NTNU

Norwegian University of
Science and Technology

Efficient Methods for Topology Optimisation of Fluid flow

Petter Johnsen Frisvåg

Master of Science in Physics and Mathematics
Submission date: July 2018
Supervisor: Anton Evgrafov, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Project description

The goal of this project is to quantify the discretisation error related to the finite element discretisation of fluid flow boundary value problems in topology optimisation, using a posteriori error analysis. This information will then be utilised in the context of approximate linear algebraic solvers needed within topology optimisation algorithms and in the context of adaptive mesh refinement.

Abstract

We consider topology optimisation of two-dimensional Stokes flow. The objective is to distribute a certain amount of solid material in a given domain, such that the total power dissipation is minimised. A generalised Stokes problem acts as the governing PDE, and both existence of solutions to the state equations and the optimisation problem is shown. The optimisation problem is solved using the optimality criteria method, and the MINRES method is used for solving the algebraic linear system arising from discretising the governing PDE with the finite element method. Residual estimates are used in order to prematurely stop MINRES, and the results indicate that the residual related to the momentum part of the Stokes equations is sufficient for formulating a meaningful stopping criterion. Through testing the algorithm on several different numerical examples, we propose a tolerance such that the total number of MINRES iterations is low, while largest observed error in the resulting objective value is 3.5 % compared to when the linear system is solved exactly. Adaptive mesh refinement based on the elementwise residual estimates is performed during the course of the optimisation, and it was found that this approach yields a significant reduction in the residuals when compared to starting with a fine mesh.

Sammendrag

Vi ser på topologioptimering av todimensjonal Stokes-flyt. Målet er å fordele en viss mengde fast materiale i et gitt domene, slik at det totale effekttapet er minimert. Et generalisert Stokes-problem opptrer som PDEen, og både eksistens og unikhet av løsninger til tilstandslikningene blir vist. Optimeringsproblemet blir løst ved bruk av optimalitetskriterier-metoden, og MINRES-metoden blir brukt for å løse det algebraiske, lineære systemet som oppstår ved å diskretisere PDEen med element-metoden. Residualestimater blir brukt for å tidlig stoppe MINRES, og resultatene indikerer at bevegelsesmengde-delen av Stokes-likningene er tilstrekkelig for å formulere et meningsfullt stoppkriterium. Ved å teste algoritmen på flere forskjellige numeriske eksempler, foreslår vi en toleranse slik at MINRES bruker få iterasjoner totalt, samtidig som den største observerte feilen i den resulterende målverdien er 3.5 % sammenliknet med når det lineære systemet løses eksakt. Adaptiv gitterforfining basert på de elementvise residualestimatene blir utført under optimeringen, og det ble funnet at denne tilnærmingen gir en markant reduksjon i residualene sammenliknet med å starte med et fint gitter.

Preface

This thesis is submitted in partial fulfilment of the requirements for a master's degree at the Department of Mathematical Sciences at the Norwegian University of Science and Technology. I would like to express my gratitude to my supervisor Professor Anton Evgrafov, for suggesting this interesting topic for my master's thesis and for his tireless support and supervision throughout this project. I also want to thank my dear friend Nicolaj Nielsen, for proofreading the thesis and suggesting numerous improving formulations.

Contents

Chapter 1

Introduction

Topology optimisation is a branch of optimal control which is concerned with how an isotropic material should be distributed in a domain such that a certain property is minimised. The fact that we want to optimise the topological properties of a domain motivates what is in the literature referred to as a 0-1 or *black-white* property, which means that we seek optimal controls which resemble an indicator function.

Historically, topology optimisation of isotropic materials has mainly been applied to the discipline of solid mechanics, with the objective of minimising the compliance in a structure exposed to external forces when only a limited amount of material is available. In recent years, however, this strategy have also been adapted to fluid mechanics, where both the Stokes and Navier-Stokes equations are among the models subject to the optimisation.

In this report we will consider an optimal control problem for minimising the energy loss of Stokes flow in a material by controlling the material distribution. This strategy was first introduced in [BP03], where the generalised Stokes problem

$$\begin{aligned} -\mu\Delta\mathbf{u} + \alpha\mathbf{u} + \nabla p &= \mathbf{f}, \\ -\nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{1.1}$$

was used as a basis for formulating an optimal control framework for Stokes flow. The details of (1.1) will be discussed in the following chapter, and this PDE system will be the state equations of the optimal control problem. The idea is that given a predetermined domain, corresponding boundary conditions and a minimum amount of solid material occupying the domain, the goal is to distribute said material such the dissipated power is at a minimum.

The power loss acts as the objective functional in the optimal control problem considered, which depends on the solution of the state equations. We use the finite element method to solve (1.1) numerically, and more precisely by using the software library FEniCS. As it is be necessary to solve (1.1) repeatedly over the course of the optimisation algorithm we will be using, it is crucial that this can be done efficiently. We therefore utilise a Krylov subspace method in order to solve

the linear system, and to this end a significant part of the report will be devoted how early this method can be stopped while still obtaining sufficiently accurate solutions of (1.1).

The report is structured as follows. In Chapter 2 the details of (1.1) is presented, and the existence and uniqueness of solutions is proved.

Chapter 3 then deals with the optimisation problem, and the necessary optimality conditions are derived. It is proven that solutions to the optimisation problem exists, and an algorithmic framework based on the optimality conditions is proposed for solving the problem.

Chapter 4 addresses the details of the finite element method for the generalised Stokes problem and the properties of the resulting linear system of algebraic equations. The concept of residuals is then introduced in order to estimate the accuracy of the numerical solution to (1.1), which will be central in later chapters. Lastly, a numerical implementation of the algorithm discussed in Chapter 3 is presented, which is then applied to several different numerical benchmarks.

Next, in Chapter 5 and adaptive mesh refinement strategy for the finite element method is introduced, based on the residuals presented in the preceding chapter. The adaptive mesh refinement is then applied to a selection of the numerical examples presented previously in order to reduce the discretisation error related to the mesh.

Chapter 6 first presents the Krylov subspace method MINRES for solving (1.1), and a stopping criterion based on the residuals from Chapter 4 is proposed. Finally, MINRES with the proposed stopping criterion is then applied to all the numerical examples considered previously, and the results are compared to those obtained in Chapter 4.

Chapter 2

The state equations

Several differential equations for modelling viscous, incompressible fluid flow exists, with the famous Navier-Stokes equations arguably being the most prevalent in both the literature and engineering. Although widely used, the non-linear nature of the Navier-Stokes equations often leads to the usage of simpler, linear models in the development phase of numerical algorithms.

2.1 The generalised Stokes equations

In this report we will focus on the *generalised Stokes equations*, stated in the previous chapter. Considering (1.1), $\mathbf{u} = \mathbf{u}(\mathbf{x})$ denotes the velocity of the fluid, where the bold font emphasises that it is a vector-valued function. Δ in turn denotes the vector Laplacian, defined by taking the Laplacian in each component of $\mathbf{u}$, while $p = p(\mathbf{x})$ the pressure in the fluid at a point $\mathbf{x} \in \Omega$. $\alpha = \alpha(\mathbf{x}) \geq 0$ is a given function which we will interpret as the inverse permeability of a porous medium, as governed by Darcy's law. With this interpretation in mind, the generalised Stokes equations (2.1) will interchangeably be referred to as the *Darcy-Stokes equations* in this report. Finally, $\mu > 0$ is the viscosity and $\mathbf{f}$ denotes the body forces acting on fluid.

The generalised Stokes equations appears when the non-linear convective term in Navier-Stokes [Qua14, Ch. 16] is neglected, a simplification which is reasonable for very slow fluids, such that the Reynold's number satisfies the condition $Re \ll 1$. It also appears as a subproblem of the Navier-Stokes equations when solved numerically using Picard linearisation [BL07].

Regarding the domain of (1.1), we let this be defined as $\Omega \subset \mathbb{R}^d$, which we will assume to be open, bounded and Lipschitz continuous on the boundary. We consider pure Dirichlet boundary conditions, such that $\mathbf{u} = \mathbf{g}(\mathbf{x})$ on $\partial\Omega$ for a given function $\mathbf{g}$. In this report we focus on $d = 2$, however we note that all the results presented will also be valid for $d = 3$.

Before we continue, note that if we define $\tilde{\mathbf{u}} = \mu\mathbf{u}$, then (1.1) can be written

$$\begin{aligned} -\Delta\tilde{\mathbf{u}} + \frac{\alpha}{\mu}\tilde{\mathbf{u}} + \nabla p &= \mathbf{f}, \\ -\nabla \cdot \tilde{\mathbf{u}} &= 0, \end{aligned}$$

and with $\tilde{\mathbf{u}} = \mu\mathbf{g}$ on $\partial\Omega$. These equations are on the same form as before, meaning α and $\mathbf{g}$ can be scaled such that μ disappears from the problem. Hence, without loss of generality we will assume $\mu = 1$ from now on, and the complete boundary value problem reads

$$\begin{cases} -\Delta\mathbf{u} + \alpha\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega, \\ -\nabla \cdot \mathbf{u} = 0 & \text{in } \Omega, \\ \mathbf{u} = \mathbf{g} & \text{on } \partial\Omega. \end{cases} \tag{2.1}$$

The first equation describes the conservation of momentum in the fluid, while second equation is the incompressibility constraint, which enforces the conservation of mass. The two first equations in (2.1) will therefore be referred to as the *momentum* and *mass* equation, respectively.

Note that p in (2.1) only appears inside a gradient, meaning that it is only uniquely determined up to a constant, provided it exists. In order to avoid this indeterminacy we add a *zero mean constraint*, that is we will require

$$\int_\Omega p = 0.$$

Throughout this report we will, for the sake of brevity, generally omit the integration variables in the integrals whenever it is clear what the domain of integration is.

For the case $\alpha(\mathbf{x}) = 0$ everywhere in Ω, the generalised Stokes equations reduces to the *pure Stokes equations*, and is the type of flow we ultimately want to model. We will expand on interpretation of α and the relationship between the Darcy-Stokes equations and the pure Stokes equations in Chapter 3. However, before doing that we address some important properties of (2.1), starting with deriving its weak form in the next section.

2.2 Variational formulation

Apart from deriving the weak form of (2.1), in this section we introduce some important notation.

First, throughout this report the velocity $\mathbf{u}$ will formally belong to the Sobolev space [Trö10, Sec. 2.2] of one time weakly differentiable functions, $H^1(\Omega, \mathbb{R}^2) = W^{1,2}(\Omega, \mathbb{R}^2)$. As all vectors will be written in boldface, there is no immediate risk for confusion, so for the sake of brevity we will generally omit the dimension and write $H^1(\Omega)$ and $L^2(\Omega)$ for $H^1(\Omega, \mathbb{R}^2)$ and $L^2(\Omega, \mathbb{R}^2)$, respectively.

Secondly, define the function spaces

$$\begin{aligned} V &= H_0^1(\Omega) = \{\, \mathbf{v} \in H^1(\Omega) \,:\, \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega \,\}, \\ V_{\text{div}} &= \{\, \mathbf{v} \in V \,:\, \nabla \cdot \mathbf{v} = 0 \,\}, \\ Q &= L_0^2(\Omega) = \{\, q \in L^2(\Omega) \,:\, \int_\Omega q = 0 \,\}, \end{aligned} \tag{2.2}$$

along with the set $H^1_{\mathbf{g}}(\Omega) = \{\, \mathbf{v} \in H^1(\Omega) \,:\, \mathbf{v} = \mathbf{g} \text{ on } \partial\Omega \,\}$. Here V_{div} is the space of divergence free functions, while Q is the space of functions satisfying the zero mean constraint mentioned in the foregoing section. All three spaces in (2.2) are Hilbert spaces, with inner products defined as

$$\begin{aligned} (\mathbf{u}, \mathbf{v})_V &= (\mathbf{u}, \mathbf{v})_{H^1(\Omega)} = \int_\Omega \mathbf{u} \cdot \mathbf{v} + \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}, \\ (p, q)_Q &= (p, q)_{L^2(\Omega)} = \int_\Omega pq, \end{aligned}$$

for functions $\mathbf{u}, \mathbf{v} \in V$ and $p, q \in Q$. The vector gradient of $\mathbf{u}$ is defined $\nabla \mathbf{u} = [\nabla u_1 \dots \nabla u_d]$, that is the $d \times d$ matrix whose columns are the gradient of the entries in $\mathbf{u}$. Consequently, $\nabla \mathbf{u} : \nabla \mathbf{v}$ denotes the Frobenius inner product of $\nabla \mathbf{u}$ and $\nabla \mathbf{v}$, defined as

$$\nabla \mathbf{u} : \nabla \mathbf{v} = \sum_{i,j=1}^{d} (\nabla \mathbf{u})_{ij} (\nabla \mathbf{v})_{ij}.$$

Similarly, the norms of V and Q are naturally defined as

$$\begin{aligned} \|\mathbf{u}\|_V &= \sqrt{(\mathbf{u}, \mathbf{u})_V}, \\ \|p\|_Q &= \sqrt{(p, p)_Q}, \end{aligned}$$

respectively.

In order to derive the variational form of the Darcy-Stokes equations, we multiply the first equation of (2.1) with a test function $\mathbf{v} \in V$. Taking the integral over Ω, integration by parts then yields

$$\int_\Omega (-\Delta \mathbf{u} \cdot \mathbf{v} + \alpha \mathbf{u} \cdot \mathbf{v} + \nabla p \cdot \mathbf{v}) = \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + \int_\Omega \alpha \mathbf{u} \cdot \mathbf{v} - \int_\Omega p \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{f} \cdot \mathbf{v}.$$

Similarly, for the second equation we have simply multiply with a test function $q \in Q$ and integrate, resulting

$$-\int_\Omega q \nabla \cdot \mathbf{u} = 0.$$

Motivated by the form of the above two equations, we define the bilinear forms

$$\begin{aligned} a_\alpha(\mathbf{u}, \mathbf{v}) &= \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + \int_\Omega \alpha \mathbf{u} \cdot \mathbf{v}, \\ b(\mathbf{v}, p) &= -\int_\Omega p \nabla \cdot \mathbf{v}, \end{aligned} \tag{2.3}$$

meaning the weak form of (2.1) now can be formulated as: Find $(\mathbf{u}, p) \in H^1_{\mathbf{g}}(\Omega) \times Q$ such that

$$\begin{aligned} a_\alpha(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} \qquad \forall \mathbf{v} \in V, \\ b(\mathbf{u}, q) &= 0 \qquad \forall q \in Q, \end{aligned} \tag{2.4}$$

where we define $\alpha \in L^\infty(\Omega)$ such that $\alpha \geq 0$ almost everywhere in Ω. As usual, "almost anywhere" and "for almost all" refers to everywhere in Ω except possibly on set of zero measure [Rud87].

It is possible to transform (2.4) to an equivalent formulation with homogeneous boundary conditions, that is such that the space for the test function $\mathbf{v}$ and trial function $\mathbf{u}$ coincide. To see this, let $\mathbf{u} = \mathring{\mathbf{u}} + \mathbf{Rg}$, where $\mathbf{Rg} \in H^1_{\mathbf{g}}(\Omega)$ is a lifting of the boundary datum $\mathbf{g}$, such that $\mathring{\mathbf{u}} \in V$. Inserting $\mathring{\mathbf{u}} + \mathbf{Rg}$ into (2.4) and rearranging, we get

$$\begin{aligned} a_\alpha(\mathring{\mathbf{u}}, \mathbf{v}) + b(\mathbf{v}, p) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} - a_\alpha(\mathbf{Rg}, \mathbf{v}) \qquad \forall \mathbf{v} \in V, \\ b(\mathring{\mathbf{u}}, q) &= -b(\mathbf{Rg}, q) \qquad \forall q \in Q. \end{aligned} \tag{2.5}$$

In order to continue, we need the surjectivity property of the divergence operator in V, which we get from the following result:

Lemma 2.1. For all $p \in Q$ there is a $\mathbf{v} \in V$ satisfying

$$\nabla \cdot \mathbf{v} = p.$$

Moreover, there exists a constant $C > 0$ such that

$$\|\mathbf{v}\|_V \leq C \|p\|_Q.$$

The lemma is a special case of [BS07, Lemma 11.2.3], where a proof can also be found. We let $\mathring{\mathbf{u}} = \mathring{\mathbf{u}}_0 + \mathring{\mathbf{u}}_1$, with $\mathring{\mathbf{u}}_0, \mathring{\mathbf{u}}_1 \in V$ and such that $\nabla \cdot \mathring{\mathbf{u}}_1 = -\nabla \cdot \mathbf{Rg}$. By definition of $\mathbf{g}$ we have

$$\int_\Omega -\nabla \cdot \mathbf{Rg} = -\int_{\partial\Omega} \mathbf{Rg} \cdot \mathbf{n} = -\int_{\partial\Omega} \mathbf{g} \cdot \mathbf{n} = 0,$$

meaning $-\nabla \cdot \mathbf{Rg} \in Q$, so $\mathring{\mathbf{u}}_1$ exists owing to Lemma 2.1. Inserting $\mathring{\mathbf{u}}_0 + \mathring{\mathbf{u}}_1$ for $\mathring{\mathbf{u}}$ in (2.5) then finally yields

$$\begin{aligned} a_\alpha(\mathring{\mathbf{u}}_0, \mathbf{v}) + b(\mathbf{v}, p) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} - a_\alpha(\mathring{\mathbf{u}}_1 + \mathbf{Rg}, \mathbf{v}) \qquad \forall \mathbf{v} \in V, \\ b(\mathring{\mathbf{u}}_0, q) &= 0 \qquad \forall q \in Q, \end{aligned}$$

showing that $\mathring{\mathbf{u}}_0$ solves the weak formulation of the Darcy-Stokes equations in the case of homogeneous boundary conditions. Because this transformation is possible, we will for the remaining of this chapter consider the case $\mathbf{g} = 0$ in (2.4).

We restate the entire problem to avoid confusion, and for future reference: Find $(\mathbf{u}, p) \in V \times Q$ such that

$$\begin{aligned} a_\alpha(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} \qquad \forall \mathbf{v} \in V, \\ b(\mathbf{u}, q) &= 0 \qquad \forall q \in Q. \end{aligned} \tag{2.6}$$

2.3 Existence and uniqueness of solutions

We prove existence and uniqueness of solutions to (2.6), which is equivalent to (2.4). Starting by discussing the continuity of a_α and b, we have the following proposition:

Proposition 2.2. a_α and b are continuous, meaning

$$\begin{aligned} a_\alpha(\mathbf{u},\mathbf{v}) &\le C_1\|\mathbf{u}\|_V\|\mathbf{v}\|_V \qquad \forall \mathbf{u},\mathbf{v}\in V,\\ b(\mathbf{v},p) &\le C_2\|\mathbf{u}\|_V\|p\|_Q \qquad \forall \mathbf{v}\in V, p\in Q \end{aligned}$$

for positive constants C_1 and C_2.

Proof. The proof follows from the Cauchy-Schwarz inequality. For the first term in a_α we have

$$\begin{aligned} \left|\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\right| &\le \left(\int_\Omega |\nabla\mathbf{u}|^2\right)^{1/2}\left(\int_\Omega |\nabla\mathbf{v}|^2\right)^{1/2}\\ &\le \left(\int_\Omega (|\nabla\mathbf{u}|^2+|\mathbf{u}|^2)\right)^{1/2}\left(\int_\Omega (|\nabla\mathbf{v}|^2+|\mathbf{v}|^2)\right)^{1/2} = \|\mathbf{u}\|_V\|\mathbf{v}\|_V. \end{aligned}$$

Similarly, for the second term

$$\begin{aligned} \left|\int_\Omega \alpha\mathbf{u}\cdot\mathbf{v}\right| &\le \|\alpha\|_{L^\infty(\Omega)}\left(\int_\Omega |\mathbf{u}|^2\right)^{1/2}\left(\int_\Omega |\mathbf{v}|^2\right)^{1/2}\\ &\le \|\alpha\|_{L^\infty(\Omega)}\left(\int_\Omega (|\mathbf{u}|^2+|\nabla\mathbf{u}|^2)\right)^{1/2}\left(\int_\Omega (|\mathbf{v}|^2+|\nabla\mathbf{v}|^2)\right)^{1/2}\\ &= \|\alpha\|_{L^\infty(\Omega)}\|\mathbf{u}\|_V\|\mathbf{v}\|_V. \end{aligned}$$

Consequently, we get

$$a_\alpha(\mathbf{u},\mathbf{v}) \le |a_\alpha(\mathbf{u},\mathbf{v})| \le C_1\|\mathbf{u}\|_V\|\mathbf{v}\|_V$$

with $C_1 = 1+\|\alpha\|_{L^\infty(\Omega)}$.

To prove continuity for $b(\mathbf{v},p)$ we need an estimate for $\nabla\cdot\mathbf{v}$. Recall that in general $\mathbf{v}:\mathbb{R}^d\to\mathbb{R}^d$, so

$$\begin{aligned} (\nabla\cdot\mathbf{v})^2 &= \left(\sum_{i=1}^d \frac{\partial v_i}{\partial x_i}\right)^2 = \sum_{i,j=1}^d \frac{\partial v_i}{\partial x_i}\frac{\partial v_j}{\partial x_j} \le \sum_{i,j=1}^d \frac{1}{2}\left(\left(\frac{\partial v_i}{\partial x_i}\right)^2+\left(\frac{\partial v_j}{\partial x_j}\right)^2\right)\\ &= d\sum_{i=1}^d \left(\frac{\partial v_i}{\partial x_i}\right)^2 \le d\sum_{i,j=1}^d \left(\frac{\partial v_i}{\partial x_j}\right)^2 = d|\nabla\mathbf{v}|^2. \end{aligned}$$

Hence, $|\nabla\cdot\mathbf{v}| \le \sqrt{d}|\nabla\mathbf{v}|$. Continuity for $b(\mathbf{v},p)$ follows immediately:

$$\begin{aligned} \left|-\int_\Omega p\nabla\cdot\mathbf{v}\right| &\le \left(\int_\Omega |\nabla\cdot\mathbf{v}|^2\right)^{1/2}\left(\int_\Omega p^2\right)^{1/2} \le \sqrt{d}\left(\int_\Omega |\nabla\mathbf{v}|^2\right)^{1/2}\|p\|_{L^2(\Omega)}\\ &\le \sqrt{d}\left(\int_\Omega |\nabla\mathbf{v}|^2+|\mathbf{v}|^2\right)^{1/2}\|p\|_{L^2(\Omega)} = \sqrt{d}\|\mathbf{v}\|_V\|p\|_Q, \end{aligned}$$

meaning $C_2 = \sqrt{d}$. □

In addition to continuity of $a_\alpha(\cdot,\cdot)$ and $b(\cdot,\cdot)$, we will also need a coercivity property. The next proposition discusses the coercivity of a_α.

Proposition 2.3. $a_\alpha(\cdot,\cdot)$ in (2.3) is coercive in V, that is there exists a constant $C > 0$ such that

$$C\|\mathbf{v}\|_V^2 \leq a_\alpha(\mathbf{v},\mathbf{v}) \quad \forall \mathbf{v} \in V. \tag{2.7}$$

Proof. Since $\alpha \geq 0$ almost everywhere, the second term in a_α is non-negative, yielding $a_0(\mathbf{v},\mathbf{v}) \leq a_\alpha(\mathbf{v},\mathbf{v}) \, \forall \mathbf{v} \in V$. Additionally, the Poincaré inequality [BS07, Proposition 5.3.5] guarantees that there exists a constant $\tilde{C} > 0$ such that

$$\|\mathbf{v}\|_V \leq \tilde{C}\|\nabla \mathbf{v}\|_{L^2(\Omega)} \quad \forall \mathbf{v} \in V,$$

so coercivity of a_α is seen by

$$a_\alpha(\mathbf{v},\mathbf{v}) \geq a_0(\mathbf{v},\mathbf{v}) = \int_\Omega |\nabla \mathbf{v}|^2 = \|\nabla \mathbf{v}\|_{L^2(\Omega)}^2 \geq \frac{1}{\tilde{C}^2}\|\mathbf{v}\|_V^2.$$

Thus, the inequality (2.7) is satisfied with $C = 1/\tilde{C}^2$. □

As the spaces in the arguments of $b : V \times Q \to \mathbb{R}$ do not coincide, another definition than (2.7) is needed in order to examine the coercivity properties of $b(\cdot,\cdot)$. Motivated by (2.7), define

$$C\|\mathbf{v}\|_V \leq \sup_{\mathbf{w} \in V} \frac{a_\alpha(\mathbf{w},\mathbf{v})}{\|\mathbf{w}\|_V},$$

which is equivalent to (2.7) when $\mathbf{w} = \mathbf{v}$. This is the motivation for the following *inf-sup* definition, also commonly referred to as the Ladyženskaja-Babuška-Brezzi (LBB) condition in the literature.

Definition 2.4 (inf-sup)**.** The bilinear functional $b : V \times Q \to \mathbb{R}$ satisfies the *inf-sup* condition if there exists a $\beta > 0$ such that

$$\beta\|p\|_Q \leq \sup_{\mathbf{v} \in V} \frac{b(\mathbf{v},p)}{\|\mathbf{v}\|_V} \quad \forall p \in Q. \tag{2.8}$$

The name "inf-sup" comes from the fact that the condition necessarily also have to hold for the p corresponding to the smallest right-hand side of (2.8), that is (2.8) is equivalently formulated as

$$\beta \leq \inf_{p \in Q} \sup_{\mathbf{v} \in V} \frac{b(\mathbf{v},p)}{\|\mathbf{v}\|_V \|p\|_Q}.$$

Having defined the inf-sup condition, we now state an important theorem that we will need in order to show existence and uniqueness of p in (2.6).

Theorem 2.5 (Babuška–Lax–Milgram)**.** *Let V and Q be Hilbert spaces. Assume that the bilinear functional $b : V \times Q \to \mathbb{R}$ is continuous and satisfies* (2.8)*, and let $F \in V'$. Then the variational problem: Find $p \in Q$ such that*

$$b(\mathbf{v}, p) = F(\mathbf{v}) \quad \forall \mathbf{v} \in V,$$

admits a unique solution.

Proof. See [EG04, Thm 2.6]. □

As the name implies, Theorem 2.5 is a generalisation of the well-known Lax-Milgram theorem. In fact, it is easy to see that if we replace Q with V in Theorem 2.5 and (2.8), Theorem 2.5 reduces to the usual Lax-Milgram theorem [BS07, Theorem (2.7.7)].

We are now ready to address the existence and uniqueness of the solutions to (2.6) in the following lemma.

Lemma 2.6. The solution $(\mathbf{u}, p) \in V \times Q$ of (2.6) exists and is unique. Furthermore we have the bound $\|\mathbf{u}\|_V \leq \|\mathbf{f}\|_{L^2(\Omega)}/C$, where C is the constant from (2.7).

Proof. In order to show existence and uniqueness of $\mathbf{u}$, note that $\mathbf{u}$ is equivalently defined as: Find $\mathbf{u} \in V_{\text{div}}$ such that

$$a_\alpha(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} \quad \forall \mathbf{v} \in V_{\text{div}}. \tag{2.9}$$

As $V_{\text{div}} \subset V$, having already shown continuity of a_α and b in V, along with coercivity of a_α in V, existence and uniqueness of $\mathbf{u}$ in (2.9) follows immediately by the Lax-Milgram theorem.

By Proposition 2.3 and (2.9), we have

$$\|\mathbf{v}\|_V^2 \leq \frac{1}{C} a_\alpha(\mathbf{v}, \mathbf{v}) = \frac{1}{C}(\mathbf{f}, \mathbf{v})_{L^2(\Omega)} \leq \frac{1}{C}\|\mathbf{f}\|_{L^2(\Omega)}\|\mathbf{v}\|_V \quad \forall \mathbf{v} \in V_{\text{div}},$$

yielding $\|\mathbf{v}\|_V \leq \|\mathbf{f}\|_{L^2(\Omega)}/C$. Here we have used the Cauchy–Schwarz inequality along with the obvious estimate $\|\mathbf{v}\|_{L^2(\Omega)} \leq \|\mathbf{v}\|_V$. Choosing $\mathbf{v} = \mathbf{u}$, we arrive at the desired bound.

Assume now that we have found the solution $\mathbf{u}$ in (2.6), for instance by solving (2.9). The pressure p is then determined by

$$b(\mathbf{v}, p) = (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} - a_\alpha(\mathbf{u}, \mathbf{v}) \quad \forall \mathbf{v} \in V, \tag{2.10}$$

where right-hand side of the above equation is a member of the dual space of V. Hence by Theorem 2.5, existence and uniqueness of p follows, provided $b(\cdot, \cdot)$ satisfies the inf-sup condition (2.8).

Using from Lemma 2.1 that there exists a $\mathbf{v} \in V$ such that $\nabla \cdot \mathbf{v} = -p$ and $1/\|p\|_Q \leq \tilde{C}/\|\mathbf{v}\|_V$, we find the upper bound

$$\|p\|_Q = \frac{\|p\|_Q^2}{\|p\|_Q} = \frac{1}{\|p\|_Q} \int_\Omega p(-\nabla \cdot \mathbf{v}) = \frac{b(\mathbf{v}, p)}{\|p\|_Q} \leq \tilde{C} \frac{b(\mathbf{v}, p)}{\|\mathbf{v}\|_V} \leq \tilde{C} \sup_{\mathbf{v} \in V} \frac{b(\mathbf{v}, p)}{\|\mathbf{v}\|_V},$$

showing that (2.8) holds when $\beta = 1/\tilde{C}$. Consequently, by Theorem 2.5 the variational problem (2.10) is uniquely solvable, meaning that p in (2.6) exists and is unique. □

Note that since the above argumentation holds for all $\alpha \in L^\infty(\Omega)$ such that $\alpha \geq 0$ almost everywhere, then if we choose $\alpha(\mathbf{x}) = 0$ for almost all $\mathbf{x} \in \Omega$, it follows that the solution to the *pure* Stokes equations exists and is unique.

Chapter 3

The optimisation problem

In the previous chapter the generalised Stokes equations were introduced, a system of two linear PDEs which can be interpreted as pure Stokes fluid governed by Darcy's law for fluids in a porous material [BP03]. In this chapter we interpret α^{-1} as the *permeability* of the material, which measures how easily a fluid can pass through it. Based on this interpretation we formulate an optimal control problem, in addition to presenting an optimisation algorithm for solving it.

3.1 Topology optimisation

We begin by introducing the objective function. In light of (2.6), the total potential power in the Darcy-Stokes equation is given by

$$J_\alpha(\mathbf{u}) = \frac{1}{2}a_\alpha(\mathbf{u},\mathbf{u}) - (\mathbf{f},\mathbf{u})_{L^2(\Omega)}, \tag{3.1}$$

which is a combination of the dissipative power in the fluid and the body forces $\mathbf{f}$ acting on the velocity. While the first term in (3.1) is non-negative, from the minus sign in front inner product involving $\mathbf{f}$ it is clear that the total power dissipation increases when the velocity $\mathbf{u}$ flows in a direction opposite of the body forces.

Note that the functional $\mathbf{v} \to J_{\alpha(\rho)}(\mathbf{v})$ is convex. To see this, observe first that

$$a_{\alpha(\rho)}(\mathbf{v},\mathbf{v}) = \int_\Omega |\nabla \mathbf{v}|^2 + \int_\Omega \alpha(\rho)|\mathbf{v}|^2$$

is continuous and convex in $\mathbf{v}$, as the operator $|\cdot|^2$ is convex and ∇ is linear. The third term is simply linear in and continuous in $\mathbf{v}$, so it is also convex.

Next we discuss the details of the inverse permeability function α. The permeability is defined such that a lower permeability yields more resistance in the material, yielding a higher energy loss. To this end, we define the $\rho(\mathbf{x})$ to be the scaled porosity of the material, that is ρ at given point $\mathbf{x}$ yields how large portion of the material is empty space, meaning that fluid can flow through it. As such, ρ

will act as the control function for our optimal control problem, which belongs to the set

$$P = \{\, \rho \in L^2(\Omega) \,:\, \rho_{\min} \leq \rho(\mathbf{x}) \leq 1 \text{ for almost all } \mathbf{x} \in \Omega \,\},$$

where the pointwise constraints $\rho_{\min} \leq \rho(\mathbf{x}) \leq 1$ are referred to as *box-constraints*. Here $0 < \rho_{\min} \ll 1$ will be a small lower bound, whose positivity will be necessary in order to address some theoretical aspects of the optimisation problem. In addition we require that the proportion of Ω occupied by fluid do not exceed some given volume fraction $\gamma \in (\rho_{\min}, 1)$, such that the set of admissible controls is defined

$$P_{ad} = \{\, \rho \in P \,:\, \int_\Omega \rho \leq \gamma |\Omega| \,\}. \tag{3.2}$$

In order to relate the porosity ρ to the inverse permeability α, we define $\alpha(\rho) : [\rho_{\min}, 1] \to [0, \bar{\alpha}]$, for some $\bar{\alpha} > 0$, to be a convex, continuously differentiable and strictly monotonically decreasing function. These properties will be necessary in order to show that our optimal control problem has solutions, where the monotonicity is intuitive considering we want a higher porosity to yield a lower permeability.

The Darcy-Stokes equations (2.6) will naturally play the role of the *state equations*, meaning that the optimal control problem reads

$$\min_{\rho \in P_{ad}} J_{\alpha(\rho)}(\mathbf{u}) \quad \text{where } (\mathbf{u}, p) \text{ solves (2.6) with } \alpha = \alpha(\rho). \tag{3.3}$$

In the next section the question regarding whether or not solutions to (3.3) exists is addressed.

3.2 Existence of optimal controls

Before discussing the optimality conditions for the optimal control problem (3.3) presented in the previous section, it is important to inspect whether or not there exists optimal controls for this problem. More precisely we want to inspect if there is a control $\rho^* \in P_{ad}$ with corresponding velocity $\mathbf{u}^*$ such that

$$J_{\alpha(\rho^*)}(\mathbf{u}^*) \leq J_{\alpha(\rho)}(\mathbf{u}) \quad \forall \rho \in P_{ad},$$

where $\mathbf{u}$ is the solution of (2.6) corresponding to ρ.

We start by showing that (3.3) is bounded from below. To do this, consider first the optimisation problem

$$\min_{\mathbf{v} \in V} J_{\alpha(\rho)}(\mathbf{v}) \quad \text{subject to} \quad \nabla \cdot \mathbf{v} = 0, \tag{3.4}$$

for some $\rho \in P$. Here we have omitted the first state equation in (2.6), and we minimise with respect to $\mathbf{v}$ rather than ρ. This optimisation problem has a unique solution, which can be seen from the following proposition.

Proposition 3.1. The optimisation problem (3.4) admits a unique solution. Additionally, the first order optimality conditions coincide with (2.6).

Proof. To address existence of stationary points of (3.4), consider instead the equivalent, unconstrained optimisation problem

$$\min_{\mathbf{v}\in V_{\text{div}}} J_{\alpha(\rho)}(\mathbf{v}).$$

We have $J_{\alpha(\rho)}(\mathbf{v}) \to \infty$ as $|\mathbf{v}| \to \infty$, which can be seen by the fact that a_α is the dominating term in J, and is coercive. By convexity and coercivity of $\mathbf{v} \to J_{\alpha(\rho)}(\mathbf{v})$, it follows that (3.4) has at least one stationary point.

Assume $q \in Q$ to be the Lagrange multiplier corresponding the incompressibility constraint, and define the Lagrangian

$$\mathcal{L}(\mathbf{v}, q) = J(\rho, \mathbf{v}) - \int_\Omega q\nabla \cdot \mathbf{v} = \frac{1}{2}a_\alpha(\mathbf{v}, \mathbf{v}) - (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} + b(\mathbf{v}, q).$$

Denote the stationary point $(\mathbf{u}, p) \in V \times Q$, which needs to satisfy the necessary conditions of (3.4), that is

$$\begin{aligned} \mathcal{L}'_{\mathbf{v}}(\mathbf{u}, p)\mathbf{v} = a_\alpha(\mathbf{u}, \mathbf{v}) - (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} + b(\mathbf{v}, p) &= 0 \quad \forall \mathbf{v} \in V, \\ \mathcal{L}'_q(\mathbf{u}, p)q = b(\mathbf{u}, q) &= 0 \quad \forall q \in Q, \end{aligned} \tag{3.5}$$

which is exactly (2.6). Since $\mathbf{v} \to J_{\alpha(\rho)}(\mathbf{v})$ is convex, (3.5) are also sufficient conditions. Owing to Lemma 2.6, the stationary point exists and is unique. □

As α is monotonically decreasing, we have $J_0(\mathbf{v}) \leq J_{\alpha(\rho)}(\mathbf{v})$ for all $\rho \in P$ and $\mathbf{v} \in V$. Hence, from the above proposition it follows that $J_0(\mathbf{u})$ is a lower bound on (3.4), where $\mathbf{u}$ now is the velocity part of the solution to the *pure* Stokes equations. Furthermore (3.4) is less constrained than (3.3), as in addition to not have the first equation in (2.6) as a constraint, we are allowed to choose ρ freely. This means that every admissible ρ and corresponding state $\mathbf{u}$ of (3.3) is also admissible for (3.4), and since (3.4) is bounded from below, then so is (3.3).

