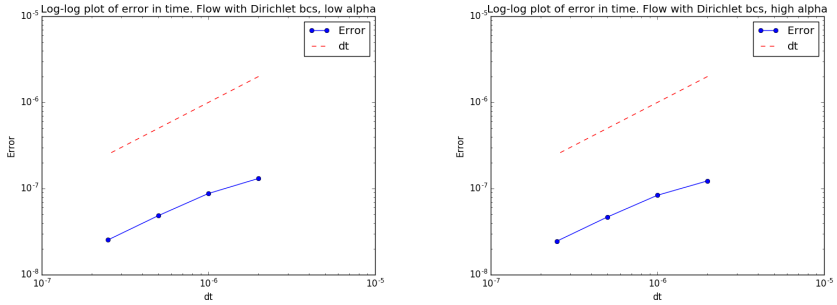
As mentioned in Section 5.5, an erroneous boundary layer with width $\approx \sqrt{\nu\Delta t}$ can occur when Dirichlet boundary conditions are applied. This prevents the scheme from being fully second-order in space and first-order in time [42]. By letting the time step Δt be very small, and increasing the kinematic viscocity to $\nu = 1.0$ to get a lower Reynold's number, we were able to compute convergence rates at end time $T = 0.1$.

The convergence of Chorin's method in space is presented in Figure 6.4 , with mesh size $N = [4, 8, 16, 32]$ and time step $\Delta t = 0.0000025$. The convergence of Chorin's method in time is presented in Figure 6.5 with time step size $\Delta t = [0.000002, 0.000001, 0.0000005, 0.00000025]$ and grid size $N = 400$. The convergence in space appears to be of third order, which may be an effect of a regular mesh. The convergence in time is of first order, as expected.



(a) Low α **(b)** High α

Figure 6.4: Chorin's scheme convergence rates in space on velocity in L^2-norm for (6.9) at $T = 0.1$.

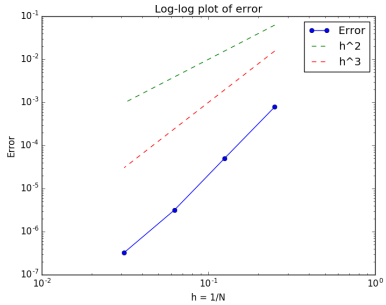


(a) Low α **(b)** High α

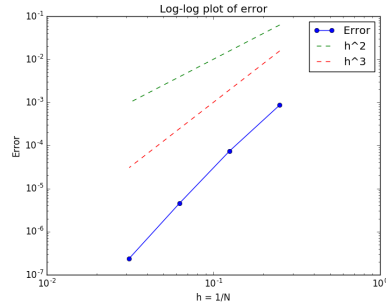
Figure 6.5: Chorin's scheme convergence rates in time on velocity in L^2-norm for (6.9) at $T = 0.1$.

Convergence was very hard to achieve for the Chorin scheme, due to the artificial boundary layer. A convergence test was also performed for the IPCS method. With this scheme, we managed to get the same order of convergence using a much bigger time step Δt, and thus calculated the error at end time $T = 1.0$. It seems that including the pressure in the tentative velocity step, as shown in equation (5.13), has a huge impact on the erroneous boundary layer when using Dirichlet boundary conditions.

The convergence of the IPCS method in space is presented in Figure 6.6, with mesh size $N = [4, 8, 16, 32]$ and time step $\Delta t = 0.001$. The convergence of the IPCS method in time is presented in Figure 6.5 with time step size $\Delta t = [0.1, 0.05, 0.025, 0.0125]$ and grid size $N = 60$. The order of convergence is of third-order in space and of first-order in time. In other words, the same result is obtained using much larger time steps Δt and smaller grid size N. This result indicates that we should proceed with the IPCS method when solving the NSD problem.

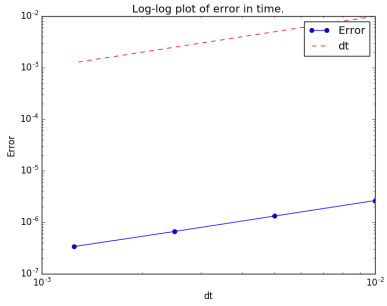


(a) Low α

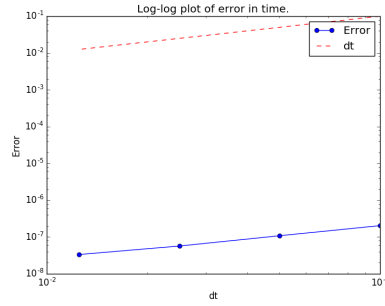


(b) High α

Figure 6.6: IPCS convergence rates in space on velocity in L^2-norm for (6.9) at $T = 1.0$.



(a) Low α



(b) High α

Figure 6.7: IPCS convergence rates in time on velocity in L^2-norm for (6.9) at $T = 1.0$.

6.3 Channel Problem with Darcy Term

To verify the accuracy of the scheme with respect to material distribution, we compare the results of the flow in a fluid channel using the Darcy term α against the reference solution of a body-fitted mesh with $\alpha = 0$. The boundary conditions are Poiseuille flow with maximum velocity 1.0 at the inflow, open boundary conditions $\mathbf{g}_N = \nu(\nabla \mathbf{u})\mathbf{n} - p\mathbf{n} = 0$ at the outflow, and no-slip on the remaining boundaries. An illustration of the problems can be seen in Figure 6.8.

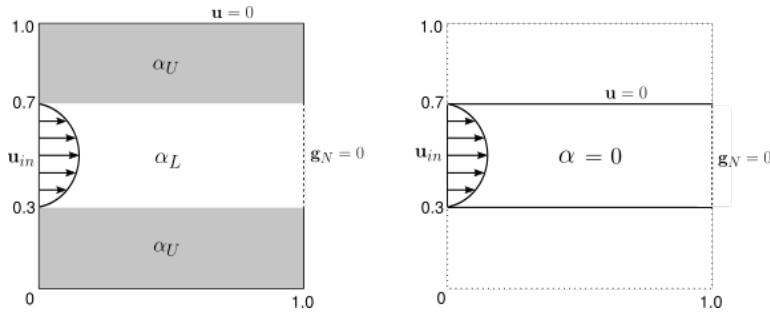


Figure 6.8: Illustration of the channel for Navier-Stokes-Darcy (left) and Navier-Stokes (right).

The solution of the channel problem is computed on a domain $\Omega = [0, 1] \times [0, 1]$ with a mesh containing 4224 cells. The reference solution is computed on a domain $\tilde{\Omega} = [0, 1] \times [0.3, 0.7]$, and the mesh is a subset of the initial mesh, thus having only 1696 nodes, as its volume is $4/10$ of the initial domain.

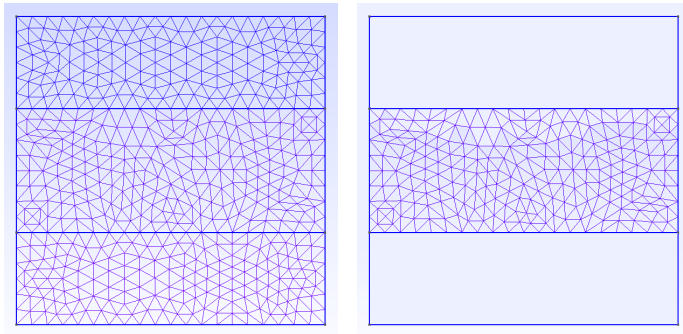
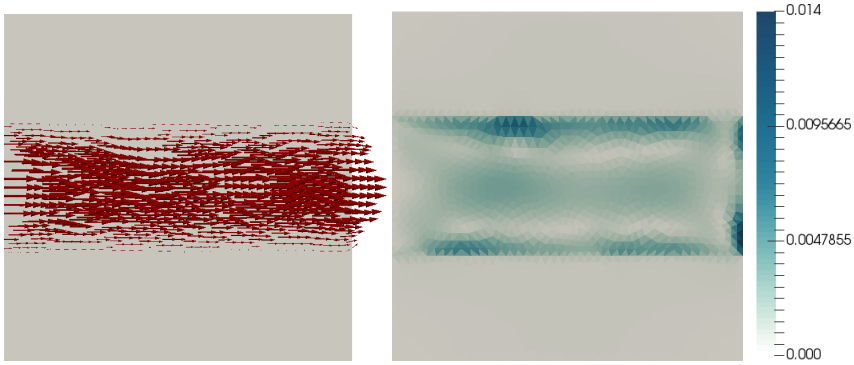


Figure 6.9: Illustration of the meshes for the channel problem.

The meshes are created in `gmsh` [48], and a coarse version can be seen in Figure 6.9. It is here easy to observe that the meshes have the exact same structure in the channel area. Both problems are solved using the IPCS-scheme with time step $\Delta t = 0.001$. The solutions are compared at end time $T = 1.0$, and the $L^2(\tilde{\Omega})$-norm of the error is 0.0018. The velocity solution to the Navier-Stokes-Darcy channel problem is plotted in Figure 6.10 together with the error with respect to the reference solution. The error is most significant in the boundary area, where the impermeability constant α changes its value.



(a) Velocity for Navier-Stokes-Darcy. **(b)** Absolute value of difference in velocities.

