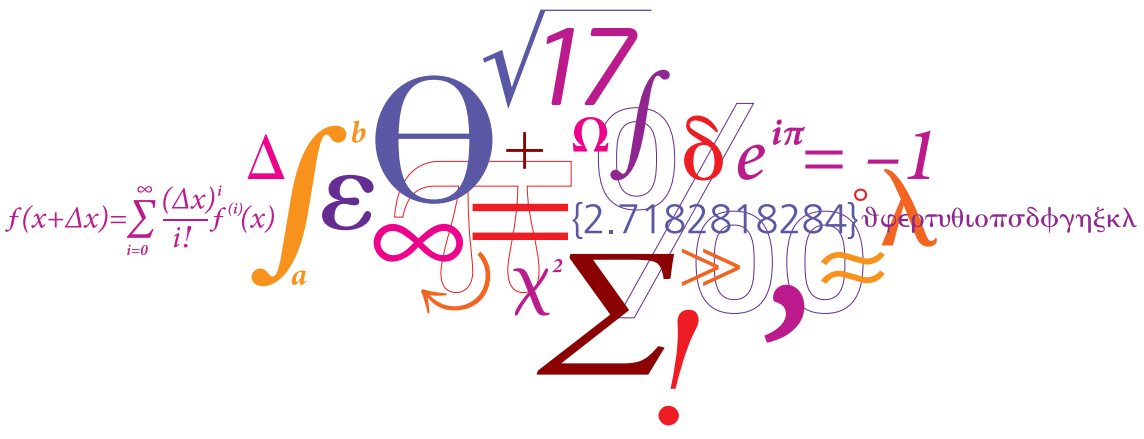
Technical University of Denmark

Jens Vallø Christiansen (s090339)

Topology optimization of flow domains

using least-squares finite element methods

Master's Thesis, June 2012



JENS VALLØ CHRISTIANSEN (S090339)

Topology optimization of flow domains

using least-squares finite element methods

Master's Thesis, June 2012

Supervisors:
Anton Evgrafov, Associate Professor at DTU Mathematics
Kim Knudsen, Associate Professor at DTU Mathematics

DTU - Technical University of Denmark, Kgs. Lyngby - 2012

DTU Mathematics
Department of Mathematics

Topology optimization of flow domains using least-squares finite element methods

This report was prepared by:
Jens Vallø Christiansen (s090339)

Advisors:
Anton Evgrafov, Associate Professor at DTU Mathematics
Kim Knudsen, Associate Professor at DTU Mathematics

Department of Mathematics
Technical University of Denmark
Matematiktorvet, Building 303 S
2800 Kgs. Lyngby
Denmark

Contact:
jenschristiansen@gmail.com

Project period: January 2012 - June 2012

ECTS: 30

Education: MSc

Field: Mathematical Modelling and Computation

Remarks: This report is submitted as partial fulfillment of the requirements for graduation in the above stated MSc programme at the Technical University of Denmark.

Table of Contents

Abstract

This paper investigates topology optimization of Stokes flow in 2D domains using two different classes of finite element methods: mixed formulations, and least-squares finite element methods (LS-FEM). The paper is organized into three main parts. The first part introduces the Stokes equations and shows the existence and uniqueness of solutions. The second part compares the performance of the two methods by doing convergence analysis in both 2D and 3D. In the third part of the paper, a methodology for topology optimization of Stokes flow is introduced, based on the paper by Borrvall & Petersson (2003, [3]). Based on the results from part two, the topology optimization problems are solved using finite elements. The numerical computations were done using FEniCS.

The main conclusions are:

- The finite element method based on a mixed formulation of the Stokes problem outperforms the least-squares methods in 2D, in terms of accuracy and speed.
- The advantages of the LS-FEM, particularly the fact that they lead to symmetric, positive definite systems of linear equations, become relevant in three dimensions. For very large problems, the ratio of accuracy to speed moves in favor of the LS-FEM.
- The least-squares finite element method analyzed in Chang & Yang (2002, [5]) is outperformed by quasi-norm-equivalent methods. Moreover, the method proves to be too inaccurate to solve topology optimization problems.
- Quasi-norm-equivalent LS-FEM methods come closer to solving topology optimization problems, but the results are still not satisfactory compared to methods based on a mixed formulation.