Since (3.3) is bounded, the infimum

$$j = \inf_{\rho\in P_{ad}} J_{\alpha(\rho)}(\mathbf{u})$$

exists, and we choose $\{(\rho_n, \mathbf{u}_n)\}_{n=1}^\infty$ to be a minimising sequence of J. That is $J_{\alpha(\mathbf{u}_n)}(\rho_n) \to j$ as $n \to \infty$, where $\mathbf{u}_n$ is the state corresponding to ρ_n. In order to discuss the properties of this sequence, we need the following proposition.

Proposition 3.2. Let $B_r \subset V$ be a closed ball [Wal14, p. 9] with radius r. The sets $V_{\text{div}} \cap B_r$ and P_{ad} are *weakly sequentially compact*, meaning that every sequence $\{\mathbf{v}_n\}_{n=1}^\infty \subset V_{\text{div}} \cap B_r$ and $\{\rho_n\}_{n=1}^\infty \subset P_{ad}$ contains a weakly convergent subsequence, with the limit being in the set.

Proof. [Trö10, Thm. 2.11] states that a bounded, convex and closed subset of a reflexive Banach space is weakly sequentially compact, so the proof amounts to showing that these properties hold for the two sets.

As a function $\rho \in P_{ad}$ is bounded from below by $\rho_{\min}$ and from above by 1 almost everywhere, we have $\|\rho\|_{L^2(\Omega)} \leq |\Omega|$, showing that P_{ad} is bounded. Furthermore, convexity of P_{ad} follows directly from the definition of convex sets, and reflexivity of $L^2(\Omega)$ is due to it being a Hilbert space.

To see that P_{ad} is closed w.r.t. the L^2 norm, assume that $\{\rho_n\}_{n=1}^{\infty} \subset P_{ad}$ is a convergent sequence, meaning $\|\rho_n - \bar{\rho}\|_{L^2(\Omega)} \to 0$ as $n \to \infty$ for some $\bar{\rho} \in L^2(\Omega)$. As $\{\rho_n\}_{n=1}^{\infty}$ converges strongly, it converges almost everywhere along a subsequence. In other words, this subsequence converges pointwise to $\bar{\rho}(\mathbf{x})$ almost everywhere, meaning $\rho_{\min} \leq \bar{\rho}(\mathbf{x}) \leq 1$ for almost all $\mathbf{x}$ in Ω. Convergence of the integral in P_{ad} follows from Lebesgue's dominated convergence theorem [Tao11, Thm. 1.4.49], implying that P_{ad} is closed.

Turning our attention to $V_{\text{div}} \cap B_r$, this set is trivially bounded by r. Regarding closedness, observe that V_{div} is by definition the null space of $\nabla\cdot : V \to L^2(\Omega)$. The divergence operator can be shown to be continuous in V, where the proof is similar to that for the continuity of $b(\cdot,\cdot)$ in the proof of Proposition 2.2. The singleton $\{0\}$ is trivially closed in $L^2(\Omega)$, meaning that V_{div} is closed. As the intersection of two closed sets is closed, it follows that $V_{\text{div}} \cap B_r$ is closed.

Lastly, convexity follows form the linearity of $\nabla\cdot$ and the fact that the intersection of two convex sets is convex, while reflexivity of V from the fact that it is a Hilbert space.

As all the properties of [Trö10, Thm. 2.11] hold for both sets, it follows that $V_{\text{div}} \cap B_r$ and P_{ad} are weakly sequentially compact. □

Using the bound $\|\mathbf{f}\|_{L^2(\Omega)}/C$ from Lemma 2.6 as r in the above proposition, we can extract a weakly convergent subsequence from $\{(\rho_n, \mathbf{u}_n)\}_{n=1}^{\infty}$. For the sake of brevity we simply reuse the index n for this new subsequence, such that

$$(\rho_n, \mathbf{u}_n) \rightharpoonup (\bar{\rho}, \bar{\mathbf{u}})$$

as $n \to \infty$ for some $(\bar{\rho}, \bar{\mathbf{u}}) \in P_{ad} \times V_{\text{div}}$, which will be our candidate for a minimiser of (3.3). In order to discuss what happens in the limit for $J_{\alpha(\rho_n)}(\mathbf{u}_n)$ as $n \to \infty$ we invoke the weakly lower semicontinuity of $(\rho, \mathbf{u}) \to J_{\alpha(\rho)}(\mathbf{u})$, which is given in the following proposition.

Proposition 3.3. The functional $(\rho, \mathbf{u}) \to J_{\alpha(\rho)}(\mathbf{u})$ is *weakly lower semicontinuous* in $P_{ad} \times V$, meaning that for every sequence $\{(\rho_n, \mathbf{u}_n)\} \subset P_{ad} \times V$ such that $(\rho_n, \mathbf{u}_n) \rightharpoonup (\bar{\rho}, \bar{\mathbf{u}}) \in P_{ad} \times V$ as $n \to \infty$ we have

$$J_{\alpha(\bar{\rho})}(\bar{\mathbf{u}}) \leq \liminf_{n\to\infty} J_{\alpha(\rho_n)}(\mathbf{u}_n).$$

Proof. Writing (3.1) out as

$$J_{\alpha(\rho)}(\mathbf{u}) = \frac{1}{2}\int_\Omega |\nabla \mathbf{u}|^2 + \frac{1}{2}\int_\Omega \alpha(\rho)|\mathbf{u}|^2 - \int_\Omega \mathbf{f}\cdot\mathbf{u}, \tag{3.6}$$

we will consider weak lower semicontinuity for each term individually.

As discussed previously, the first term in (3.6) is continuous and convex in $\mathbf{u}$. The third term is simply linear in and continuous in $\mathbf{u}$, so it is also convex. [Trö10, Thm 2.12] states that every continuous and convex functional in a Banach space is weakly lower semicontinuous, yielding

$$\begin{aligned} \frac{1}{2}\int_\Omega |\nabla \bar{\mathbf{u}}|^2 &\leq \liminf_{n\to\infty} \frac{1}{2}\int_\Omega |\nabla \mathbf{u}_n|^2, \\ -\int_\Omega \mathbf{f}\cdot\bar{\mathbf{u}} &\leq \liminf_{n\to\infty} -\int_\Omega \mathbf{f}\cdot\mathbf{u}_n. \end{aligned} \tag{3.7}$$

In order to show weakly lower semicontinuity for the second term in (3.6), write

$$\int_\Omega \alpha(\rho_n)|\mathbf{u}_n|^2 - \int_\Omega \alpha(\bar{\rho})|\bar{\mathbf{u}}|^2 = \int_\Omega \alpha(\rho_n)(|\mathbf{u}_n|^2 - |\bar{\mathbf{u}}|^2) + \int_\Omega (\alpha(\rho_n)-\alpha(\bar{\rho}))|\bar{\mathbf{u}}|^2. \tag{3.8}$$

The first term on the right hand-side can be estimated as

$$\begin{aligned} &\left|\int_\Omega \alpha(\rho_n)(|\mathbf{u}_n|^2 - |\bar{\mathbf{u}}|^2)\right| \leq \|\alpha(\rho_n)\|_{L^\infty(\Omega)} \int_\Omega \left|(|\mathbf{u}_n|^2 - |\bar{\mathbf{u}}|^2)\right| \\ &= C_1 \int_\Omega |(\mathbf{u}_n - \bar{\mathbf{u}})\cdot(\mathbf{u}_n + \bar{\mathbf{u}})| \leq C_1 \|\mathbf{u}_n - \bar{\mathbf{u}}\|_{L^2(\Omega)} \|\mathbf{u}_n + \bar{\mathbf{u}}\|_{L^2(\Omega)} \\ &= C_2 \|\mathbf{u}_n - \bar{\mathbf{u}}\|_{L^2(\Omega)}, \end{aligned}$$

where we have used the bound on $\|\mathbf{u}\|_V$ from Lemma 2.6 to conclude that $\|\mathbf{u}_n + \bar{\mathbf{u}}\|_{L^2(\Omega)}$ is bounded. By the compact embedding of $V \subset H^1(\Omega)$ in $L^2(\Omega)$ given by [Trö10, Thm 7.3 (Rellich)], the weak convergence $\mathbf{u}_n \rightharpoonup \bar{\mathbf{u}}$ in V implies that $\mathbf{u} \to \bar{\mathbf{u}}$ strongly in $L^2(\Omega)$. As $\|\mathbf{u}_n - \bar{\mathbf{u}}\|_{L^2(\Omega)} \to 0$, the first term on the right-hand side of (3.8) tends to 0 as $n \to \infty$.

For the second term in (3.8), since α is differentiable we have

$$\alpha'(\bar{\rho})(\rho_n - \bar{\rho}) = \lim_{t\to 0} \frac{\alpha(\bar{\rho} + t(\rho_n - \bar{\rho})) - \alpha(\bar{\rho})}{t},$$

and by using that $\bar{\rho} + t(\rho_n - \bar{\rho}) = t\rho_n + (1-t)\bar{\rho}$ in the right-hand side of the above equation, invoking the convexity of α yields

$$\alpha'(\bar{\rho})(\rho_n - \bar{\rho}) \leq \lim_{t\to 0} \frac{t\alpha(\rho_n) + (1-t)\alpha(\bar{\rho}) - \alpha(\bar{\rho})}{t} = \alpha(\rho_n) - \alpha(\bar{\rho}),$$

meaning $\alpha'(\bar{\rho})(\rho_n - \bar{\rho}) \leq \alpha(\rho_n) - \alpha(\bar{\rho})$ almost everywhere in Ω. Furthermore, since $|\bar{\mathbf{u}}|^2 \geq 0$ almost everywhere in Ω, the second term on the right-hand side in (3.8) can be estimated from below as

$$\int_\Omega (\alpha(\rho_n) - \alpha(\bar{\rho}))|\bar{\mathbf{u}}|^2 \geq \int_\Omega (\rho_n - \bar{\rho})\alpha'(\bar{\rho})|\bar{\mathbf{u}}|^2. \tag{3.9}$$

The right-hand side of the above estimate tends to zero as $\rho_n \rightharpoonup \bar{\rho}$ in $L^2(\Omega)$ provided $\alpha'(\bar{\rho})|\bar{\mathbf{u}}|^2 \in L^2(\Omega)$. In order to see this, consider the factors $\alpha'(\rho)$ and $|\mathbf{u}|^2$ separately.

First, as α is continuously differentiable in $[\rho_{\min}, 1]$, then $\alpha'(\rho(\cdot))$ attains a maximum and minimum in this interval for all $\rho \in P$, meaning $\alpha'(\rho) \in L^\infty(\Omega)$.

To see that $|\bar{\mathbf{u}}|^2 \in L^2(\Omega)$, it is necessary to use an embedding result for Sobolev spaces. [Trö10, Thm. 7.1] provides conditions for when Sobolev spaces are continuously embedded in Lebesgue spaces, and in particular for the case $\Omega \subset \mathbb{R}^2$ we have that $H^1(\Omega, \mathbb{R}^2) \hookrightarrow L^q(\Omega, \mathbb{R}^2)$ for all $1 \leq q < \infty$. Using $q = 4$, $\bar{\mathbf{u}} \in L^4(\Omega, \mathbb{R}^2)$ implies that $|\bar{\mathbf{u}}|^2 \in L^2(\Omega, \mathbb{R})$.

Finally, as $\alpha'(\rho) \in L^\infty(\Omega)$ and $|\bar{\mathbf{u}}|^2 \in L^2(\Omega)$, it follows that the product $\alpha'(\rho)|\bar{\mathbf{u}}|^2 \in L^2(\Omega)$, meaning

$$\lim_{n\to\infty} \int_\Omega (\rho_n - \bar{\rho})\alpha'(\bar{\rho})|\mathbf{u}|^2 = 0.$$

As this holds for all weakly convergent sequences $\{\rho_n\}_{n=1}^\infty$ it also holds for a minimising sequence, so (3.9) becomes

$$\frac{1}{2}\int_\Omega \alpha(\bar{\rho})|\bar{\mathbf{u}}|^2 \leq \liminf_{n\to\infty} \int_\Omega \frac{1}{2}\alpha(\rho_n)|\bar{\mathbf{u}}|^2, \tag{3.10}$$

showing that the second term on the right-hand side of (3.8) is weakly lower semicontinuous.

Finally, adding together the three inequalities in (3.7) and (3.10) yields the desired result. □

The rest of the proof follows from the above proposition. We have

$$J_{\alpha(\bar{\rho})}(\bar{\mathbf{u}}) \leq \liminf_{n\to\infty} J_{\alpha(\rho_n)}(\mathbf{u}_n) = j,$$

where the last equality comes from the fact that $\{\rho_n\}_{n=1}^\infty$ is a minimising sequence. This confirms that $(\bar{\rho}, \bar{\mathbf{u}})$ is a minimiser of (3.3).

Until now, the only time we relied on the dimension of Ω was in the proof of Proposition 3.3, where we assumed $\Omega \subset \mathbb{R}^2$ in order to conclude that $\mathbf{u} \in L^4(\Omega, \mathbb{R}^2)$. However, for the case $\Omega \subset \mathbb{R}^3$ we note that we still have from [Trö10, Thm. 7.1] that $H^1(\Omega, \mathbb{R}^3) \hookrightarrow L^q(\Omega, \mathbb{R}^3)$ for $1 < q \leq 6$, meaning that $\mathbf{u} \in L^4(\Omega, \mathbb{R}^3)$ also in this case. In other words, the proof presented in this section also holds in three dimensions.

3.3 Necessary optimality conditions

Having seen that there exists optimal controls for (3.3), we formulate the optimality conditions such optimal controls has to satisfy. To this end we will use *the formal Lagrange method* [Trö10, Sec. 2.10], so we formulate a Lagrangian of (3.3) as

$$\begin{aligned}\mathcal{L}(\rho, \mathbf{u}, p, \mathcal{P}_\mathbf{u}, \mathcal{P}_p) = {} & J_{\alpha(\rho)}(\mathbf{u}) \\ & + a_{\alpha(\rho)}(\mathbf{u}, \mathcal{P}_\mathbf{u}) + b(\mathcal{P}_\mathbf{u}, p) - (\mathbf{f}, \mathcal{P}_\mathbf{u})_{L^2(\Omega)} \\ & + b(\mathbf{u}, \mathcal{P}_p).\end{aligned} \tag{3.11}$$

Here $\mathcal{P}_\mathbf{u} \in V$ and $\mathcal{P}_p \in Q$ are the Lagrange multipliers corresponding to the state equations.

In order to proceed, we derive *the adjoint equations*, which are the differential equations $\mathcal{P}_\mathbf{u}$ and $\mathcal{P}_p$ need to satisfy. This is done by finding the stationary points of (3.11) w.r.t. to the Lagrange multipliers, that is

$$\begin{aligned} \mathcal{L}'_\mathbf{u}(\rho, \mathbf{u}, p, \mathcal{P}_\mathbf{u}, \mathcal{P}_p)\mathbf{v} = 0 \quad \forall \mathbf{v} \in V, \\ \mathcal{L}'_p(\rho, \mathbf{u}, p, \mathcal{P}_\mathbf{u}, \mathcal{P}_p)q = 0 \quad \forall q \in Q, \end{aligned} \tag{3.12}$$

in directions $\mathbf{v}$ and q. The linear terms in (3.11) are easy to differentiate, as the directional derivative of a linear mapping is just the mapping itself. I.e. in general we have $(Au)'v = Av$ for a linear mapping $u \to Au$ in a direction v. For the non-linear term $a_{\alpha(\rho)}(\mathbf{u}, \mathbf{u})$ in $J_{\alpha(\rho)}(\mathbf{u})$, it can be verified that $(a_{\alpha(\rho)}(\mathbf{u}, \mathbf{u})'_\mathbf{u})\mathbf{v} = a_{\alpha(\rho)}(\mathbf{u}, \mathbf{v})$ using the chain rule from differentiation in Banach spaces [Trö10, Thm. 2.20], along with the fact that ∇ is linear.

With this in mind, the adjoint equations (3.12) become

$$\begin{aligned} a_{\alpha(\rho)}(\mathcal{P}_\mathbf{u}, \mathbf{v}) + a_{\alpha(\rho)}(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, \mathcal{P}_p) = (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} \quad \forall \mathbf{v} \in V, \\ b(\mathbf{u}, q) = 0 \quad \forall q \in Q. \end{aligned} \tag{3.13}$$

Since $(\mathbf{u}, p)$ is the solution to the state equations (2.6), (3.13) reduces to

$$a_{\alpha(\rho)}(\mathcal{P}_\mathbf{u}, \mathbf{v}) = \mathbf{0} \quad \forall \mathbf{v} \in V, \tag{3.14}$$

with $\mathcal{P}_p = p$. By the Lax-Milgram theorem (3.14) is uniquely solvable, and it is trivial to see that $\mathcal{P}_\mathbf{u} = \mathbf{0}$ is the solution.

Having derived the forms of $\mathcal{P}_\mathbf{u}$ and $\mathcal{P}_p$, (3.11) reduces to

$$\mathcal{L}(\rho, \mathbf{u}, p) = J_{\alpha(\rho)}(\mathbf{u}) + b(\mathbf{u}, p). \tag{3.15}$$

By the formal Lagrange method, the optimality conditions of (3.3) can be stated in terms of (3.15) by

$$\begin{aligned} \mathcal{L}'_\mathbf{u}(\rho^*, \mathbf{u}^*, p^*)\mathbf{v} = 0 \quad \forall \mathbf{v} \in V, \\ \mathcal{L}'_p(\rho^*, \mathbf{u}^*, p^*)q = 0 \quad \forall q \in Q, \\ \mathcal{L}'_\rho(\rho^*, \mathbf{u}^*, p^*)(\rho - \rho^*) \geq 0 \quad \forall \rho \in P_{ad}, \end{aligned} \tag{3.16}$$

for an optimal control ρ^* with associated state $(\mathbf{u}^*, p^*)$. Here the first two equations in (3.16) are simply the state equations, while the third equations is the variational inequality, written out as

$$\mathcal{L}'_\rho(\rho^*, \mathbf{u}^*, p^*)(\rho - \rho^*) = \frac{1}{2} \int_\Omega \alpha'(\rho^*) |\mathbf{u}^*|^2 (\rho - \rho^*). \tag{3.17}$$

In treating of the optimal control problem (3.3) and corresponding optimality conditions (3.16), it is useful to introduce the notion of the *reduced objective functional*. As there exists a unique state corresponding to each control, we can

eliminate the state variables from the problem in favour for an operator $\rho \to \mathbf{u}(\rho)$. Hence it takes a control ρ as input and returns the corresponding velocity $\mathbf{u}$ from solving the state equations. From the context it should be clear that $\mathbf{u}(\rho)$ is not to be confused with the value $\mathbf{u}(\mathbf{x})$ at the point $\mathbf{x} \in \Omega$.

We define the reduced objective functional

$$f(\rho) = J_{\alpha(\rho)}(\mathbf{u}(\rho)), \tag{3.18}$$

meaning (3.3) can be written

$$\min_{\rho \in P_{ad}} f(\rho). \tag{3.19}$$

Although yielding a slight simplification of the optimisation problem, the main motivation for introducing f is to be able to derive its gradient. The formal Lagrange method enables us to formulate the gradient $f'(\rho) \in L^2(\Omega)'$ in terms of the Lagrangian $\mathcal{L}$, more precisely by the simple rule $f'(\rho) = \mathcal{L}'_\rho(\rho, \mathbf{u}, p)$ [Trö10, p. 88]. Furthermore, by the Riesz representation theorem [BS07, Thm. 2.4.2] we can identify $f'(\rho)$ by an element in $L^2(\Omega)$, and having calculated $\mathcal{L}'_\rho$ in (3.17) we see that it takes the form

$$f'(\rho) = \frac{1}{2}\alpha'(\rho)|\mathbf{u}(\rho)|^2. \tag{3.20}$$

Finally, using (3.20) and writing out the Lagrangian, (3.16) can be written more explicitly as

$$\begin{aligned} a_{\alpha(\rho^*)}(\mathbf{u}^*, \mathbf{v}) + b(\mathbf{v}, p^*) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} && \forall \mathbf{v} \in V, \\ b(\mathbf{u}^*, q) &= 0 && \forall q \in Q, \\ (f'(\rho^*), \rho - \rho^*)_{L^2(\Omega)} &\geq 0 && \forall \rho \in P_{ad}. \end{aligned} \tag{3.21}$$

3.4 The optimisation algorithm

We will now formulate a strategy for finding optimal controls ρ^*. Several algorithms for solving (3.3) exist, like the separable sequential quadratic programming (SQP) approach, the method of moving asymptotes (MMA) used in [BP03, Sec. 3.4], the projected gradient method [Trö10, Sec. 3.7.1] or the optimality criteria method (OC). Here we will use the *optimality criteria method*, which is presented in the following subsection.

3.4.1 The optimality criteria method

As the name suggests, the optimality criteria method [BS11] is based on the optimality conditions of (3.19). More precisely, we create a fixed-point scheme based on (3.21), and to this end it is useful to enforce the volume constraint in P_{ad} explicitly using a Lagrange multiplier. (3.19) is then formulated

$$\min_{\rho \in P} f(\rho) \quad \text{subject to} \quad \int_\Omega \rho \leq \gamma|\Omega|, \tag{3.22}$$

and the optimality conditions of (3.22) become

$$\begin{aligned}(f'(\rho^*)+\lambda,\rho-\rho^*)_{L^2(\Omega)} &\geq 0 \quad \forall \rho \in P,\\ \int_\Omega(\gamma-\rho^*) &\geq 0,\\ \lambda &\geq 0,\\ \lambda\int_\Omega(\gamma-\rho^*) &= 0,\end{aligned} \tag{3.23}$$

for a minimum $(\mathbf{u}^*, p^*, \lambda)$. $\lambda \in \mathbb{R}$ is the Lagrange multiplier corresponding to volume constraint, whose non-negativity follows from the standard KKT conditions, stated in [NW06, Thm. 12.1] and generalised for optimal control in [Trö10, Thm. 6.1].

First, we give an alternative formulation of the variational inequality in (3.23). Assuming ρ^* is an optimal control, split Ω into the three domains $\Omega_{\rho_{\min}} = \{\mathbf{x} \in \Omega : \rho^*(\mathbf{x}) = \rho_{\min}\}$, $\Omega_{(\rho_{\min},1)} = \{\mathbf{x} \in \Omega : \rho_{\min} < \rho^*(\mathbf{x}) < 1\}$ and $\Omega_1 = \{\mathbf{x} \in \Omega : \rho^*(\mathbf{x}) = 1\}$. Choosing $\rho \in P$ such that $\rho(\mathbf{x}) = \rho^*(\mathbf{x})$ for $\mathbf{x} \in \Omega_{\rho_{\min}} \cup \Omega_1$, and either identically $\rho_{\min}$ or 1 in $\Omega_{(\rho_{\min},1)}$, then for the first of (3.23) to hold we need to have $f'(\rho^*) + \lambda = 0$ for $\mathbf{x} \in \Omega_{(\rho_{\min},1)}$. Similarly, by choosing $\rho \in P$ such that $\rho(\mathbf{x}) = \rho^*(\mathbf{x})$ for $\mathbf{x} \in \Omega_{(\rho_{\min},1)} \cup \Omega_1$, varying $\rho(\mathbf{x})$ for $\mathbf{x} \in \Omega_{\rho_{\min}}$, we get that $f'(\rho^*) + \lambda \geq 0$ for $\mathbf{x} \in \Omega_{\rho_{\min}}$. Equivalently we arrive at $f'(\rho^*) + \lambda \leq 0$ for $\mathbf{x} \in \Omega_1$. To summarise, at a minimum ρ^* we have

$$\begin{aligned}f'(\rho^*) &= -\lambda \text{ for } \rho_{\min} < \rho^*(\mathbf{x}) < 1,\\ f'(\rho^*) &\geq -\lambda \text{ for } \rho^*(\mathbf{x}) = \rho_{\min},\\ f'(\rho^*) &\leq -\lambda \text{ for } \rho^*(\mathbf{x}) = 1.\end{aligned} \tag{3.24}$$

To get the actual scheme, instead of using ρ^*, we instead consider an iterate ρ_i for $i = 0, 1, \ldots$, and write the first equation of (3.24) as

$$\rho_i = -\frac{f'(\rho_i)}{\lambda}\rho_i.$$

Defining the left-hand side to be the next iterate ρ_{i+1}, we have

$$\rho_{i+1} = \Pi_P\left(-\frac{f'(\rho_i)}{\lambda}\rho_i\right), \tag{3.25}$$

where $\Pi_P : L^2(\Omega) \to P$ is the projection onto P. In the case of box-constraints, this projection can easily be expressed

$$\Pi_P(\rho) = \min\{1, \max\{\rho_{\min}, \rho\}\},$$

see [Trö10, p. 71]. To ensure that $\rho_{i+1} \in P_{ad}$, λ is defined by

$$\int_\Omega \Pi_P\left(-\frac{f'(\rho_i)}{\lambda}\rho_i\right) = \gamma|\Omega|.$$

Using equality in the above expression, we restrict the OC method to only generate iterates where the volume constraint is binding. This means that if there is any hope for the OC method to converge towards a solution of (3.22), it necessary that there actually exists an optimal control where the volume constraint is in fact binding. This assumption is justified in the following proposition.

Proposition 3.4. There exists a solution $\rho^* \in P_{ad}$ to the optimal control problem (3.22) such that

$$\int_\Omega \rho^* = \gamma|\Omega|.$$

Proof. Let $\rho^* \in P_{ad}$ be a *global* solution to (3.22) with $\mathbf{u}(\rho^*) = \mathbf{u}^*$, and assume that

$$\int_\Omega \rho^* < \gamma|\Omega|.$$

From the last of the KKT conditions stated in (3.23), it follows that $\lambda = 0$. The third of (3.23) becomes

$$\frac{1}{2}\int_\Omega \alpha'(\rho^*)|\mathbf{u}^*|(\rho - \rho^*) \geq 0 \quad \forall \rho \in P, \tag{3.26}$$

using the definition (3.20) of $f'(\rho)$. Splitting Ω into the three domains $\Omega_{\rho_{\min}} = \{\mathbf{x} \in \Omega : \rho^*(\mathbf{x}) = \rho_{\min}\}$, $\Omega_{(\rho_{\min},1)} = \{\mathbf{x} \in \Omega : \rho_{\min} < \rho^*(\mathbf{x}) < 1\}$ and $\Omega_1 = \{\mathbf{x} \in \Omega : \rho^*(\mathbf{x}) = 1\}$ as before, we consider the three cases separately.

On $\Omega_{\rho_{\min}}$, $\alpha'(\rho) < 0$ for values of $\rho \in [\rho_{\min}, 1]$ as α is strictly monotonically decreasing, meaning (3.26) can only hold if $\mathbf{u}^* = \mathbf{0}$ on this part of the domain. Considering $\Omega_{(\rho_{\min},1)}$, setting first $\rho(\mathbf{x}) = 0$ and then $\rho(\mathbf{x}) = 1$ for almost all $\mathbf{x}$ in $\Omega_{(\rho_{\min},1)}$, the only way the variational inequality (3.26) can hold in both cases is if $\mathbf{u}^* = \mathbf{0}$ also here. Lastly, for Ω_1 (3.26) is trivially satisfied.

Define a new control $\hat{\rho} \in P_{ad}$ with $\hat{\rho}(\mathbf{x}) = \rho^*(\mathbf{x})$ for almost all $\mathbf{x} \in \Omega_1$ and $\hat{\rho}(\mathbf{x}) \geq \rho^*(\mathbf{x})$ for almost all $\mathbf{x} \in \Omega_{\rho_{\min}} \cup \Omega_{(\rho_{\min},1)}$ such that

$$\int_\Omega \hat{\rho} = \gamma|\Omega|.$$

From the above discussion we then have that $\mathbf{u}^* = \mathbf{0}$ when $\hat{\rho} \neq \rho^*$, and by the definition of $a_{\alpha(\rho)}(\cdot,\cdot)$ we see that the only term containing ρ also contains $\mathbf{u}$, meaning $\mathbf{u}(\hat{\rho}) = \mathbf{u}^*$. Furthermore, $a_{\alpha(\hat{\rho})}(\mathbf{u}^*,\cdot) = a_{\alpha(\rho^*)}(\mathbf{u}^*,\cdot)$, such that $f(\hat{\rho}) = J_{\alpha(\hat{\rho})}(\mathbf{u}^*) = J_{\alpha(\rho^*)}(\mathbf{u}^*)$. As $f(\hat{\rho}) = f(\rho^*)$, $\hat{\rho} \in P_{ad}$ is another global solution of (3.22). □

Numerical experience has shown that the (3.25) is not feasible in its current form, so we are going to make two changes to the scheme. In order for the method to converge, it is necessary to enforce move limits such that the next iterate does not change too much relative to the current iterate. These limits are introduced as pointwise upper and lower bounds, and if we let $\zeta > 0$ be the fraction ρ_{i+1} is allowed to differ from ρ_i, we can express these move limits as $(1-\zeta)\rho_i \leq \rho_{i+1} \leq (1+\zeta)\rho_i$ almost everywhere in Ω.

The second change we want to make is to enforce some sort of damping on $-f'(\rho_i)/\lambda$. It is generally preferable that this factor should be close to 1, as this indicates that we have converged. Hence we want values less than 1 to be amplified, and values greater than 1 to be decreased. By taking $(-f'(\rho_i)/\lambda)^\xi$ for some $\xi \in (0,1)$ we get exactly this effect, and a smaller value yields a greater damping around 1.

Finally, these two changes can be combined into a new operator $Z_\lambda : L^2(\Omega) \to L^2(\Omega)$ defined by

$$Z_\lambda(\rho_i) = \min\left\{1+\zeta, \max\left\{1-\zeta, \left(-\frac{f'(\rho_i)}{\lambda}\right)^\xi\right\}\right\}\rho_i,$$

such that the fixed-point scheme reads

$$\rho_{i+1} = \Pi_P Z_\lambda(\rho_i) \quad \text{with } \lambda \text{ such that} \quad \int_\Omega \Pi_P Z_\lambda(\rho_i) = \gamma|\Omega|. \tag{3.27}$$

How accumulation points of (3.27) relates to the optimisation problem (3.22), provided they exist, is addressed in the following proposition.

Proposition 3.5. Let $\bar\rho \in P_{ad}$ be an accumulation point of (3.27), that is

$$\bar\rho = \Pi_P Z_\lambda(\bar\rho). \tag{3.28}$$

Then $\bar\rho$ satisfies the optimality conditions (3.23).

Proof. First, note that the move limits in Z_λ are nowhere binding in an accumulation point. To see this, assume the contrary. When the move limits are binding, then $Z_\lambda(\bar\rho) = (1 \pm \zeta)\bar\rho$, where the positive and negative case corresponds to the upper limit and lower limit, respectively. For the case where the box constraints in Π_P are binding, the move limits cannot be, as they in this case are a distance $\zeta > 0$ outside the box constraints. If the box constraints in Π_P are not binding, then $\Pi_P Z_\lambda(\bar\rho) = Z_\lambda(\bar\rho)$. Hence $\bar\rho = (1 \pm \zeta)\bar\rho$, which is a contradiction. This means that in an accumulation point we have

$$\bar\rho = \Pi_P\left(\left(-\frac{f'(\bar\rho)}{\lambda}\right)^\xi \bar\rho\right). \tag{3.29}$$

Now, consider the case where $\rho_{\min} \le (-f'(\rho_i)/\lambda)^\xi < 1$, that is the box constraints are not binding. (3.29) yields $\bar\rho = (-f'(\bar\rho)/\lambda)^\xi \bar\rho$, and by dividing by $\bar\rho$ we see that the equation $(-f'(\bar\rho)/\lambda)^\xi = 1$ is only satisfied when $-f'(\bar\rho)/\lambda = 1$, retrieving the first of (3.24).

For the case $(-f'(\bar\rho)/\lambda)^\xi \bar\rho \ge 1$ it follows from (3.28) that $\bar\rho = 1$. Hence we have $(-f'(\bar\rho)/\lambda)^\xi \ge 1$. This is equivalent to $-f'(\bar\rho)/\lambda \ge 1$, retrieving the third of (3.24). For $(-f'(\bar\rho)/\lambda)^\xi \bar\rho \le \rho_{\min}$, the argumentation is completely equivalent. □

3.4.2 The stopping criterion

The last thing we will do in this section is to present a criterion for stopping the iterations (3.27), which we will need when implementing the optimality criteria method numerically in the coming chapter.

The simplest example of a stopping criterion is to check how much the iterates between to consecutive iterations differ, that is for an iteration i we take $\|\rho_i - \rho_{i-1}\|_{L^2(\Omega)}$. Although this approach can often work fairly well in practice, there is always the possibility that for some reason the fixed-point scheme (3.27) takes a small step in some iteration despite not having reached an accumulation point, meaning the algorithm will terminate prematurely.

A more robust stopping criterion is to use the direction of steepest descent $-f'(\rho_i)$, and check how much moving in this direction will differ from the current iterate ρ_i. This point is given by $\rho_i + (-f'(\rho_i)) = \rho_i - f'(\rho_i)$, but as it might not be admissible, it is necessary to project back onto P_{ad}.

To this end, write the L^2 projection $\Pi_{P_{ad}} : L^2(\Omega) \to P_{ad}$ in a point $\hat{\rho} \in L^2(\Omega)$ onto P_{ad} as the solution to the optimisation problem

$$\min_{\rho \in P} \frac{1}{2}\|\rho - \hat{\rho}\|^2_{L^2(\Omega)} \quad \text{subject to} \quad \int_\Omega \rho \leq \gamma|\Omega|.$$

Existence and uniqueness of the above problem is seen by convexity, coercivity and continuity of $\|\cdot\|^2_{L^2(\Omega)}$, along with convexity of P_{ad}. Furthermore, the solution is $\rho^* = \hat{\rho} - \lambda$, where λ again is the Lagrange multiplier for the volume constraint. Assuming $\hat{\rho} \notin P_{ad}$, the projection $\Pi_{P_{ad}} : L^2(\Omega) \to P_{ad}$ is given by

$$\Pi_{P_{ad}}(\hat{\rho}) = \Pi_P(\hat{\rho} - \lambda), \quad \text{with } \lambda \text{ such that} \quad \int_\Omega \Pi_P(\hat{\rho} - \lambda) = \gamma|\Omega|.$$

In the case $\hat{\rho} \in P_{ad}$, obviously $\Pi_{P_{ad}}(\hat{\rho}) = \hat{\rho}$, that is constraint on λ in the (3.2) is not enforced.

Having defined $\Pi_{P_{ad}}$, we can define the stopping criterion as

$$\|\rho_i - \Pi_{P_{ad}}(\rho_i - f'(\rho_i))\|_{L^2(\Omega)} < \epsilon$$

for some sufficiently small $\epsilon > 0$, and the projection onto P_{ad} is illustrated in Figure 3.1.

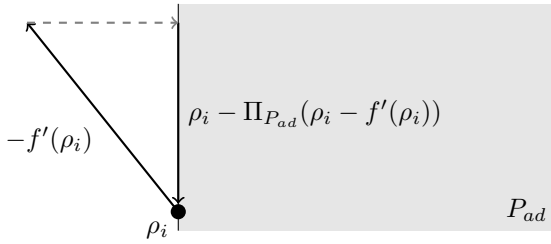


Figure 3.1: Illustration of the stopping criterion for an iteration i. The grey, dashed line denotes the projection of the point $\rho_i - f'(\rho_i)$ onto P_{ad}.

Chapter 4

Numerical implementation

In the previous chapter an optimal control problem for minimising the energy loss in Stokes flow was introduced, and the optimality criteria method was outlined as a strategy for finding optimal controls for the optimisation problem.

However several questions regarding actually implementing the method in practice were left unanswered, like how the state equations should be solved and how to estimate the Lagrange multipliers appearing in the OC method. In this chapter these details will be addressed, and we also introduce several numerical examples.

4.1 Finite element discretisation details

The finite element method (FEM) is the most commonly used discretisation technique in engineering and analysis [Qua14], and has proven well-suited for a wide variety of differential equations, including fluid mechanics equations like the generalised Stokes equations discussed in this report.

Due to this, we will also use FEM for the numerical implementation considered, and more precisely we will use the *FEniCS* platform, a collection of programs for automatic solving of partial differential equations [LMW+12]. FEniCS is supported with both Python and C++, where we have chosen to use the former for our implementation. All parts of the code has been made fully MPI compliant [Nie16], and in fact all the computations were performed in parallel on a dual-core processor.

4.1.1 Galerkin formulation of the state equations

In order to discretise the (2.6), we define $\mathcal{T}_h$ to be a discretisation of Ω, with discretisation parameter h. At the time of writing FEniCS only supports triangular elements, so we assume that $\mathcal{T}_h$ is a triangular. As such, h will denote the largest diameter of the triangles. Lastly, if we let V_h and Q_h be the discretised spaces for the velocity space $V = H_0^1(\Omega)$ and pressure space $Q = L_0^2(\Omega)$, respectively, then

the Galerkin approximation of (2.6) yields: Find $(\mathbf{u}_h, p_h) \in V_h \times Q_h$ such that

$$\begin{aligned} a_\alpha(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) &= (\mathbf{f}, \mathbf{v}_h)_{L^2(\Omega)} \quad \forall \mathbf{v}_h \in V_h, \\ b(\mathbf{u}_h, q_h) &= 0 \qquad\qquad\qquad\quad \forall q_h \in Q_h. \end{aligned} \tag{4.1}$$

The exact definition of the two families of finite-dimensional spaces V_h and Q_h will be introduced later in the next section.