Figure 6.10: Velocity plots for the channel problem with Darcy term. The l^2-norm of the difference in velocities is 0.0018

6.4 Verification of Gradient Using dolfin-adjoint

Dolfin-adjoint [45] is a software developed for computing adjoints and functional gradients numerically. In order to use the gradient computed in dolfin-adjoint, we will verify it by comparing it to the topological derivative derived in Section 4.7, where the objective is the dissipation energy functional averaged over all time steps,

$$J(\mathbf{u}, \alpha) = \Delta t \sum_{n=1}^{N_t} \left(\int_\Omega \left(\nu |\nabla \mathbf{u}|^2 + \alpha(\psi) |\mathbf{u}^n|^2 \right) + \beta |\omega| \right), \tag{6.14}$$

where N_t is the number of time steps, $|\omega|$ is the volume of the fluid domain and β is a penalty parameter. The dolfin-adjoint gradient will be computed using the python commands

```
J = Functional((alpha(psi)*inner(u, u) + nu*inner(grad(u), grad(u)))*dx*dt
               + beta * fluid_domain(psi) * dx * dt)
g_a = compute_gradient(J, Control(psi))
```

while the topological gradient will be computed using the expression derived in Section 4.7, with the additional fluid volume penalization,

$$g_t = \Delta t \sum_{n=1}^{N_t} \left(-(\alpha_U - \alpha_L) \mathbf{u}^n (\mathbf{u} - \mathbf{u}^a)^n + \beta \right). \tag{6.15}$$

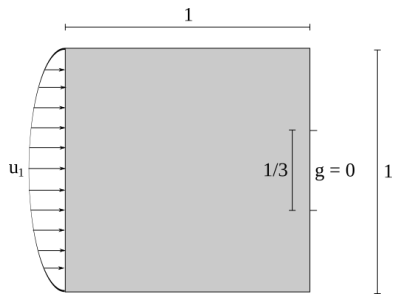
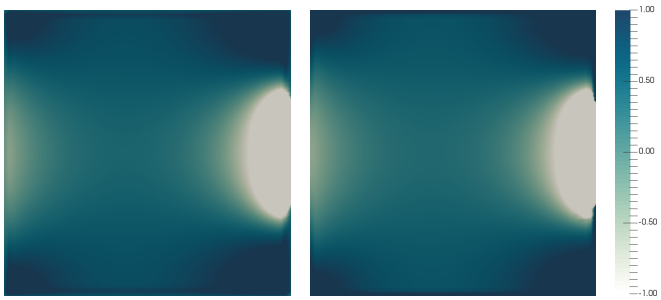


Figure 6.11: Computational domain for the diffuser problem.

Figure 6.11 shows an illustration of the design domain. The kinematic viscosity is set to $\nu = 0.01$ and the penalty parameter $\beta = 90.0$. The outlet has traction-free Neumann boundary conditions (3.9), $\mathbf{g} = \mathbf{0}$, and the inlet has a time-varying parabolic velocity with value

$$u_{\text{in}}(t) = \sin\left(\frac{\pi}{2T}t\right) \tag{6.16}$$

The gradients are computed on a $[0, 1] \times [0, 1]$ domain with 60×60 triangular elements. The time step size is $\Delta t = 0.0001$ and number of time steps $N_t = 60$.



(a) Dolfin-adjoint gradient **(b)** Topological gradient **(c)**

Figure 6.12: Comparison of gradients for the diffuser problem.

The L^2-norm between the two gradients, $\|g_a - g_t\| = 0.1889$, and looking at the plot of the differences in Figure 6.13, it is obvious that the biggest difference is in the active areas. The relatively big difference in norm can be explained by the fact that dolfin-adjoint uses the trapezoidal rule to compute the time integral (6.14), while the topological gradient has been computed at every time step. Figure 6.14 shows a plot of the l^2-norm for different grid sizes, and a convergence of order $1/2$ is observed.

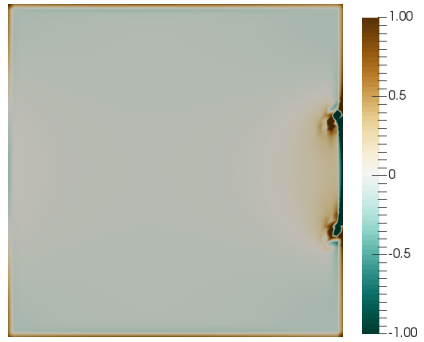


Figure 6.13: Difference between gradients for the diffuser problem.

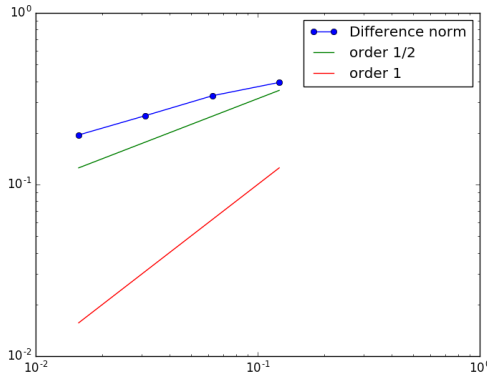


Figure 6.14: L2-norm of the difference in gradients for different mesh sizes $N = [16, 32, 64, 128]$ with time step $\Delta t = 0.01$ and total time $T = 0.15$. A convergence of order $1/2$ is observed.

Chapter 7

Numerical Experiments

In this chapter, numerical experiments are presented to demonstrate the features and utility of the proposed topological optimization algorithm for unsteady Navier-Stokes-Darcy flow. In particular, two examples from Deng et al.[24] and Kreissl et al.[12] are implemented to verify the topological optimization method for unsteady flow. At the end of this chapter, the optimization method is applied to a biomedical flow problem to optimize the topology of a coronary artery bypass anastomosis. The exact topological derivative for dissipation, see Section 4.7, and the topological gradient computed using dolfin-adjoint [45], will be applied in the experiments. It will be specified in each case which one is used. Listings of the code can be found in Appendix A.

The values of the impermeability constants are set to $\alpha_L = 2.5\nu/100^2$ and $\alpha_U = 2.5\nu/0.01^2$, as discussed in Section 6. The stopping criterion on the angle θ, presented in Section 3.6, is $\varepsilon_\theta = 0.01$. For the line search in Section 3.5, the step size tolerance is set to $\varepsilon_\kappa = 0.0001$.

At the end of every experiment, a discussion of the results is included, and a summary of the discussions is presented in Chapter 8.

7.1 Double Pipe with Analytical Topological Derivative

The double pipe optimization problem with varying inflow condition is proposed in [24], and will be reconstructed here in order to compare the resulting topology. The design domain is shown in Figure 7.1, and the mesh consists of 43×43 triangular elements, see Figure 7.2. The kinematic viscosity is $\nu = 1$, the density $\rho = 1$ and the velocities imposed at the two inlets are time-varying, given by

$$\mathbf{u}_1 = -144(y - 4/6)(y - 5/6)\cos(t)\mathbf{n}, \tag{7.1}$$

$$\mathbf{u}_2 = -144(y - 1/6)(y - 2/6)\sin(t)\mathbf{n}, \tag{7.2}$$

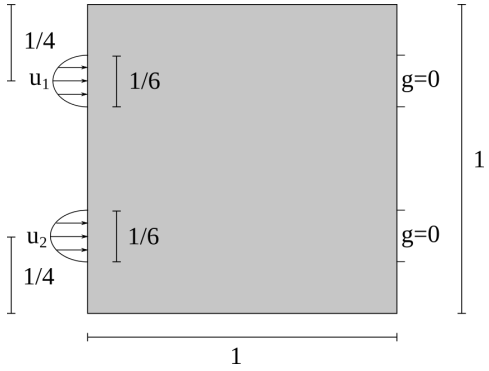


Figure 7.1: Design domain for the double pipe problem

with $t \in [0, 2\pi]$. At the outlets, traction free Neumann boundary conditions are imposed, $\mathbf{g} = \mathbf{0}$, see (3.9). The time step used is $\Delta t = 0.00002$.

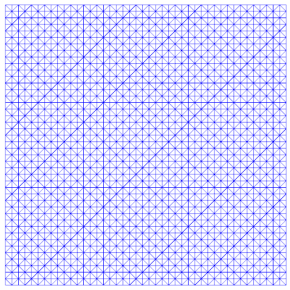


Figure 7.2: Mesh for the double pipe problem.

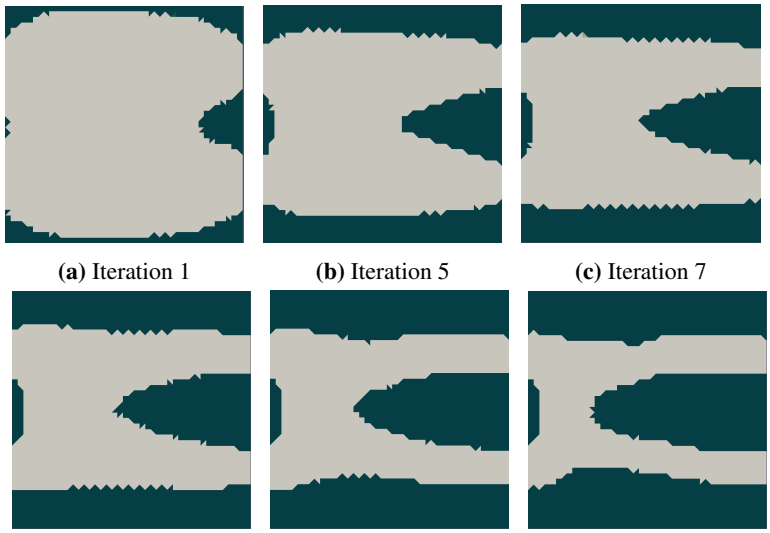
The objective functional used in this case is the same as in equation (4.64), with an extra term for fluid volume penalization

$$J(\mathbf{u}, \alpha) = \int_0^T \int_\Omega \left(\nu |\nabla \mathbf{u}|^2 + \alpha |\mathbf{u}|^2 \right) \mathrm{d}t + \beta |\omega|, \tag{7.3}$$

where $|\omega|$ denotes the fluid volume. The penalty parameter β is set to 200, and the gradient is the topological derivative in equation (4.7). In order to increase the accuracy of the adjoint solution, a coupled solver is used, see Appendix A.2 for the implementation. The impermeability parameter α is not interpolated, corresponding to a penalty value of $q = 0$ in equation (2.12), since the exact topological derivative does not require it to be differentiable. In other words, the material distribution is discontinuous, which can be observed in the resulting shape.

The stopping criterion for the optimization procedure, is that the angle θ reaches below the tolerance $\varepsilon_\theta = 0.01$. The optimization algorithm terminated after 20 iterations, and some

snapshots of the optimization procedure are presented in Figure 7.3. The evolvement of the objective function J and angle θ are shown in Figure 7.4.



(a) Iteration 1 **(b)** Iteration 5 **(c)** Iteration 7

(d) Iteration 9 **(e)** Iteration 13 **(f)** Iteration 20

Figure 7.3: Optimization procedure for double pipe problem.

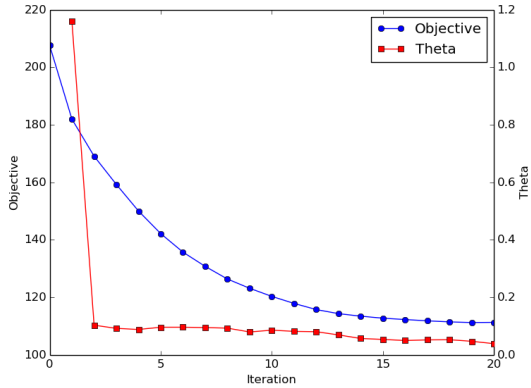


Figure 7.4: Evolvement of objective function J and angle θ in the optimization procedure for the double pipe problem.

The optimal design is presented together with the optimal design proposed in [24], in Figure 7.5, and the values for the objective functional J and fluid fraction γ of these are presented in Table 7.1. The values of J come from eq. (7.3) and Table 2 in [24].

Table 7.1: Functional value J and fluid fraction γ for the optimal design and design in [24] for the double pipe problem.

	J	γ
Fig. 7.5a	111.3	0.3797
Fig. 7.5b	113.8	0.3333

(a) Optimal design. **(b)** Optimal design in [24].

Figure 7.5: Optimal designs for the double pipe problem.

The two optimal shapes presented in Figure 7.5 have been found using different optimization algorithms, and the most obvious difference is the smoothness, or lack of, in the material distribution. Nevertheless, the shapes are quite similar, and the objective values in Table 7.1 likewise, so it is reasonable to assume that the gradient-based algorithm with the exact topological derivative as gradient is capable of optimizing a shape for unsteady fluid flow.

7.2 Diffuser with Approximate Topological Derivative

The diffuser optimization problem, originally from Borrvall and Peterson [15], is proposed for unsteady flow in Kreissl et al.[12], and will be reconstructed here in order to compare the optimization algorithms. The design domain is a unit square, $[0,1] \times [0,1]$, shown in Figure 7.6, and the mesh consists of 42×42 triangular elements. The kinematic viscosity is $\nu = 0.01$, the density $\rho = 1.0$ and the velocity at the inlet is time dependent and parabolic shaped, given by

$$\mathbf{u}_1(t) = \sin\left(\frac{\pi}{2T}t\right)^2 (1 - (2y - L)^2)\mathbf{n} \tag{7.4}$$

where the inlet length $L = 1.0$ in our case. At the outlet, traction-free Neumann boundary conditions are imposed, $\mathbf{g} = \mathbf{0}$, see (3.9). The time step used is $\Delta t = 0.0001$, and total time $T = 15\Delta t$, as suggested in [12].

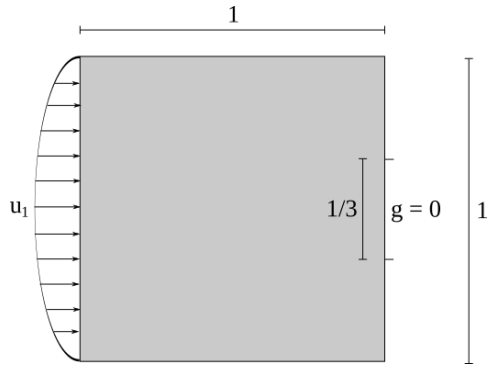


Figure 7.6: Computational domain for the diffuser problem.

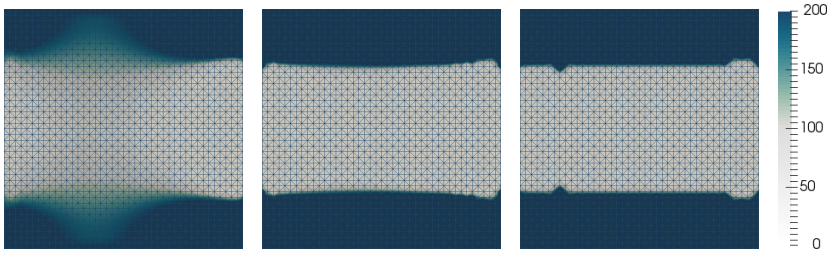
The objective functional is the pressure difference between the inlet and outlet, and the fluid volume penalization term,

$$J(\mathbf{u}, \alpha) = \int_0^T p_{in}(t) - p_{out}(t)\mathrm{d}t + \beta|\omega(\psi)|, \tag{7.5}$$

where $p_{in}(t)$ is the pressure averaged spatially over the inlet and correspondingly for the outlet in p_{out}. The gradient is computed using the dolfin-adjoint python call

```
J = Functional(( p * ds(inlet) - p * ds(outlet) / (1./3)) * dt
               + beta * fluid_domain(psi) * dx * dt)
g = compute_gradient(J, Control(psi))
```

where `ds(inlet)` and `ds(outlet)` are the inlet and outlet boundary measures, respectively. The factor $1/3$ comes from the averaging over the outlet length, see Figure 7.6. The penalty parameter β is set to 95.0. Because we use dolfin-adjoint for the gradient computation, we need the impermeability on the domain to be differentiable. Hence, we apply an interpolation of α described in (2.12), with penalty parameter $q = 0.1$. Another observation, is that the optimization algorithm is unstable if the step size $\kappa = 1.0$ is used, so the backtracking line search algortihm, see Section 3.5, is applied in every iteration to find a decrease in the functional J.



(a) Iteration 1 **(b)** Iteration 2 **(c)** Iteration 15 **(d)**

Figure 7.7: Optimization procedure for diffusion problem.

The impermeability distribution in the two first and last iterations are presented in Figure 7.7. The evolvement of the objective function J and angle θ is shown in Figure 7.8. As θ did not converge to a tolerance ε_θ when applying the numerically computed gradient, insufficient change in the functional $\|J_{i+1} - J_i\|_{L^2} < \varepsilon_J = 0.01$ was used as stopping criterion. The optimization algorithm terminated after 15 iterations.

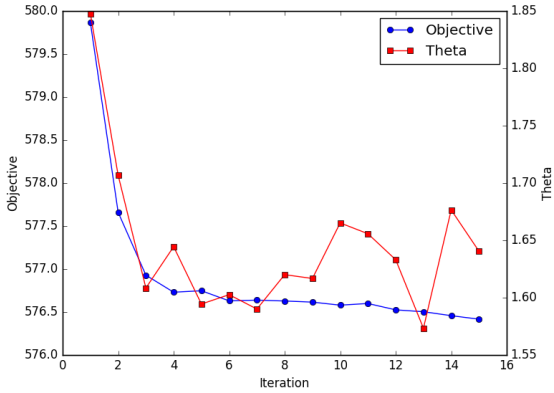


Figure 7.8: Evolvement of objective function J and angle θ in the optimization procedure for the diffuser problem.

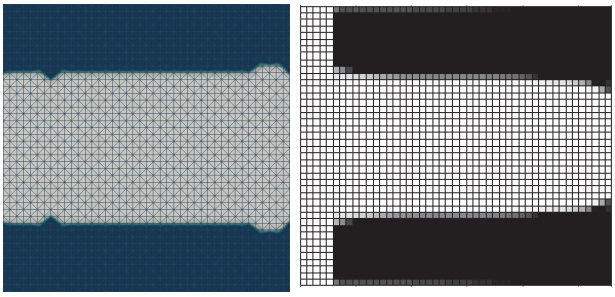
The optimal design is presented together with the optimal design from [12] in Figure 7.9, and the values for the objective functional $\tilde{J}$ and fluid fraction γ of these are presented in Table 7.2. The values of $\tilde{J}$ are computed by the formula

$$\tilde{J} = \frac{1}{N_t} \sum_{n=1}^{N_t} (p_{\text{in}}^{\text{tot}}(t^n) - p_{\text{out}}^{\text{tot}}(t^n)), \tag{7.6}$$

where $N_t = 15$ is the number of time steps, and the inlet and outlet pressures, $p_{\text{in}}^{\text{tot}}$ and $p_{\text{out}}^{\text{tot}}$ are averaged spatially over the inlet and outlet ports. (7.6) is collected from equation (35), and the value of the objective function is obtained from Table III, both in [12].