For now, assume that V_h has basis $\boldsymbol{\varphi}_i$, $i = 1, \ldots, n$ and Q_h has basis ϕ_j, $j = 1, \ldots, m$, that is $V_h = \operatorname{span}\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_n\}$ and $Q_h = \operatorname{span}\{\phi_1, \ldots, \phi_m\}$. This means that generic functions $\mathbf{v}_h \in V_h$ and $q_h \in Q_h$ can be expressed as

$$\mathbf{v}_h(\mathbf{x}) = \sum_{j=1}^{n} v_j \boldsymbol{\varphi}_j(\mathbf{x}), \quad q_h(\mathbf{x}) = \sum_{l=1}^{m} q_l \phi_l(\mathbf{x}), \tag{4.2}$$

that is $\mathbf{v}_h$ and q_h is represented as a linear combination of the basis functions with coefficients v_i and q_j, respectively. Similarly, writing the discrete solution $\mathbf{u}_h$ and p_h as

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{n} u_i \boldsymbol{\varphi}_i(\mathbf{x}), \quad p_h(\mathbf{x}) = \sum_{k=1}^{m} p_k \phi_k(\mathbf{x}), \tag{4.3}$$

inserting (4.2) and (4.3) into (4.1) yields the linear system

$$\sum_{j=1}^{m} \left(u_i a_\alpha(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j) + p_k b(\boldsymbol{\varphi}_j, \phi_k)\right) = (\mathbf{f}, \boldsymbol{\varphi}_j)_{L^2(\Omega)}$$

$$\sum_{l=1}^{m} u_i b(\boldsymbol{\varphi}_i, q_l) = 0$$

with $i = 1, \ldots, n$ and $k = 1, \ldots, m$. Furthermore, organising $\mathbf{U} = (u_1, \ldots, u_n)^T$ and $\mathbf{P} = (p_1, \ldots, p_m)^T$, the above system can be written in matrix form as

$$\begin{aligned} A_\alpha \mathbf{U} + B^T \mathbf{P} &= \mathbf{F}, \\ B\mathbf{U} &= \mathbf{0}. \end{aligned} \tag{4.4}$$

Here $A_\alpha \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{m \times n}$ and $\mathbf{F} \in \mathbb{R}^n$, and are defined as

$$A_\alpha = [a_{ij}] = [a_\alpha(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)], \quad B = [b_{kj}] = [b(\boldsymbol{\varphi}_j, \phi_k)], \quad \mathbf{F} = [f_j] = [(\mathbf{f}, \boldsymbol{\varphi}_j)_{L^2(\Omega)}]. \tag{4.5}$$

Lastly, it is trivial to see that the two linear systems in (4.4) can be written as a single system with the *system matrix* $S \in \mathbb{R}^{(m+n)\times(m+n)}$, defined as

$$S = \begin{bmatrix} A_\alpha & B^T \\ B & 0 \end{bmatrix}.$$

We inspect the properties of the matrix S in the following subsection.

4.1.2 Properties of the system matrix

Proposition 4.1. The matrix S is block symmetric and indefinite.

Proof. Block symmetry of S is easily seen by the fact that A is symmetric, which can be seen from (4.5) as $a_\alpha(\cdot,\cdot)$ is symmetric.

Showing that S is indefinite amounts to finding two vectors such that the quadratic form associated with S has different signs. In order to do this, we will first show that the matrix A_α is positive definite. Let $\mathbf{V} \in \mathbb{R}^n$ be the column vector with entries corresponding to the coefficients v_i in (4.2), yielding

$$\begin{aligned}\mathbf{V}^T A_\alpha \mathbf{V} &= \sum_{i=1}^{n}\sum_{j=1}^{n} v_i a_{ij} v_j = \sum_{i=1}^{n}\sum_{j=1}^{n} v_i a_\alpha(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j) v_j \\ &= a_\alpha\left(\sum_{i=1}^{n} v_i\boldsymbol{\varphi}_i, \sum_{j=1}^{n} v_j\boldsymbol{\varphi}_j\right) = a_\alpha(\mathbf{v}_h, \mathbf{v}_h) \geq 0.\end{aligned}$$

As $\alpha \geq 0$ in $a_\alpha(\cdot,\cdot)$, it is obvious that $a_\alpha(\mathbf{v}_h, \mathbf{v}_h) = 0$ only when $\mathbf{v}_h = 0$, meaning $\mathbf{V} = \mathbf{0}$. This shows that A_α is positive definite.

Similarly as for $\mathbf{V}$, let $\mathbf{Q} \in \mathbb{R}^m$ be a column vector with elements q_i from (4.2), such that the quadratic form associated with S reads

$$\begin{bmatrix}\mathbf{V}^T & \mathbf{Q}^T\end{bmatrix}\begin{bmatrix}A_\alpha & B^T \\ B & 0\end{bmatrix}\begin{bmatrix}\mathbf{V} \\ \mathbf{Q}\end{bmatrix} = \mathbf{V}^T A_\alpha \mathbf{V} + 2\mathbf{V}^T B^T \mathbf{Q}. \tag{4.6}$$

Choosing $\mathbf{V} \neq \mathbf{0}$ and $\mathbf{Q} = \mathbf{0}$, by the positive definiteness of A_α it is easy to see that the right-hand side of (4.6) is greater than zero.

To find a vector such that (4.6) is less than zero, choose $\mathbf{Q} \neq \mathbf{0}$ and note that the right-hand side of (4.6) can be written $\mathbf{V}^T(A_\alpha \mathbf{V} + 2B^T\mathbf{Q})$. If we now define $A\mathbf{V} + 2B^T\mathbf{Q} = -\mathbf{V}$, solving for $\mathbf{V}$ yields $\mathbf{V} = -2(I + A_\alpha)^{-1}B^T\mathbf{Q}$, where I is the identity matrix. This is possible as both I and A_α are symmetric and positive definite, meaning $I + A_\alpha$ is also symmetric and positive definite, hence invertible. The right-hand side of (4.6) is now

$$\mathbf{V}^T(A_\alpha \mathbf{V} + 2B^T\mathbf{Q}) = \mathbf{V}^T(-\mathbf{V}) = -\|\mathbf{V}\|_2^2 < 0$$

for $\mathbf{V} \neq \mathbf{0}$, showing that S is indefinite. □

Having seen that the matrix S is indefinite, we can not guarantee that the linear system (4.4) is uniquely solvable. We need to enforce another condition for this to be the case, which is discussed in the following proposition.

Proposition 4.2. S is non-singular if and only if $b : V_h \times Q_h \to \mathbb{R}$ satisfies the inf-sup condition (2.8).

Proof. From (2.8) it is clear that the inf-sup condition is violated if and only if there exists a non-zero function $p_h^* \in Q_h$ such that

$$b(\mathbf{v}_h, p_h^*) = 0 \quad \forall \mathbf{v}_h \in V_h.$$

We let $\mathbf{V}$ and $\mathbf{P}^*$ denote the column vectors whose entries are the coefficients when $\mathbf{v}_h$ and p_h^* are represented in terms of their bases $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_n$ and $\phi_1, \ldots, \phi_m$, respectively. The above equation is written

$$B^T \mathbf{P}^* = \mathbf{0},$$

meaning that the inf-sup condition is violated iff the kernel of B^T is non-trivial. In other words, the inf-sup condition holds iff

$$\ker B^T = \{\, \mathbf{0} \,\}. \tag{4.7}$$

If we now can prove that (4.7) holds iff S is non-singular, we are done. As A_α is invertible, we can write the first equation of (4.4) as

$$\mathbf{U} = A_\alpha^{-1}(\mathbf{F} - B^T \mathbf{P})$$

which inserted into the second equation and rearranging yields

$$R\mathbf{P} = BA_\alpha^{-1}\mathbf{F}, \quad \text{where} \quad R = BA_\alpha^{-1}B^T.$$

$\mathbf{P}$, and by extension, $\mathbf{U}$, are uniquely determined provided R is invertible, so proving the proposition amounts to showing that R is invertible iff $b : V_h \times Q_h \to \mathbb{R}$ satisfies 2.8.

To this end, assume first that $\ker B^T$ is non-trivial, and consider the equation

$$R\mathbf{Q} = BA_\alpha^{-1}B^T\mathbf{Q} = \mathbf{0}$$

for some $\mathbf{Q} \in \mathbb{R}^m$. In this case any $\mathbf{Q} \in \ker B^T$ would solve the equation, and since $\ker B^T$ is non-trivial this violates the injectivity property, a necessary condition for invertibility.

On the other hand, if $\ker B^T$ is trivial, then B^T is injective. Additionally, since A_α is positive definite then so is A_α^{-1}, and we have

$$\mathbf{Q}^T R\mathbf{Q} = \mathbf{Q}^T BA_\alpha^{-1}B^T\mathbf{Q} = (B^T\mathbf{Q})^T A_\alpha^{-1}B^T\mathbf{Q} \geq 0,$$

with equality only when $B^T\mathbf{Q} = \mathbf{0}$. Since $\ker B^T = \{\, \mathbf{0} \,\}$, this means that $\mathbf{Q}^T R\mathbf{Q} = 0$ only when $\mathbf{Q} = \mathbf{0}$, showing that R is positive definite, hence non-singular. □

To get a better understanding of the above proposition, let us examine the case where a pair of finite dimensional function spaces V_h and Q_h are chosen such that $b : V_h \times Q_h \to \mathbb{R}$ violates the inf-sup condition. As we saw in the proof of Proposition 4.2, this means that there exists a $p_h^* \in Q_h$ such that $b(\mathbf{v}_h, p_h^*) = 0$ for all $\mathbf{v}_h \in V_h$. In this case, if $(\mathbf{u}_h, p_h)$ is a solution to the Galerkin problem (4.1), then so is $(\mathbf{u}_h, p_h + p_h^*)$, meaning that the pressure is not uniquely determined, giving rise to numerical instabilities in the form of S being singular. Because of this, we refer to pairs of spaces satisfying inf-sup as *stable*, while spaces that do not, to be *unstable*.

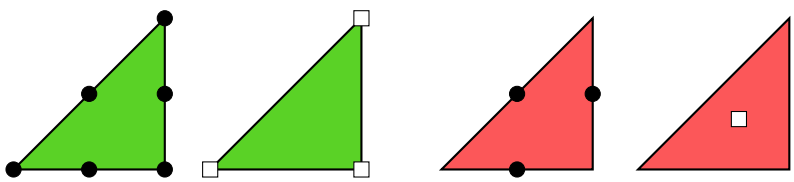
4.2 Choice of elements

The spaces V_h and Q_h are in the finite element method decided by the choice of element types used for each space, and in light of previous section we present two different pairs of elements which we will use for comparison in our numerical implementation.

4.2.1 Taylor-Hood

The Taylor-Hood (TH) elements are a family of elements where both V_h and Q_h are continuous piecewise polynomials. In the generalised Stokes equations the pressure p is of order one and the velocity $\mathbf{u}$ of order two in terms of differentiability, so to that end it seems reasonable to let V_h be of a higher order than Q_h. More precisely, the TH elements are commonly written as the pair $\mathbb{P}_k - \mathbb{P}_{k-1}$, $k \geq 2$, denoting that the basis functions of V_h should be of one order higher than those of Q_h. Note that in the case $k = 1$, corresponding to piecewise linear and continuous velocity and piecewise constant pressure, is not part of the definition, and in fact it can be shown that this pair of elements are unstable in terms of the inf-sup condition [Qua14, p. 449]. For $k \geq 2$ however, the inf-sup condition holds, and the TH elements have been proven to be both stable and convergent as the discretisation parameter h tends to zero [BF91].

In our numerical implementation we have chosen to use the smallest representative of TH, that is the pair $\mathbb{P}_2 - \mathbb{P}_1$, corresponding to piecewise continuous quadratic basis functions in V_h and piecewise continuous linear basis functions for Q_h. The TH elements are illustrated in Figure 4.1a.



(a) Taylor-Hood **(b)** Crouzeix–Raviart

Figure 4.1: The two pairs of elements. The black circles and white squares denotes the degrees of freedom for the velocity and pressure, respectively.

4.2.2 Crouzeix–Raviart

The second pair of elements we are going to consider are the *Crouzeix–Raviart* (CR) elements, which were first introduced in [CR73] for solving the Stokes equations. These elements uses linear Lagrange elements, but as opposed to TH they are not continuous on the facets. Instead they are only required to be continuous at the midpoint of the facet between two neighbouring elements. Because of this discontinuity, the CR elements are not weakly differentiable in $L^2(\Omega)$, meaning

these functions are not in $H^1(\Omega)$. We say that CR elements are *not* H^1*-conforming*. Although non-conforming, the CR elements are convergent.

As the bilinear forms $a_\alpha(\cdot,\cdot)$ and $b(\cdot,\cdot)$ contains a weak gradient and a weak divergence operator, respectively, we can not evaluate $a_\alpha(\mathbf{u}_h, \mathbf{v}_h)$ and $b(\mathbf{v}_h, \cdot)$ for two functions $\mathbf{u}_h, \mathbf{v}_h \in V_h \not\subset H^1(\Omega)$. Formally, for the previous section to be meaningful it would be necessary define a_α and b element-wise. That is, we would need to replace the weak gradient operator ∇ with ∇_h, the *piecewise* weak gradient operator, defined by evaluating ∇ element-wise on $\mathcal{T}_h$. However, for the sake of brevity we simply assume that the derivatives are understood element-wise for the CR elements.

Furthermore, for the pressure space Q_h we use piecewise constant polynomials, meaning this also consists of discontinuous functions. However as p and q are not weakly differentiated in (2.6), we have $Q_h \subset Q$. The elements are illustrated in Figure 4.1b.

Formally "Crouzeix–Raviart elements" refer to the velocity elements only, as we here always use piecewise constant functions for the pressure elements along with the CR elements, we refer to CR as the pair V_h and Q_h.

4.3 A posteriori residual estimates

A crucial part of solving differential equations numerically is the question of how accurate a computed solution is. For instance, say we have computed a solution $(\mathbf{U}, \mathbf{P})$ for the algebraic system of equations (4.4), it would be natural to define the residuals for this linear system as

$$\begin{aligned}\tilde{\mathbf{R}}^{mo} &= \mathbf{F} - \left(A_\alpha \mathbf{U} + B^T \mathbf{P}\right),\\ \tilde{\mathbf{R}}^{ma} &= -B\mathbf{U}.\end{aligned}$$

Here the indices *mo* and *ma* are short for *momentum* and *mass*, as they correspond to the momentum and the mass equation in (2.6), respectively. Even though we would have solved (4.4) accurately such that $\tilde{\mathbf{R}}^{mo} = \mathbf{0}$ and $\tilde{\mathbf{R}}^{ma} = \mathbf{0}$ in the above equations, this would yield no information about how well the functions $(\mathbf{u}_h, p_h)$ of (4.1) resembles the true solution $(\mathbf{u}, p)$ of (2.6), which are the equations we want to solve in the first place.

To this end, let us define the residuals of (2.6) as the linear functionals $R^{mo} \in V'$ and $R^{ma} \in Q'$ given by

$$\begin{aligned}R^{mo}_{\mathbf{u},p}(\mathbf{v}) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} - a_\alpha(\mathbf{u}, \mathbf{v}) - b(\mathbf{v}, p),\\ R^{ma}_{\mathbf{u}}(q) &= -b(\mathbf{u}, q),\end{aligned} \tag{4.8}$$

for a provided pair of functions $(\mathbf{u}, p) \in V \times Q$. The dual norms of R^{mo} and R^{ma} are defined

$$\begin{aligned}\left\|R^{mo}_{\mathbf{u},p}\right\|_{V'} &= \sup\left\{\, \left|R^{mo}_{\mathbf{u},p}(\mathbf{v})\right| \,:\, \mathbf{v} \in V,\, \|\mathbf{v}\|_V \leq 1 \,\right\},\\ \|R^{ma}_{\mathbf{u}}\|_{Q'} &= \sup\left\{\, |R^{ma}_{\mathbf{u}}(q)| \,:\, q \in Q,\, \|q\|_Q \leq 1 \,\right\},\end{aligned}$$

whose value can bee seen as how well $(\mathbf{u}, p)$ solves (2.6), and with $\|R^{mo}\|_{V'} = \|R^{ma}\|_{Q'} = 0$ iff $(\mathbf{u}, p)$ is the solution. Unfortunately, these dual norms are difficult to compute directly, so in practice we need some other way of estimating them. Recalling that V and Q are Hilbert spaces, invoking the Riesz representation theorem yields

$$\begin{aligned} (\mathbf{r}^{mo}, \mathbf{v})_V &= R^{mo}_{\mathbf{u},p}(\mathbf{v}) \quad \forall \mathbf{v} \in V, \\ (r^{ma}, q)_Q &= R^{ma}_{\mathbf{u}}(q) \quad \forall q \in Q, \end{aligned} \tag{4.9}$$

for some $\mathbf{r}^{mo} \in V$ and $r^{ma} \in Q$. Now, using that

$$\begin{aligned} \|\mathbf{r}^{mo}\|_V &= \left\|R^{mo}_{\mathbf{u},p}\right\|_{V'}, \\ \|r^{ma}\|_Q &= \|R^{ma}_{\mathbf{u}}\|_{Q'}, \end{aligned}$$

we get estimates for $\left\|R^{mo}_{\mathbf{u},p}\right\|_{V'}$ and $\|R^{ma}_{\mathbf{u}}\|_{Q'}$ by solving (4.9), which we see are just linear variational problem with unknowns $\mathbf{r}^{mo}$ and r^{ma}. Furthermore, by the Lax-Milgram theorem it is trivial to see that both variational problems are uniquely solvable. Note also that the two equations are decoupled, as opposed to the state equations, meaning they can be solved separately.

Having introduced the concept of residuals, let us adapt (4.9) to the finite element method. Again, assume we have computed the solution $(\mathbf{u}_h, p_h)$ of (4.1) on some triangulation $\mathcal{T}_h$ of Ω with maximum triangle diameter h. Motivated by (4.8), we want to estimate how well $(\mathbf{u}_h, p_h)$ solves (4.1), but at the same time we want to capture some the discretisation error resulting from restricting the test and trial space from the continuous space $V \times Q$ to the finite dimensional space $V_h \times Q_h$. In order to do the latter, it is necessary to evaluate $(\mathbf{u}_h, p_h)$ on a space with higher accuracy than $V_h \times Q_h$. In the finite element method there are mainly two ways of achieving this, either by h-enrichment, where h in $\mathcal{T}_h$ is decreased, or by p-enrichment, where the order of the basis functions spanning $V_h \times Q_h$ is increased.

Here we use the former approach, that is $\mathcal{T}_h$ will be uniformly refined by dividing each triangular cell into four. This results in a finer mesh $\mathcal{T}_{h/2}$, as illustrated in Figure 4.2. Next, define the corresponding trial and test space $V_{h/2} \times Q_{h/2}$ on

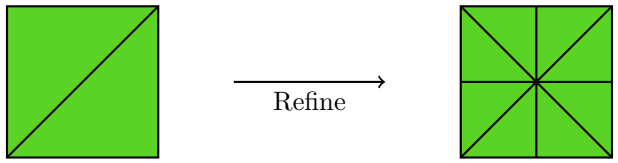


Figure 4.2: Refinement of two triangular cells.

$\mathcal{T}_{h/2}$, consisting of the same type of basis functions as for $V_h \times Q_h$ on $\mathcal{T}_h$. We are now ready to discretise (4.9), which we will do on $\mathcal{T}_{h/2}$. Letting $\mathbf{r}^{mo}_{h/2} \in V_{h/2}$ and $r^{ma}_{h/2} \in Q_{h/2}$ be trial functions and $\mathbf{v}_{h/2} \in V_{h/2}$ and $q_{h/2} \in Q_{h/2}$ be test functions,

the Galerkin formulation of (4.9) becomes

$$\begin{aligned}(\mathbf{r}^{mo}_{h/2}, \mathbf{v}_{h/2})_V &= R^{mo}_{\mathbf{u}_h, p_h}(\mathbf{v}_{h/2}) \quad \forall \mathbf{v}_{h/2} \in V_{h/2}, \\ (r^{ma}_{h/2}, q_{h/2})_Q &= R^{ma}_{\mathbf{u}_h}(q_{h/2}) \quad \forall q_{h/2} \in Q_{h/2},\end{aligned} \tag{4.10}$$

on the finer mesh $\mathcal{T}_{h/2}$. Here the integrals containing $\mathbf{u}_h$ and p_h from the coarser mesh $\mathcal{T}_h$ can be evaluated numerically on the finer mesh $\mathcal{T}_{h/2}$ simply by evaluating the polynomials making up $(\mathbf{u}_h, p_h)$ on the new quadrature points of $\mathcal{T}_{h/2}$.

At first glance it might seem counter-intuitive to compute the residuals $\mathbf{r}^{mo}$ and r^{mo} on a finer mesh in order to get an estimate of the discretisation error related to $(\mathbf{u}_h, p_h)$. After all, why not just solve (4.1) on $\mathcal{T}_{h/2}$ in the first place? The reason is that the left-hand side of the equations (4.10) are independent of the state equations and the control. This means that the matrix in the linear system arising from applying the finite element method to (4.10) does not change either, only the right-hand sides. We can therefore perform the most computationally heavy parts in the first optimisation iteration, and then reuse these computations in the later iterations.

It can easily be shown that the matrices arising from applying the finite element method to (4.10), as done for the generalised Stokes equations in the previous section, are symmetric and positive definite, so for instance we could perform Cholesky factorisation [HJ90] once on the matrices in order to later get the exact solution for arbitrary right-hand sides relatively inexpensive. Another option is to use an *algebraic multigrid method* (AMG), which is what we will use in this report due to the generally lower memory footprint than full matrix factorisations. As we will use the *HYPRE* algebraic multigrid implementation provided by FEniCS, we do not go into the concept of multigrid methods in general here.

Lastly, we introduce the notion of *relative residuals*. Considering how $R^{mo}_{\mathbf{u},p}$ and $R^{ma}_{\mathbf{u}}$ depends on $\mathbf{u}$ in (4.8), and the fact that $\mathbf{u} = \mathbf{g}$ on $\partial\Omega$, it is not unreasonable to expect that the norm of the residuals will increase with the magnitude of $\mathbf{g}$. We generally do not want the values of residuals to depend too strongly on $\mathbf{g}$, as it will make difficult to compare the residuals for different numerical examples. In order to relate the magnitude of $\mathbf{g}$ to the value of the residuals, we therefore define the relative residuals to be

$$\eta^{mo} = \frac{\|\mathbf{r}^{mo}_{h/2}\|_V}{\|\mathbf{g}\|_{L^2(\partial\Omega)}} \quad \text{and} \quad \eta^{ma} = \frac{\|r^{ma}_{h/2}\|_Q}{\|\mathbf{g}\|_{L^2(\partial\Omega)}} \tag{4.11}$$

in hopes of having a quantity that will not differ too much for the different numerical experiments that will be introduced in Section 4.6.

4.4 Convergence test

In order check the correctness of our program, let us apply the refinement on a test example. To this end, the domain Ω, the porosity distribution ρ, the source function $\mathbf{f}$ and the boundary conditions in (2.6) can be chosen arbitrarily. For

simplicity we let Ω be the unit square and $\rho = 1$ everywhere in Ω, that is we recover the pure Stokes equations. Furthermore we continue to use homogeneous Dirichlet boundary conditions, and we let the source function be defined

$$\mathbf{f}(x, y) = \begin{bmatrix} \sin(x + 2y) \\ \sin(2x + y) \end{bmatrix}$$

for $(x, y) \in \Omega$.

Having chosen a test problem, we solve the state equations for different mesh sizes $n \times n$ exactly using a direct solver. The triangle diameter h of the mesh is inversely proportional to n, and we use the acquired solution solution $(\mathbf{u}_h, p_h)$ in the framework developed in Section 4.3 to estimate the residuals $\mathbf{r}^{mo}_{h/2}$ and $r^{ma}_{h/2}$. More precisely we apply the finite element method to (4.10) and solve the resulting linear system, again using a direct solver. We solve the test problem using both CR and TH type elements, and the result is depicted in Figure 4.3.

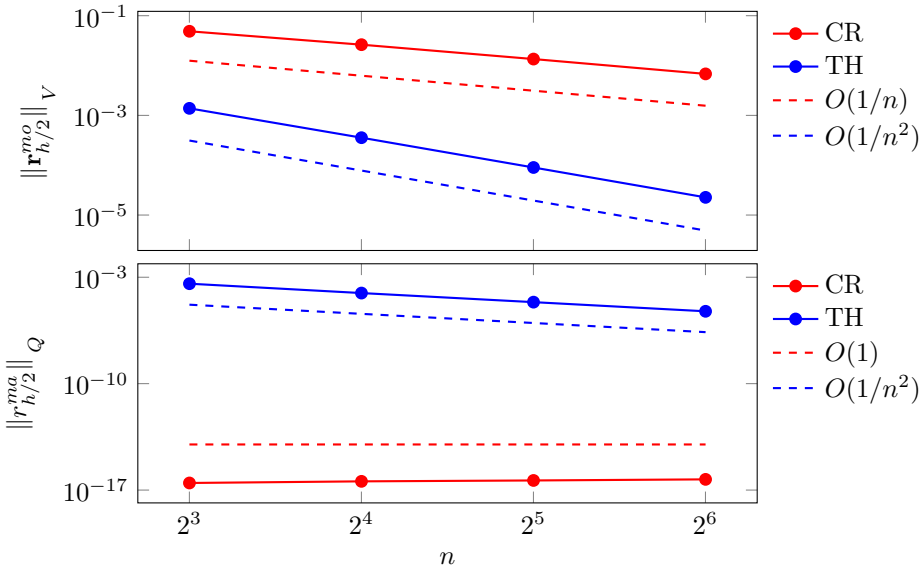


Figure 4.3: Log-log plot of the two residual norms for the test problem as a function of the discretisation size n for CR TH. n ranges from 8 to 64.

Starting with the TH elements, the convergence rate appears to be quadratic for both residuals norms. For the CR the convergence rate appears linear in for the momentum conservation equation, while being constant for the mass conservation equation. However, by noting the values of the axis in the second graph of Figure 4.3 we see that $\|r^{ma}_{h/2}\|_Q \sim 10^{-16}$ in this case, which corresponds to machine precision. In other words, second of (2.6) is solved exactly for the case of CR type elements, owing to the piecewise constant elements for the pressure.

When defining the general TH elements $\mathbb{P}_k - \mathbb{P}_{k-1}$, $k \geq 2$ in Subsection 4.2.1 we claimed that the lower degree for the pressure elements were necessary in order

for the pair of elements to be stable, that is the space V_h needs to be sufficiently "rich" compared to Q_h. However we de not discuss whether or not the TH family is optimal in terms of convergence rate, which in this situation amounts to whether or not we obtain any additional accuracy by increasing the order of the basis functions spanning V_h. For instance, would the pair $\mathbb{P}_3 - \mathbb{P}_1$ yield a more rapid convergence rate than the TH element considered above? To test this, we solve the test problem for the same n as previously using this new pair of elements, and the result can be seen in Figure 4.4. From Figure 4.4 we see that the mass conservation equation

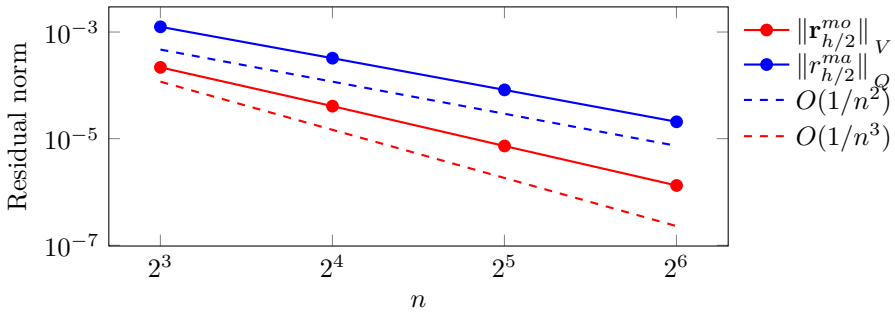


Figure 4.4: Log-log plot of the two residual norms for the element pair $\mathbb{P}_3 - \mathbb{P}_1$.

converges with the same rate as before for cubic basis functions for the velocity, so the higher order basis functions did not result in a higher accuracy in this case. For the momentum conservation equation we can see that there have been a slight increase in convergent rate, appearing to converge with a rate somewhere between cubic and quadratic. Unfortunately, cubic basis function results in a significant increase in the number of degrees of freedom for our problem, which is noticed as an increase in the size of the linear system (4.4). This makes the problem significantly more costly to solve, both computationally and storage wise. To this end we see that it is practical to continue using the TH elements.

In Section 4.3 where we introduced the residual estimates, we did not justify whether or not these estimates sufficiently approximates the true residuals. To address this question, we compare the residuals computed on $\mathcal{T}_{h/2}$ to the residuals evaluated on an even finer mesh $\mathcal{T}_{h/4}$, that is where we have performed the refinement illustrated in Figure 4.2 twice. To capture the difference in discretisation error, we evaluate $\|\mathbf{r}^{mo}_{h/2} - \mathbf{r}^{mo}_{h/4}\|_V$ on $\mathcal{T}_{h/4}$ for the test example, and the results can be seen in Figure 4.5. It is clear that between the two element types, the TH elements yield a small change in the residuals when refining the mesh one and two times, with about a 5 % decrease when doing two refinements compared to one. This indicates that for the TH elements, one refinement is sufficient in order to estimate $\mathbf{r}^{mo}$ accurately. For the CR elements, the change is larger at about 25 %, meaning that there still is a not insignificant discretisation error in $\mathbf{r}^{mo}_{h/2}$. This is something we will have to keep in mind when the residuals are used in Chapters 5 and 6.

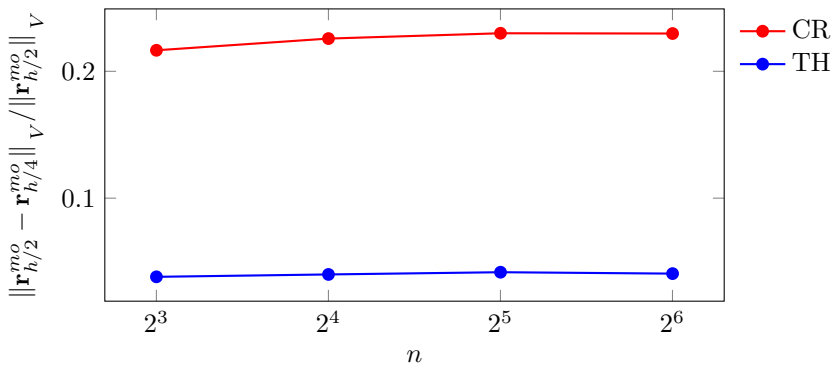


Figure 4.5: Plot of the relative difference in the norms for the momentum residual evaluated on $\mathcal{T}_{h/2}$ and $\mathcal{T}_{h/4}$ as a function of the grid size n.

4.5 Implementation of the optimisation problem

4.5.1 The discrete optimisation problem

Having introduced the finite element implementation of the generalised Stokes equations in Section 4.1 and inspected the convergence rates of the residuals, we now discretise the optimisation problem considered in Chapter 3.

As we already have defined the finite-dimensional spaces V_h and Q_h, where h is the maximum triangle diameter for a triangulation of $\mathcal{T}_h$ for Ω, we additionally have to define an appropriate space for the discretised controls ρ_h. Recall that in topology optimisation we prefer ρ to resemble a characteristic function, meaning that they will be piecewise constant, but with sudden jumps between 0 and 1. As such, piecewise constant functions is a natural choice, that is we define the space for the controls as

$$W_h = \{\, \rho_h : \Omega \to \mathbb{R} \;:\; \rho(\mathbf{x}_K) = \rho_K \,\forall \mathbf{x}_K \in K,\, \forall K \in \mathcal{T}_h,\, \rho_K \in \mathbb{R} \,\},$$

along with the set of box-constrained functions P_h and the set of admissible controls $P_{h,ad}$ by

$$P_h = \{\, \rho_h \in W_h \;:\; \rho_{\min} \leq \rho(\mathbf{x}) \leq 1 \,\forall \mathbf{x} \in \Omega \,\},$$
$$P_{h,ad} = \{\, \rho_h \in P_h \;:\; \int_\Omega \rho_h \leq \gamma |\Omega| \,\}.$$

By the boundedness of Ω we see that for a $\rho_h \in W_h$ we have

$$\left(\int_\Omega |\rho_h|^2 \right)^{1/2} = \left(\sum_{K \in \mathcal{T}_h} \int_K |\rho_K|^2 \right)^{1/2} \leq \sqrt{|\Omega|} \max_{K \in \mathcal{T}_h} |\rho_K| < \infty,$$

meaning $W_h \subset L^2(\Omega)$, which in turn implies that $P_h \subset P$ and $P_{h,ad} \subset P_{ad}$. Finally, as for (3.3), the discretised optimisation problem reads

$$\min_{\rho_h \in P_{h,ad}} J_{\alpha(\rho_h)}(\mathbf{u}_h) \quad \text{where } \mathbf{u}_h \text{ solves (4.1) with } \alpha = \alpha(\rho_h).$$

4.5.2 The projection onto the set of admissible controls

Before stating the algorithm for the complete optimisation algorithm in the next subsection, we present here the details for implementing the stopping criterion from Subsection 3.4.2.

The projection $\Pi_{P_h} : W_h \to P_h$ can be defined as for P, that is

$$\Pi_{P_h}(\rho_h) = \min\{1, \max\{\rho_{\min}, \rho_h\}\}. \tag{4.12}$$

Similarly, $\Pi_{P_{h,ad}} : W_h \to P_{h,ad}$ is given by

$$\Pi_{P_{h,ad}}(\rho_h) = \Pi_{P_h}(\rho_h - \lambda) \quad \text{with } \lambda \text{ such that} \quad \int_\Omega \Pi_{P_h}(\rho_h - \lambda) = \gamma|\Omega|. \tag{4.13}$$

In (4.13) it is necessary to estimate the Lagrange multiplier λ for the volume constraint in order to project onto $P_{h,ad}$, and although we have provided the equation λ needs to satisfy, we have not specified how this equation can be solved. To this end, we denote $\Delta^{ad}_{\text{vol}}(\rho_h, \lambda)$ as the residual of the equation for λ in (4.13), i.e.

$$\Delta^{ad}_{\text{vol}}(\rho_h, \lambda) = \int_\Omega \left(\gamma - \Pi_{P_h}(\rho_h - \lambda)\right).$$

Assuming now that we have been given some $\rho_h \in W_h$, estimating λ for (4.13) amounts to finding the roots of the function $\lambda \to \Delta^{ad}_{\text{vol}}(\rho_h, \lambda)$.

Bisection [Sau14, Sec. 1.1] is an algorithm for finding roots of continuous functions in an interval, and is what we will use to estimate λ. Generally, for a continuous function g we provide two bounds $\lambda_{\min}$ and $\lambda_{\max}$ such that $g(\lambda_{\min}) < 0$ and $g(\lambda_{\max}) > 0$. We can without loss of generality assume that $\lambda_{\min} < \lambda_{\max}$, as g can be multiplied with -1 without changing its roots. Then, by the intermediate value theorem there exists $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ such that $g(\lambda) = 0$. Bisection works by dividing the interval in half, and by the sign of the function value in the midpoint determine which half contains a root. The algorithm can then be performed recursively on this new interval until the interval containing a root is sufficiently small. Here we use

$$\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} < \epsilon_\lambda$$

for some $\epsilon_\lambda > 0$ as the stopping criterion for the bisection algorithm, which is depicted in Algorithm 1.