The two optimal shapes presented in Figure 7.9 are computed with the same interpolation penalty for the impermeability α, but in Figure 7.9b a fluid layer has been added in order to resolve the unsteady flow close to the inlet. This was not done in our case, and in addition our mesh has twice as many elements, due to the fact that FEniCS does not support quadrilateral elements, only simplices. The objective values presented in Table 7.2 are highly different, and must be a result of different ways of averaging the pressure. The author has so far not assimilated how this has been calculated in [12]. The gradient-based algorithm with a numerically computed gradient (using dolfin-adjoint), does not behave in the same way as with the analytical topological gradient, as it is unstable for step size $\kappa = 1.0$. A backtracking line search is applied in order to ensure sufficient decrease in the objective J, but the algorithm performes better with a constant step size $\kappa < 1$. Combining a default step size of $\kappa = 0.5$ with a backtracking line search show better results

in few iterations, and is applied to the algorithm when using the numerically computed gradient. In addition, the optimality criterion θ did not behave as expected when using the numerically computed gradient, see Figure 7.8. This is another indication that the numerical gradient is not fully a descent direction, and caution should be made when applying this. Further work should investigate how the gradient behaves during mesh refinement. The advantage of using the numerically computed gradient, is that it is not necessary to derive the topological derivative when applying different objective functionals.



(a) Optimized diffuser design. **(b)** Diffuser design from [12].

Figure 7.9: Optimal designs for the diffuser problem.

Table 7.2: Functional value J and fluid fraction γ for the optimal design and design in [12] for the diffuser problem.

	$\tilde{J}$	γ
Fig. 7.9a	576.36	0.527
Fig. 7.9b	36 400	0.500

7.3 Coronary Artery Bypass Anastomosis Models

The gradient-based optimization algorithm is applied to a coronary artery bypass anastomosis with the purpose of finding an optimal shape. We refer the reader to Section 1.2 for background information and terminology on coronary artery bypass anastomoses.

In the literature, Quarteroni, Rozza and Agoshov [1, 8, 9, 10] have investigated the reduction in vorticity when optimizing a known shape of the toe of the bypass. Since we are dealing with topology optimization, we have no prior information about the shape of the bypass. This opens up for the possibility to optimize the whole domain of the bypass, starting with just the obstructed artery. Instead of looking at the relative reduction of vorticity, as we have no reference shape, we will investigate the optimal shapes with respect to minimizing different objective functionals, including vorticity, for different domains.

The inlet velocity used in the bypass experiments is based on data from [11], and is plotted in Figure 7.10, with a time period of $T = 0.8$ seconds, which corresponds to a pulse of 75

beats per minute. Notice that the velocity is negative at around $t = 0.1\ s$.

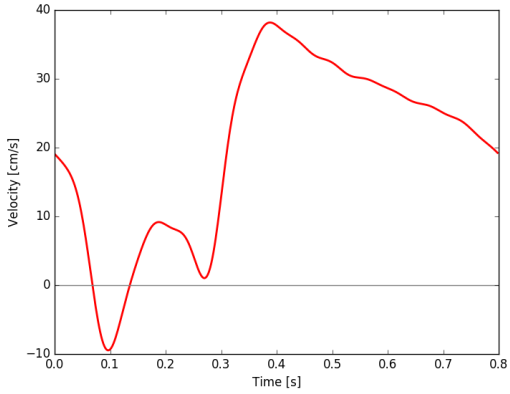


Figure 7.10: Velocity profile for the left anterior descending coronary artery. Data taken from [11].

The kinematic viscosity is $\nu = 0.04\ \mathrm{cm}^2/\mathrm{s}$, the blood density $\rho = 1.0\ \mathrm{g}/\mathrm{cm}^3$, and the arterial diameter is $D = 0.35\ \mathrm{cm}$ [2, 9]. The mean velocity is set to $\bar{u} = 20.0\ \mathrm{cm}/\mathrm{s}$, which corresponds to a Reynolds number $Re = \bar{u} \cdot D/\nu$ of 175. This is reasonable according to [3].

In order to model a bypass, we have made some assumptions. The design domain for the bypass is set to $L = 6.0\ \mathrm{cm}$, and the inlet velocity has a Poiseuille profile. We assume that the artery is fully occluded. The fluid penalization parameter β is affected by the size of the domain, velocity magnitude and kinematic viscosity, and will be adjusted in every case such that the bypass channel is sufficiently narrow. For the material distribution α, we have chosen a penalty factor $q = 0.1$.

We will consider the energy dissipation functional

$$J_1(\mathbf{u}, \alpha) = \int_0^T \int_\Omega \left(\nu |\nabla \mathbf{u}|^2 + \alpha |\mathbf{u}|^2 \right) \mathrm{d}t + \beta |\omega|, \tag{7.7}$$

and the vorticity functional

$$J_2(\mathbf{u}, \alpha) = \int_0^T \int_\Omega \left(\nu |\nabla \times \mathbf{u}|^2 + \alpha |\mathbf{u}|^2 \right) \mathrm{d}t + \beta |\omega|, \tag{7.8}$$

where $|\omega|$ is the volume of the fluid in the design domain. For efficient switching between functionals, all gradients are computed numerically using dolfin-adjoint [45].

7.3.1 Simple Bypass Model

A simple model, illustrated in Figure 7.11, is implemented for the bypass problem. The area under the dashed line represents the coronary artery, and the area above is the design

domain. A rectangle part of width 0.2 cm and height 0.35 cm is removed where the occlusion is. The time-dependent inlet velocity presented in Figure 7.10 is applied with a Poiseuille profile at the inlet, and a traction-free boundary condition is imposed at the outlet. For every iteration, the level set function in the coronary artery sub-domain is enforced to have negative value, ensuring fluid representation in this area. The mesh for this model is created using `gmsh` [48], with a spatial step size of 0.05 cm around the occlusion, and 0.1 cm elsewhere, in total 4534 cells. Figure 7.12 shows an illustration of the mesh.

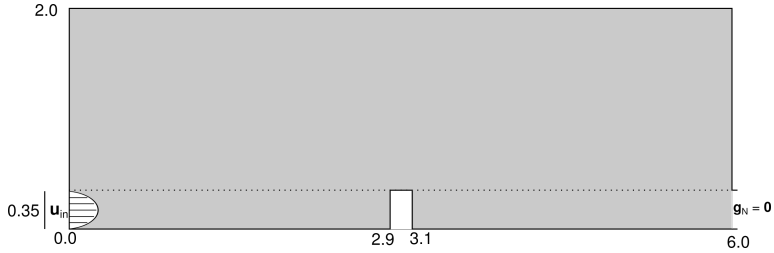


Figure 7.11: Illustration of the simple bypass model. All spatial measures are in cm.

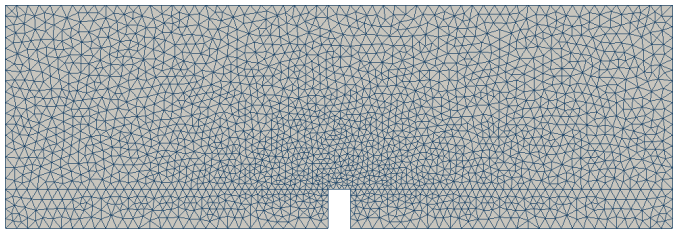


Figure 7.12: Grid for the simple bypass model.

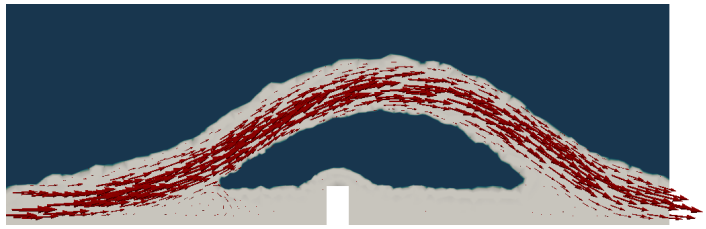
A preliminary experiment for the energy dissipation functional (7.7) with 10% of the inlet velocity magnitude was implemented, corresponding to a Reynolds number of $Re = 17.5$. Figure 7.13 shows the optimal shape. A small opening around the occlusion is observed, although little fluid movement is detected in the area. The opening is suspected to come from the update of the level set function in the coronary artery domain at every iteration, thus affecting the topological gradient. Experiments without the manual level-set update were tested without satisfactory results, see Section 8.2 for a suggestion on further investigation.

The simple model for the bypass was implemented for the energy dissipation functional with mean velocity $\bar{u} = 20.0$. The penalty parameter used was $\beta = 500.0$ and the time step size $\Delta t = 0.002$. The optimal shape was found after 20 iterations, and is presented in Figure 7.14, together with an illustration of the fluid flow at end time $T = 0.8$ s and the optimal shape with the mesh. It is easily observed that the optimal shape in Figure 7.14a differs from the one in Figure 7.13 with lower Reynolds number. This observation enhances the importance of implementing a model that represents the actual physical

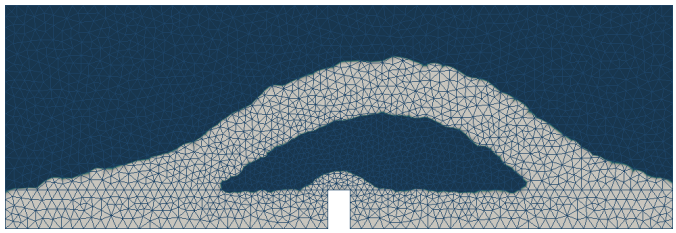
Figure 7.13: Optimal shape for the bypass problem with respect to the energy dissipation functional, with 10 % of the velocity magnitude and $\beta = 10.0$.

properties of the system with the correct scaling of variables.

(a) Optimal shape iterations.



(b) Fluid flow in optimal shape at time $T = 0.8$ s.



(c) Optimal shape with mesh.

Figure 7.14: Optimial shape for the simple bypass model with respect to the energy dissipation functional (7.7).

The simple model for the bypass was also implemented for the vorticity functional (7.8) with mean velocity $\bar{u} = 20.0$. The penalty parameter used was $\beta = 500.0$ and the time step size $\Delta t = 0.002$. The optimal shape was found after 20 iterations, and is presented in Figure 7.15 with the fluid flow at end time $T = 0.8$ s.

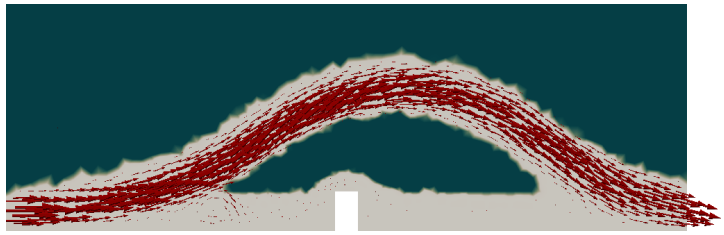


Figure 7.15: Fluid flow at time $T = 0.8$ s in optimal shape for the simple bypass model with respect to the vorticity functional (7.8).

The optimal shapes for the two functionals are very similar. Figure 7.16 shows the functional values for the energy dissipation functional J_1 and the vorticity functional J_2 at each iteration. It seems that the functional values at early iterations are strongly dependent on the fluid penalization term, $\beta|\omega|$, as the functional values follow the same pattern.

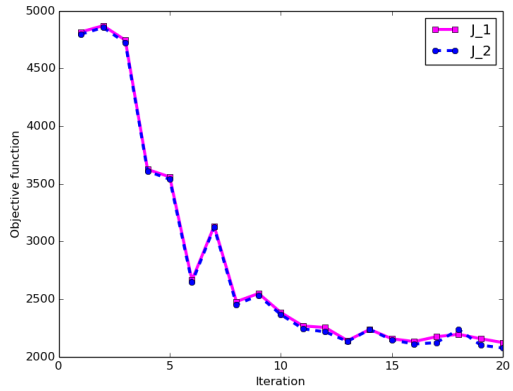


Figure 7.16: Evolvement of the objective functionals (7.7) and (7.8) for the simple model.

7.3.2 Extended Bypass Model

An extended model, illustrated in Figure 7.17, is also implemented for the bypass problem, where 1.0 cm is added to the coronary artery on each side, in order to resolve the unsteady flow close to the inlet and outlet. The mesh for the extended model is created using `gmsh` [48], with a spatial step size of 0.05 cm along the upper part of the coronary artery, and 0.1 cm elsewhere, in total 6626 cells. Figure 7.18 shows an illustration of the mesh.

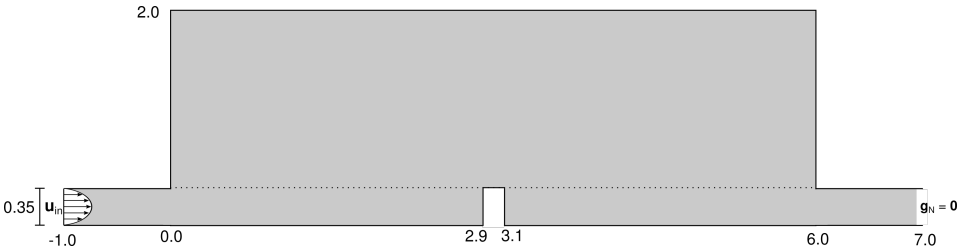


Figure 7.17: Illustration of the extended bypass model.

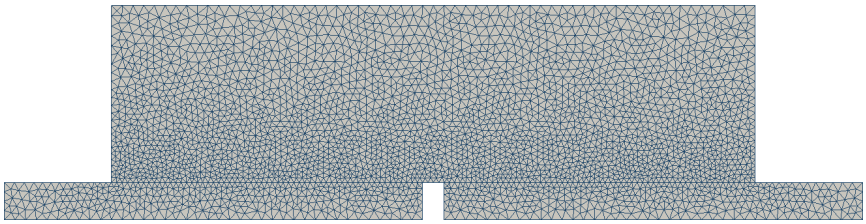


Figure 7.18: Mesh for the extended bypass model.

The extended model for the bypass was used with the energy dissipation functional and vorticity functional, with mean velocity $\bar{u} = 20.0$. The penalty parameter is set to $\beta = 500.0$ and the time step size $\Delta t = 0.002$. The optimal shape for the energy dissipation functional J_1 was found after 19 iterations, and the optimal shape for the vorticity functional J_2 was found after 17 iterations. The optimal shapes are presented with fluid flow at end time $T = 0.8$. in Figure 7.20. The optimal shapes for the two functionals are in this case also very similar.

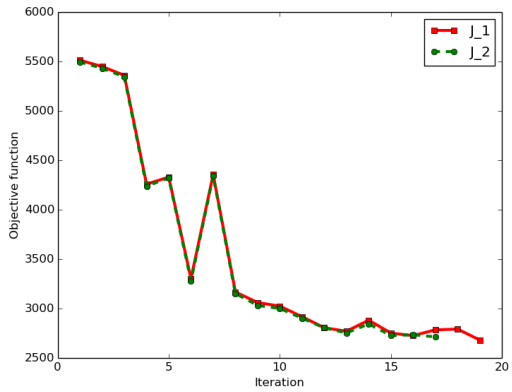
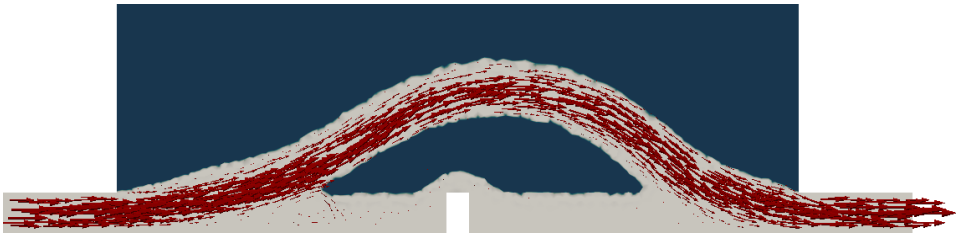
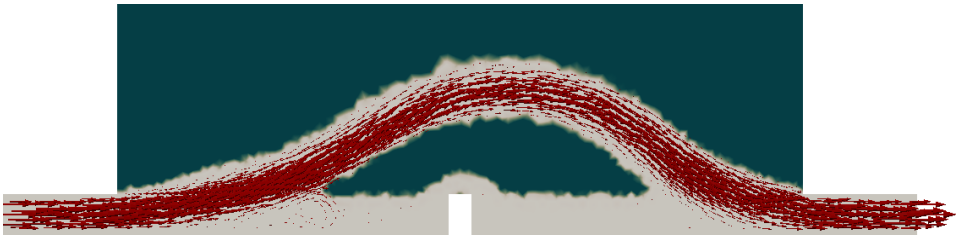


Figure 7.19: Evolvement of the objective functionals (7.7) and (7.8) for the extended model.

Figure 7.19 shows the functional values for the energy dissipation functional J_1 and the vorticity functional J_2 at each iteration. Also for the extended model, the functional values follow the same pattern at early iterations, which means that also in this model the fluid volume penalization term $\beta|\omega|$ dominates the functional value.



(a) Energy dissipation functional



(b) Vorticity functional

Figure 7.20: Fluid flow at time $T = 0.8$ s in optimal shape for the extended bypass model for the different functionals.

7.3.3 Comparison of the Two Models

The evolvement of the objective function during each optimization procedure is presented in Figure 7.21. The objective values seem to follow a similar pattern, even for the two different models. The peak at iteration 7 is in all four cases caused by intermediate impermeability values inside of the channel. As this is not beneficial, the impermeability in the channel is restored by the algorithm at iteration 8.

Table 7.3: Numerical results of the different bypass models with respect to the energy dissipation functional J_1 and vorticity functional J_2.