Having introduced the bisection algorithm, we want to apply Algorithm 1 to $\lambda \to \Delta^{ad}_{\text{vol}}(\rho_h, \lambda)$. In order to do this, we first find values for $\lambda_{\min}$ and $\lambda_{\max}$ such that $\Delta^{ad}_{\text{vol}}(\rho_h, \lambda_{\min})$ and $\Delta^{ad}_{\text{vol}}(\rho_h, \lambda_{\max})$ have opposite signs. We assume that

$$\int_\Omega \Pi_{P_h}(\rho_h) > \gamma|\Omega|, \tag{4.14}$$

as $\Pi_{P_{h,ad}}(\rho_h) = \Pi_{P_h}(\rho_h)$ if this is not the case, so using bisection is unnecessary. Starting with $\lambda_{\min}$, using $\lambda_{\min} = 0$ gives that $\Delta^{ad}_{\text{vol}}(\rho_h, \lambda_{\min}) < 0$ by (4.14), meaning

Algorithm 1 Bisection

Require: $\lambda_{\min} < \lambda_{\max}$, $g \in C([\lambda_{\min}, \lambda_{\max}])$, $g(\lambda_{\min}) \leq 0$, $g(\lambda_{\max}) \geq 0$, $\epsilon_\lambda > 0$

procedure BISECTION(g, $\lambda_{\min}$, $\lambda_{\max}$)
  $\lambda \leftarrow (\lambda_{\max} + \lambda_{\min})/2$
  if $(\lambda_{\max} - \lambda_{\min})/(\lambda_{\max} + \lambda_{\min}) < \epsilon_\lambda$ **or** $|g(\lambda)| < \epsilon_\lambda$ **then**
    return λ
  end if
  if $g(\lambda) < 0$ **then**
    return BISECTION($g, \lambda, \lambda_{\max}$)
  else
    return BISECTION($g, \lambda_{\min}, \lambda$)
  end if
end procedure

$\lambda_{\max}$ must be chosen such that $\Delta_{\text{vol}}^{ad}(\rho_h, \lambda_{\max}) > 0$. To see how this can be done, write the box-constraints in (4.12) as

$$\Pi_{P_h}(\rho_h) = \begin{cases} \rho_{\min} & \text{for } \rho_h \leq \rho_{\min}, \\ \rho_h & \text{for } \rho_{\min} < \rho_h < 1, \\ 1 & \text{for } \rho_h \geq 1, \end{cases}$$

meaning $\Pi_P(\rho_h - \lambda)$ can be written

$$\Pi_{P_h}(\rho_h - \lambda) = \begin{cases} \rho_{\min} & \text{for } \rho_h \leq \lambda + \rho_{\min}, \\ \rho_h - \lambda & \text{for } \lambda + \rho_{\min} < \rho_h < 1 + \lambda, \\ 1 & \text{for } \rho_h \geq 1 + \lambda. \end{cases} \tag{4.15}$$

Choosing $\lambda_{\max} = \max_{\mathbf{x}\in\Omega}\{\rho_n(\mathbf{x})\}$, then from (4.15) we see that $\Pi_{P_h}(\rho_h - \lambda_{\max}) = 0$ for all $\mathbf{x} \in \Omega$, meaning $\Delta_{\text{vol}}^{ad}(\rho_h, \lambda_{\max}) = \gamma|\Omega| > 0$.

Continuity of $\lambda \to \Delta_{\text{vol}}^{ad}(\rho_h, \lambda)$ can be seen by first observing continuity of $\lambda \to \Pi_{P_h}(\rho_h - \lambda)$. This follows from the fact that $\lambda \to \min\{h(\lambda), g(\lambda)\}$ and $\lambda \to \max\{h(\lambda), g(\lambda)\}$ are continuous for arbitrary continuous functions h and g. Next, ensuring that the integral in (4.13) is continuous, it is easiest to simply rely on the fact that the integration will be performed numerically, where we evaluate the integral as a finite sum over some given quadrature points.

As the requirements in Algorithm 1 are satisfied, the projection onto $P_{h,ad}$ is formulated as Algorithm 2.

4.5.3 The optimality criteria method

To address the implementation details of the optimality criteria method, let us summarise the most important details from Section 3.4 in terms of the discretised function and corresponding sets defined in Subsection 4.5.1.

Algorithm 2 Projection onto P_{ad}

Require: $\rho_h \in W_h$

procedure PROJECTION(ρ_h)
  if $\Delta_{\text{vol}}^{ad}(\rho_h, 0) \geq 0$ **then**
    return $\Pi_{P_h}(\rho_h)$ ▷ $\Pi_{P_h}(\rho_h)$ already satisfies the volume constraint
  end if
  $\lambda_{\min} \leftarrow 0$, $\lambda_{\max} \leftarrow \max_{\mathbf{x}\in\Omega}\{\rho_h(\mathbf{x})\}$
  Set $g(\lambda) = \Delta_{\text{vol}}^{ad}(\rho_h, \lambda)$
  $\lambda \leftarrow$ BISECTION($g, \lambda_{\min}, \lambda_{\max}$)
  return $\Pi_{P_h}(\rho_h - \lambda)$
end procedure

The move limits and damping factor are given as

$$Z_\lambda(\rho_h) = \min\left\{1+\zeta, \max\left\{1-\zeta, \left(-\frac{f'(\rho_h)}{\lambda}\right)^{\xi}\right\}\right\}\rho_h, \tag{4.16}$$

while the reduced gradient is

$$f'(\rho_h) = \frac{1}{2}\alpha'(\rho_h)|\mathbf{u}_h(\rho_h)|^2. \tag{4.17}$$

Again, $\mathbf{u}_h(\rho_h)$ is understood as the solution $\mathbf{u}_h$ to (4.1) with $\alpha = \alpha(\rho_h)$. Practical testing of the optimality criteria method showed that $\zeta = 0.4$ and $\xi = 0.5$ in (4.16) worked well overall, so these are the values we will use in the implementation. Finally, the fixed-point scheme for the discretised problem is written

$$\rho_{h,k+1} = \Pi_{P_h} Z_\lambda(\rho_{h,k}) \quad \text{with } \lambda \text{ such that} \quad \int_\Omega \Pi_{P_h} Z_\lambda(\rho_{h,k}) = \gamma|\Omega|, \tag{4.18}$$

with $\rho_{h,k} \in P_{h,ad}$ for $k = 0, 1, \ldots$.

Equivalently as for (4.13), we define

$$\Delta_{\text{vol}}^{oc}(\rho_h, \lambda) = \int_\Omega (\gamma - \Pi_{P_h} Z_\lambda(\rho_h)) \tag{4.19}$$

for (4.18), and we will again use bisection in order to compute λ. However, as opposed to (4.13), in (4.18) it is difficult to find explicit expressions for $\lambda_{\min}$ and $\lambda_{\max}$ such that $\Delta_{\text{vol}}^{oc}(\rho_h, \lambda)$ have different signs for a given ρ_h. Instead we only show that there exists a $\lambda > 0$ such that $\Delta_{\text{vol}}^{oc}(\rho_h, \lambda) = 0$.

Proposition 4.3. Assume that $\rho_h \in P_{h,ad}$ with

$$\int_\Omega \rho_h = \gamma|\Omega|. \tag{4.20}$$

We then have

$$\lim_{\lambda\to 0^+} \Delta_{\text{vol}}^{oc}(\rho_h, \lambda) < 0 \quad \text{and} \quad \lim_{\lambda\to\infty} \Delta_{\text{vol}}^{oc}(\rho_h, \lambda) > 0.$$

Proof. First, consider the situation $\lambda \to 0^+$ in (4.16) and assume that $\mathbf{u}_h(\rho_h) \neq \mathbf{0}$ everywhere in Ω. We do not prove the validity of this assumption, but note that the contrary was not observed during our numerical experiments. Under this assumption, from (4.17) we see that $f'(\rho_h) \neq 0$ everywhere in Ω as well, meaning that $(-f'(\rho_h)/\lambda)^\xi \to \infty$ in (4.16) yields

$$\lim_{\lambda \to 0^+} Z_\lambda(\rho_h) = (1+\zeta)\rho_h$$

for all $\mathbf{x} \in \Omega$. Furthermore, as $\zeta > 0$ and $\rho_h \in P_h$ we have $\rho_h < (1+\zeta)\rho_h$,

$$\lim_{\lambda \to 0^+} \Pi_{P_h} Z_\lambda(\rho_h) = \Pi_{P_h}((1+\zeta)\rho_h) \geq \Pi_{P_h}(\rho_h) = \rho_h$$

for all $\mathbf{x} \in \Omega$. The inequality in the above equation is strict on the set where $\rho_h < 1$, which is satisfied on a non-negligible part of Ω as $\gamma < 1$. Using the above estimate along with the dominated convergence theorem [Tao11, Thm. 1.4.49], we therefore get

$$\lim_{\lambda \to 0^+} \int_\Omega \Pi_{P_h} Z_\lambda(\rho_h) = \int_\Omega \lim_{\lambda \to 0^+} \Pi_{P_h} Z_\lambda(\rho_h) > \int_\Omega \rho_h = \gamma|\Omega|,$$

which along with (4.20) implies that the limit as $\lambda \to 0$ in (4.19) results in

$$\lim_{\lambda \to 0^+} \Delta^{oc}_{\mathrm{vol}}(\rho_h, \lambda) = \gamma|\Omega| - \lim_{\lambda \to 0^+} \int_\Omega \Pi_{P_h} Z_\lambda(\rho_h) < \gamma|\Omega| - \gamma|\Omega| = 0.$$

Next, consider $\lambda \to \infty$ in (4.16). In this case $(-f'(\rho_h)/\lambda)^\xi \to 0$, meaning (4.16) yields

$$\lim_{\lambda \to 0^+} Z_\lambda(\rho_h) = (1-\zeta)\rho_h.$$

In addition, since $\rho_h \in P_h$ we have $\rho_h > (1-\zeta)\rho_h$, meaning

$$\lim_{\lambda \to \infty} \Pi_{P_h} Z_\lambda(\rho_h) = \Pi_{P_h}((1-\zeta)\rho_h) \leq \Pi_{P_h}(\rho_h) = \rho_h$$

for all $\mathbf{x} \in \Omega$. Again, the inequality in the above equation is strict on the set $\rho_h > \rho_{\min}$, which is satisfied on a non-negligible part of Ω as $\gamma > \rho_{\min}$. The dominated convergence theorem gives

$$\lim_{\lambda \to \infty} \int_\Omega \Pi_{P_h} Z_\lambda(\rho_h) = \int_\Omega \lim_{\lambda \to \infty} \Pi_{P_h} Z_\lambda(\rho_h) < \int_\Omega \rho_h = \gamma|\Omega|,$$

and we arrive at the estimate

$$\lim_{\lambda \to \infty} \Delta^{oc}_{\mathrm{vol}}(\rho_h, \lambda) = \gamma|\Omega| - \lim_{\lambda \to \infty} \int_\Omega \Pi_{P_h} Z_\lambda(\rho_h) > \gamma|\Omega| - \gamma|\Omega| = 0.$$

□

As for (4.13), continuity of $\lambda \to \Delta_{\text{vol}}^{oc}(\rho_h, \lambda)$ follows from the continuity of $\lambda \to \min\{h(\lambda), g(\lambda)\}$ and $\lambda \to \max\{h(\lambda), g(\lambda)\}$ when h and g are two arbitrary, continuous functions. The integral is naturally also in this case evaluated numerically, implying continuity. We can therefore set $\lambda_{\min} = 0$, while for $\lambda_{\max}$ we can only conclude that since $\lim_{\lambda\to\infty} \Delta_{\text{vol}}^{oc}(\rho_h, \lambda) < 0$, $\lambda \to \Delta_{\text{vol}}^{oc}(\rho_h, \lambda)$ changes sign for some $\lambda > 0$, that is (4.18) has a root.

One way to find a valid $\lambda_{\max}$ could be to first define it to be some initial value, and then increase it with a fixed amount until $\Delta_{\text{vol}}^{oc}(\rho_h, \lambda_{\max})$ becomes positive. For instance, for a constant $c > 0$ we let $\lambda_{\max} \leftarrow \lambda_{\max} + c$ until $\Delta_{\text{vol}}^{oc}(\rho_h, \lambda_{\max}) \geq 0$, such that we know the interval $[\lambda_{\max} - c, \lambda_{\max}]$ contains a root. However, in our numerical experiment we found that simply setting $\lambda_{\max}$ sufficiently large was feasible. As such, we use will use $\lambda_{\min} = 0$ and $\lambda_{\max} = 10^4$ in our implementation. Regarding the lower bound $\rho_{\min}$, numerical experience showed that setting $\rho_{\min} = 0$ did not pose any apparent problems, so that is the value we will use in practice.

Finally, the implementation of the optimality criteria method is given in Algorithm 3. In our implementation we use $i_{max} = 500$, that is the iterations terminate after 500 iterations if the stopping criteria has not been met. For the stopping criterion itself we will use $\epsilon_{sc} = 0.1$. This value was chosen by visually inspecting the design $\rho_{h,i}$ and corresponding value $\|\rho_{h,i} - \text{PROJECTION}(\rho_{h,i} - f'(\rho_{h,i}))\|_{L^2(\Omega)}$, and it was found that when this value was less than 0.1 there were no visible changes in $\rho_{h,i}$.

Algorithm 3 The optimality criteria method

Require: $\rho_{h,0} \in W_h$, $\lambda_{\max} \in \mathbb{R}$, $i_{max} \in \mathbb{N}$, $\epsilon_{sc} > 0$
for $i \leftarrow 0, 1, \ldots, i_{max}$ **do**
  Get $\mathbf{u}_{h,i}$ by solving (4.4)
  Get $f'(\rho_{h,i})$ by using $\rho_{h,i}$ and $\mathbf{u}_{h,i}$ in (4.17).
  if $\|\rho_{h,i} - \text{PROJECTION}(\rho_{h,i} - f'(\rho_{h,i}))\|_{L^2(\Omega)} < \epsilon_{sc}$ **then**
    break
  end if
  $\lambda_{\min} \leftarrow 0$
  Set $g(\lambda) = \Delta_{\text{vol}}^{oc}(\rho_{h,i}, \lambda)$
  $\lambda_i \leftarrow \text{BISECTION}(g, \lambda_{\min}, \lambda_{\max})$
  $\rho_{h,i+1} \leftarrow \Pi_{P_h} Z_{\lambda_i}(\rho_{h,i})$
end for

4.5.4 The interpolation function

We end this section discussing the details of the inverse permeability α. Several different choices for the choice of α exist, with the most simple being that α is linear. [BP03] inspected this choice of α, but found that it induced a too severe penalty on the design, often resulting in non-global solutions. Instead they proposed a

q-parametrised function $\alpha_q[0,1] \to [\bar{\alpha}, \underline{\alpha}]$ defined as

$$\alpha_q(\rho) = \bar{\alpha} + (\underline{\alpha} - \bar{\alpha})\rho\frac{1+q}{\rho+q}$$

for a penalty parameter $q > 0$ and lower and upper bounds $\underline{\alpha}$ and $\bar{\alpha}$, respectively. It is easy to see that α_q is continuously differentiable, convex and strictly monotonically decreasing. Additionally, $\alpha_q(\rho) = \bar{\alpha}(1-\rho) + \rho\underline{\alpha}$ as $q \to \infty$, meaning that α_q will be close to linear for large q. An important result of [BP03] is that no regularisation is required to guarantee existence of solutions to the optimal control problem (3.3), meaning we can set $\underline{\alpha} = 0$ in α_q to get

$$\alpha_q(\rho) = \bar{\alpha}\left(1 - \rho\frac{1+q}{\rho+q}\right), \tag{4.21}$$

being the function we will use in the rest of this report. For $\bar{\alpha}$ we adopt the value chosen in [BP03], that is we set $\bar{\alpha} = 2.5/0.01^2$. To better understand how the value of q affects (4.21), α_q is plotted in Figure 4.6 with four different values of q. In our

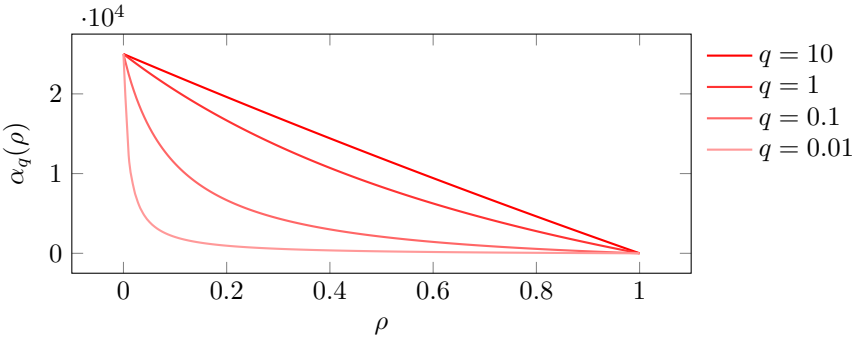


Figure 4.6: The interpolation function (4.21).

experiments we use $q = 0.1$ unless other is specified.

4.6 Numerical examples

We now introduce a set of numerical benchmarks, which will be used for the rest of this report. These benchmarks were initially introduced in [BP03], which can be used as a reference for confirming that the results from the optimality criteria method are reasonable.

In practice, problems are usually given in terms of $\mathbf{g}$, describing the inflow, outflow and no-slip parts of $\partial\Omega$. It is therefore useful to consider the inhomogeneous formulation (2.4) of the Darcy-Stokes equations. With this formulation in mind, we present the benchmarks along with the results obtained by application of the OC method discussed in Section 4.5.

4.6.1 A diffuser

The first example we present is a *diffuser*, a common benchmark in engineering applications and fluid mechanics. A diffuser is a device which controls the velocity of a fluid, usually designed as an open chamber where the cross section of the inlet is larger than that of the outlet. The design of the diffuser considered in this report is depicted Figure 4.7, and is modelled by the domain Ω being 1×1 box with flow entering from the left and exiting out of a smaller outlet on the right of the box.

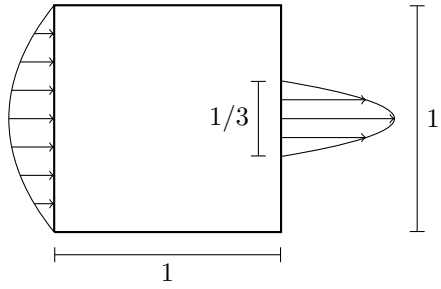


Figure 4.7: Design domain of the diffuser.

The boundary function is defined

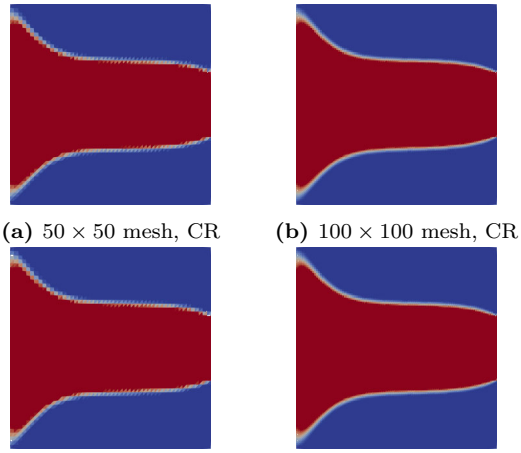
$$\mathbf{g}(x,y) = \begin{cases} (4y(1-y),0) & \text{for } x = 0, 0 \leq y \leq 1, \\ (108(y-\frac{1}{3})(\frac{2}{3}-y),0) & \text{for } x = 1, \frac{1}{3} \leq y \leq \frac{2}{3}, \\ (0,0), & \text{else.} \end{cases}$$

where we have chosen to use quadratic functions on the non-zero parts of $\mathbf{g}$ such that the maximum magnitude of the inlet velocity is 1. This models laminar flow in a straight pipe with no-slip conditions on the walls, meaning that the inlet in Figure 4.7 can be interpreted as being connected to a straight pipe with cross section 1, while the outlet is connected to a smaller pipe with cross section 1/3. In this example we let $\gamma = 0.5$ and $\mathbf{f} = \mathbf{0}$. As the initial guess for the OC method, we use the constant function $\rho_{h,0} = \gamma$.

We apply the method for a 50×50 mesh as well as a 100×100 mesh, using both the CR and the TH type elements. The final designs are presented in Figure 4.8 and the properties of the optimisation algorithm are summarised in Table 4.1.

Table 4.1: Computational data for the diffuser.

Element type	Mesh size	Iterations	Objective value
CR	50×50	43	30.44
	100×100	49	30.43
TH	50×50	44	31.02
	100×100	45	30.62



(a) 50×50 mesh, CR **(b)** 100×100 mesh, CR

(c) 50×50 mesh, TH **(d)** 100×100 mesh, TH

Figure 4.8: Resulting design of the optimality criteria method for a diffuser.

Considering Figure 4.8 and Table 4.1, the results seems to be fairly consistent for both mesh sizes and types of elements used. In Figure 4.8, the red colour is 1 and the blue is 0. These colours will be used for all the designs depicted in this report.

4.6.2 A pipe bend

The next benchmark we will present is the *pipe bend* example. Again, we consider Ω to be the unit square, but now with an inlet on the left boundary and an equal sized outlet on the bottom. The cross section of the inlet and outlet will be equal, but placed such that the domain is meant to model a 90° pipe bend, and can be seen in Figure 4.9. Here the prescribed volume fraction $\gamma = 0.08\pi$ is chosen to be

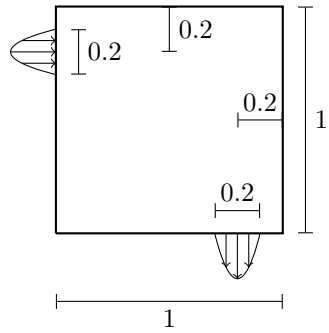


Figure 4.9: Design domain of the pipe bend.

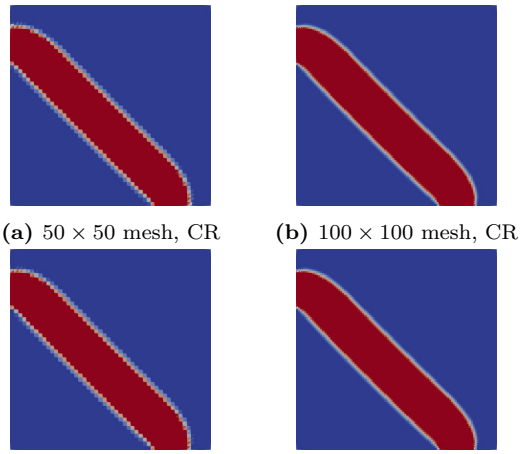
the area of a quarter annulus with inner and outer radius 0.7 and 0.9, respectively.

This would correspond to a circular pipe, which is a natural design to consider for this problem. We let $\mathbf{f} = \mathbf{0}$, and the flow on the boundary is defined

$$\mathbf{g}(x,y) = \begin{cases} (100(y-\frac{7}{10})(\frac{9}{10}-y),0) & \text{for } x=0,\ \frac{7}{10} \leq y \leq \frac{9}{10}, \\ (0,-100(x-\frac{7}{10})(\frac{9}{10}-x)) & \text{for } \frac{7}{10} \leq x \leq \frac{9}{10}, y=0, \\ (0,0) & \text{else,} \end{cases}$$

where again $\mathbf{g}$ is defined such that the maximum inflow is 1.

As before, the initial design $\rho_{h,0} = \gamma$ in the optimality criteria method and we consider a 50×50 and 100×100 mesh for both the TH and CR elements. The resulting designs are depicted in Figure 4.10.



(a) 50×50 mesh, CR **(b)** 100×100 mesh, CR

(c) 50×50 mesh, TH **(d)** 100×100 mesh, TH

Figure 4.10: Resulting design for the pipe bend benchmark.

The optimal design of a pipe bend appears to be wide, straight pipe between the inlet and the outlet, rather than a torus shaped pipe arguably more common in the fluid mechanics literature. In order to inspect this matter, we consider the reduced objective functional f. Since $\mathbf{f} = \mathbf{0}$ in this example, the objective functional (3.18) becomes

$$f(\rho) = \frac{1}{2}\int_\Omega \left(|\nabla \mathbf{u}|^2 + \alpha(\rho)|\mathbf{u}|^2 \right),$$

meaning that the dissipated power is due to shears only. The shears in a fluid are large when the pipe is thin, and small when the pipe is wide. A straight, wide pipe between the inlet and the outlet is therefore natural to consider yielding minimal shear in the fluid.

The properties of the optimisation algorithm are summarised in Table 4.2, and although the optimisation algorithm uses a few more iterations for the 50×50 mesh than for the 100×100 mesh for both element types, both the designs and objective values are fairly consistent for the four experiments.

Table 4.2: Computational data for the pipebend.

Element type	Mesh size	Iterations	Objective value
CR	50×50	41	9.70
	100×100	58	9.67
TH	50×50	44	9.96
	100×100	63	9.74

4.6.3 A rugby ball

The next example is somewhat different from the preceding two. In this example we will restrict ourselves to only look at at a 100×100 grid, and we will instead vary γ for the two element types. The domain is depicted in Figure 4.11, where the boundary function now is defined as

$$\mathbf{g}(x, y) = \begin{cases} (0, 1) & \text{for } 0 \leq x \leq 1, y \in \{0, 1\}, \\ (0, 0) & \text{else,} \end{cases}$$

and again $\mathbf{f} = \mathbf{0}$. For the different values for the volume fraction, we will use the three different values $\gamma = 0.8$, 0.9 and 0.95.

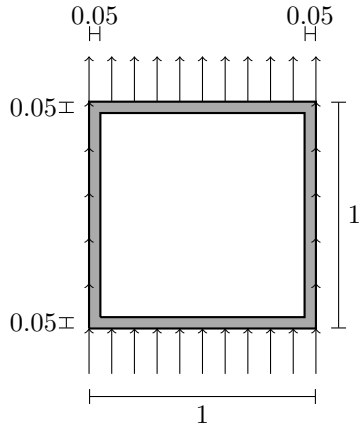


Figure 4.11: Design domain of the rugby ball. The grey area is prescribed to fluid.

For this problem it was observed that in the presumed global minimum all the solid material was distributed along the left and right boundary in Figure 4.11. As a measure to obtain a minimum where the solid material is located in the centre of the domain, a small area close to the boundary is prescribed to fluid, i.e. $\rho_h = 1$ here.

Prescribing the area around the boundary to fluid, we found that the optimisation algorithm still would not converge to the preferred minimum when using

$\rho_{h,0} = \gamma$ as an initial guess. Instead, for this benchmark we choose

$$\rho_{h,0}(x, y) = 0.1 + (x - 0.5)^2 + (y - 0.5)^2, \tag{4.22}$$

which we found gave satisfactory result for the three values of γ considered.

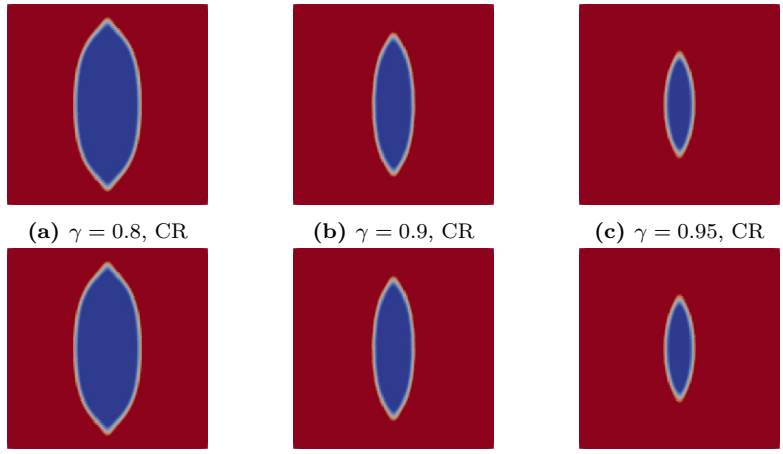
Several initial guesses were tried and, unsurprisingly, functions which were close to 0 around the centre of the domain and close to 1 near the boundary usually resulted in the methods converging to the preferred local minimum. In this sense there is nothing special about the exact choice of $\rho_{h,0}$, which resembles an elliptic parabola with centre in the middle of Ω. However we do note the constant term, which is necessary in order to have $\rho_{h,0} > 0$.

Lastly, as (4.22) is independent of γ, it is obvious that we generally do *not* have

$$\int_\Omega \rho_{h,0} = \gamma|\Omega|$$

when γ varies, which we initially required for $\rho_{h,0}$ in the optimality criteria method. However, this is a sufficient condition for existence of a λ satisfying the equation in (3.27), and looking at the box-constraints in (3.27), especially the definition of Z_λ, we can convince ourselves that if the move limit ζ is not too small, the equation in (3.27) can still be solved for λ even though $\rho_{h,0}$ does not satisfy the requirements exactly. We do not go into a deeper analysis of how much $\rho_{h,0}$ can deviate from the requirements for the optimality criteria method, but only note that we could successfully apply the bisection algorithm using $\rho_{h,0}$ as defined in (4.22) for the three values of γ.

The optimality criteria method is performed with the initial guess discussed above, and the result is given in Figure 4.12. Again, the resulting designs looks



(a) $\gamma = 0.8$, CR **(b)** $\gamma = 0.9$, CR **(c)** $\gamma = 0.95$, CR

(d) $\gamma = 0.8$, TH **(e)** $\gamma = 0.9$, TH **(f)** $\gamma = 0.95$, TH

Figure 4.12: Resulting design for a the rugby ball benchmark.

identical for the two types of elements, and their properties can be seen in Table

Table 4.3: Computational data for the rugby ball.

Element type	γ	Iterations	Objective value
CR	0.8	32	68.36
	0.9	46	43.29
	0.95	27	34.61
TH	0.8	31	76.58
	0.9	45	51.39
	0.95	26	42.68

4.3. As we would expect, the objective value decreases as γ increases, yielding a smaller rugby ball in the centre. However, despite the results looking visually similar in Figure 4.12 for the two elements, the objective values for the TH elements are consistently slightly larger than those for the CR elements. As the value of the objective functional is dependent on $\mathbf{u}$, it is naturally to assume that the TH elements will yield the most correct objective values, as the velocity elements are of order 2, while the CR elements only are of order 1.

4.6.4 A double pipe

The last example we consider is a model of a box of height 1 and width δ which is connected to two parallel pipes. The domain can be viewed in Figure 4.13, and in this example we will vary the width δ of the box in order to see how the solutions of the optimisation problem changes.

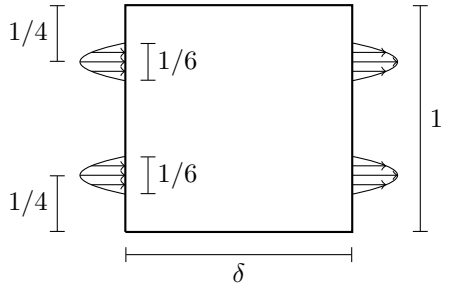


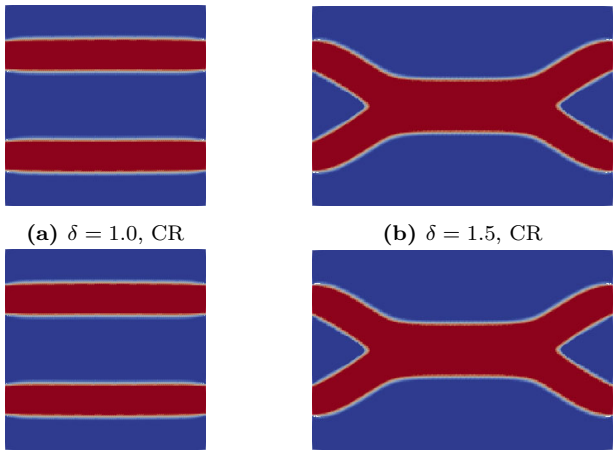
Figure 4.13: Design domain of the double pipe example.

The flow function is in this example defined

$$\mathbf{g}(x,y) = \begin{cases} (144(y-\frac{1}{6})(\frac{1}{3}-y),0) & \text{for } x \in \{0,1\},\ \frac{1}{6} \le y \le \frac{1}{3}, \\ (144(y-\frac{2}{3})(\frac{5}{6}-y),0) & \text{for } x \in \{0,1\},\ \frac{2}{3} \le y \le \frac{5}{6}, \\ (0,0) & \text{else,} \end{cases}$$

and we test the algorithms for values $\delta = 1$ and $\delta = 1.5$, with grid resolution 100×100 and 150×100, respectively. We will also use $\gamma = \delta/3$, which corresponds

to the area used by two straight pipes through the domain. From a numerical point of view, acquiring good local minima has proven itself difficult for this problem, and to that end we adopt the technique from [BP03] for finding a good initial guess. Using an initial control $\rho_{h,0} = \gamma$, we perform a certain number of iterations with $q = 0.01$ in (4.21). We then use the function obtained from these iterations as the initial guess, and proceed normally with $q = 0.1$. We found that 50 iterations were sufficient in order to obtain the preferred solutions, and the results can be seen in Figure 4.14. For $\delta = 1.0$ the optimal solution is two simply straight pipes, while



(a) $\delta = 1.0$, CR **(b)** $\delta = 1.5$, CR

(c) $\delta = 1.0$, TH **(d)** $\delta = 1.5$, TH

Figure 4.14: Resulting design for the double pipe benchmark.

for $\delta = 1.5$ the minimum seems to be attained when the two pipes joins together to form a single, wider pipe. In light of our previous discussion regarding how wider pipes results in a lower velocity, it is reasonable that when δ is large enough, a single, wider pipe is optimal. The numerical results of the optimisation algorithm are seen in Table 4.4. The first 40 iterations performed with $q = 0.01$ are included in the values for the *Iterations* column.

Table 4.4: Computational data for the double pipe benchmark.

Element type	δ	Iterations	Objective value
CR	1.0	44	21.88
	1.5	269	24.00
TH	1.0	59	22.13
	1.5	384	24.81

Considering Figure 4.14a and 4.14c, and assuming the optimal solution is in fact two completely straight pipes with width 1/6, the objective value can be computed analytically. In this case we can consider one of the pipes as the rectangle $[0, 1] \times$

$[0, 1/6]$ containing pure Stokes flow, with inflow and outflow on the left and right side, respectively, and no-slip conditions on the top and bottom. $\mathbf{u}$ is then trivially given from the boundary function as

$$\mathbf{u}(x, y) = \begin{bmatrix} 144y\left(\frac{1}{6} - y\right) \\ 0 \end{bmatrix},$$

and the dissipated power of the two pipes becomes

$$J_0(\mathbf{u}) = 2\left(\frac{1}{2}\int_\Omega |\nabla \mathbf{u}|^2\right) = \int_0^{1/6} \left[144\left(\frac{1}{6} - 2y\right)\right]^2 \mathrm{d}y = 32.$$

Comparing this value to the computed values in Table 4.4, we see that the computed value is significantly smaller than the real value. [BP03] addresses the question of how this deviation in objective value affects the design obtained from the optimisation algorithm by increasing $\bar{\alpha}$ in (4.21) and decreasing q in order to get a more apparent 0-1 design. It is concluded that it has no visible impact on the optimal design, meaning that we although we can not expect the computed objective value to accurately reflect the real value, the computed design is expected to resemble the true solution of the optimisation problem.

Chapter 5

Adaptivity

Having introduced residuals in the previous chapter, we saw that we can get an estimate for how well discretised a pair of discretised functions $(\mathbf{u}_h, p_h)$ satisfies the continuous state equations (2.6). In this chapter we use these residuals to refine the mesh locally during optimisation, in hopes of significantly improving the residuals norms over the whole domain.

5.1 Adaptive mesh refinement

Using the Riesz representation theorem to identify functions $\mathbf{r}^{mo} \in V$ and $r^{ma} \in Q$ as the residuals, we can estimate how well $(\mathbf{u}_h, p_h)$ satisfies (2.6) on different parts of the mesh. This is done by calculating *local* norms $\|\mathbf{r}^{mo}\|_{H^1(K)}$ and $\|r^{ma}\|_{L^2(K)}$ for $K \in \mathcal{T}_h$, meaning we consider the norm of the residuals on each cell of the mesh.

Assuming $(\mathbf{u}_h, p_h)$ solves the discretised state equations (4.1) so accurately that the discretisation error dominates in the values of the residuals, the local residual norms give estimates of how large this discretisation error is on different parts on the domain. This information can in turn be used to determine where reducing the mesh size locally will yield the most significant reduction in $\|\mathbf{r}^{mo}\|_V$ and $\|r^{ma}\|_Q$, so it makes sense to refine the mesh in these regions. Figure 5.1 illustrates how the this process is performed in FEniCS when two cells marked red are subject to refinement, reducing the local mesh size by half.

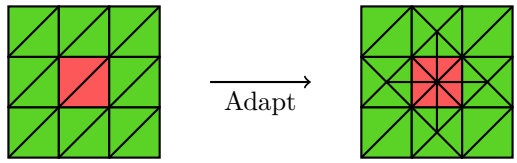


Figure 5.1: A 3×3 structured mesh with two marked cells before and after refinement.

How to exactly to choose which cells to be subject to the refinement can be done in a multiple of ways. One option could for instance be to refine a fixed portion of

the cells with the largest local residuals in each refinement. This strategy gives a very predictable increase in the number of cells for a refinement, which is useful in situations where it is essential that the size of the linear system does not become too large. This is especially the case when using higher order basis functions, where the number degrees of freedom increases rapidly with the number of cells in the mesh. In order to implement this strategy for a distributed program, it is necessary to be able to select the cells with largest local residual norms distributed among all processes. Unfortunately, FEniCS does not provide access to advanced MPI functionality through its API, making this strategy difficult to implement in practice.

Instead, our approach will be to define a global threshold, such that all cells whose residual local norms exceeds this value will be refined. We derive this threshold only for $\mathbf{r}^{mo}$, as the case r^{ma} is completely equivalent. To this end, assume that $\mathcal{T}_{h/2}$ is the triangulation obtained when uniformly refining some triangulation $\mathcal{T}_h$ of Ω, that is when each cell in $\mathcal{T}_h$ is refined according to Figure 4.2. Furthermore, let $\mathbf{r}^{mo}_{h/2} \in V_{h/2}$ be defined as in the first of (4.10) for some $(\mathbf{u}_h, p_h) \in V_h \times Q_h$. We have

$$\|\mathbf{r}^{mo}_{h/2}\|^2_V = \|\mathbf{r}^{mo}_{h/2}\|^2_{H^1(\Omega)} = \sum_{K \in \mathcal{T}_h} \|\mathbf{r}^{mo}_{h/2}\|^2_{H^1(K)},$$

where dividing by $\|\mathbf{r}^{mo}_{h/2}\|^2_V$ yields

$$\sum_{K \in \mathcal{T}_h} \frac{\|\mathbf{r}^{mo}_{h/2}\|^2_{H^1(K)}}{\|\mathbf{r}^{mo}_{h/2}\|^2_V} = 1.$$

The mean of the terms in the above sum is $1/|\mathcal{T}_h|$, where $|\mathcal{T}_h|$ is the number of cells in the mesh. We define the threshold for the cells to be subject to refinement if the local residual norm is larger than some factor $c_{mo} > 0$ of the mean, so a cell K will be refined if

$$\frac{\|\mathbf{r}^{mo}_{h/2}\|_{H^1(K)}}{\|\mathbf{r}^{mo}_{h/2}\|_V} > \sqrt{\frac{c_{mo}}{|\mathcal{T}_h|}}. \tag{5.1}$$

Similarly, if we instead consider r^{ma} we have

$$\frac{\|r^{ma}_{h/2}\|_{L^2(K)}}{\|r^{mo}_{h/2}\|_Q} > \sqrt{\frac{c_{ma}}{|\mathcal{T}_h|}} \tag{5.2}$$

for a constant $c_{mo} > 0$. The choice of values for the parameters c_{mo} and c_{ma} will be addressed through numerical experiments in the coming section.

We recall from Figure 4.5 that there was still a significant discretisation error related to the momentum residuals for the CR elements after refining once. Again, addressing this problem is outside the scope of this project, so we will for practical reasons only consider the residuals evaluated on the one time uniformly refined mesh, and see if yields reasonable results.

5.2 Numerical experiments

From the figures in Section 4.6 we got a good impression about what the optimal design for the different benchmarks should look like. For the diffuser design in Figure 4.8 and pipe bend design in Figure 4.10 it is clear that the designs for the 100×100 meshes are more smooth in the transition between the Stokes flow and solid material parts of the domain than the 50×50 mesh, which is natural due to the higher resolution of the mesh. However, on the interior of the two regions the designs for the two mesh sizes look identical, as the design is uniformly 0 and 1 here. This means that especially for the 100×100 mesh there might be an unnecessary amount of cells in these regions, which could have been modelled just as well with the mesh resolution corresponding to the 50×50 mesh, or potentially even lower.

5.2.1 The diffuser

We consider the diffuser example. Using the adaptive refinement approach presented in Section 5.1, we start with the 50×50 mesh and will during the optimisation algorithm locally refine the mesh where the local residual norms are largest, according to (5.1) and (5.2). However, it is not clear what the parameters c_{mo} and c_{ma} should be chosen as. One thing we immediately see is that $c_{mo} = c_{ma} = 1$ corresponds to locally refining all cells where the value of the local residuals are above average among all the residuals.

We will therefore perform the optimisation with adaptive refinement for the three values 1, 2.5 and 5 for c_{mo} and c_{ma}, and observe how the number of cells increases during the course of optimisation. An important property for the a posteriori residuals was that the matrix arising from applying the finite element method to the two systems in (4.10) remains constant throughout the OC iterations, such that the heavy lifting ideally needs to be performed only once in the beginning. When the mesh is refined, these matrices will need to be assembled on the new mesh, so refining the mesh in every OC iteration would not allow us to reuse the matrices for computing the residuals. Hence, we will instead perform adaptive refinement every 10'th iteration.

We start with considering the momentum residual, meaning we use (5.1) in order to determine which cells to refine. In Figure 5.2 we have used the CR elements, and we plot the number of cells along with the relative momentum residual η^{mo} during the optimisation procedure. The first thing to note is that smaller c_{mo} yields larger increase in the number cells that are refined, which is reasonable considering smaller c_{mo} implies a lower threshold for cells to be subject to refinement. Furthermore, the difference between $c_{mo} = 1$ and $c_{mo} = 2.5$ is quite substantial. Where the mesh corresponding to the latter value has about 1.6×10^4 cells towards the end of the optimisation procedure, the mesh corresponding to the former value of about 1.4×10^5 cells. For reference, the 100×100 mesh has $2 \times 100^2 = 2 \times 10^4$ cells.

Considering the plot for η^{mo} in Figure 5.2, observe that for all three values of c_{mo}, the relative residual norm is in fact smaller than that for the 100×100 mesh towards the end. Note also that for $c_{mo} = 2.5$ and $c_{mo} = 5$ the number of cells in the refined mesh is smaller than in the 100×100 mesh. This means that by

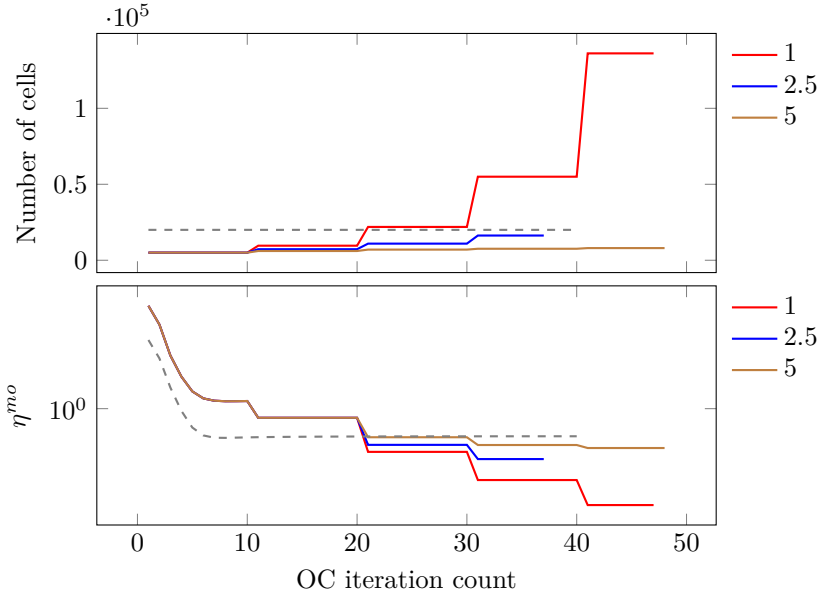
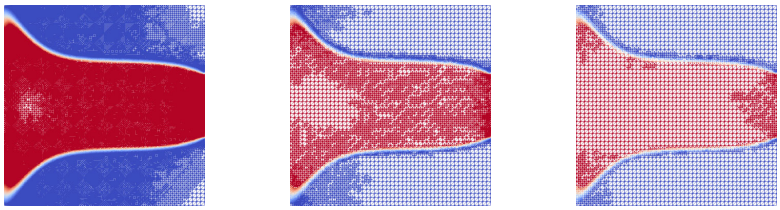


Figure 5.2: Number of cells in mesh and relative residual norm η^{mo} during the optimisation algorithms for three different values of c_{mo}, using CR elements. The grey, dashed line corresponds to the 100×100 mesh without adaptive mesh refinement.

using the refinement approach during the optimisation iterations, we are in fact able to solve the continuous equation (2.6) with greater accuracy, with a smaller linear system in (4.4).

In Figure 5.3 the refined meshes is depicted for the three values of c_{mo}. As



(a) $c_{mo} = 1$ **(b)** $c_{mo} = 2.5$ **(c)** $c_{mo} = 5$

Figure 5.3: The facets of the meshes refined using the momentum residual with the CR elements. The colour corresponds to ρ_h.

reflected in Figure 5.2 regarding the number of cells for $c_{mo} = 1$, we see from Figure 5.3a that the mesh has a large amount of cells. In fact it can be seen that it has been refined on most parts of the domain, except for in a small area in the upper and lower right corner. As the Considering Figure 5.3b, there has still been a significant amount of refinement on the red part of the domain, corresponding to Stokes flow. However, we also clearly see that the mesh has also been refined

around the transition parts of the domain. For Figure 5.3c, the refinement is located mainly in the transition parts of the domain, although there has been some refinement on the red parts as well, especially around the inlet and outlet of the diffuser.

Having performed adaptive refinement using the CR elements, we perform the same experiment again with the TH elements. We start using (5.1) as a threshold, and perform refinement with the values 1, 2.5 and 5 for c_{mo}. The results are depicted in Figure 5.4. The results are fairly similar to the case with CR elements,

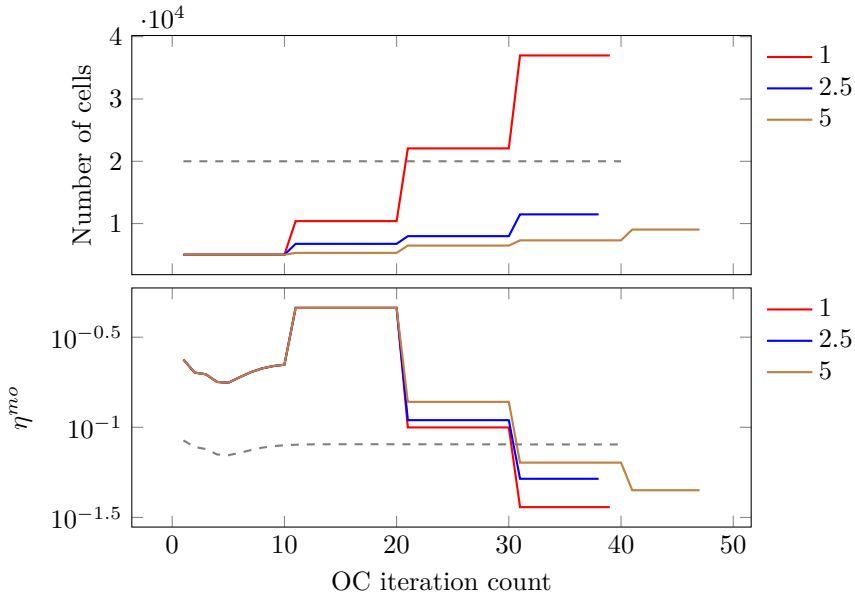


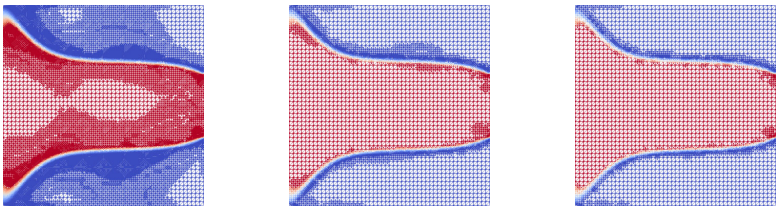
Figure 5.4: Number of cells in mesh and relative residual norm η^{mo} during the optimisation algorithms for three different values of c_{mo}, using TH elements. The grey, dashed line corresponds to the 100×100 mesh without adaptive refinement.

where $c_{mo} = 1$ yields a significantly finer mesh than the 100×100 mesh in the later iterations of the OC method. For $c_{mo} = 2.5$ and $c_{mo} = 5$ the resulting mesh is coarser, but from η^{mo} in Figure 5.4 we see that the resulting residual norm is overall smaller for these two values compared to the 100×100 mesh not subject to refinement, meaning that we again get a better approximation of the continuous equation with a smaller linear system.

Note that for $c_{mo} = 5$, the mesh is subject to an extra refinement at iteration 40 of the OC method, as the method uses about 7 to 8 extra iterations to satisfy the stopping criterion compared to the other meshes. After this additional refinement, the mesh still has fewer cells than towards the end than for $c_{mo} = 2.5$, but from the plot of the residual norm we can see that the resulting value of η^{mo} for $c_{mo} = 5$ is slightly smaller than that $c_{mo} = 2.5$. This means that in this case, the cost of performing an extra refinement and 8 additional iterations, we get a resulting design where the solution to the state equations is slightly better approximated

than for $c_{mo} = 2.5$.

In Figure 5.5 the meshes from Figure 5.4 can bee seen. Again, the case $c_{mo} = 1$



(a) $c_{mo} = 1$ **(b)** $c_{mo} = 2.5$ **(c)** $c_{mo} = 5$

Figure 5.5: The facets of the meshes refined using the momentum residual with the TH elements. The colour corresponds to the values of the resulting design ρ_h.

is quite different than for the two other values, as in this case a significant amount refinement have occurred in the interior of the solid and flow regions of the solution. This is reflected in the number of cells seen in Figure 5.4, which is significantly more than for the other meshes. As the residual uses quadratic basis functions, the number of degrees of freedom increases even more rapidly with the number of cells compared to linear elements, which dictates the size of the linear system in (4.4). Figure 5.5b and 5.5c look more similar in terms of that only small regions of the interior flow and solid regions have been refined. Most of the refinement have occurred on the transition between the regions, where the main difference between $c_{mo} = 2.5$ and $c_{mo} = 5$ appears to be that the former has been refined more extensively. However we emphasise again that the residual norm for the latter is the smaller one of the two due to the extra refinement at iteration 40.

We now turn our attention to the residual associated with the mass conservation equation. As we have seen previously, for the CR elements the mass conservation equation in (2.6) is expected to be satisfied to machine accuracy for all mesh sizes, so using the values of r^{ma} to determine what cells to refine does not make sense in this case. For the TH elements however, there is a discretisation error related to the mass conservation equation, so using the estimate of the mass conservation residual for marking cells subject to refinement is a reasonable approach. We perform the same experiment as for $\mathbf{r}^{mo}$ above, but now we use the same values for c_{ma} as we did for c_{mo}, and the results can be seen in Figure 5.6.

The first thing we note is that for $c_{ma} = 1$, the number of cells towards the end of the iterations is about 2.1×10^4, just slightly more than for the 100×100 mesh. This is significantly less than when using the momentum residual in Figure 5.4, whose mesh had about twice the number of cells towards the end of the iterations. Furthermore, for $c_{ma} = 1$ and $c_{ma} = 2.5$ it is actually necessary to perform an additional refinement at iteration 40 as opposed to $c_{ma} = 5$, which is the opposite of what is observed Figure 5.4. Consequently, η^{ma} is significantly lower for $c_{ma} = 1$ and $c_{ma} = 2.5$ than for $c_{ma} = 5$ in their respective design. That being said, comparing η^{ma} in the last iteration for $c_{ma} = 1$ with the value of η^{ma} for $c_{ma} = 2.5$ and $c_{ma} = 5$ in the corresponding iteration, the two latter values are still smaller.

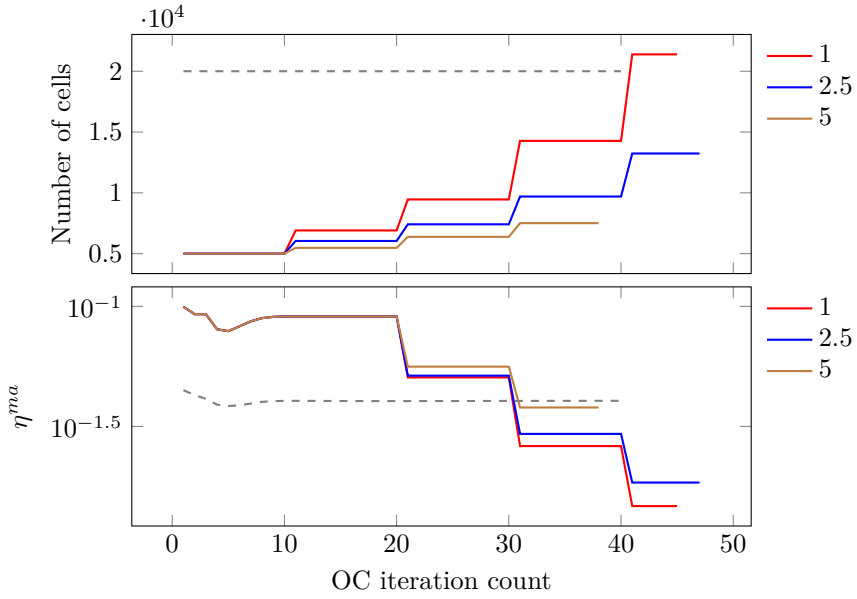
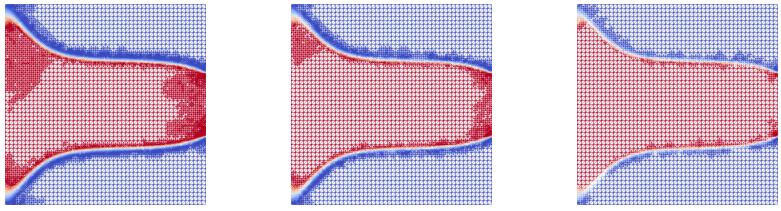


Figure 5.6: Number of cells in mesh and relative residual norm η^{ma} during the optimisation algorithms for three different values of c_{ma}, using TH elements. The grey, dashed line corresponds to the 100×100 mesh without adaptive refinement.

We also note for all the three values of c_{ma}, the resulting η^{ma} is smaller than that for the 100×100 mesh.

In Figure 5.7 the meshes from Figure 5.4 have been plotted. The tendencies



(a) $c_{ma} = 1$ **(b)** $c_{ma} = 2.5$ **(c)** $c_{ma} = 5$

Figure 5.7: The facets of the meshes refined using the mass residual with the TH elements. The colour corresponds to ρ_h.

can be seen to similar as for the momentum residuals considered previously, and the three meshes in Figure 5.7 are qualitatively similar in terms of where they have been refined. For $c_{ma} = 1$ and $c_{ma} = 2.5$ a certain amount of refinement has occurred around the interior of the free flow, more precisely around the inlet and outlet. Although noticeably less than in Figure 5.5b, Figure 5.7c shows that almost all the refinement has occurred in the transition region.

Considering that both Figure 5.5 and 5.7 show similar tendencies in terms of

refinement occurring in the transition region, it is to see how this affects the value of that residual *not* subject to refinement. More precisely, we also compute η^{mo} when (5.2) is used as the refinement criterion, and the results can be seen in Figure 5.8. Comparing Figure 5.8 and the η^{mo} plot in Figure 5.4, the behaviour of η^{mo}

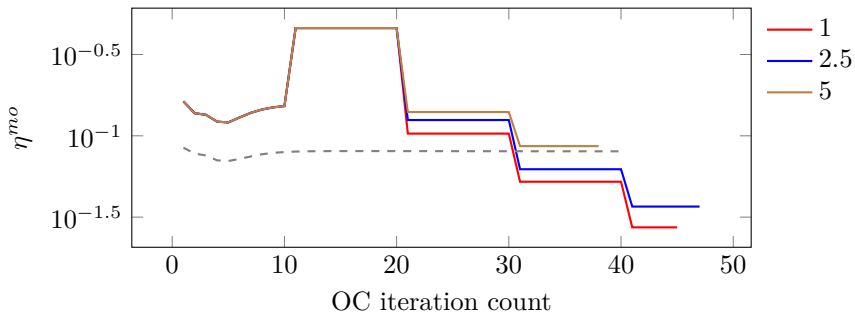


Figure 5.8: The relative residual norm η^{mo} during the optimisation algorithms for three different values of c_{ma}, using TH elements. The grey, dashed line corresponds to η^{mo} for the 100×100 mesh without adaptive refinement.

seems very similar for (5.1) and (5.2), at least until iteration 30 of the OC method. For $c_{mo} = 5$, the algorithm stops after 38 iterations, where η^{mo} is now slightly larger than for the 100×100. Although this difference is rather small, we have in Figure 5.4 that for the corresponding iteration η^{mo} was smaller than for the 100×100 mesh when $c_{mo} = 5$. As discussed previously, the OC method needs to perform some additional iterations when (5.2) is used as the refinement criterion for $c_{mo} = 2.5$ and $c_{mo} = 5$, compared to when (5.1) is used. As this naturally leads to an additional reduction of η^{mo}, the consequence is that η^{mo} for the resulting designs for $c_{mo} = 2.5$ and $c_{mo} = 5$ are approximately the same for when (5.1) and (5.2) is used for refinement.

In light of the above experiments, let us summarise our findings so far. The first thing to note is regarding the number of optimisation iterations necessary to satisfy the stopping criterion when refining the mesh during the iterations. From the findings above it appears that for the diffuser example the number of necessary iterations was not significantly affected by ongoing refinement in any situation. Furthermore, a significant improvement in the value of η^{mo} and η^{ma} was observed for all the experiments, resulting in a smaller residual norm than for the 100×100 mesh, using fewer cells in the refined mesh. The exception was using (5.1) with $c_{mo} = 1$, where the resulting meshes had significantly more cells than 100×100 mesh.

As the 100×100 is already quite large considering we also perform a uniform refinement of the mesh when calculating the residuals, a rule of thumb could be that we ideally do not want the refined mesh such that at we at any time exceed the number of cells corresponding to the 100×100 mesh. This means that c_{mo} and c_{ma} being 1 results in a too low threshold for marking cells subject to refinement. Additionally, considering the meshes when $c_{mo} = c_{ma} = 5$, it appears that choosing

a value larger than this would generally result in a too high threshold, meaning we can not expect to get a significant decrease in the residual norms in this case. We conclude that a value between 2.5 and 5 is appropriate for c_{mo} and c_{ma}, where the exact value will probably depend on the benchmark considered, along with how large an increase in the size of the linear system (4.4) is tolerable.

5.2.2 The double pipe

We conclude this section by applying the adaptive refinement approach the double pipe benchmark from Subsection 4.6.4 with $\delta = 1.5$. This experiment is significantly more difficult to solve numerically than the diffuser benchmark, needing between 400 and 500 iterations according to Table 4.4, due to slow convergence. We start with 75×50 mesh, and as we expect the method to perform a large number of iterations, we refine the mesh every 50'th iteration. The refinement is performed using (5.1), and numerical experience showed that $c_{mo} = 4$ was appropriate for both the CR and TH elements.

The numerical values are listed in Table 5.1, and the refined meshes are depicted in Figure 5.9. Comparing the refined and non-refined method using the CR ele-

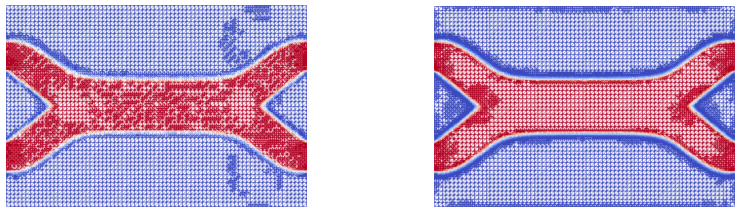
Table 5.1: Computational data for the double pipe benchmark with $\delta = 1.5$. When using the adaptive refinement approach, a 75×50 mesh is used in the beginning, while the 150×100 mesh is used when not using refinement.

Element type	Refined	Iters.	J	η^{mo}	η^{ma}	Cells
CR	Yes	375	23.99	1.36	1.53×10^{-13}	15231
	No	269	24.57	1.33	1.32×10^{-13}	30000
TH	Yes	384	24.02	0.04	0.03	26540
	No	384	24.81	0.22	0.14	30000

ments, we see from Table 5.1 that the difference in number of iterations necessary is significantly larger, using about 40 % more iterations. This is in contrast to the TH elements, where the number is the same as for the non-refined method. There has also been a slight decrease in the objective value J for both element type when using adaptive refinement, with a difference in about 2.4 % and 3.2 % for the CR and TH elements, respectively. Looking at the columns for the relative residual norms in 5.1, the values for η^{mo} is similar for the CR elements, with a slight increase of 2.2 %. As $\eta^{ma} \approx 0$ for the CR elements, small differences are likely to occur due to round-off errors, which are of little interest. For the TH elements, the decrease in the relative residual norms are significantly reduced when using refinement, with an 81.8 % decrease in η^{mo} and a 78.6 % decrease in η^{ma}. Comparing this to the number cells in the mesh, we see that this decrease is achieved using 11.5 % fewer cells in the mesh. Finally, for the CR elements we see that the resulting mesh 49.2 % smaller when using adaptive refinement, a significant reduction in the size of the linear system.

Considering 5.9, the refined meshes have the same tendencies as for the diffuser

example. When using the CR elements, more of the refinement occurs in the



(a) CR **(b)** TH

Figure 5.9: Resulting refined meshes corresponding to the data presented in Table 5.1, beginning with a 75×50 mesh.

free flow regions of the domain. Although refinement also occurs in the transition region, it is not as prominent as when using the TH elements, where the refinement mainly occurs in the transition region.

Chapter 6

Iterative approach

Among the Krylov subspace methods, the conjugate gradient method is arguably the most well-known, and usually the preferred choice when applicable. However, the conjugate gradient method requires the matrix in question to be symmetric and positive definite. As we saw in Section 4.1, although the system matrix S in (4.6) is symmetric, it is indefinite. Since the conjugate gradient method cannot be used, we will have to rely on some other method. Preferably we still want to take advantage of the symmetry of our linear system, and to this end we will use the *MINRES* method.

6.1 The MINRES method

The name "MINRES" is a contraction of "Minimal residual", whose name come from the fact that the 2-norm of the residual of the current iterate is minimised over the current search space in each iteration. In the following two subsections we give a detailed background of the MINRES method, such that if reader is already familiar with MINRES, or is not concerned with the derivation of the method, he or she can skip to Subsection 6.1.3.

6.1.1 The method

During our research, we were not able to find a suitable implementation of the MINRES method readily available in the literature. While one algorithmic formulation can be found in [CS06, Tab. 3.4], this implementation requires the zero vector to be used as the initial guess. As the linear system (4.4) will be solved in each step of the OC method, using an initial guess for MINRES based on previous iterates could potentially reduce the number of MINRES iterations necessary to perform in each OC iteration. Hence, we now extend the implementation given in [CS06] to utilise a non-zero vector as the initial guess. We assume that the reader is familiar with the concept of Krylov subspace methods and Arnoldi's procedure for constructing orthogonal bases of Krylov subspaces. If not, we refer to [Saa03,

Ch. 6] for an introduction to these concepts.

Consider the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{6.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric. Given an initial guess $\mathbf{x}_0$ to (6.1), we define the order-k Krylov subspace as

$$\mathcal{K}_k(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\},$$

with $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ being the residual associated with $\mathbf{x}_0$. Stated in [Saa03, Alg. 6.1], Arnoldi's procedure can be used to construct an orthonormal basis $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of $\mathcal{K}_k(A, \mathbf{r}_0)$, along with an $(k+1) \times k$ upper Hessenberg matrix $\bar{H}_k$. Defining the $n \times k$ matrix $V_k = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_k\end{bmatrix}$ and H_k by deleting the last row of $\bar{H}_k$, it can be shown that

$$V_k^T A V_k = H_k. \tag{6.2}$$

Note that since A is symmetric, it can be seen from (6.2) that $H_k^T = H_k$. Furthermore, since H_k is symmetric and has all-zero entries below the subdiagonal by construction, it follows that it is tridiagonal. As such, it is standard notation to denote $\bar{H}_k$ as $\bar{T}_k$, whose entries we can define as

$$\bar{T}_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{bmatrix}.$$

Similarly, we identify T_k with H_k by

$$\bar{T}_k = \begin{bmatrix} T_k \\ \beta_{k+1}\mathbf{e}_k^T, \end{bmatrix}, \quad T_k = \begin{bmatrix} T_{k-1} & \beta_k \mathbf{e}_{k-1} \\ \beta_k \mathbf{e}_{k-1}^T & \alpha_k \end{bmatrix}$$

where $\mathbf{e}_k$ is the k-th unit vector. A being symmetric gives rise to the symmetric *Lanczos algorithm* [Saa03, Alg. 6.15], a simplification of Arnoldi's procedure. The symmetry of T_k leads to the three-term recurrence

$$\beta_{k+1}\mathbf{v}_{k+1} = A\mathbf{v}_k - \alpha_k \mathbf{v}_k - \beta_k \mathbf{v}_{k-1}, \tag{6.3}$$

where $\alpha_k = \mathbf{v}_k^T A \mathbf{v}_k$, $\beta_k = 1/\|\mathbf{v}_k\|_2$, and the initial vectors are $\mathbf{v}_0 = \mathbf{0}$ and $\mathbf{v}_1 = \mathbf{r}_0$. Equation (6.3) can be written in matrix form as

$$AV_k = V_{k+1}\bar{T}_k,$$

and by using that the Arnoldi's algorithm is designed to stop when $\beta_{k+1} = 0$, we get

$$AV_k = V_k T_k.$$

In an iteration k, Krylov subspace methods seek to find the best estimate of the solution of (6.1) in $\mathcal{K}_k(A, \mathbf{r}_0)$, in some sense. Any vector of the subspace $\mathcal{K}_k(A, \mathbf{r}_0)$ can be written

$$\mathbf{x} = V_k \mathbf{y},$$

with $y \in \mathbb{R}^k$. Defining the residual of $\mathbf{x}$ as $\mathbf{r} = \mathbf{b} - A\mathbf{x}$, we have

$$\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{b} - AV_k\mathbf{y} = \beta_1\mathbf{v}_1 - V_{k+1}\bar{T}_k\mathbf{y} = V_{k+1}(\beta_1\mathbf{e}_1 - \bar{T}_k\mathbf{y}).$$

Furthermore, as the column vectors of V_{k+1} are orthonormal,

$$\|\mathbf{r}\|_2 = \left\|\beta_1\mathbf{e}_1 - \bar{T}_k\mathbf{y}\right\|_2.$$

In the MINRES method $\|\mathbf{r}\|_2$ in the equation above is minimised in each iteration k, hence its name. The iterates can then be written

$$\begin{aligned}\mathbf{x}_k &= V_k\mathbf{y}_k, \quad \text{where} \\ \mathbf{y}_k &= \arg\min_{\mathbf{y}\in\mathbb{R}^k} \left\|\beta_1\mathbf{e}_1 - \bar{T}_k\mathbf{y}\right\|_2^2.\end{aligned}$$

The normal equation for $\mathbf{y}_k$ is

$$\bar{T}_k^T\bar{T}_k\mathbf{y}_k = \bar{T}_k^T\beta_1\mathbf{e}_1, \tag{6.4}$$

which MINRES uses QR decomposition of $\bar{T}_k$ in order to solve. In general, the QR decomposition of the rectangular matrix $\bar{T}_k$ is given by

$$\bar{T}_k = \tilde{Q}_k\begin{bmatrix}R_k \\ \mathbf{0}^T\end{bmatrix}$$

for an upper triangular matrix R_k and orthogonal matrix $\tilde{Q}_k$. $\tilde{Q}_k^T$ will occur several times in what follows, so we define $Q_k = \tilde{Q}_k^T$. This gives the QR decomposition of $\bar{T}_k$ the form

$$Q_k\bar{T}_k = \begin{bmatrix}R_k \\ \mathbf{0}^T\end{bmatrix} = \begin{bmatrix}
\gamma_1^{(1)} & \delta_2^{(1)} & \varepsilon_3^{(1)} & & & & \\
 & \gamma_2^{(2)} & \delta_3^{(2)} & \varepsilon_4^{(1)} & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \varepsilon_k^{(1)} & \\
 & & & & \ddots & \delta_k^{(2)} & \\
 & & & & & \gamma_k^{(2)} & \\
 & & & & & 0 &
\end{bmatrix}. \tag{6.5}$$

Using the above equation, (6.4) becomes

$$\begin{bmatrix}R_k^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}R_k \\ \mathbf{0}^T\end{bmatrix}\mathbf{y}_k = \begin{bmatrix}R_k^T & \mathbf{0}\end{bmatrix}Q_k\beta_1\mathbf{e}_1,$$

which formulated as a least-squares problem becomes

$$\mathbf{y}_k = \arg\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \begin{bmatrix} \mathbf{t}_k \\ \phi_k \end{bmatrix} - \begin{bmatrix} R_k \\ \mathbf{0}^T \end{bmatrix} \mathbf{y}_k \right\|_2^2, \quad \begin{bmatrix} \mathbf{t}_k \\ \phi_k \end{bmatrix} = Q_k \beta_1 \mathbf{e}_1.$$

Here we define $\mathbf{t}_k = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_k \end{bmatrix}^T$. In order to perform the QR decomposition, let $Q_k = Q_{k,k+1} \cdots Q_{2,3} Q_{1,2}$ be the product of $(k+1) \times (k+1)$ Householder reflections [QSS10, Sec. 5.6], used to eliminate the sub diagonal of $\bar{T}_k$ in order to find the values of the entries for R_k in (6.5). Defining

$$\rho_k = \sqrt{(\gamma_k^{(1)})^2 + \beta_{k+1}^2}, \quad c_k = \frac{\gamma_k^{(1)}}{\rho_k}, \quad s_k = \frac{\beta_{k+1}}{\rho_k},$$

the action of $Q_{k,k+1}$ applied to $\bar{T}_k$ and $\beta_1 \mathbf{e}_1$ is given by

$$\begin{bmatrix} \gamma_k^{(2)} & \delta_{k+1}^{(2)} & \varepsilon_{k+1}^{(1)} & \tau_k \\ 0 & \gamma_{k+1}^{(1)} & \varepsilon_{k+2}^{(1)} & \phi_k \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \begin{bmatrix} \gamma_k^{(1)} & \delta_{k+1}^{(1)} & 0 & \phi_{k-1} \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} & 0 \end{bmatrix},$$

where the values in the second matrix on the right-hand side are known from the last iteration. In practice, however, we will compute c_k, s_k and $\gamma_k^{(2)}$ from the more stable implementation stated in Algorithm 4. That is, we use $c_k, s_k, \gamma_k^{(2)} \leftarrow$

Algorithm 4 Stable computation of the orthogonal transformation related to the Householder reflections.

```
1:  procedure SymOrtho(a, b)
2:      if b = 0 then
3:          s ← 0, r ← |a|
4:          if a = 0 then
5:              c ← 1
6:          else
7:              c ← sign(a)
8:          end if
9:      else if a = 0 then
10:         c ← 0, s ← sign(b), r ← |b|
11:     else if |b| = |a| then
12:         τ = a/b, s = sign(b)/√(1 + τ²), c = sτ, r = b/s
13:     else if |a| = |b| then
14:         τ = b/a, c = sign(a)/√(1 + τ²), s = cτ, r = a/c
15:     end if
16:     return c, s, r
17: end procedure
```

SymOrtho$(\gamma_k^{(1)}, \beta_{k+1})$.

Defining the matrix $D_k = V_k R_k^{-1}$ with columns $D_k = \begin{bmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_k \end{bmatrix}$, it can be shown that

$$\gamma_k^{(2)} \mathbf{d}_k = \mathbf{v}_k - \delta_k^{(2)} \mathbf{d}_{k-1} - \varepsilon_k^{(1)} \mathbf{d}_{k-2}.$$

Finally, the iterate $\mathbf{x}_k$ can then be written

$$\mathbf{x}_k = V_k \mathbf{y}_k = V_k R_k^{-1} \mathbf{t}_k = D_k \begin{bmatrix} \mathbf{t}_{k-1} \\ \tau_k \end{bmatrix} = \begin{bmatrix} D_{k-1} & \mathbf{d}_k \end{bmatrix} \begin{bmatrix} \mathbf{t}_{k-1} \\ \tau_k \end{bmatrix} = \mathbf{x}_{k-1} + \tau_k \mathbf{d}_k.$$

In a practical implementation it is usually recommended to apply some sort of preconditioning technique to MINRES, which we will do next.

6.1.2 Preconditioned formulation

We consider a symmetric and positive definite preconditioning M, which will be applied as a two-sided preconditioner in order to preserve the symmetry required by the Lanczos algorithm [CS06, p. 62]. Since M is symmetric, its eigendecomposition is given by $M = VDV^T$, with D being diagonal. The square root is given as $M^{1/2} = VD^{1/2}V^T$, which exists due to the positive definiteness of M. Furthermore, from its eigendecomposition it is trivial to see that $M^{1/2}$ is also symmetric and positive definite. Next, define $\tilde{A} = M^{-1/2} A M^{1/2}$ and $\tilde{\mathbf{b}} = M^{-1/2}\mathbf{b}$, such that the new linear system

$$\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \quad M^{1/2}\mathbf{x} = \tilde{\mathbf{x}} \tag{6.6}$$

is equivalent to (6.1).