	Objective value	β	Number of iterations
Simple J_1	2123	500.0	20
Simple J_2	2076	500.0	20
Extended J_1	2680	500.0	19
Extended J_2	2711	500.0	17

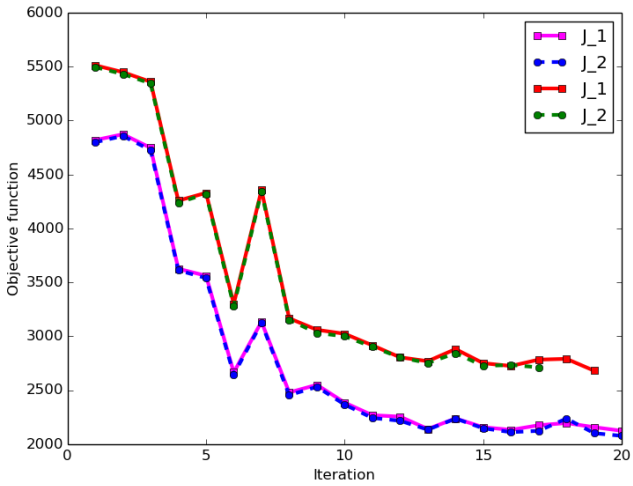
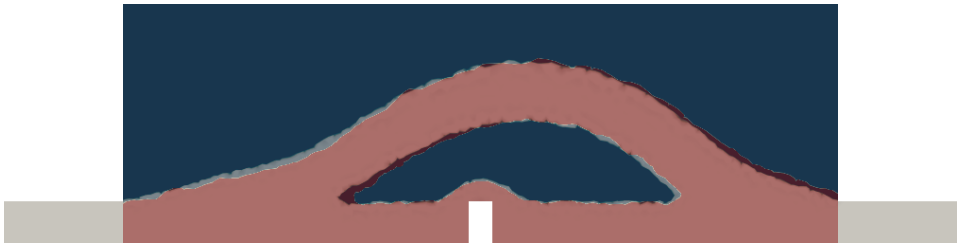
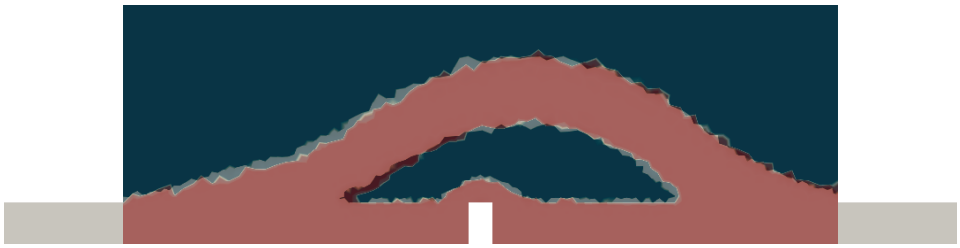


Figure 7.21: Evolvement of the objective functionals J_1 (7.7) and J_2 (7.8) for the simple and extended model.

The objective functional values for the four different cases are presented in Table 7.3 together with penalty value β and number of iterations.



(a) Energy dissipation functional



(b) Vorticity functional

Figure 7.22: Comparing of the optimal shapes for the simple and extended bypass model. The channel for the simple model is colored red.

The optimal shapes for the simple and extended model are compared for both functionals in Figure 7.22. Small differences, especially at the inlet and outlet of the design domain, are observed. The optimal shapes for the simple model have a wider outlet than for the extended model, which favors the extended model. As the dissipation energy and vorticity was calculated on the whole domain, the functional values are not comparable in the two models. This can easily be computed by calculating the energy dissipation and vorticity in the common domains. See Section 8.2 for further discussion on improvement of these models.

Chapter 8

Conclusion and Recommendations for Further Work

8.1 Conclusion

In this thesis, we have focused on developing a stable and efficient topology optimization method for determining the shape design of unsteady two-dimensional biomedical flows. Using the numerical method presented in this thesis, the optimal shape with respect to an arbitrary objective functional of for instance a bypass can be found, given no information about the initial shape.

To model the fluid flow, the unsteady incompressible Navier-Stokes equations have been combined with Darcy's equation. The inclusion of the Darcy term makes it possible to determine the impermeability in the domain material, distinguishing solid from fluid. The Navier-Stokes-Darcy (NSD) system has been discretized and solved using a finite element approach with the numerical software FEniCS [46] and an incremental pressure correction scheme (IPCS), see Chapter 5.

A gradient-based topology optimization algorithm proposed in [23] has been successfully implemented. The algorithm is based on a level set method that determines the impermeability in every cell, and is updated using the topological derivative. For the NSD system, the topological derivative associated with the energy dissipation functional has been derived in Section 4.7 and used as descent direction in the gradient-based algorithm. Knowledge of the adjoint solution for the NSD system is necessary in order to calculate the topological derivative. The adjoint equations for the NSD system have been derived and solved using a finite element method and a coupled solver. The implementation details can be found in Appendix A.2.

Deriving the adjoint model and topological derivative can be a very difficult and time

consuming task. Using the functionality of dolfin-adjoint [45], we have been able to investigate other functionals than the energy dissipation functional in a fast and easy way. In few lines of code, we have computed the topolgical gradient numerically, based on the forward solution of the Navier-Stokes-Darcy system. The core of the implementation using dolfin-adjoint is listed in Appendix A.4.

The IPCS solver has been verified numerically on Taylor-Green flow and a problem with manufactured solution in Chapter 6, showing convergence of 2nd order in space and 1st order in time. For high values of the kinematic viscosity in problems with Dirichlet boundary conditions, an erroneous boundary layer was discovered. This was diminished by lowering the time step size sufficiently. The gradient computed using dolfin-adjoint has been compared to the topolgical derivative, showing some dissimilarities, but a convergence in space of order 1/2. The optimization algorithm has been applied and compared to examples in other publications on topology optimization for unsteady flow [12, 24], showing similar results.

The optimization algorithm has been applied to a coronary artery bypass anastomosis in Section 7.3 in order to optimize the bypass shape with respect to energy dissipation or vorticity. This has been done for two different models of the bypass, and the optimal shapes are very similar. A differently shaped channel was detected for lower Reynolds numbers, which indicates that a simplified model with respect to length scales should be avoided in order to get reliable results. An additional small channel close to the occlusion is observed in the optimal shapes, but little fluid flow is detected through it. This is due to a weakness in the algorithm. Apart from this erroneousness, in conclusion the topological optimization algorithm seems to handle the biomedical fluid flow problem adequately.

8.2 Further Work

The further work of this thesis should include applying the optimization method to the bypass problem with higher Reynolds number, as the highest number in this thesis was $Re = 175$. In the literature [1, 8, 9, 10], a Reynolds number of order 10^3 is suggested. In addition a longer model for the bypass problem should be tested, in order to find the dependency in optimal shape with respect to the length of the design domain. Also, a more detailed model, with for instance removal of some of the domain above the coronary artery could be implemented, correcting the opening close to the occlusion. Due to time constraints, this was not carried out in this thesis.

Another interesting topic would be to extend the model to 3D, and investigate techniques for reducing the computational costs. Implementation in three dimensions is generally not a trivial case, as one has to code in parallel and handle the large amount of data. Not all models require three-dimensional simulations to give useful information about the problem. However, for realistic simulations of blood flow, a three-dimensional model is necessary. An extension to three dimensions would also enable the topology optimization method for other problems where a more realistic model is required. As the mathematics in this thesis are dimension-independent, only the implementation has to be extended.

It would also be interesting to further investigate the difference between the gradient computed using dolfin-adjoint and the topolgical derivative, and what this is caused by, as they differed more than was expected. For instance, one could look at the solution with a very fine grid, as the numerically computed gradient should approach the analytical when the space step size becomes small. One could also apply the numerical gradient to the double-pipe problem in Section 7.1 and compare the solution to the optimal shape and functional evolvement computed with the topological derivative.

Bibliography

[1] Alfio Quarteroni and Gianluigi Rozza. Optimal control and shape optimization of aorto-coronaric bypass anastomoses. *Mathematical Models and Methods in Applied Sciences*, 13(12):1801–1823, 2003.

[2] J.S. Cole, J.K. Watterson, and M.J.G. O'Reilly. Is there a haemodynamic advantage associated with cuffed arterial anastomoses? *Journal of Biomechanics*, 35(10):1337 – 1346, 2002.

[3] Armin Leuprecht, Karl Perktold, Martin Prosi, Thomas Berk, Wolfgang Trubel, and Heinrich Schima. Numerical study of hemodynamics and wall mechanics in distal end-to-side anastomoses of bypass grafts. *Journal of Biomechanics*, 35(2):225–236, 2002.

[4] J.S. Cole, J.K. Watterson, and M.J.G. O'Reilly. Numerical investigation of the haemodynamics at a patched arterial bypass anastomosis. *Medical Engineering & Physics*, 24(6):393 – 401, 2002.

[5] JS Cole, LD Wijesinghe, JK Watterson, and DJA Scott. Computational and experimental simulations of the haemodynamics at cuffed arterial bypass graft anastomoses. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 216(2):135–143, 2002.

[6] http://thoracickey.com/infrainguinal-disease-2/. Accessed 31-January-2017.

[7] Bruce Blaus. https://commons.wikimedia.org/w/index.php?curid=29140356. Accessed 17-January-2017.

[8] Gianluigi Rozza. On optimization, control and shape design of an arterial bypass. *International Journal for Numerical Methods in Fluids*, 47(10-11):1411–1419, 2005.

[9] Valery Agoshkov, Alfio Quarteroni, and Gianluigi Rozza. Shape design in aorto-coronaric bypass anastomoses using perturbation theory. *SIAM Journal on Numerical Analysis*, 44(1):367–384, 2006.

[10] Valery Agoshkov, Alfio Quarteroni, and Gianluigi Rozza. A mathematical approach in the design of arterial bypass using unsteady Stokes equations. *Journal of Scientific Computing*, 28(2):139–165, 2006.

[11] Jonathan B. Thomas, Luca Antiga, Susan L. Che, Jaques S. Milner, Dolores A. Hangan Steinman, J. David Spence, Brian K. Rutt, and David A. Steinman. Variation in the carotid bifurcation geometry of young versus older adults. *Stroke*, 36(11):2450–2456, 2005.