To derive the preconditioned Lanczos algorithm, define new vectors

$$\mathbf{z}_k = \tilde{\beta}_k M^{1/2} \tilde{\mathbf{v}}_k, \quad \mathbf{q}_k = \tilde{\beta}_k M^{-1/2} \tilde{\mathbf{v}}_k,$$

meaning $M\mathbf{q}_k = \mathbf{z}_k$. Here $\tilde{\mathbf{v}}_k$ are the orthonormal Lanczos vectors, and as before we set $\tilde{\mathbf{v}}_0 = \mathbf{0}$ and $\tilde{\beta}_1 \tilde{\mathbf{v}}_1 = \tilde{\mathbf{b}} - \tilde{A}\tilde{\mathbf{x}}_0$. As $\|\tilde{\mathbf{v}}_k\|_2 = 1$ by construction,

$$\tilde{\beta}_k^2 = \left\|\tilde{\beta}_k \tilde{\mathbf{v}}_k\right\|_2^2 = \|M^{-1/2}\mathbf{z}_k\|_2^2 = (M^{-1/2}\mathbf{z}_k)^T M^{-1/2}\mathbf{z}_k = \mathbf{z}_k^T M^{-1} \mathbf{z}_k = \mathbf{z}_k^T \mathbf{q}_k,$$

meaning that $\tilde{\beta}_k = \sqrt{\mathbf{z}_k^T \mathbf{q}_k}$. Similarly, the expression for $\tilde{\alpha}_k$ becomes

$$\tilde{\alpha}_k = \tilde{\mathbf{v}}_k^T \tilde{A} \tilde{\mathbf{v}}_k = (M^{-1/2}\tilde{\mathbf{v}}_k)^T A M^{-1/2} \tilde{\mathbf{v}}_k = (\frac{1}{\tilde{\beta}_k}\mathbf{q}_k)^T A \frac{1}{\tilde{\beta}_k}\mathbf{q}_k = \frac{1}{\tilde{\beta}_k^2}\mathbf{q}_k^T A \mathbf{q}_k,$$

and as in (6.3), three-term recurrence in the Lanczos algorithm for (6.6) is

$$\tilde{\beta}_{k+1}\tilde{\mathbf{v}}_{k+1} = \tilde{A}\tilde{\mathbf{v}}_k - \tilde{\alpha}_k \tilde{\mathbf{v}}_k - \tilde{\beta}_k \tilde{\mathbf{v}}_{k-1}. \tag{6.7}$$

Finally, multiplying (6.7) with $M^{1/2}$ from the left and writing $\tilde{\mathbf{v}}_k$ in terms of $\mathbf{z}_k$ and $\mathbf{q}_k$, (6.7) becomes

$$\mathbf{z}_{k+1} = \frac{1}{\tilde{\beta}_k} A\mathbf{q}_k - \frac{\tilde{\alpha}_k}{\tilde{\beta}_k}\mathbf{z}_k - \frac{\tilde{\beta}_k}{\tilde{\beta}_{k-1}}\mathbf{z}_{k-1}.$$

Note that in order compute $\mathbf{q}_k$, we have to solve the system $M\mathbf{q}_k = \mathbf{z}_k$ in each iteration.

The next step would now be to perform the QR factorisation of the tridiagonal $(k+1) \times k$ upper Hessenberg matrix arising from the Lanczos algorithm. Since this is equivalent to the non-preconditioned case considered previously, we omit the details. The result is

$$\tilde{\gamma}_k^{(2)} \tilde{\mathbf{d}}_k = \tilde{\mathbf{v}}_k - \tilde{\delta}_k^{(2)} \tilde{\mathbf{d}}_{k-1} - \tilde{\varepsilon}_k^{(1)} \tilde{\mathbf{d}}_{k-2}, \quad \tilde{\mathbf{x}}_k = \tilde{\mathbf{x}}_{k-1} + \tilde{\tau}_k \tilde{\mathbf{x}}_k,$$

and again multiplying the above two equations with $M^{1/2}$ from the left and writing $\tilde{\mathbf{v}}_k$ in terms of $\mathbf{z}_k$ and $\mathbf{q}_k$, we arrive at

$$\tilde{\gamma}_k^{(2)} \mathbf{d}_k = \mathbf{q}_k - \tilde{\delta}_k^{(2)} \mathbf{d}_{k-1} - \tilde{\varepsilon}_k^{(1)} \mathbf{d}_{k-2}, \quad \mathbf{x}_k = M^{-1/2} \tilde{\mathbf{x}}_k = \mathbf{x}_{k-1} + \tilde{\tau}_k \mathbf{x}_k.$$

Here $\mathbf{d}_k = M^{-1/2} \tilde{\mathbf{d}}_k$, and $\mathbf{x}_k$ are the iterates from the original problem.

We are now ready to present the preconditioned MINRES algorithm used, and it can be seen in Algorithm 5. When implementing iterative algorithms in practice, it is generally not preferable to store all the vectors previously calculated, so in our final algorithm we have omitted the subscript k. We will also like to elaborate on

Algorithm 5 A storage efficient implementation of the preconditioned MINRES algorithm with initial guess $\mathbf{x}_0$. The "$\leftrightarrow$" denotes that the content of the variables should be swapped. $\phi = \|\mathbf{b} - A\mathbf{x}\|_2$ is the 2-norm of the residual.

Require: $\epsilon > 0$, $k_{\max} \in \mathbb{Z}$
procedure MINRES($A, \mathbf{b}, M, \mathbf{x}_0$)
  $\mathbf{x} \leftarrow \mathbf{x}_0$, $\mathbf{z} \leftarrow \mathbf{b} - A\mathbf{x}$, $\mathbf{q} \leftarrow M^{-1}\mathbf{z}$, $\tilde{\beta}_{\text{old}} \leftarrow 1$, $\tilde{\beta} \leftarrow \sqrt{\mathbf{z}^T \mathbf{q}}$, $\tilde{\delta}^{(1)} \leftarrow 0$
  $\tilde{c}_{\text{old}} \leftarrow -1$, $\tilde{s}_{\text{old}} \leftarrow 0$, $\phi_0 \leftarrow \|\mathbf{z}\|_2$, $\phi \leftarrow \phi_0$, $\varepsilon \leftarrow 0$, $k \leftarrow 0$
  while no stopping criterion is true **and** $k < k_{\max}$ **do**
    $\mathbf{p} \leftarrow A\mathbf{q}$
    $\tilde{\alpha} \leftarrow \mathbf{q}^T \mathbf{p} / \tilde{\beta}^2$
    $\mathbf{z}_{\text{old}} \leftarrow \frac{1}{\tilde{\beta}} \mathbf{p} - \frac{\tilde{\alpha}}{\tilde{\beta}} \mathbf{z} - \frac{\tilde{\beta}}{\tilde{\beta}_{\text{old}}} \mathbf{z}_{\text{old}}$, $\mathbf{z} \leftrightarrow \mathbf{z}_{\text{old}}$
    $\mathbf{q}_{\text{old}} \leftarrow M^{-1}\mathbf{z}$, $\mathbf{q} \leftrightarrow \mathbf{q}_{\text{old}}$ ▷ Preconditioning step
    $\tilde{\beta}_{\text{old}} \leftarrow \sqrt{\mathbf{q}^T \mathbf{z}}$, $\tilde{\beta} \leftrightarrow \tilde{\beta}_{\text{old}}$
    $\tilde{\delta}^{(2)} \leftarrow \tilde{c}_{\text{old}} \tilde{\delta}^{(1)} + \tilde{s}_{\text{old}} \tilde{\alpha}$, $\tilde{\gamma}^{(1)} \leftarrow \tilde{s}_{\text{old}} \tilde{\delta}^{(1)} - \tilde{c}_{\text{old}} \tilde{\alpha}$
    $\tilde{c}, \tilde{s}, \tilde{\gamma}^{(2)} \leftarrow \textsc{SymOrtho}(\tilde{\gamma}^{(1)}, \tilde{\beta})$
    $\tilde{\tau} \leftarrow \tilde{c}\phi$, $\phi \leftarrow \tilde{s}\phi$
    if $\tilde{\gamma}^{(2)} \neq 0$ **then**
      $\mathbf{d}_{\text{old}} \leftarrow \frac{1}{\tilde{\gamma}^{(2)}} \left(\frac{1}{\tilde{\beta}_{\text{old}}} \mathbf{q}_{\text{old}} - \tilde{\delta}^{(2)} \mathbf{d} - \tilde{\varepsilon} \mathbf{d}_{\text{old}} \right)$, $\mathbf{d} \leftrightarrow \mathbf{d}_{\text{old}}$
      $\mathbf{x} \leftarrow \mathbf{x} + \tilde{\tau}\mathbf{d}$
    end if
    $\varepsilon \leftarrow \tilde{s}_{\text{old}} \tilde{\beta}$, $\tilde{\delta}^{(1)} \leftarrow -\tilde{c}_{\text{old}} \tilde{\beta}$
    $\tilde{c}_{\text{old}} \leftarrow \tilde{c}$, $\tilde{s}_{\text{old}} \leftarrow \tilde{s}$
    $k \leftarrow k + 1$
  end while
  return x
end procedure

the double arrow notation "$\leftrightarrow$" used, which denotes that we swap the content of

two variables. In several places of Algorithm 5 we use variables from the previous iterations, denoted by the subscript "old". Some of these variables are vectors, which if implemented naively would need to be copied to another location in order to not be overwritten by the next iteration. In most modern languages, including Python used in this report, vectors are implemented with pointers. The swapping of two vectors can therefore be implemented efficiently by simply reassigning the pointers pointing to the data in question, avoiding any unnecessary copying of data in memory.

6.1.3 MINRES for the state equations

Finding a good preconditioner is arguably one of the most difficult aspects of solving sparse linear systems. For the pure Stokes equations, a commonly used preconditioning matrix is obtained from the Galerkin FEM discretisation of the variational form

$$\int_\Omega (\nabla \mathbf{u} : \nabla \mathbf{v} + pq)\,,$$

which we recognise as

$$a_0(\mathbf{u}, \mathbf{v}) + (p, q)_Q.$$

The preconditioner is derived in [BGL05, Sec. 10.1.1], and although not directly applicable to the Darcy-Stokes equations (2.6), a natural extension is to define

$$m_\alpha(\mathbf{u}, p, \mathbf{v}, q) = a_\alpha(\mathbf{u}, \mathbf{v}) + (p, q)_Q \tag{6.8}$$

as the variational form of a possible preconditioner to the linear system (4.4). Assuming we have chosen $V_h = \operatorname{span}\{\boldsymbol{\varphi}_1^V, \ldots, \boldsymbol{\varphi}_n^V\}$ and $Q_h = \operatorname{span}\{\phi_1^Q, \ldots, \phi_m^Q\}$ in (4.1) to generate S in (4.6), the preconditioning matrix arising from (6.8) takes the form

$$M_\alpha = \begin{bmatrix} A_\alpha & 0 \\ 0 & P \end{bmatrix}. \tag{6.9}$$

Here

$$A_\alpha = [a_\alpha(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)], \quad P = [(\phi_k, \phi_l)_Q],$$

with $i, j = 1, 2, \ldots, n$ and $k, l = 1, 2, \ldots, m$. It is trivial to see that M_α is block symmetric and positive definite, which we recall is a requirement for Algorithm 5.

As the choice of M_α is not founded in any theoretical results, we will instead compare the MINRES method numerically, with and without preconditioning. The preconditioner M in Algorithm 5 will naturally be M_α from (6.9) in the preconditioned case, while we use the identity matrix in the non-preconditioned case. Additionally, we want to investigate how the MINRES method behaves in practice, so to this end we apply the method to the resulting designs from the diffuser example from Subsection 4.6.1. More precisely, we apply MINRES to the linear system (4.4) for both the CR and the TH elements using the designs from Figure 4.8a and 4.8c, respectively.

We use the HYPRE algebraic multigrid method to perform the preconditioning step in Algorithm 5, and observe how the relative residuals of *the linear system*

decreases with the iterations. This residual is denoted ϕ in Algorithm 5, and we stop the iterations when either $\phi/\phi_0 < 1 \times 10^{-15}$ or $k > k_{\max} = 2000$.

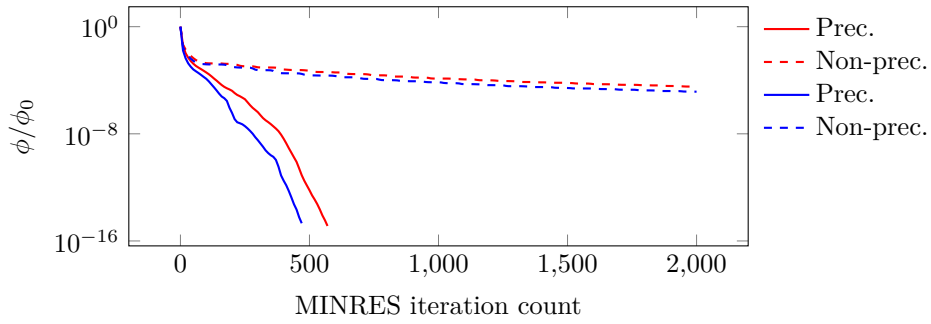


Figure 6.1: The relative residual of MINRES applied to (4.4) for the designs in Figure 4.8a and 4.8c, using $\mathbf{x}_0 = \mathbf{0}$ in Algorithm 5. The red graphs are the CR elements and the blue graphs are the TH elements.

The results are can be seen in Figure 6.1, and it is clear that it is not only preferable, but in practice strictly necessary to use preconditioning for both element types if we want to solve (4.4) accurately.

6.2 Behaviour of the residual estimates

An advantage of MINRES being an iterative method means that we can solve (4.4) down to some preferred accuracy, potentially saving computational effort by reducing the number of iterations. The question is *how* accurate is sufficiently accurate, and to assess this question we will use the theory of residuals developed in Section 4.3. By observing the residual norms $\|\mathbf{r}^{mo}_{h/2}\|_V$ and $\|r^{ma}_{h/2}\|_Q$ from (4.10) during the MINRES iterations, we can detect when the discretisation error dominates the error associated with the linear system as the residuals should no longer change. As such, solving (4.4) down to any further accuracy will not result in the numerical solution $(\mathbf{u}_h, p_h)$ approximating the solution to the continuous equation (2.6) any more accurately, meaning we can stop the MINRES iterations.

We recall from Figure 4.5 in Section 4.4 that for the CR elements there was still a certain discretisation error after refining the mesh once. To see how this affects the behaviour of the residuals during MINRES iterations, we include both the residuals evaluated on $\mathcal{T}_{h/2}$ and $\mathcal{T}_{h/4}$ for the CR elements in this section. The relative residuals for the one time refined mesh will therefore be denoted

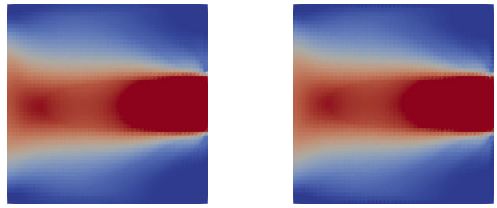
$$\eta^{mo}_{h/2} \quad \text{and} \quad \eta^{ma}_{h/2}$$

for η^{mo} and η^{ma} in (4.11), respectively. Similarly, for the two times refined mesh we write

$$\eta^{mo}_{h/4} = \frac{\|\mathbf{r}^{mo}_{h/4}\|_V}{\|\mathbf{g}\|_{L^2(\partial\Omega)}} \quad \text{and} \quad \eta^{ma}_{h/4} = \frac{\|r^{ma}_{h/4}\|_Q}{\|\mathbf{g}\|_{L^2(\partial\Omega)}}$$

in order to separate the two.

As a starting point, consider the diffuser benchmark on the 50×50 mesh presented in the previous section. In order to understand how the residuals behave in the different states of the optimisation algorithm, we look at an iterate in the early stages of the OC method and one towards the end. More precisely, in order to see how the residuals behave early on, we perform the 50×50 diffuser experiment using the direct solver. However, now we perform 100 MINRES iterations at OC iteration 3, whose corresponding design $\rho_{h,3}$ is depicted in Figure 6.2 for the two element types. The residual norms can be seen in Figure 6.3, where we have used



(a) CR **(b)** TH

Figure 6.2: The design of the 50×50 diffuser example in iteration 3 of the OC method.

$\mathbf{x}_0 = \mathbf{0}$ in Algorithm 5.

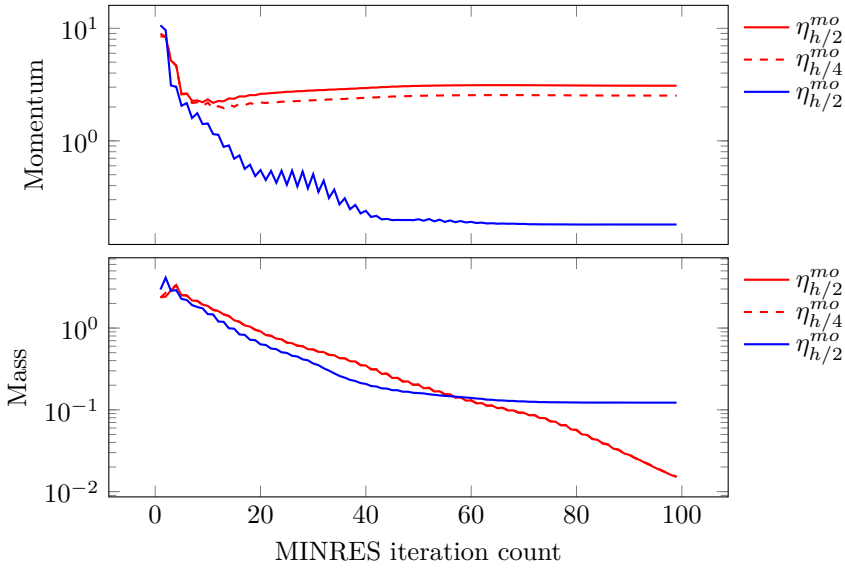


Figure 6.3: Relative residual norms as a function of MINRES iterations for iteration 3 of the OC method for the 50×50 diffuser example. The red plots corresponds to the CR elements while the blue plot correspond to the TH elements. Zero is used as the initial guess for MINRES.

The first thing to note from Figure 6.3 is that for $\eta^{mo}_{h/2}$ the discretisation error

seems to dominate after about 80 MINRES iterations for both the CR and TH elements. The TH elements converge to a value smaller than that for the CR elements, which is expected as the TH elements using higher order basis functions for $\mathbf{u}_h$ and p_h than CR, both appearing in the definition (4.10) of $\mathbf{r}^{mo}_{h/2}$. For $\eta^{ma}_{h/2}$ in Figure 6.3, it seems to be the same tendency with graph for the TH elements, flattening out after about 80 iterations. The CR elements, on the other hand, seems to decrease more or less steadily. This is due to the fact that for the CR elements, the divergence error is within machine precision for a solution of (4.1), which was observed during the convergence test. Hence, we can generally expect that the mass residual norm will continue to decrease as the MINRES iterates approach the exact solution of (4.4). Comparing $\eta^{mo}_{h/2}$ and $\eta^{mo}_{h/4}$ for the CR elements, they seemingly behave very similarly, only differing by a constant after about 10 MINRES iterations. For $\eta^{ma}_{h/2}$ and $\eta^{ma}_{h/4}$, the plots completely coincide.

Instead of using zero as the initial guess for MINRES, we can instead try to use the solution of (4.4) in the previous iteration. The matrix A_α in (4.4) depends on the design $\rho_{h,i}$ in OC iteration i, as $\alpha = \alpha(\rho_{h,i})$. It is reasonable to assume that if the previous design $\rho_{h,i} \approx \rho_{h,i-1}$, then $(\mathbf{u}_{h,i}, p_{h,i}) \approx (\mathbf{u}_{h,i-1}, p_{h,i-1})$ as well.

Seeing how he residuals change during the MINRES using zero as the initial guess for OC iteration 3, we redo the experiment above, but now using the solution $(\mathbf{u}_{h,2}, p_{h,2})$ as the initial guess for MINRES. The results can be seen in Figure 6.4, and by comparison with Figure 6.3, appear to be similar to the case using zero as the initial guess in MINRES. Again, the residuals seems to become constant after

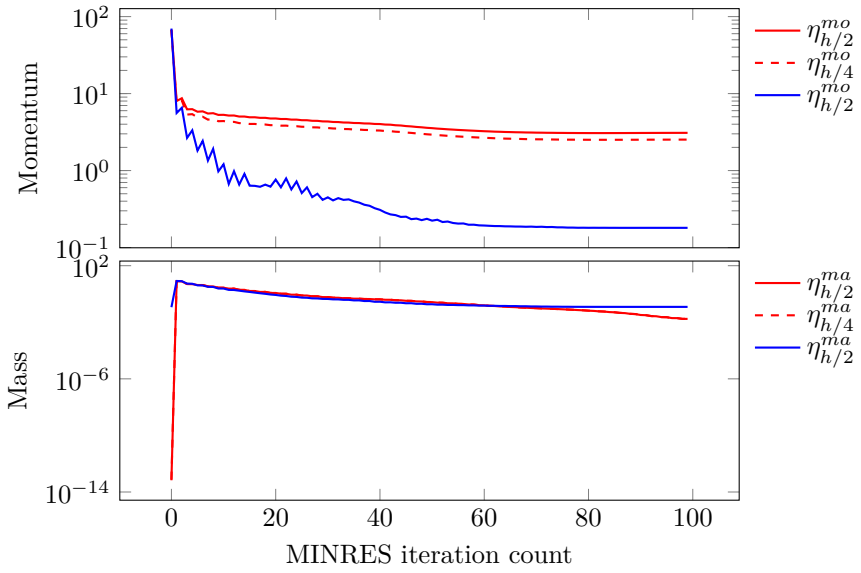


Figure 6.4: Relative residual norms as a function of MINRES iterations for iteration 3 of the OC method for the 50×50 diffuser example. The red plots corresponds to the CR elements while the blue plot correspond to the TH elements. The previous iterate is used as the initial guess for MINRES.

about 80 MINRES iterations, except for the mass residual using CR elements, which continue to decrease. Furthermore, the residuals in Figure 6.4 seems to converge to the same values as in Figure 6.3, which is expected as MINRES should converge to the solution of (4.4) regardless of the initial guess. For the CR elements, $\eta^{ma}_{h/2} \sim 10^{-14}$ and $\eta^{ma}_{h/4} \sim 10^{-14}$ in iteration 0, which is due to the initial guess being $(\mathbf{u}_{h,2}, p_{h,2})$, which was computed with a direct solver. As the matrix B in (4.4) is independent of the porosity distribution, it is constant in each iteration, so $\mathbf{u}_{h,2}$ accurately solves the second equation in (4.1) also for the new porosity distribution $\rho_{h,3}$.

The fact that using the previous iterate did not yield any faster convergence of the residuals in MINRES for OC iteration 3 is probably due to the fact that we are in the early stages of the optimisation algorithm, where we can expect the designs $\rho_{h,i}$ to change significantly in one iteration to the next. This means is we can not assume that $\rho_{h,i} \approx \rho_{h,i-1}$, and consequently we do not have $A_{\alpha(\rho_{h,i})} \approx A_{\alpha(\rho_{h,i-1})}$ in (4.4) either.

To this end, let us consider an iteration in the later stage of the OC method for the 50×50 diffuser example. We use a direct solver for solving the state equations in the OC method, but this time we additionally use the MINRES in the very last iteration, that is the iteration where the stopping criterion is satisfied. For the CR and TH elements we see from Table 4.1 that this corresponds to iteration 43 and 44, respectively. Using zero as the initial guess, the results Figure 6.5.

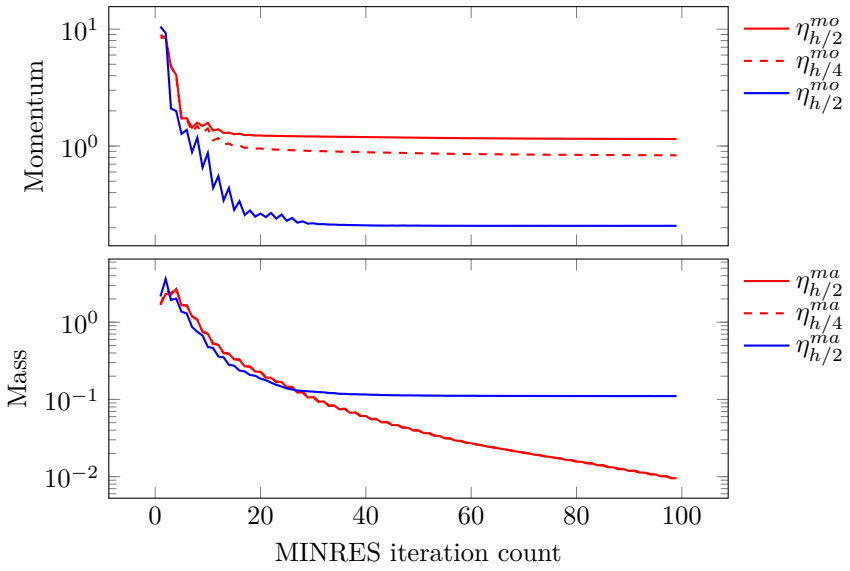


Figure 6.5: Relative residual norms as a function of MINRES iterations for iteration 3 of the OC method for the 50×50 diffuser example. The red plots corresponds to the CR elements while the blue plot correspond to the TH elements. Zero is used as the initial guess for MINRES.

The first thing to note in Figure 6.5 is when the residuals become constant. Apart from the mass residual in the case of the CR elements, which as before continue to decrease, the other graphs seems to become constant after about 40 MINRES iterations. This is significantly lower than for the OC iteration 3 considered above, which used about 80 iterations. One possible explanation for this difference in convergence could be the properties of ρ_h. The 0-1 design represent pure Stokes flow in the regions where $\rho_h = 1$ and zero velocity when $\rho_h = 0$. As the preconditioning matrix M_α was initially derived for the pure Stokes equations, it may be that M_α will be more efficient for the later iterations of the OC method where the state equations models regions of pure Stokes flow. The 0-1 property is not particularly prominent in the early stages of the optimisation algorithm, as Figure 6.2 illustrates.

Next, let us look at the case where the previous state is used as the initial guess in MINRES. Using the direct solver 100 MINRES iterations are performed in the very last OC iteration for both the CR and TH elements, and the result is seen in Figure 6.6. Comparing Figure 6.5 and 6.6, it is clear that the previous iterate

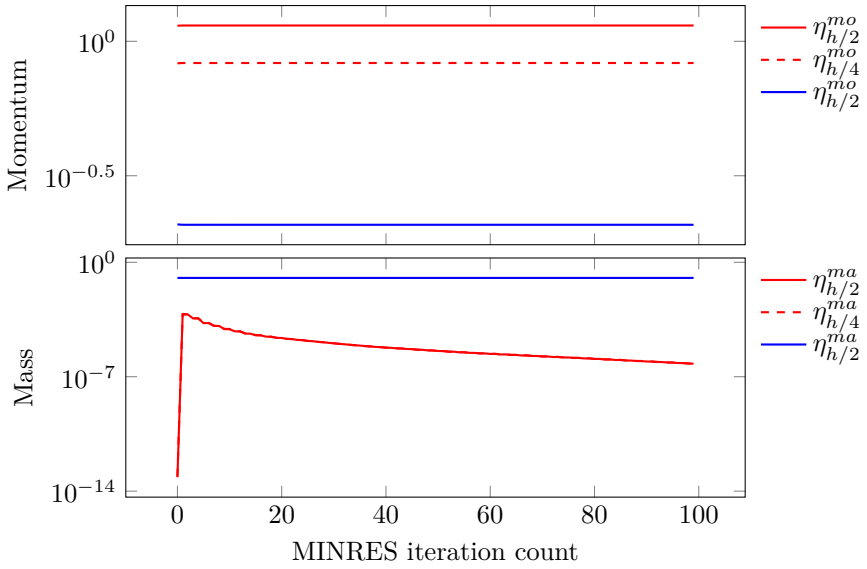


Figure 6.6: Relative residual norms as a function of MINRES iterations for the last iteration of the OC method for the 50×50 diffuser example. The red plots corresponds to the CR elements while the blue plots correspond to the TH elements. The previous iterate is used as the initial guess for MINRES.

is a good initial guess for MINRES in the later stages of the OC method. The residuals for both the CR and TH elements now converges already from the very first iteration, with the exception naturally being the mass residual for the CR elements.

The last question to be addressed is how the residuals depend on the mesh size of the problem. In addition to the 50×50 mesh, consider a 25×25 and a 100×100

mesh. We use iteration 3 of the OC method with the previous iterate as the initial guess for MINRES, and the sequence of refined grids for both CR and TH elements is depicted in Figure 6.7. We omit the case $\eta^{mo}_{h/4}$ and $\eta^{ma}_{h/4}$ for the CR elements due to their similar behaviour as $\eta^{mo}_{h/2}$ and $\eta^{ma}_{h/2}$. The first thing to note from Figure 6.7

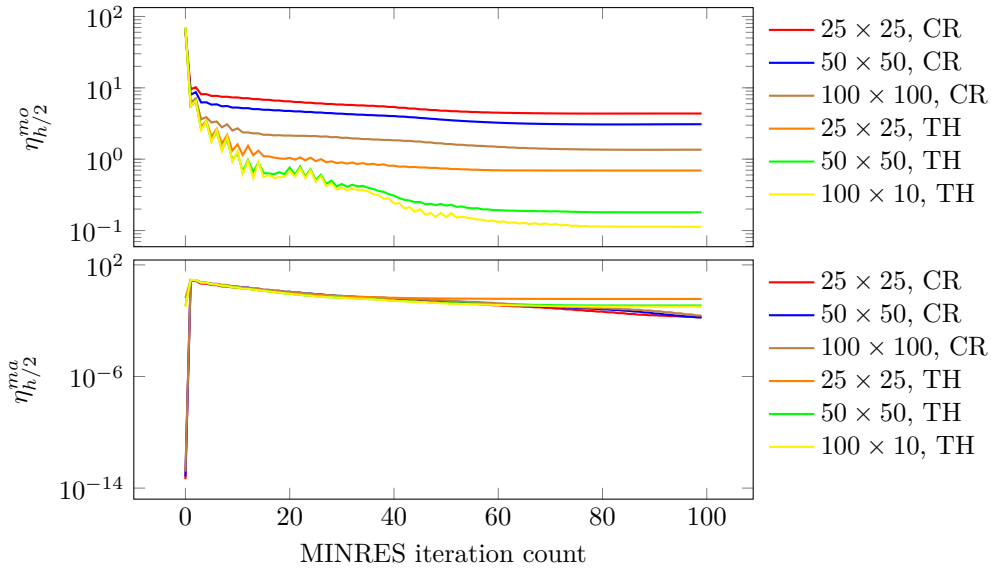


Figure 6.7: Relative residual norms as a function of MINRES iterations for iteration 3 of the OC method for the diffuser example with varying grid sizes. Zero is used as the initial guess for MINRES.

is that $\eta^{mo}_{h/2}$ generally decreases as the mesh becomes finer, which is reasonable as the discretisation error is expected to decrease. Furthermore, for the three different mesh sizes $\eta^{mo}_{h/2}$ seems to become constant after about 80 iterations for both the CR and TH elements, suggesting that convergence of the residuals depends rather weakly on the mesh size. Considering $\eta^{ma}_{h/2}$, the trend seems to be the same for the TH elements, that is $\eta^{ma}_{h/2}$ decreases with increasing mesh size and becoming constant after about 60 iterations. For the CR elements, the mesh size seems not to affect the convergence of $\eta^{ma}_{h/2}$, which continuously decreases at the same rate for the three mesh sizes considered.

6.3 Premature termination of MINRES

Considering the MINRES method in Algorithm 5, a possible stopping criterion could be to stop the iterations when $\|\mathbf{b} - A\mathbf{x}\|_2$ is smaller than some fixed tolerance. As we generally do not know how accurate the linear system (4.4) needs to be solved in each OC iteration, one option could be to solve the system to machine accuracy, which from the perspective of the OC method would be equivalent to using a direct

solver. With this approach however, we must expect to perform an unnecessarily large number of MINRES iterations, which is not desirable.

Instead we will compute the a posteriori residuals using the strategy formulated in Section 4.3 in order to propose a stopping criterion for MINRES. As we have both the mass and the momentum residuals, it is not readily obvious which one that is appropriate to use for stopping MINRES. We therefore let $\eta_k^{(s)}$ be a linear combination of the two residuals, defined by

$$\eta_k^{(s)} = s\eta_k^{mo} + (1-s)\eta_k^{ma} \tag{6.10}$$

for some $s \in [0, 1]$. Here k denotes a MINRES iteration, and we have omitted the subscript $h/2$ as we only consider residuals evaluated on a once uniformly refined mesh. From the previous section we saw that both residuals for the once and twice refined mesh behaved very similarly in terms of when becoming constant during the MINRES iterations. Based on this behaviour we therefore justify to consider only the one time refinement, meaning we will *not* consider $\eta_{h/4}^{mo}$ and $\eta_{h/4}^{ma}$ for either element type.

As the residual norms are expected to converge towards a constant, as seen in the previous section for all residual norms except η^{ma} for the CR elements, we define the relative change of the residual over one MINRES iteration

$$\tau_k^{(s)} = \frac{|\eta_k^{(s)} - \eta_{k-1}^{(s)}|}{\eta_k^{(s)}}. \tag{6.11}$$

We will stop the MINRES when the change in the relative residual norm between two consecutive iterations are sufficiently small, meaning when

$$\tau_k^{(s)} < \epsilon_\tau \tag{6.12}$$

for fixed tolerance ϵ_τ.

Having defined a stopping criterion, there are two questions we need to address. The first question is regarding what the value of the tolerance of ϵ_τ should be. As the direct solver solves the linear system (4.4) down to machine accuracy, this is what we will use as a reference. Ideally we want properties like the objective value and the residuals to coincide during iterations when using the direct solver and when using prematurely stopped MINRES, so by executing the algorithm for different values of ϵ_τ we can decide how small ϵ_τ needs to be for this to be the case.

The second question is what the value of s in (6.11) should be. Each value η_k^{mo} and η_k^{ma} requires the solution of a linear system, so if for instance it appears that only one of the residuals is sufficient in order for the OC method to converge to the desired solution, it may be the case that the other residual is unnecessary to compute. As it in Section 4.6 has proven itself to be the easiest among the benchmarks, we start by consider the 50×50 diffuser example when addressing these questions.

We will use the solution computed in the previous OC iteration as the initial guess for MINRES, as this was seen to reduce the necessary number of MINRES iterations significantly in the later stages of the OC method.

6.3.1 The diffuser

We will start by considering the CR elements. In this case η_k^{ma} continues to decrease with the MINRES iterations, so it is immediately clear that using $s = 0$ in (6.11) does not make sense. Looking at the Momentum plot in Figure 6.4 and 6.6 we see that $\eta_{h/2}^{mo}$ appears to converge to a constant, so this appear as a viable choice. Regarding the choice of ϵ_τ, we have chosen the three values 1×10^{-2}, 1×10^{-4} and 1×10^{-8}, and the results can be seen in Figure 6.8. In the plots we have included the

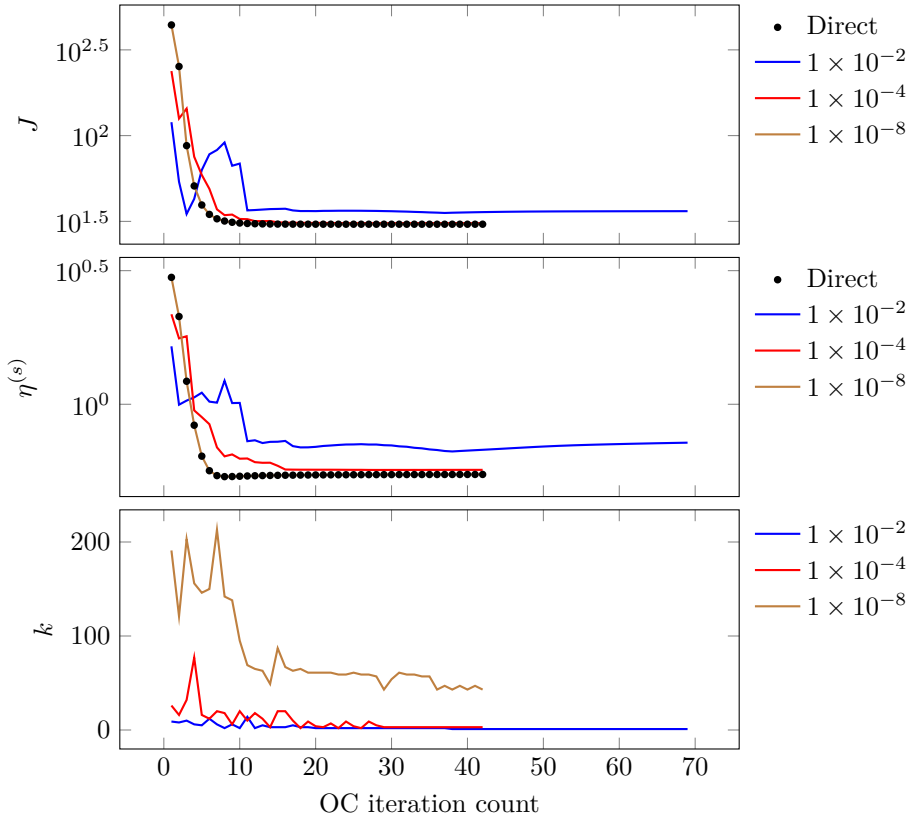


Figure 6.8: 50×50 diffuser example with the CR elements for three different values of ϵ_τ, and with $s = 1$ in (6.10). J is the objective value, $\eta^{(1)} = \eta^{mo}$ is the relative momentum residual norm and k is the number of MINRES iterations.

objective value, the residual and how the number of necessary MINRES iterations changes over the course of the OC iterations. Furthermore, we have included the results from using the direct solver for comparison.

First, considering $\epsilon_\tau = 1 \times 10^{-2}$ we see that the objective value J in the beginning deviates from the value obtained with direct solver, but appears to become fairly similar after about 10-15 iterations. For the residual norm $\eta^{(s)}$, however, $\epsilon_\tau = 1 \times 10^{-2}$ deviates noticeably from the direct solver. Despite being consider-

ably larger after about 5 iterations, we see that the OC method requires almost 70 iterations when using $\epsilon_\tau = 1 \times 10^{-2}$ as a stopping criterion.

The two other values $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-8}$ seems to yield more accurate results, where the latter in particular appears to coincide completely in the plots for J and η^{mo}. Looking at $\epsilon_\tau = 1 \times 10^{-4}$, the difference from the direct solver is most apparent in the first 20 iterations, and it seems to converge fairly accurately thenceforth. As the number of MINRES iterations k is significantly lower for $\epsilon_\tau = 1 \times 10^{-4}$ than for $\epsilon_\tau = 1 \times 10^{-8}$, the former may therefore be an enticing configuration despite not being completely coincident with the direct solver during the earlier iterations.

Next we consider the case when $s \in (0, 1)$ in (6.10), such that both $s\eta_k^{mo}$ and $(1 - s)\eta_k^{ma}$ are non-zero in η_k. Generally it is expedient to choose s such that η^{mo} and η^{ma} are of the same order, that is when η_k^{mo} and η_k^{ma} have converged. However, considering that η_k^{ma} will only decrease with the MINRES iterations, we can generally not expect this to be the case. More precisely, if $s\eta_k^{ma}$ is of the same order as $(1-s)\eta_k^{mo}$ in a MINRES iteration k, we can expect it to eventually become lower order for later iterations. It is not obvious how this will affect the stopping criterion for MINRES, so we will for simplicity use $s = 0.5$. We still consider the CR elements with the same three tolerances for ϵ_τ as found previously, and the results can be seen in Figure 6.9.

Qualitatively, the behaviour for the three values of ϵ_τ appears to be very similar as for the case $s = 1$. For $\epsilon_\tau = 1 \times 10^{-2}$, J seems to converge towards the value calculated using the direct solver, while for $\eta^{(s)}$ the values deviate substantially. Additionally, the OC method also uses several extra iterations in this case. Looking at $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-8}$, both the graphs for both J and $\eta^{(s)}$ behaves more similarly to the direct solver, with $\epsilon_\tau = 1 \times 10^{-4}$ converging for $\eta^{(s)}$ after about 25 iterations. Again, for $\epsilon_\tau = 1 \times 10^{-8}$, the results appear to coincide with the direct solver.

To more easily be able to compare the differences in the behaviour of the stopping criterion (6.12) for $s = 0.5$ and $s = 1$, we present their properties in Table 6.1. Considering $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-8}$ in particular, we note a significant

Table 6.1: Computational data the for the 50×50 diffuser example using CR elements.

s	ϵ_τ	OC iters.	J	$\eta^{(s)}$	k_{tot}
0.5	1×10^{-2}	56	35.05	3.90×10^{-1}	179
	1×10^{-4}	41	30.59	2.80×10^{-1}	454
	1×10^{-8}	42	30.44	2.73×10^{-1}	4384
1	1×10^{-2}	69	36.27	7.18×10^{-1}	175
	1×10^{-4}	42	30.87	5.68×10^{-1}	434
	1×10^{-8}	42	30.44	5.46×10^{-1}	3384

difference in the total number of MINRES iterations k_{tot} in Table 6.1 for both values of s. As our main goal should be to minimise the total number of MINRES iterations over the whole optimisation algorithm, we see that we pay a substantial

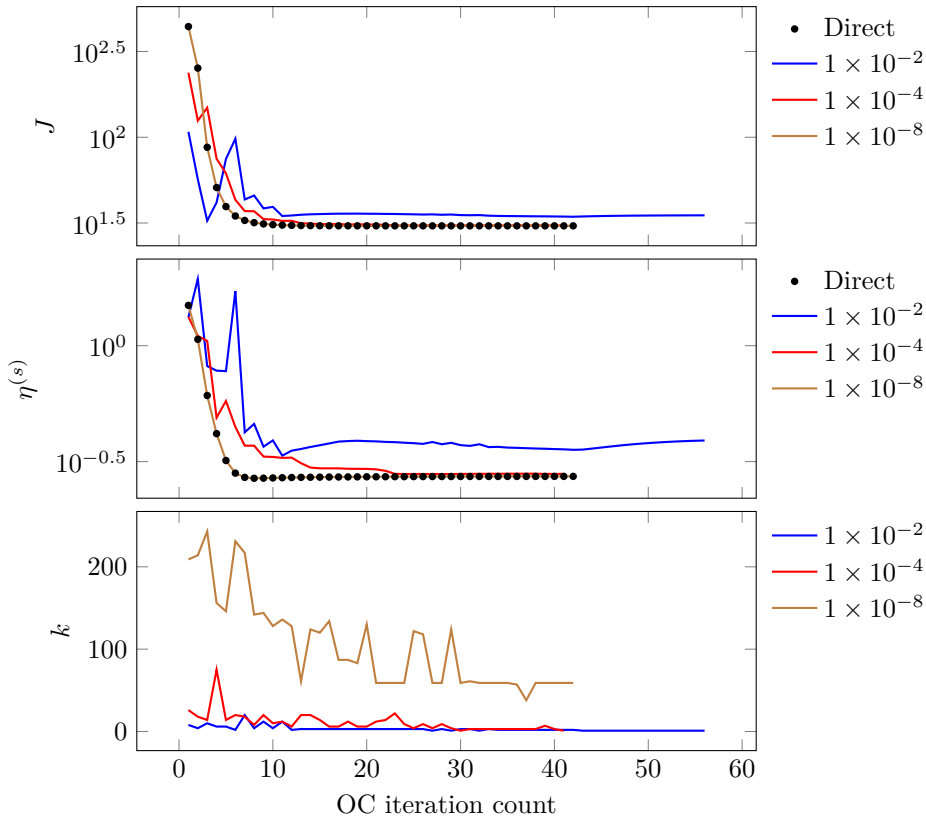


Figure 6.9: 50×50 diffuser example with the CR elements for three different values of ϵ_τ, and with $s = 0.5$ in (6.10). The objective value J, the relative momentum residual norm $\eta^{(0.5)}$ and the number of MINRES iterations k have been plotted for three different values of ϵ_τ.

price for the additional accuracy $\epsilon_\tau = 1 \times 10^{-8}$ offers over $\epsilon_\tau = 1 \times 10^{-4}$. Furthermore, we see that the objective values are very similar between the two cases, with the objective value differing with only 0.5 % and 1.4 % for $s = 0.5$ and $s = 1$, respectively. From these observations it appears that there is not much to gain from increasing the accuracy from $\epsilon_\tau = 1 \times 10^{-4}$ to $\epsilon_\tau = 1 \times 10^{-8}$ for this example, despite the value of the relative residual norm deviating somewhat from that of the direct solver in the first 20 iterations.

Comparing each ϵ_τ with respect to the two values of s, we see that the two stopping criteria yield similar values for the OC iterations and objective values. Looking at $\eta^{(s)}$, we see that the values for $\eta^{(0.5)}$ are about a half of $\eta^{(1)}$, which is due to η^{mo} being multiplied by 0.5 in (6.10) for the former. The tendency seems to be that the term $s\eta^{ma}$ does not contribute significantly to the value of $\eta^{(s)}$, indicating that for the stopping criterion, η^{mo} is the more important residual of the two. Lastly, we also see that when $s = 1$, the total number of MINRES

iterations is 1000 less for $\epsilon_\tau = 1 \times 10^{-8}$ than for $s = 0.5$, despite both the OC iterations and objective value being identical.

Having considered the CR elements, we perform the same experiment using the TH elements. We start by considering $s = 1$, and again choose three different values for ϵ_τ. We still use $\epsilon_\tau = 1 \times 10^{-2}$ and $\epsilon_\tau = 1 \times 10^{-4}$, however his time we found that $\epsilon_\tau = 1 \times 10^{-6}$ was sufficient in order to have the values for J and $\eta^{(s)}$ coincide with the values obtained with the direct solver.

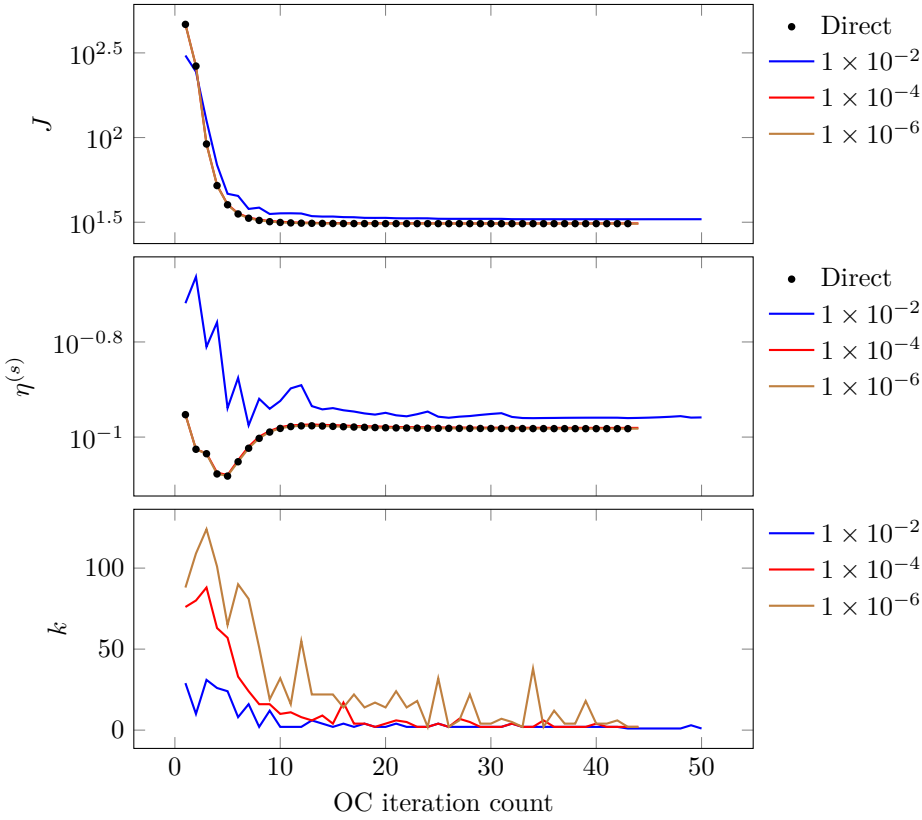


Figure 6.10: 50×50 diffuser example with the TH elements for three different values of ϵ_τ, and with $s = 1$ in (6.10). J is the objective value, η^{mo} is the relative momentum residual norm and k is the number of MINRES iterations.

Starting with $\epsilon_\tau = 1 \times 10^{-2}$, the trend seems to be similar to that for the CR elements. For J, although $\epsilon_\tau = 1 \times 10^{-2}$ resembles the direct solver, it becomes clear from the plot for $\eta^{(s)}$ that there is a significant difference. Again, $\epsilon_\tau = 1 \times 10^{-2}$ deviates for the first 20 iterations, while converging towards a value slightly larger than for the direct solver. Additionally, the optimisation algorithms needs to perform some additional iterations in order to satisfy the stopping criterion. Looking at $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-6}$, we note in particular that the former also seems to coincide almost completely with the direct solver for both J

and $\eta^{(s)}$, as opposed to the CR elements considered previously.

Turning our attention to the value of s, for TH elements we also want to consider the case $s < 1$. As discussed previously, we want to choose s such that $s\eta_k^{mo}$ and $(1-s)\eta_k^{ma}$ are of the same order in (6.10) in order to not have one of the terms dominating the other. We can use Figure 6.6 in order to address this question, as we can read off the values of the relative residual norms in the later iterations when the design has more or less settled. As the y-axes in Figure 6.6 are logarithmic, it is difficult to read off the values of the straight lines exactly. Instead we inform the reader that the exact values are 0.21 and 0.10 for η^{mo} and η^{ma}, respectively, which are clearly of the same order. This means that we can use $s = 0.5$ for $\eta_k^{(s)}$, and the results are seen in Figure 6.11.

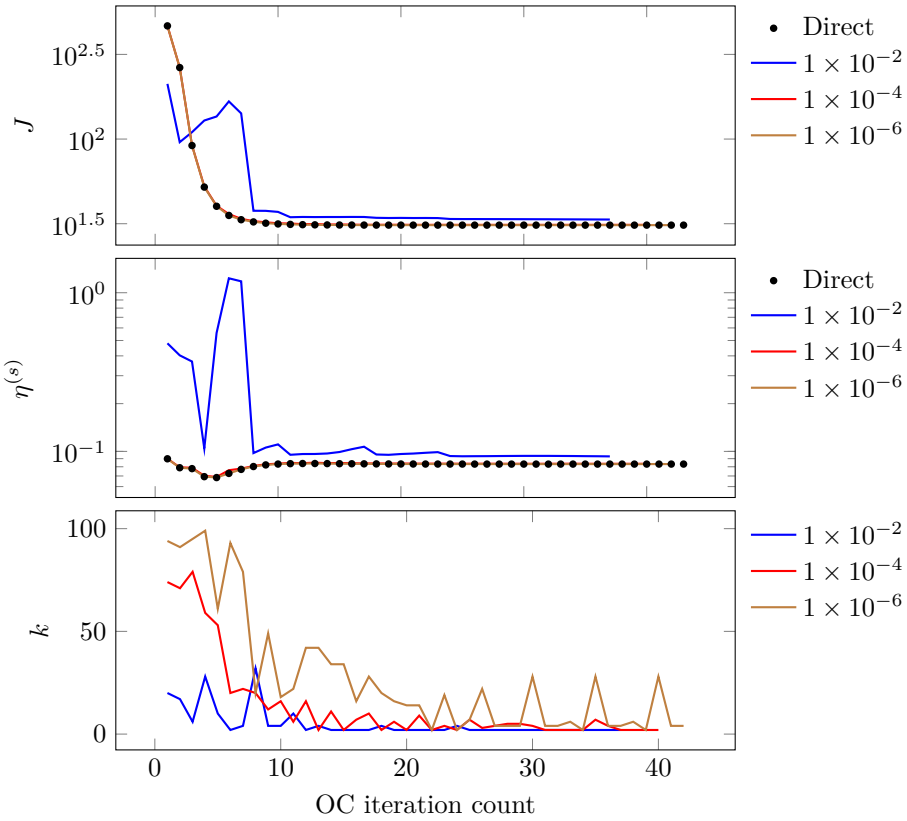


Figure 6.11: 50×50 diffuser example with the TH elements for three different values of ϵ_τ, and with $s = 0.5$ in (6.10). J is the objective value, η^{mo} is the relative momentum residual norm and k is the number of MINRES iterations.

Starting with J in Figure 6.11, we see that this time $\epsilon_\tau = 1 \times 10^{-2}$ deviates a fair amount from the direct solver for the first 6 iterations, but converges fairly accurately in the 7'th iteration and forward. The same is true for $\eta^{(s)}$, where apart

from some small zigzags the relative residual norm appears to converge towards a value slightly larger than that of the direct solver. Furthermore, note that $\epsilon_\tau = 1 \times 10^{-2}$ stop before the direct solver, which is opposite of what we have seen previously. For $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-6}$ are the same as when considering $\eta^{(s)}$, where both coincides very accurately with the direct solver.

To better be able to compare $\eta^{(1)}$ and $\eta^{(0.5)}$, we plot their main properties in Table 6.2. Looking at the larger of the three values for ϵ_τ, the objective value is

Table 6.2: Computational data the for the 50×50 diffuser example using TH elements.

s	ϵ_τ	OC iters.	J	$\eta^{(s)}$	k_{tot}
0.5	1×10^{-2}	37	33.48	9.30×10^{-2}	195
	1×10^{-4}	40	31.14	8.35×10^{-2}	562
	1×10^{-6}	42	31.02	8.32×10^{-2}	1169
1	1×10^{-2}	50	32.95	1.10×10^{-1}	250
	1×10^{-4}	44	31.12	1.05×10^{-1}	605
	1×10^{-6}	44	31.02	1.04×10^{-1}	1216

about 7.9 % higher for $s = 0.5$ and 6.2 % higher for $s = 1$ than that we get when using the direct solver, which we from Table 4.1 know is 31.02. For $\epsilon_\tau = 1 \times 10^{-4}$, the difference in the objective value from the direct solver is 0.4 % for $s = 0.5$ and 0.3 % for $s = 0.5$, which is considerably smaller. Additionally, when using $\epsilon_\tau = 1 \times 10^{-6}$, the values obtained for J coincide with the direct solver for both values of s. Unsurprisingly, this requires the algorithm to perform several extra MINRES iterations in total, which we from k_{tot} in Table 6.2 see is about twice the number of iterations used for $\epsilon_\tau = 1 \times 10^{-6}$.

Summarising what we have found so far, the most important observation for both the CR and TH elements is arguably that the behaviour of the stopping criterion $s = 0.5$ and $s = 1$ were fairly similar. Comparing the two values for s, we saw from Table 6.1 and 6.2 that among each tolerance ϵ_τ the objective value, relative residual norm and total number of MINRES iterations did not vary significantly. This suggests that using $s = 1$, such that $\eta_i^{(s)} = \eta_i^{mo}$, is sufficient as a stopping criterion, meaning we only need to solve the first of (4.10). However we emphasise that this will usually be the larger one of the two linear systems, as $\mathbf{r}_h^{mo} \in V_h$ and $r_h^{ma} \in Q_h$. This follows from the fact that the order of the basis functions for V_h will generally be higher than those for Q_h, which we recall was necessary in order for a given pair of elements to be stable. Nevertheless, it is certainly desirable to avoid having to compute r_h^{ma} if possible.

Concerning the value of ϵ_τ in (6.12), the experiments suggests that using 1×10^{-2} is too large for either type of the two types of elements considered, if we want the iterates obtained from using MINRES to solve the linear system (4.4) to resemble those obtained using the direct solver. For the TH elements it appears that 1×10^{-4} is sufficient, and certainly preferable over 1×10^{-6}. That is considering the latter doubles the number of MINRES iterations overall, in addition to yielding little additional accuracy in the objective value and residual. For the CR elements,

however, it is less clear what an appropriate value of ϵ_τ is. On the one hand, $\epsilon_\tau = 1 \times 10^{-8}$ was necessary for both values of s in order to have the iterates generated with MINRES to coincide with those generated by the direct solver. On the other hand, this resulted in a very large number of MINRES iterations compared to $\epsilon_\tau = 1 \times 10^{-4}$, which appeared to generate almost as accurate solutions of (4.4) after about 20 iterations.

It is important to note that these observations are only valid for the 50×50 diffuser example, which we used specifically because it was found to be one of the more numerically straightforward examples considered. In order to address how the premature termination of MINRES affects more numerically challenging problems, we will now consider the double pipe benchmark.

6.3.2 The double pipe

With the rather lengthy discussion about appropriate values for s in (6.10) and ϵ_τ in (6.12) for the 50×50 diffuser example, we will now use these results for the double pipe example considered in Subsection 4.6.4. More precisely we consider the example where $\delta = 1.5$, such that we expect the solution to be the joining pipe depicted in Figure 4.14. As we experienced in Subsection 4.6.4, this problem is tricky to solve numerically. First, for the algorithm to yield the preferred design it was necessary to first perform 50 iterations using $q = 0.01$ in the definition of α_q in (4.21). Secondly, the algorithm needed perform 269 and 384 iterations with the direct solver for the CR and the TH elements, respectively, in order to converge.

Equation (6.12) is used as a stopping criterion for MINRES when solving (4.4), and as we found $s = 1$ in (6.10) to sufficient and preferable for the diffuser example, we only consider this case here. Regarding the values for ϵ_τ, we will use the same three values for the CR and TH elements used for the diffuser example.

Starting with the CR elements, we execute the OC method with the stopping criterion (6.12) for MINRES, and results are depicted in Figure 6.12. The first thing to note from Figure 6.12 is the behaviour of the OC method for $\epsilon_\tau = 1 \times 10^{-2}$. For the diffuser example considered previously, we recall that this choice for ϵ_τ yielded inaccurate estimates for J and $\eta^{(s)}$ compared to the direct solver. Now, we see that the method stops after 20 iterations, and it goes without saying the algorithm does not yield a meaningful resulting design. This only strengthens our impression that $\epsilon_\tau = 1 \times 10^{-2}$ is a too lenient threshold for stopping the MINRES iterations.

Turning our attention to the two smaller values of ϵ_τ, the tendencies appears to be similar to those from Figure 6.8. For J and $\eta^{(s)}$, the case $\epsilon_\tau = 1 \times 10^{-4}$ deviates somewhat from the direct solver, while $\epsilon_\tau = 1 \times 10^{-8}$ appears to coincide. However, we see that $\epsilon_\tau = 1 \times 10^{-4}$ needs 163 additional OC iterations compared to when using the direct solver. To address this issue further, we summarise the results in Table 6.3. Looking at the column k_{tot} for the total number of MINRES iterations performed, it is clear that even though the case $\epsilon_\tau = 1 \times 10^{-4}$ uses more OC iterations, the overall number of MINRES iterations is significantly lower than for $\epsilon_\tau = 1 \times 10^{-8}$. More precisely, the case $\epsilon_\tau = 1 \times 10^{-8}$ uses over 11 times more MINRES iterations in total, while the objective value is only about 0.9 % smaller. Finally, to confirm that the resulting designs for $\epsilon_\tau = 1 \times 10^{-4}$ and $\epsilon_\tau = 1 \times 10^{-8}$

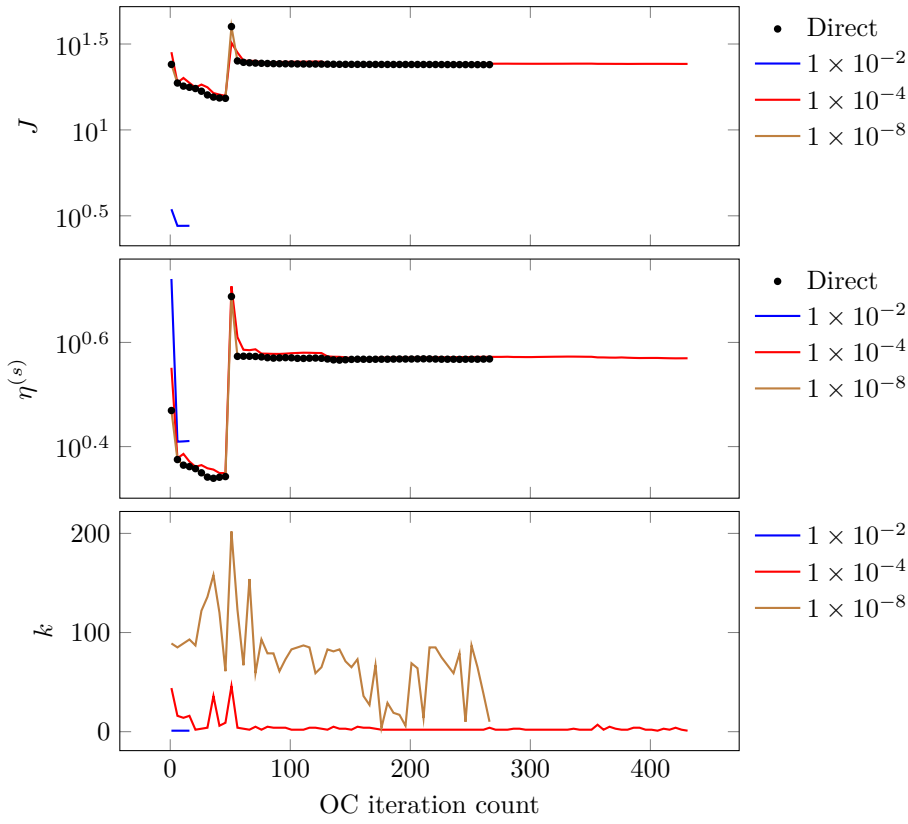


Figure 6.12: Double pipe example with $\delta = 1.5$ using CR elements for three different values of ϵ_τ, and with $s = 1$ in (6.10). J is the objective value, $\eta^{(1)}$ is the combined relative residual norm and k is the number of MINRES iterations.

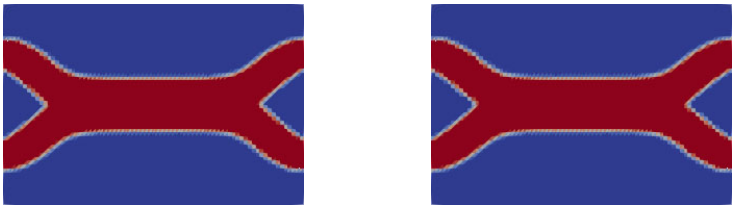
look in fact similar, we depict them in Figure 6.13. We have omitted $\epsilon_\tau = 1 \times 10^{-2}$ as it is of little interest.

Considering now the TH elements, we redo the joining pipe example as before. We still use 1×10^{-2}, 1×10^{-4}, and 1×10^{-6} for ϵ_τ, and the results are seen in Figure 6.14. Looking at the graphs for $\epsilon_\tau = 1 \times 10^{-2}$, we see that this time the OC method appears to converge to an objective value and relative residual norm slightly larger than when using the direct solver, which is similar to the diffuser example. $\epsilon_\tau = 1 \times 10^{-4}$ resembles the direct solver more accurately, but does not coincide completely for neither J nor $\eta^{(s)}$. Although this difference can be argued to be fairly small, we see that it is significantly larger than for Figure 6.10 depicting the results for the diffuser example. For $\epsilon_\tau = 1 \times 10^{-6}$, however, the results still seem to coincide completely with the direct solver.

To get a better understanding of how much accuracy is gained using $\epsilon_\tau = 1 \times 10^{-6}$ compared to $\epsilon_\tau = 1 \times 10^{-4}$, we list the computational data for Figure 6.14 in Table 6.4. For reference, recall that the objective obtained when using

Table 6.3: Computational data the for the double pipe example with $\delta = 1.5$ using CR elements. $s = 1$ in (6.10).

ϵ_τ	OC iters.	J	$\eta^{(s)}$	k_{tot}
1×10^{-2}	20	2.77	2.57	22
1×10^{-4}	432	24.22	3.71	1682
1×10^{-8}	268	24.00	3.70	19154



(a) $\epsilon_\tau = 1 \times 10^{-4}$ **(b)** $\epsilon_\tau = 1 \times 10^{-8}$

Figure 6.13: Resulting design for the double pipe example with $\delta = 1.5$ using CR elements and the stopping criterion (6.12) for MINRES.

the direct solver is 24.81 by Table 4.4. To this end, the objective value obtained with $\epsilon_\tau = 1 \times 10^{-6}$ is 0.4 % smaller, while for $\epsilon_\tau = 1 \times 10^{-4}$ it is 1.4 % larger. Comparing the total number of MINRES iterations between these two values for ϵ_τ, we see that the case $\epsilon_\tau = 1 \times 10^{-6}$ uses approximately 2.8 times more iterations than $\epsilon_\tau = 1 \times 10^{-4}$. Although this factor is significantly smaller than that for the CR elements, it is a noticeable improvement if we accept the 1.4 % error in the objective value.

6.3.3 All examples

Having considered both the 50×50 diffuser example and the joining double pipe example in details, it appears that $s = 1$ in (6.10) provides a reasonable stopping criterion for MINRES for the CR and TH elements. Furthermore, based on observing the objective value and relative residual norm, it appears that by using $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12) we retain a reasonable level of accuracy in the method, while at the same time not needing to perform an excessive amount of MINRES iterations. To complete this chapter we will therefore apply the stopping criterion with the proposed value for s and ϵ_τ for all the examples presented in Section 4.6.

As we are mainly interested in confirming that the premature stopping of MINRES yield satisfactory results also for the other experiments from Section 4.6, we do not discuss the details of each experiment in detail. Starting with the CR elements, we therefore summarise the computational results for all the experiments, and the results is seen in Table 6.5. Looking at the column J err. in particular, we can compare the algorithm using the stopping criterion with that using the direct solver. We see that the largest difference in J is for the 50×50 pipe bend example,

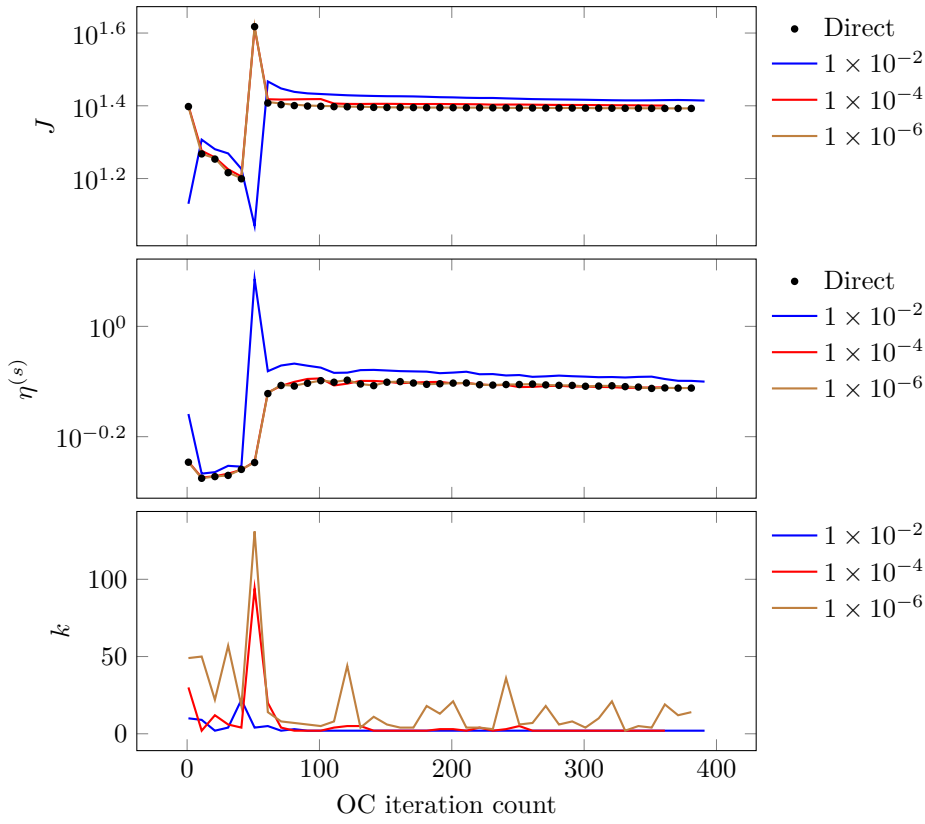


Figure 6.14: Double pipe example with $\delta = 1.5$ using TH elements for three different values of ϵ_τ, and with $s = 1$ in (6.10). J is the objective value, $\eta^{(1)}$ is the combined relative residual norm and k is the number of MINRES iterations.

where the resulting objective value is 2.78 % larger than for the direct solver.

Next we perform the experiments using the TH elements, and we have listed the results in Table 6.6. Again, the 50×50 pipe bend example appears to yield the largest error in J with a difference of 3.51 %. From the two preceding subsections we saw that setting ϵ_τ too small increased the total number of MINRES iterations substantially, while setting it too large resulted in inaccurate or even non-convergent resulting designs. Based on Table 6.5 and 6.6, it therefore appears that using $s = 1$ in (6.10) and $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12) yield a reasonable balance between the number of MINRES iterations necessary to perform and the accuracy of the resulting design.

Table 6.4: Computational data the for the double pipe example with $\delta = 1.5$ using TH elements. $s = 1$ in (6.10).

ϵ_τ	OC iters.	J	$\eta^{(s)}$	k_{tot}
1×10^{-2}	398	26.00	8.07×10^{-1}	1184
1×10^{-4}	361	25.16	7.75×10^{-1}	1771
1×10^{-6}	384	24.71	7.73×10^{-1}	5048

Table 6.5: Computational data the for all the experiments from Section 4.6 using CR elements. $s = 1$ in (6.10) and $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12). The J err. column lists the error in the objective value compared to the results in Section 4.6.

Example	Property	OC iters.	J	J err.	$\eta^{(s)}$	k_{tot}
Diffuser	50×50 mesh	42	30.87	1.41 %	0.57	434
	100×100 mesh	48	30.54	0.36 %	0.29	467
Pipe bend	50×50 mesh	36	9.97	2.78 %	3.79	501
	100×100 mesh	87	9.82	1.55 %	2.00	991
Rugby ball	$\gamma = 0.8$	23	68.41	0.07 %	0.71	214
	$\gamma = 0.9$	51	43.18	−0.25 %	0.65	279
	$\gamma = 0.95$	28	34.51	−1.24 %	0.64	251
Double pipe	$\delta = 1.0$	46	22.06	0.82 %	2.08	409
	$\delta = 1.5$	432	24.22	−2.38 %	3.71	1682

Table 6.6: Computational data the for all the experiments from Section 4.6 using TH elements. $s = 1$ in (6.10) and $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12). The J err. column lists the error in the objective value compared to the results in Section 4.6.

Example	Property	OC iters.	J	J err.	$\eta^{(s)}$	k_{tot}
Diffuser	50×50 mesh	44	31.12	0.32 %	0.11	605
	100×100 mesh	40	30.62	0.00 %	0.04	742
Pipe bend	50×50 mesh	38	10.31	3.51 %	1.01	746
	100×100 mesh	85	9.82	0.82 %	0.40	978
Rugby ball	$\gamma = 0.8$	37	78.25	2.18 %	1.99	247
	$\gamma = 0.9$	39	51.36	−0.05 %	1.99	251
	$\gamma = 0.95$	37	42.55	−0.30 %	1.99	251
Double pipe	$\delta = 1.0$	49	22.26	0.59 %	0.30	755
	$\delta = 1.5$	361	25.16	1.41 %	0.78	1771

Chapter 7

Concluding remarks

7.1 Conclusion

In order to sum up our findings, we discuss them in the order they appeared in this report.

Starting with Chapter 5, in retrospect one could argue that the numerical experiments we performed could have been carried out in a more appropriate manner. Based on our findings in Chapter 6, it appears that the mass residual was the less prominent of the two residuals when it comes to deciding whether or not the discretisation error dominates in a numerical solution, so considering the mass residual in isolation might not be the most expedient approach. As this corresponds to $s = 0$ in (6.10), we could therefore instead to adopt the strategy from Chapter 6 and consider $s = 0.5$. However, as our findings in 6 indicated that using $s = 0.5$ and $s = 1$ in (6.10) resulted in very similar behaviour of the stopping criterion, it is not unreasonable to expect the same tendencies when also applied in the adaptive mesh refinement.

We therefore focus mainly on the results from using (5.1) as the threshold for refinement. For both the CR and TH elements, based on varying c_{mo} we saw that the region of transition between solid and flow was the first part of the design to be subject to refinement. Although *only* refining in this region is not the goal of using the residual estimates for adaptive mesh refinement, we saw that refinement in the transition region generally yielded the largest reduction in the residual compared to the increase in the number of cells. This observation was seen for both the CR elements and TH elements, although the tendency of refining mainly occurring in the transition region was more evident for the latter. As the solid and flow regions are usually much larger than the transition region, extensive refinement in these two regions yielded a significant increase in the number of cells in the mesh, which in turn leads to a large linear system. In that sense, since the strategy for adaptively refining the mesh proposed here does not provide any control over the number of cells marked as subject to refinement, the algorithm was rather sensitive to the choice of c_{mo}. That being said, when the value of c_{mo} was chosen carefully, we

observed a significant decrease in the residual norms without an immense increase in the number of cells in the mesh.

Lastly, it could of course be interesting to apply the adaptive mesh refinement to the other numerical examples from Section 4.6, and in fact these experiments are ready to be carried out using the code provided. We chose however not to go into such comprehensive examination of all the numerical examples in this report.

Turning our attention to the premature termination of MINRES in Chapter 6, we focus mainly on the case $s = 1$ in (6.10). The experiments in Section 6.3 indicates that $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12) yields a reasonable trade-off between the number of MINRES iterations perform and the accuracy of solution for both types of elements. In this case, the largest deviation in the objective value of the resulting design compared to when using the direct solver was about 3.5 %, among all the numerical experiments considered. A significantly larger value for ϵ_τ resulted in either inaccurate designs, or even the algorithm failing to converge to any reasonable solution at all. And naturally, while a much smaller value ϵ_τ gave more coinciding results to the direct solver, a significant increase in MINRES iterations was necessary. In terms the OC iterations, the number necessary when using MINRES with $\epsilon_\tau = 1 \times 10^{-4}$ in (6.12), did not appear to deviate significantly when compared to the direct solver, strengthening our belief that this value for ϵ_τ can be appropriate.

Finally, solving a linear system on a refined mesh in each MINRES iterations is of course computationally expensive, so the efficiency of prematurely terminating MINRES based on the stopping criterion proposed in this report, comes down to how efficiently the residuals can be computed. As such, for methods where the residuals are easily obtained, for instance like those based on least-squared finite elements [BG09], the strategies presented in Sections 5.1 and 6.3 might be more suitable.

7.2 Further research

We end this report by proposing some further research suggestions.