[12] Sebastian Kreissl, Georg Pingen, and Kurt Maute. Topology optimization for unsteady flow. *International Journal for Numerical Methods in Engineering*, 87(13):1229–1253, 2011.

[13] Diego Esteves Campeão, Sebastian Miguel Giusti, and Andre Antonio Novotny. Topology design of plates considering different volume control methods. *Engineering Computations*, 31(5):826–842, 2014.

[14] J.A. Sethian and Andreas Wiegmann. Structural boundary design via level set and immersed interface methods. *Journal of Computational Physics*, 163(2):489 – 528, 2000.

[15] Thomas Borrvall and Joakim Petersson. Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids*, 41(1):77–107, 2003.

[16] James K. Guest and Jean H. Prévost. Topology optimization of creeping fluid flows using a Darcy–Stokes finite element. *International Journal for Numerical Methods in Engineering*, 66(3):461–484, 2006.

[17] Martin Burger, Benjamin Hackl, and Wolfgang Ring. Incorporating topological derivatives into level set methods. *J. Comput. Phys.*, 194(1):344–362, February 2004.

[18] Samuel Amstutz and Heiko Andrä. A new algorithm for topology optimization using a level-set method. *J. Comput. Phys.*, 216(2):573–588, August 2006.

[19] Vivien J. Challis and James K. Guest. Level set topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Engineering*, 79(10):1284–1308, 2009.

[20] A. Gersborg-Hansen, O. Sigmund, and R.B. Haber. Topology optimization of channel flow problems. *Structural and Multidisciplinary Optimization*, 30(3):181–192, 2005.

[21] J. Sokolowski and A. Zochowski. On the topological derivative in shape optimization. *SIAM Journal on Control and Optimization*, 37(4):1251–1272, 1999.

[22] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, 58(8):861–877, 2008.

[23] L. F. N. Sá, R. C. R. Amigo, A. A. Novotny, and E. C. N. Silva. Topological derivatives applied to fluid flow channel design optimization problems. *Structural and Multidisciplinary Optimization*, 54(2):249–264, 2016.

[24] Yongbo Deng, Zhenyu Liu, Ping Zhang, Yongshun Liu, and Yihui Wu. Topology optimization of unsteady incompressible Navier-Stokes flows. *J. Comput. Phys.*, 230(17):6688–6708, July 2011.

[25] Yongbo Deng, Zhenyu Liu, and Yihui Wu. Topology optimization of steady and unsteady incompressible Navier–Stokes flows driven by body forces. *Structural and Multidisciplinary Optimization*, 47(4):555–570, 2013.

[26] Antonio André Novotny and Jan Sokołowski. *Topological derivatives in shape optimization*. Springer Science & Business Media, 2012.

[27] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, November 1988.

[28] Samuel Amstutz. Analysis of a level set method for topology optimization. *Optimization Methods and Software*, 26(4-5):555–573, 2011.

[29] Donald A Nield and Adrian Bejan. *Convection in porous media*. Springer Science & Business Media, 2006.

[30] H. C. Brinkman. A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. *Flow, Turbulence and Combustion*, 1(1):27, 1949.

[31] Aycil Cesmelioglu, Vivette Girault, and Béatrice Rivière. Time-dependent coupling of Navier–Stokes and Darcy flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(2):539–554, 003 2013.

[32] Alfio Quarteroni. *Numerical models for differential problems*, volume 2. Springer Science & Business Media, 2010.

[33] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[34] Bryan P Rynne and Martin A Youngson. *Linear functional analysis*. Springer Science & Business Media, 2000.

[35] Simon W Funke and Patrick E Farrell. A framework for automated PDE-constrained optimisation. *arXiv preprint arXiv:1302.3894*, 2013.

[36] Kai A. James and Haim Waisman. Topology optimization of viscoelastic structures using a time-dependent adjoint method. *Computer Methods in Applied Mechanics and Engineering*, 285:166–187, 3 2015.

[37] W. Rudin. *Real and complex analysis*. Mathematics series. McGraw-Hill, 1987.

[38] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.

[39] H. P. Langtangen, K.-A. Mardal, and R. Winther. Numerical methods for incompressible viscous flow. *Advances in Water Resources*, 25:1125–1146, 2002.

[40] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

[41] Michel O Deville, Paul F Fischer, and Ernest H Mund. *High-order methods for incompressible fluid flow*, volume 9. Cambridge University Press, 2002.

[42] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44–47):6011 – 6045, 2006.

[43] Hans Johnston and Jian-Guo Liu. Accurate, stable and efficient Navier–Stokes solvers based on explicit treatment of the pressure term. *Journal of Computational Physics*, 199(1):221 – 259, 2004.

[44] F.M. White. *Viscous Fluid Flow*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1991.

[45] Patrick E Farrell, David A Ham, Simon W Funke, and Marie E Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393, 2013.

[46] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015.

[47] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: Symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[48] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

Appendix

Appendix A

Numerical Implementation

A.1 Navier-Stokes-Darcy Solver

```
from dolfin import *

class NSDSolver:

    def __init__(self, problem):
        self.mesh = problem.mesh
        self.boundaries = problem.boundaries
        self.T = problem.T
        self.dt = problem.dt

    def solve_IPCS(self, psi):

        # Function spaces
        V = VectorFunctionSpace(self.mesh, "CG", 2)
        Q = FunctionSpace(self.mesh, "CG", 1)

        ds = Measure("ds")(subdomain_data=self.boundaries)
        n = FacetNormal(self.mesh)

        # Define trial and test functions
        u = TrialFunction(V)
        p = TrialFunction(Q)
        v = TestFunction(V)
        q = TestFunction(Q)

        u0 = Function(V, name='u0')
        u1 = Function(V, name='u1')
        p0 = Function(Q, name='p0')
        p1 = Function(Q, name='p1')

        k = Constant(self.dt)
```

```
        # Tentative velocity step
        F1 = (1. / k) * inner(u - u0, v) * dx \
            + inner(grad(u0) * u0, v) * dx \
            + nu * inner(grad(u), grad(v)) * dx \
            + inner(grad(p0), v) * dx \
            + alpha(psi) * inner(u, v) * dx \
            - nu * inner(dot(n, nabla_grad(u)), v) * ds(3) \
            + inner(p0 * n, v) * ds(3)

        a1 = lhs(F1)
        L1 = rhs(F1)

        # Pressure update
        a2 = inner(grad(p), grad(q)) * dx
        L2 = inner(grad(p0), grad(q)) * dx \
            - ((1. / k) + alpha(psi)) * div(u1) * q * dx

        # Velocity update
        a3 = (1. / k + alpha(psi)) * inner(u, v) * dx
        L3 = (1. / k + alpha(psi)) * inner(u1, v) * dx \
            - inner(grad(p1 - p0), v) * dx

        # Assemble system
        A1 = assemble(a1)
        A2 = assemble(a2)
        A3 = assemble(a3)

        b1 = None
        b2 = None
        b3 = None

        t = 0.0

        # Save forward solution
        u_list = []

        while t < self.T:
            bcu, bcp = problem.boundary_conditions(V, Q, t)
            t += self.dt

            # Tentative velocity
            b1 = assemble(L1, tensor=b1)
            [bc.apply(A1, b1) for bc in bcu]
            solve(A1, u1.vector(), b1)

            # Pressure correction
            b2 = assemble(L2, tensor=b2)
            [bc.apply(A2, b2) for bc in bcp]
            solve(A2, p1.vector(), b2)

            # Velocity update
            b3 = assemble(L3, tensor=b3)
            [bc.apply(A3, b3) for bc in bcu]
            solve(A3, u1.vector(), b3)

            # Ensure deep copy:
```

```
            u_temp = Function(V)
            u_temp.assign(u1)
            u_list.append(u_temp)

            u0.assign(u1)
            p0.assign(p1)

        return u_list
```

A.2 Adjoint Solver

```
from dolfin import *

class AdjointSolver:

    def __init__(self, problem):
        self.mesh = problem.mesh
        self.boundaries = problem.adjoint_boundaries
        self.T = problem.T
        self.dt = problem.dt

    def solve_coupled(self, psi, u_list):

        # psi is the level set function
        # u_list contains forward solution

        # Adjoint coupled:
        P2 = VectorElement('P', 'triangle', 2)
        P1 = FiniteElement('P', 'triangle', 1)
        TH = P2 * P1
        W = FunctionSpace(self.mesh, TH)

        ds = Measure("ds")(subdomain_data=self.boundaries)
        n = FacetNormal(self.mesh)

        (u, p) = TrialFunctions(W)
        (v, q) = TestFunctions(W)
        w = Function(W)
        (ua, pa) = split(w)

        # in order to assign w.sub(0) to u_a
        V = VectorFunctionSpace(self.mesh, "CG", 2)
        assigner = FunctionAssigner(V, W.sub(0))

        # Adjoint system with Neumann boundary conditions
        F_a = (1. / k) * inner(u - ua, v) * dx \
                  - inner(dot(grad(u), u0), v) * dx \
                  - inner(dot(u0, grad(u)), v) * dx \
                  + nu * inner(grad(u), grad(v)) * dx \
                  + alpha(psi) * inner(u, v) * dx \
                  - div(u) * q * dx - div(v) * p * dx \
                  - 2 * nu * inner(grad(u0), grad(v)) * dx \
                  - 2 * alpha(psi) * inner(u0, v) * dx \
```

```
                - inner(p * n, v) * ds(3) \
                + nu * inner(dot(n, grad(u)), v) * ds(3) \
                + inner(dot(u0, u) * n, v) * ds(3) \
                + inner(dot(u0, n) * u, v) * ds(3)
            a = lhs(F_a)
            L = rhs(F_a)

        # save adjoint solution
        ua_list = []

        # Constant Dirichlet boundary conditions
        bcu = problem.adjoint_boundary_conditions(V)

        k = Constant(self.dt)
        t = self.T

        for u0 in reversed(u_list):
            t -= self.dt

            problem = LinearVariationalProblem(a, L, w, bcu)
            solver = LinearVariationalSolver(problem)
            solver.solve()
            (ua, pa) = split(w)

            # ensure deep copy
            u_a = Function(V)
            assigner.assign(u_a, w.sub(0))
            ua_list.append(u_a)

        # return reversed ua_list such that it has the same ordering
        # as u_list
        return ua_list[::-1]
```