In chapter 4 it was seen that (4.18) could be used to generate iterates for the discrete implementation of the OC method. By addressing the continuity of the operators Π_P and Z_λ from Chapter 3, we believe that Proposition 4.3 can be extended to the continuous formulation in Chapter 3, showing existence of λ in (3.27).

Next, in the beginning of Chapter 2 we mentioned that the generalised Stokes equations appears as a subproblem of the Navier-Stokes equations. As the Navier-Stokes equations are significantly more used in engineering applications, it would be reasonable to apply the proposed methods to Navier-Stokes.

Lastly, having examined two-dimensional flow, a natural extension is consider three dimensions. As we recall, all results presented in this report also holds in three dimensions, and extending the code to this case should be straight-forward in FEniCS. The number of degrees of freedom tends to increase dramatically when going from two to three dimensions, meaning the amount of memory available can quickly

become a limitation. However, both MINRES and the algebraic multigrid method used for the preconditioning and for the residuals, are both generally considered to be memory efficient methods, such that they are expected to perform well also in three dimensions. A significant amount of effort was put into making the code MPI compliant, so it should be readily deployable to large high-performance computers. As such, performing the algorithm on larger three dimensional problems should be achievable in practice.

Appendices

Appendix A

The zero mean constraint

During the report, how to actually enforce the zero mean constraint in order to guarantee uniqueness of solutions for the continuous Darcy-Stokes (2.6) equations and the Galerkin formulation (4.1), was not addressed. The space $Q = L_0^2(\Omega)$ is not readily available in FEniCS and common FEM software, meaning that in order to have $p \in Q$, it is necessary to enforce

$$\int_\Omega p = 0 \tag{A.1}$$

explicitly.

We let $p \in L^2(\Omega)$ and $\bar{p} \in \mathbb{R}$ be the corresponding Lagrange multiplier to the zero mean constraint. Additionally, the variational form of (A.1) is trivially defined as

$$\bar{q} \int_\Omega p = 0 \quad \forall \bar{q} \in \mathbb{R},$$

where $\bar{q}$ is the test function. The new variational formulation of (4.1) becomes: Find $(\mathbf{u}, p, \bar{p}) \in V \times L^2(\Omega) \times \mathbb{R}$ such that

$$\begin{aligned} a_\alpha(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} && \forall \mathbf{v} \in V, \\ b(\mathbf{u}, q) + \bar{p} \int_\Omega q &= 0 && \forall q \in L^2(\Omega), \\ \bar{q} \int_\Omega p &= 0 && \forall \bar{q} \in \mathbb{R}. \end{aligned}$$

The details are completely equivalent as for extending the Galerkin formulation (4.1), such that the linear system (4.4) becomes

$$\begin{aligned} A_\alpha \mathbf{U} + B^T \mathbf{P} &= \mathbf{F}, \\ B\mathbf{U} + \mathbf{\Phi}\bar{p} &= \mathbf{0}, \\ \mathbf{\Phi}^T \mathbf{P} &= 0, \end{aligned}$$

with

$$\mathbf{\Phi} = [\Phi_l] = \left[\int_\Omega \phi_l\right]$$

for $l = 1, \ldots, m$. The rest of the variables remain the same as in Section 4.1. This yields the new system matrix $\bar{S} \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$ defined as

$$\bar{S} = \begin{bmatrix} A_\alpha & B^T & 0 \\ B & 0 & \mathbf{\Phi} \\ 0 & \mathbf{\Phi}^T & 0 \end{bmatrix},$$

which can easily be verified to still be symmetric and indefinite, and non-singular when Proposition (4.2) holds. In other words, $\bar{S}$ is the matrix S with an additional row and column. As this implementation detail does not affect the values of the numerical solution $(\mathbf{u}_h, p_h)$, it is not included in the main part of the report.

Appendix B

Code

The file `main.py` contains the optimisation algorithm, and takes four arguments. The first argument is a number between `1` and `9` denoting the experiment number, being the order in which they appear in Table 6.5 and 6.6. The second argument is either `CR` or `TH`, denoting the element type. The third argument is either `direct` or `minres`, denoting the solver. The fourth argument is either `0` or `1`, denoting whether or not to use adaptivity.

In order to run in parallel, execute
`mpirun -np` ***n*** `python3 main.py` ***arg1*** `...`,
where ***n*** is the number of processes.

The code has been tested with the 2017.1.0 version of the FEniCS Docker container, and is given in its entirety below.

Code B.1: `main.py`. The main script containing the optimisation algorithm.

```
from dolfin import *
import numpy as np
from common import *
from sys import argv,stdout

comm = mpi_comm_world()
rank = MPI.rank(comm)

# prob,L,H,nx,ny,gamma
experiments = [
        ("diffuser", 1, 1, 50, 50, 0.5),
        ("diffuser", 1, 1, 100, 100, 0.5),
        ("pipebend", 1, 1, 50, 50, 0.08*np.pi),
        ("pipebend", 1, 1, 100, 100, 0.08*np.pi),
        ("rugbyball", 1, 1, 100, 100, 0.8),
        ("rugbyball", 1, 1, 100, 100, 0.9),
        ("rugbyball", 1, 1, 100, 100, 0.95),
        ("doublepipe", 1, 1, 100, 100, 1/3),
        ("doublepipe", 1.5, 1, 150, 100, 1/3)
]

```

```
if len(argv) == 1:
        experiment = 0
        element_type = "CR"
        solver_type = "direct"
        perform_adaptivity = False
else:
        experiment = int(argv[1])-1
        element_type = argv[2].upper()
        solver_type = argv[3].lower()
        perform_adaptivity = bool(int(argv[4]))

(prob,L,H,nx,ny,gamma) = experiments[experiment]

# move limits, stopping criterion etc
sc_eps = 0.1
k_max = 500
prec_solver = "hypre_amg"
c_param = 4
s_param = 1

mesh = RectangleMesh(comm, Point(np.array([0.0,0.0])),
↪  Point(np.array([L,H],dtype='double')), nx, ny)
bndry = MeshFunction("size_t", mesh, mesh.topology().dim() - 1)

if element_type == "CR":
        V = VectorElement("CR", mesh.ufl_cell(), 1)
        Q = FiniteElement("DG", mesh.ufl_cell(), 0)
elif element_type == "TH":
        V = VectorElement("CG", mesh.ufl_cell(), 2)
        Q = FiniteElement("CG", mesh.ufl_cell(), 1)
else:
        raise ValueError("No element type %s" % element_type)

R = FiniteElement("R", mesh.ufl_cell(), 0)
W = FunctionSpace(mesh, MixedElement(V,Q,R))
P = FunctionSpace(mesh, "DG", 0)

bc_func = {
        "diffuser": ("near(x[0],0)*4.0*x[1]*(1.0-x[1])+\
        (near(x[0],1) && 1.0/3 <= x[1] && x[1] <= 2.0/3)*108.0*(x[1]-1.0/3.0)*\
        (2.0/3.0-x[1])","0"),
        "pipebend": ("(near(x[0],0) && fabs(0.8-x[1]) <= 0.1)*\
        (1-100*(x[1]-0.8)*(x[1]-0.8))","-(near(x[1],0) && fabs(0.8-x[0]) <= 0.1)*\
        (1-100*(x[0]-0.8)*(x[0]-0.8))"),
        "rugbyball": ("0","near(x[1],0) || near(x[1],1)"),
        "doublepipe": ("(fabs(0.25-x[1]) <= 1.0/12)*\
        (1-144*(x[1]-0.25)*(x[1]-0.25))+(fabs(0.75-x[1]) <= 1.0/12)*\
        (1-144*(x[1]-0.75)*(x[1]-0.75))","0.0")
}

bcs_val = Expression(bc_func[prob], element=V)
bcs_expr = lambda W: DirichletBC(W, bcs_val, "on_boundary")

# integrate over the boundary to check that inflow == outflow
n_vec = FacetNormal(mesh)
bndry_integral = assemble(inner(bcs_val,n_vec)*ds(mesh))
```

```
77  # for the diffuser problem the integral over the outflow subdomain is not very
    ↪  accurate unless
78  # n is chosen such that the boundary nodes "line up" with the edges of the
79  # outlet (for instance if n = 99 or 51).
80  if abs(bndry_integral) > 1E-2: #DOLFIN_EPS
81      raise ValueError("The boundary integral of u, %f, is non-zero" % bndry_integral)
82
83  q_param = Constant(0.01)
84  f_src = Constant((0,0))
85
86  alpha = lambda rho:
    ↪  Constant(2.5/1E-4)*(Constant(1)-rho*(Constant(1)+q_param)/(rho+q_param))
87  a = lambda rho,u,v: (inner(grad(u),grad(v)) + alpha(rho)*inner(u,v))*dx
88  b1 = lambda v,p: -p*div(v)*dx
89  b2 = lambda p,q: p*q*dx
90  F = lambda v: inner(f_src,v)*dx
91  f_expr = lambda rho,u: Constant(0.5)*a(rho,u,u) - F(u)
92  grad_f_expr = lambda rho,u: Constant(0.5)*diff(alpha(rho), rho)*inner(u,u)
93
94  prob_name = "output/" + prob
95  if prob == "diffuser":
96          prob_name += "_%d" % nx
97  elif prob == "pipebend":
98          prob_name += "_%d" % nx
99  elif prob == "rugbyball":
100         prob_name += "_%d" % int(100*gamma)
101 elif prob == "doublepipe":
102         prob_name += "_%d" % int(10*L)
103 else:
104         raise ValueError("Invalid benchmark %s" % prob)
105 prob_name = "_".join((prob_name, element_type, solver_type, "ad",
    ↪  str(int(perform_adaptivity))))
106
107 file = XDMFFile(comm, prob_name+".xdmf")
108
109 rho = Function(P)
110
111 rho_v = variable(rho)
112 (u,p,p0) = TrialFunctions(W)
113 (v,q,q0) = TestFunctions(W)
114 A_ = a(rho,u,v) + b1(v,p) + b1(u,q) + b2(p,q0) + b2(p0,q)
115 M_ = a(rho,u,v) + b2(p,q) + b2(p0,q0)
116 ell = F(v)
117 w = Function(W)
118
119
120 (u,p,p0) = w.split()
121 rho.rename("rho", "Control")
122 u.rename("u", "Velocity")
123 p.rename("p", "Pressure")
124 f = f_expr(rho,u)
125 grad_f_form = grad_f_expr(rho_v,u)
126 grad_f = Function(P)
127 bcs = bcs_expr(W.sub(0))
128
129 if prob == "doublepipe" and L/H >= 1.5:
130         pre_iters = 50
```

```
131         ref_freq = 50
132 else:
133         pre_iters = 0
134         ref_freq = 10
135
136 rho_presc = Function(P)
137 if prob == "rugbyball":
138         rho.assign(Expression("0.1+(4.0/(L*L))*(x[0]-L/2)*(x[0]-L/2)", L=float(L),
            ↪  element=P.ufl_element()))
139         rho_presc.assign(Expression("x[0] < 0.25 or x[0] > L-0.25", L=L,
            ↪  element=P.ufl_element()))
140 else:
141         rho.assign(Constant(gamma))
142         rho_presc.assign(Constant(0))
143
144 vol_constr = gamma*L*H
145 if assemble(rho_presc*dx) > vol_constr:
146         raise ValueError("The area of prescribed fluid exceeds the volume
            ↪  constraint")
147
148 if solver_type == "minres" or perform_adaptivity:
149         g_val = sqrt(assemble(inner(bcs_val,bcs_val)*ds(mesh)))
150         res = Residual(w, alpha(rho), f_src, s_param, g_val,
            ↪  solver_type=prec_solver, fine_mesh=None)
151
152 if solver_type == "minres":
153         solver = MINRESSolver(prec_solver)
154         solver.parameters['nonzero_initial_guess'] = True
155         solver.parameters['maximum_iterations'] = 2000
156         solver.parameters['relative_tolerance'] = DOLFIN_EPS
157         solver.parameters['relative_residual_tolerance'] = 1E-4
158
159         solver.set_res(res)
160         solver.parameters['use_residual'] = True
161         M = assemble_system(M_, ell, bcs)[0]
162 elif solver_type == "direct":
163         solver = LUSolver()
164 else:
165         raise ValueError("Invalid solver %s" % solver_type)
166
167 A,b = assemble_system(A_, ell, bcs)
168
169 if rank == 0:
170         data = np.empty([k_max+1,4])
171         print("============== Optimisation start =============")
172         stdout.flush()
173
174 for k in range(0, k_max+1):
175         if k == pre_iters:
176                 q_param.assign(Constant(0.1))
177
178         if solver_type == "minres":
179                 solver.set_operators(A, M)
180                 ms_iters = solver.solve(w.vector(), b)
181         elif solver_type == "direct":
182                 solver.set_operator(A)
183                 ms_iters = 0
```

```
184                 solver.solve(w.vector(), b)
185
186         f_val = assemble(f)
187         project(grad_f_form, P, function=grad_f)
188         sc_val = sc(rho, rho_presc, grad_f, vol_constr)
189
190         if rank == 0:
191                 data[k,:] = np.array([k,f_val,sc_val,ms_iters])
192                 print("k = %3d, f = %7.3f, sc = %6.4f, ms_iters = %d" %
                    ↪  tuple(data[k,:]))
193                 stdout.flush()
194
195         if  k % 5 == 0 or sc_val < sc_eps: # save sol'n to file every 5'th iteration
196                 file.write(rho,k,file.Encoding_HDF5)
197                 file.write(u,k,file.Encoding_HDF5)
198                 file.write(p,k,file.Encoding_HDF5)
199         if sc_val < sc_eps and k > 20:
200                 break
201
202         rho.assign(OC(rho,rho_presc,grad_f,vol_constr))
203
204         if perform_adaptivity and k > pre_iters and k % ref_freq == 0:
205                 mesh = adapt_mesh(mesh, res, c_param)
206                 W = adapt(W, mesh)
207                 P = adapt(P, mesh)
208                 rho = adapt(rho, mesh)
209                 rho_v = variable(rho)
210                 rho_presc = adapt(rho_presc, mesh)
211                 (u,p,p0) = TrialFunctions(W)
212                 (v,q,q0) = TestFunctions(W)
213                 A_ = a(rho,u,v) + b1(v,p) + b1(u,q) + b2(p,q0) + b2(p0,q)
214                 M_ = a(rho,u,v) + b2(p,q) + b2(p0,q0)
215
216                 ell = F(v)
217                 w = adapt(w, mesh)
218                 (u,p,p0) = w.split()
219                 rho.rename("rho", "Control")
220                 u.rename("u", "Velocity")
221                 p.rename("p", "Pressure")
222                 f = f_expr(rho,u)
223                 grad_f_form = grad_f_expr(rho_v,u)
224                 grad_f = Function(P)
225
226                 bcs = bcs_expr(W.sub(0))
227                 A,b = assemble_system(A_, ell, bcs)
228                 res = Residual(w, alpha(rho), f_src, s_param, g_val,
                    ↪  solver_type=prec_solver, fine_mesh=None)
229
230                 if solver == "minres":
231                         solver = MINRESSolver(prec_solver)
232                         solver.parameters['nonzero_initial_guess'] = True
233                         solver.parameters['maximum_iterations'] = 2000
234                         solver.parameters['relative_tolerance'] = DOLFIN_EPS
235                         solver.parameters['relative_residual_tolerance'] = 1E-4
236
237                         solver.set_res(res)
238                         solver.parameters['use_residual'] = True
```

```
239                         M = assemble_system(M_, ell, bcs)[0]
240                 else:
241                         solver = LUSolver()
242
243         else:
244                 assemble_system(A_, ell, bcs, A_tensor=A, b_tensor=b)
245                 if solver_type == "minres":
246                         assemble_system(M_, ell, bcs, A_tensor=M)
247
248 if rank == 0:
249         data = data[0:k,:]
250         np.savetxt(prob_name+".csv", data,\
251                 fmt=['%d', '%E', '%E', '%d'], delimiter=' ', header='k f sc
                ↪  ms_iters', comments='')
252
253 dump_timings_to_xml(prob_name+"_timings.xml", TimingClear_clear)
254 file.close()
255
256 if rank == 0:
257         print("============== Optimisation done ==============")
258         stdout.flush()
```

Code B.2: `common.py`. Functions and classes used by the main script.

```
1  from dolfin import *
2  import numpy as np
3  from petsc4py import PETSc
4
5  parameters["allow_extrapolation"] = True
6  parameters["refinement_algorithm"] = "plaza_with_parent_facets"
7
8  def OC(rho,rho_presc,grad_f,vol_constr):
9          zeta = 0.4
10         l1 = 0
11         l2 = 1E4
12         rho_vec = rho.vector().get_local()
13         rho1 = Function(rho.function_space())
14         vec = np.empty_like(rho1.vector().get_local())
15
16         while (l2-l1)/max(l2+l1,1) > 1E-10:
17                 lmid = (l2+l1)/2
18                 vec = np.sqrt(-grad_f.vector().get_local()/lmid)*rho_vec
19                 vec = np.minimum(1,vec)
20                 vec = np.maximum(rho_presc.vector().get_local(),vec)
21                 vec = np.minimum(rho_vec*(1+zeta),vec)
22                 vec = np.maximum(rho_vec*(1-zeta),vec)
23                 rho1.vector().set_local(vec)
24                 rho1.vector().apply('insert')
25
26                 vol_diff = vol_constr - assemble(rho1*dx)
27                 if abs(vol_diff) < 1E-10:
28                         break
29                 if vol_diff < 0.0:
30                         l1 = lmid
31                 else:
```

```
32                          l2 = lmid
33          return rho1
34
35  def proj(rho, rho_presc, vol_constr):
36          # projects rho onto the simplex {0 <= rho <= 1, integral rho <= vol_constr}
37          rho_vec = rho.vector().get_local()
38          rho1 = Function(rho.function_space())
39
40          vec = rho_vec
41          vec = np.minimum(1,vec)
42          vec = np.maximum(rho_presc.vector().get_local(),vec)
43          rho1.vector().set_local(vec)
44          rho1.vector().apply('insert')
45          if assemble(rho1*dx) <= vol_constr:
46                  return rho1
47
48          l1 = 0
49          l2 = rho.vector().max()
50
51          while (l2-l1)/max(l2+l1,1) > 10E-10:
52                  lmid = (l2+l1)/2
53                  vec = rho_vec - lmid
54                  vec = np.minimum(1,vec)
55                  vec = np.maximum(rho_presc.vector().get_local(),vec)
56                  rho1.vector().set_local(vec)
57                  rho1.vector().apply('insert')
58
59                  vol_diff = vol_constr - assemble(rho1*dx)
60                  if abs(vol_diff) < 10E-10:
61                          break
62                  if vol_diff < 0.0:
63                          l1 = lmid
64                  else:
65                          l2 = lmid
66          return rho1
67
68  def sc(rho, rho_presc, grad_f, vol_constr):
69          rho_tmp = Function(rho.function_space())
70          rho_tmp.assign(rho - grad_f)
71          rho_tmp.assign(rho - proj(rho_tmp, rho_presc, vol_constr))
72          return norm(rho_tmp, 'L2')
73
74  def adapt_mesh(mesh_coarse, res, c_param = 1):
75          mesh_fine = res.mesh
76          markers_coarse = MeshFunction("size_t", mesh_coarse,
            ↪  mesh_coarse.topology().dim())
77
78          for c in cells(mesh_coarse):
79                  markers_coarse[c] = c.index()
80
81          markers_fine = adapt(markers_coarse, mesh_fine)
82
83          res.compute()
84          loc_r_norm_fine = res.norm_loc_squared()
85
86          loc_r_norm_coarse = assemble(TestFunction(FunctionSpace(mesh_coarse, "DG",
            ↪  0))*Constant(0)*dx(mesh_coarse))
```

```
87          tmp_vec = np.zeros_like(loc_r_norm_coarse.get_local())
88          for c in cells(mesh_fine):
89                  tmp_vec[markers_fine[c]] += loc_r_norm_fine[c.index()][0]
90          tmp_vec = np.sqrt(tmp_vec)
91          loc_r_norm_coarse.set_local(tmp_vec)
92          loc_r_norm_coarse.apply('insert')
93          treshold =
            ↪  res.norm()*np.sqrt(c_param/mesh_coarse.num_entities_global(mesh_coarse.topology().dim()))
94          markers = MeshFunction("bool", mesh_coarse, mesh_coarse.topology().dim())
95
96          for c in cells(mesh_coarse):
97                  markers[c] = loc_r_norm_coarse.get_local()[c.index()] > treshold
98
99          mesh_adapted = refine(mesh_coarse, markers, redistribute=False)
100         return mesh_adapted
101
102 class ResidualBase():
103         def set_solver(self, A, solver_type):
104                 solver_type = solver_type.lower()
105
106                 if solver_type == "lu":
107                         solver = LUSolver(A)
108                         solver.parameters['reuse_factorization'] = True
109                 elif solver_type == "hypre_amg":
110                         solver = AMGSolver()
111                         solver.set_operator(A)
112                 else:
113                         raise NameError("Illegal solver type")
114
115                 return solver
116
117         def rel_norm(self):
118                 return self.norm()/self.g_val
119
120 class ResidualMo(ResidualBase):
121         def __init__(self, fine_mesh, u, p, alpha, f_src, g_val = 1,
            ↪  solver_type='lu'):
122                 V = u.function_space()
123                 if fine_mesh == None:
124                         fine_mesh = refine(V.mesh(), redistribute=False)
125
126                 deg = V.ufl_element().degree()
127                 self.g_val = g_val
128                 self.u = u
129                 self.grad_u = Function(TensorFunctionSpace(V.mesh(), "DG", deg))
130                 self.mesh = fine_mesh
131                 V_fine = adapt(V, self.mesh)
132                 r_mo = TrialFunction(V_fine)
133                 v = TestFunction(V_fine)
134
135                 mo_form = (inner(r_mo,v) + inner(grad(r_mo),grad(v)))*dx(self.mesh)
136                 A = assemble(mo_form)
137                 self.A = A
138                 self.bcs = DirichletBC(V_fine, Constant((0,0)), "on_boundary")
139                 self.bcs.apply(A)
140                 self.solver = self.set_solver(A, solver_type)
141                 self.r_mo = Function(V_fine)
```

```
142                 self.r_mo.rename("r_mo", "Momentum residual")
143
144                 self.L = (inner(f_src,v) - inner(self.grad_u,grad(v)) -
                    ↪  alpha*inner(u,v) + p*div(v))*dx(self.mesh)
145
146         def compute(self):
147                 project(grad(self.u), self.grad_u.function_space(),
                    ↪  function=self.grad_u)
148                 b = assemble(self.L)
149                 self.bcs.apply(b)
150                 self.solver.solve(self.r_mo.vector(), b)
151                 return self.r_mo
152
153         def norm_loc_squared(self):
154                 r_mo = self.r_mo
155                 v = TestFunction(FunctionSpace(self.mesh, "DG", 0))
156                 vec = assemble((inner(r_mo,r_mo) +
                    ↪  inner(grad(r_mo),grad(r_mo)))*v*dx)
157                 return vec
158
159         def norm(self):
160                 return norm(self.r_mo, 'H1')
161
162 class ResidualMa(ResidualBase):
163         def __init__(self, fine_mesh, u, p, p0, g_val = 1, solver_type='lu'):
164                 Q = p.function_space()
165                 if fine_mesh == None:
166                         fine_mesh = refine(Q.mesh(), redistribute=False)
167
168                 deg = Q.ufl_element().degree()
169                 self.g_val = g_val
170                 self.u = u
171                 self.div_u = Function(FunctionSpace(Q.mesh(), "DG", deg))
172                 self.mesh = fine_mesh
173                 Q_fine = adapt(Q, self.mesh)
174                 r_ma = TrialFunction(Q_fine)
175                 q = TestFunction(Q_fine)
176
177                 ma_form = r_ma*q*dx(self.mesh)
178                 A = assemble(ma_form)
179                 self.solver = self.set_solver(A, solver_type)
180                 self.r_ma = Function(Q_fine)
181                 self.r_ma.rename("r_ma", "Mass residual")
182
183                 self.L = (self.div_u*q - p0*q)*dx(self.mesh)
184
185         def compute(self):
186                 project(div(self.u), self.div_u.function_space(),
                    ↪  function=self.div_u)
187                 b = assemble(self.L)
188                 self.solver.solve(self.r_ma.vector(), b)
189                 return self.r_ma
190
191         def norm_loc_squared(self):
192                 q = TestFunction(FunctionSpace(self.mesh, "DG", 0))
193                 vec = assemble(self.r_ma**2*q*dx)
194                 return vec
```

```
195
196         def norm(self):
197                 return norm(self.r_ma, 'L2')
198
199 class Residual():
200         def __init__(self, w, alpha, f_src, s, g_val = 1,  solver_type='lu',
        ↪  fine_mesh=None):
201                 if fine_mesh == None:
202                         fine_mesh = refine(w.function_space().mesh(),
                        ↪  redistribute=False)
203
204                 self.mesh = fine_mesh
205                 (u,p,p0) = w.split()
206                 self.s = s
207                 if self.s > DOLFIN_EPS:
208                         self.res_mo = ResidualMo(fine_mesh, u, p, alpha, f_src,
                        ↪  g_val, solver_type)
209                 if 1-self.s > DOLFIN_EPS:
210                         self.res_ma = ResidualMa(fine_mesh, u, p, p0, g_val,
                        ↪  solver_type)
211
212         def compute(self):
213                 if self.s > DOLFIN_EPS:
214                         self.res_mo.compute()
215                 if 1-self.s > DOLFIN_EPS:
216                         self.res_ma.compute()
217
218         def rel_norm(self):
219                 norm_val = 0
220                 if self.s > DOLFIN_EPS:
221                         norm_val += self.s*self.res_mo.rel_norm()
222                 if 1-self.s > DOLFIN_EPS:
223                         norm_val += (1-self.s)*self.res_ma.rel_norm()
224
225                 return norm_val
226
227         def norm(self):
228                 norm_val = 0
229                 if self.s > DOLFIN_EPS:
230                         norm_val += self.s*self.res_mo.norm()
231                 if 1-self.s > DOLFIN_EPS:
232                         norm_val += (1-self.s)*self.res_ma.norm()
233
234                 return norm_val
235
236         def norm_loc_squared(self):
237                 norm_val = assemble(TestFunction(FunctionSpace(self.mesh, "DG",
                ↪  0))*Constant(0)*dx(self.mesh))
238                 if self.s > DOLFIN_EPS:
239                         norm_val.axpy(self.s, self.res_mo.norm_loc_squared())
240                 if 1-self.s > DOLFIN_EPS:
241                         norm_val.axpy(1-self.s, self.res_ma.norm_loc_squared())
242
243                 return norm_val
244
245
246 class MINRESSolver():
```

```
247         def __init__(self, prec = None):
248                 global parameters
249                 self.parameters = parameters['krylov_solver'].copy()
250                 self.parameters.add('relative_residual_tolerance', 1E-10)
251                 self.parameters.add('use_residual', False)
252 
253                 if isinstance(prec, str):
254                         prec = prec.lower()
255 
256                 self.set_preconditioner(prec)
257 
258         def set_operators(self, A, M = None):
259                 self.A = A
260                 self.prec.set_operator(M)
261 
262         def set_res(self, res):
263                 self.res = res
264 
265         def set_preconditioner(self, prec = None):
266                 if prec == None or prec == "none":
267                         self.prec = NoPreconditioner()
268                 elif prec == "lu":
269                         self.prec = LUSolver()
270                         self.prec.parameters['reuse_factorization'] = True
271                         self.prec.parameters['symmetric'] = True
272                 elif prec == "hypre_amg":
273                         self.prec = AMGSolver()
274                 else:
275                         raise NameError("Invalid preconditioning method %s" %
                        ↪  str(prec_method))
276 
277         def solve(self, x, b):
278                 if self.parameters['maximum_iterations']:
279                         max_iters = self.parameters['maximum_iterations']
280                 else:
281                         max_iters = 10000
282 
283                 if self.parameters['relative_tolerance']:
284                         tol = self.parameters['relative_tolerance']
285                 else:
286                         tol = 1E-10
287 
288                 A = self.A
289                 prec = self.prec
290 
291                 if not self.parameters['nonzero_initial_guess']:
292                         A.init_vector(x, 1)
293 
294                 p = Vector()
295                 A.init_vector(p, 0)
296                 A.mult(x, p)
297 
298                 z = Vector(b)
299                 z.axpy(-1.0, p)
300 
301                 q = Vector()
302                 A.init_vector(q, 1)
```

```
303                 prec.solve(q, b)
304                 phi_0 = sqrt(q.inner(b))
305                 prec.solve(q, z)
306                 beta = sqrt(q.inner(z))
307                 phi = beta
308
309                 z_old = Vector()
310                 A.init_vector(z_old, 0)
311                 q_old = Vector()
312                 A.init_vector(q_old, 1)
313                 d = Vector()
314                 A.init_vector(d, 1)
315                 d_old = Vector()
316                 A.init_vector(d_old, 1)
317                 beta_old = 1
318                 c_old = -1
319                 s_old = 0
320                 delta = np.zeros(2)
321                 gamma = np.zeros(2)
322                 epsilon = 0
323
324                 if self.parameters['use_residual']:
325                         self.res.compute()
326                         eta = self.res.rel_norm()
327
328                 k = 0
329                 while phi/phi_0 > tol:
330                         if k >= max_iters:
331                                 raise RuntimeError("Maximum number of MINRES
                                    ↪  iterations %d exceeded. Relative tolerance =
                                    ↪  %.3E" % (k,phi/phi_0))
332
333                         A.mult(q, p)
334                         alpha = 1/beta**2*q.inner(p)
335                         z_old.set_local(1/beta*p.get_local() -
                                ↪  alpha/beta*z.get_local() -
                                ↪  beta/beta_old*z_old.get_local())
336                         z_old.apply('insert')
337                         z,z_old = z_old,z
338
339                         self.prec.solve(q_old, z) # preconditioning
340                         q,q_old = q_old,q
341                         beta_old = sqrt(z.inner(q))
342                         beta,beta_old = beta_old,beta
343
344                         delta[1] = c_old*delta[0] + s_old*alpha
345                         gamma[0] = s_old*delta[0] - c_old*alpha
346
347                         c,s,gamma[1] = SymOrtho(gamma[0],beta)
348
349                         tau = c*phi
350                         phi = s*phi
351
352                         if abs(gamma[1]) > DOLFIN_EPS:
353                                 d_old.set_local((1/beta_old*q_old.get_local() -
                                        ↪  delta[1]*d.get_local() -
                                        ↪  epsilon*d_old.get_local())/gamma[1])
```

```
354                                 d_old.apply('insert')
355                                 d,d_old = d_old,d
356                                 x.axpy(tau,d)
357
358                         epsilon = s_old*beta
359                         delta[0] = -c_old*beta
360                         c_old = c
361                         s_old = s
362                         k += 1
363
364                         if self.parameters['use_residual'] == True:
365                                 self.res.compute()
366                                 eta_old = self.res.rel_norm()
367                                 eta,eta_old = eta_old,eta
368                                 if abs(eta-eta_old)/eta <
                                    ↪  self.parameters['relative_residual_tolerance']:
369                                         break
370
371                 return k
372
373 class NoPreconditioner():
374                 def set_operator(self, M = None):
375                         return
376
377                 def solve(self, q, z):
378                         q.set_local(np.zeros_like(q.get_local()))
379                         q.apply('insert')
380                         q.axpy(1, z)
381
382 class AMGSolver():
383         def __init__(self):
384                 self.prec = PETSc.PC()
385                 self.prec.create()
386                 self.prec.setType(PETSc.PC.Type.HYPRE)
387
388         def set_operator(self, M):
389                 self.prec.setOperators(as_backend_type(M).mat(),
                        ↪  as_backend_type(M).mat())
390                 PETSc.Options().setValue("pc_hypre_type", "boomeramg")
391                 self.prec.setFromOptions()
392                 self.prec.setUp()
393
394         def solve(self, q, z):
395                 self.prec.apply(as_backend_type(z).vec(), as_backend_type(q).vec())
396
397 def SymOrtho(a, b):
398         if abs(b) < DOLFIN_EPS:
399                 s = 0
400                 r = abs(a)
401                 if abs(a) < DOLFIN_EPS:
402                         c = 1
403                 else:
404                         c = np.sign(a)
405         elif abs(a) < DOLFIN_EPS:
406                 c = 0
407                 s = np.sign(b)
408                 r = abs(b)
```

```
409         elif abs(b) > abs(a):
410                 tau = a/b
411                 s = np.sign(b)/np.sqrt(1+tau**2)
412                 c = s*tau
413                 r = b/s
414         elif abs(a) > abs(b):
415                 tau = b/a
416                 c = np.sign(a)/np.sqrt(1+tau**2)
417                 s = c*tau
418                 r = a/c
419         return c,s,r
```

Bibliography

[BF91] Franco Brezzi and Michel Fortin, eds. *Mixed and Hybrid Finite Element Methods*. Springer New York, 1991. DOI: 10.1007/978-1-4612-3172-1.

[BG09] Pavel B. Bochev and Max D. Gunzburger. *Least-Squares Finite Element Methods: 166 (Applied Mathematical Sciences)*. Springer, 2009.

[BGL05] Michele Benzi, Gene H. Golub and Jörg Liesen. 'Numerical solution of saddle point problems'. In: *Acta Numerica* 14 (2005), pp. 1–137. DOI: 10.1017/S0962492904000212.

[BL07] Michele Benzi and Jia Liu. 'An Efficient Solver for the Incompressible Navier–Stokes Equations in Rotation Form'. In: *SIAM Journal on Scientific Computing* 29.5 (Jan. 2007), pp. 1959–1981. DOI: 10.1137/060658825.

[BP03] Thomas Borrvall and Joakim Petersson. 'Topology optimization of fluid in Stokes flow'. In: 41 (Jan. 2003), pp. 77–107.

[Bra07] Dietrich Braess. *Finite Elements*. Cambridge University Press, 2007. DOI: 10.1017/cbo9780511618635.

[Bre14] Susanne C. Brenner. 'Forty Years of the Crouzeix-Raviart element'. In: *Numerical Methods for Partial Differential Equations* 31.2 (July 2014), pp. 367–396. DOI: 10.1002/num.21892.

[Bre74] F. Brezzi. 'On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers'. In: *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique* 8.R2 (1974), pp. 129–151. DOI: 10.1051/m2an/197408r201291.

[Bru03] Bruce van Brunt. *The Calculus of Variations (Universitext)*. Springer, 2003. ISBN: 0387402470.

[BS07] Susanne Brenner and Ridgway Scott. *The Mathematical Theory of Finite Element Methods (Texts in Applied Mathematics)*. Springer, 2007.

[BS11] Martin Philip Bendsoe and Ole Sigmund. *Topology Optimization: Theory, Methods, and Applications*. Springer, 2011. ISBN: 9783662050866.

[CF89] Michel Crouzeix and Richard S. Falk. 'Nonconforming finite elements for the Stokes problem'. In: *Mathematics of Computation* 52.186 (May 1989), pp. 437–437. DOI: 10.1090/s0025-5718-1989-0958870-8.

[CR73] M. Crouzeix and P.-A. Raviart. 'Conforming and nonconforming finite element methods for solving the stationary Stokes equations I'. In: *Revue française d'automatique informatique recherche opérationnelle. Mathématique* 7.R3 (1973), pp. 33–75. DOI: 10.1051/m2an/197307r300331.

[CS06] Sou-Cheng Choi and Michael Saunders. 'Iterative methods for singular linear equations and least-squares problems'. In: (2006).

[EG04] Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*. Springer New York, 2004. DOI: 10.1007/978-1-4757-4355-5.

[HJ90] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. ISBN: 0521386322.

[LMW+12] Anders Logg, Kent-Andre Mardal, Garth N. Wells et al. *Automated Solution of Differential Equations by the Finite Element Method*. Ed. by Anders Logg, Kent-Andre Mardal and Garth N. Wells. Springer, 2012. ISBN: 978-3-642-23098-1. DOI: 10.1007/978-3-642-23099-8.

[Nie16] Frank Nielsen. *Introduction to HPC with MPI for Data Science (Undergraduate Topics in Computer Science)*. Springer, 2016.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer, 2006.

[QSS10] Alfio Quarteroni, Riccardo Sacco and Fausto Saleri. *Numerical Mathematics: 37 (Texts in Applied Mathematics)*. Springer, 2010.

[Qua14] Alfio Quarteroni. *Numerical Models for Differential Problems (MS&A)*. Springer, 2014.

[Rud87] Walter Rudin. *Real & Complex Analysis*. MHHE, 1987. ISBN: 0070619875.

[Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, 2003. ISBN: 0898715342.

[Sau14] Tim Sauer. *Numerical analysis*. Harlow, Essex: Pearson, 2014. ISBN: 1292023589.

[Tao11] Terence Tao. *An Introduction to Measure Theory (Graduate Studies in Mathematics)*. American Mathematical Society, 2011. ISBN: 0821869191.

[Trö10] Fredi Tröltzsch. *Optimal Control of Partial Differential Equations*. American Mathematical Society, Apr. 2010. DOI: 10.1090/gsm/112.

[Wal14] Stefan Waldmann. *Topology : an introduction*. Cham: Springer, 2014. ISBN: 978-3-319-09679-7.