A.3 Gradient-Based Method

```
from dolfin import *

class GradientMethod:

    def __init__(self, problem):
        self.problem = problem
        self.V0 = FunctionSpace(problem.mesh, "CG", 1)
        self.PSI = "-1"
        self.n_max = 20
        self.tol_theta = 0.01
        self.tol_kappa = 0.001

    def functional(self, psi, u_list):
        J = 0.0
        N = len(u_list)
        for u in u_list:
            J += problem.dt * assemble(alpha(psi) * inner(u, u) * dx
                                       + nu * inner(grad(u), grad(u)) * dx)
        J += assemble(self.beta * fluid_domain(psi) * dx)
```

```
        return J

    def gradient(self, g, u_, ua_):
        N = len(u_)
        g.vector().zero()
        for i in range(N):
            u = u_[i]
            ua = ua_[i]
            direction = - (ALPHA_U - ALPHA_L) * inner(u, u - ua)
            g_temp = project(direction + self.beta, self.V0)
            g.vector().axpy(1./N, g_temp.vector())

    def angle(self, g, psi):
        dot_prod = assemble(dot(g, psi) * dx)
        nrm_g = sqrt(assemble(dot(g, g) * dx))
        nrm_psi = sqrt(assemble(dot(psi, psi) * dx))
        return acos(dot_prod / (nrm_g * nrm_psi))

    def new_psi(self, psi, kappa, theta, psi_ref, g):
        nrm_g = sqrt(assemble(dot(g, g) * dx))
        nrm_psi = sqrt(assemble(dot(psi_ref, psi_ref) * dx))
        k1 = sin((1 - kappa) * theta) / (sin(theta) * nrm_psi)
        k2 = sin(kappa * theta) / (sin(theta) * nrm_g)
        psi.vector().zero()
        psi.vector().axpy(k1, psi_ref.vector())
        psi.vector().axpy(k2, g.vector())

    def run_gradient_method(self):

        kappa = 1.0

        # Design parameter
        psi_ref = project(Expression(self.PSI, degree=1), self.V0)
        nrm_psi = sqrt(assemble(dot(psi_ref, psi_ref) * dx))
        psi = project(Expression(self.PSI) / nrm_psi, self.V0)
        g = Function(self.V0)

        NSD = NSDSolver(self.problem)
        adjoint = AdjointSolver(self.problem)

        u_list = NSD.solve_IPCS(psi)
        ua_list = adjoint.solve_coupled(psi, u_list)

        J = self.functional(psi, u_list)

        # Iterate until convergence
        for n in range(self.n_max):

            print "Iteration #: " + str(n+1)

            self.gradient(g, u_list, ua_list)
            theta = self.angle(g, psi)
            if theta < self.tol_theta:
                print "Theta smaller that tolerance!"
                break

            kappa = min(1.0, kappa*1.5)
```

```
            # Store old psi and update new psi
            psi_ref.assign(psi)
            self.new_psi(psi, kappa, theta, psi_ref, g)

            # Solve forward and adjoint system
            u_list = NSD.solve_IPCS(psi)
            ua_list = adjoint.solve_coupled(psi, u_list)

            # calculate new functional
            J_new = self.functional(psi, u_list)


            # Line search
            while J_new > J and kappa > self.tol_kappa:
                # Did not find smaller J, decreasing kappa."
                kappa = kappa*0.5

                # Update psi and calculate new J
                self.new_psi(psi, kappa, theta, psi_ref, g)
                u_list = NSD.solve_IPCS(psi)
                J_new = self.functional(psi, u_list)

            if J_new < J:
                J = J_new
            else:
                # Did not find smaller J, terminating
                return psi, J

        return psi, J
```

A.4 Implementation with dolfin-adjoint

```
from dolfin import *
from dolfin_adjoint import *

class NSDsolver:

    def __init__(self, problem):
        self.problem = problem
        self.mesh = problem.mesh
        self.boundaries = problem.boundaries

    def solve(self, psi):

        adj_reset()
        timestep = problem.dt

        # Function spaces
        V = VectorFunctionSpace(self.mesh, "CG", 2)
        Q = FunctionSpace(self.mesh, "CG", 1)

        ds = Measure("ds")(subdomain_data=self.boundaries)
```

```
        n = FacetNormal(self.mesh)

        # Initial velocity and pressure
        u0, p0 = problem.initial_values()

        # Declare velocity functions
        u = Function(V, name="u")
        v = TestFunction(V)

        # Weak form for tentative velocity
        F1 = (1./timestep) * inner(u - u0, v) * dx \
            + inner(grad(u) * u0, v) * dx \
            + nu * inner(grad(u), grad(v)) * dx \
            + alpha(psi) * inner(u, v) * dx \
            + inner(grad(p0), v) * dx \
            - nu * inner(dot(n, nabla_grad(u)), v) * ds(2) \
            + inner(p0 * n, v) * ds(2)


        # Declare pressure functions
        p = Function(Q, name="p")
        q = TestFunction(Q)

        # Pressure correction weak form
        F2 = inner(grad(p - p0), grad(q)) * dx \
             + (1. / timestep) * div(u0) * q * dx


        # Velocity update weak form
        F3 = (1./timestep)*inner(u - u0, v) * dx \
             + inner(grad(p - p0), v) * dx

        t = 0.0
        psi_tmp = psi.copy(deepcopy=True)

        adj_time = True
        annotate = True
        adj_start_timestep()

        while t < problem.T:

            # u and p are trial functions
            # u0 and p0 are newest value

            # Boundary conditions, Pousille flow at inlet
            bcu, bcp = problem.boundary_conditions(V, Q, t)

            solve(F1 == 0, u, bcu)
            u0.assign(u, annotate=annotate)
            solve(F2 == 0, p, bcp)
            solve(F3 == 0, u, bcu)

            psi.assign(psi_tmp, annotate=annotate)
            u0.assign(u, annotate=annotate)
            p0.assign(p, annotate=annotate)

            # plot(u0, interactive=True)
```

```
            t += float(timestep)
            if adj_time:
                adj_inc_timestep(t, t > problem.T)

        # dolfin-adjoint only needs the last solution of u and p
        return u0, p0


class GradientMethod:
    def __init__(self, problem):
        self.problem = problem
        self.mesh = problem.mesh
        self.boundaries = problem.boundaries
        self.V0 = FunctionSpace(self.mesh, "CG", 1)
        self.PSI = "-1"
        self.n_max = 20
        self.kappa_tol = 0.001

    def new_psi(self, kappa, psi, g):

        dot_prod = assemble(dot(g, psi) * dx)
        nrm_g = sqrt(assemble(dot(g, g) * dx))
        nrm_psi = sqrt(assemble(dot(psi, psi) * dx))

        theta = acos(dot_prod / (nrm_g * nrm_psi))
        k1 = sin((1 - kappa) * theta) / (sin(theta) * nrm_psi)
        k2 = sin(kappa * theta) / (sin(theta) * nrm_g)

        psi = project(k1 * psi - k2 * g, self.V0, annotate=False)
        return psi

    def run_gradient_method(self):

        ds = Measure("ds")(subdomain_data=self.boundaries)
        kappa = 0.5

        # Design parameter
        psi_ref = project(Expression(self.PSI, degree=1), self.V0)
        nrm_psi = sqrt(assemble(dot(psi_ref, psi_ref) * dx))
        psi = project(Expression(self.PSI, degree=2) / nrm_psi,
                      self.V0, annotate=True)

        # Initialize problem and Navier-Stokes-Darcy solver
        nsd = NSDsolver(self.problem)
        u, p = nsd.solve(psi)

        # Dolfin-adjoint functional
        J = Functional((p * ds(1) - 3. * p * ds(2)) * dt
                       + beta * fluid_domain(psi) * dx * dt)

        for i in range(self.n_max):
            print "\nIteration #" + str(i+1)

            g = compute_gradient(J, Control(psi), forget=False)
```

```
            # Store old values of J and psi:
            Jm = ReducedFunctional(J, Control(psi))
            J_old = Jm(psi)
            psi_ref.assign(psi)

            # Update psi and calculate new J
            kappa = min(0.5, kappa * 1.5)
            psi = self.new_psi(kappa, psi_ref, g)
            u, p = nsd.solve(psi)

            J = Functional((p * ds(1) - 3. * p * ds(2)) * dt
                           + beta * fluid_domain(psi) * dx * dt)
            Jm = ReducedFunctional(J, Control(psi))

        return psi, J
```