Key words: Stokes equations, variational methods, topology optimization, finite element method, mixed formulation, least-squares FEM, convergence analysis, FEniCS.

Preface

The present thesis was written at the Department of Mathematics of the Technical University of Denmark during the spring of 2012 under the supervision of Anton Evgrafov and Kim Knudsen. The idea for the subject of the thesis came from Anton Evgrafov, who suggested a number of possible topics; this one was chosen because it combines functional analysis, PDE theory, optimization and numerical calculations in an interesting way. Mr. Evgrafov also suggested that the open source library FEniCS be used to do the numerical experiments. Although it took some time to learn how to work in a Linux (Ubuntu) environment, FEniCS proved to be a great choice due to its power and flexibility, and it was certainly worth the effort.[1]

Intented audience

The reader is assumed to have a mathematical background on the level of a bachelor degree, and experience with PDEs and numerical methods, particularly the finite element method, will be helpful. Knowledge of measure theory is also helpful, but not a requirement. The paper has an extensive appendix that reviews much of the theory needed to understand the material in the paper. The notes [18] give an excellent introduction to the finite element method.[2]

Acknowledgements

I would like to thank Anton and Kim for their enthusiastic guidance during my work with the thesis, for giving me lots of great feedback, and for patiently answering my many questions. It was Kim's course on PDEs and Anton (et al)'s course on scientific computing that got me interested in the two subjects in the first place, and made this project possible. Thanks to Johan Hake, Anders Logg and the other people on the FEniCS message board for promptly answering my numerous questions, this was very helpful. And thanks to Jakob Goldbach for helping me get started with Linux, and for giving useful comments on my Python code.

Lyngby, June 2012

Jens Vallø Christiansen (s090339)

[1] The code used in the paper can be found at `https://github.com/jenschristiansen/MMC-thesis-code.git`.
[2] They are available for download at `http://people.maths.ox.ac.uk/suli/fem.pdf`.

Chapter 1
Introduction

This paper investigates topology optimization of Stokes flow in 2D domains using two different classes of finite element methods: mixed formulations, and least-squares finite element methods (LS-FEM). Each class takes a different approach to the problem, and each has its own strengths and weaknesses. The main goal of the paper is to identify these strengths and weaknesses, and compare them.

The paper is organized into three main parts. The first part gives a brief introduction to fluid mechanics, and the Stokes equations are derived as a special case of the Navier-Stokes equations. To show the existence and uniqueness of solutions to the Stokes equations, we exploit the fact that the equations result from minimizing a certain "energy" functional. So solutions are given by the Euler-Lagrange equation from the calculus of variations. We combine this with a result from functional analysis known as the Lax-Milgram theorem.

The second part gives an introduction to the finite element method, and derives the mixed formulation for the Stokes equation. Then the least-squares finite element method is introduced, and three different formulations are derived, including one from Chang and Yang (2002, [5]). We then compare the performance of the methods by doing convergence analysis in both 2D and 3D, and the results are compared with [5]. Since the goal is to use the methods for topology optimization, we pay particular attention to how accurately the methods calculate certain energy functionals. Since the method of topology optimization (that we will be using) is based on minimizing such functionals under certain constraints, it is critical that they are calculated accurately.

In the third part of the paper, a methodology for topology optimization of Stokes flow is introduced, based on the paper by Borrvall & Petersson (2003, [3]). In order to control the fluid flow in a given domain, a penalty term is added to the Stokes equations. The penalty term will emulate the presence of a solid in the domain, and can thus control where the fluid flows or doesn't flow. To avoid trivial solutions, the solid is only allowed to fill part of the domain. The problem is solved numerically using finite elements, and the penalty term is specified to be element-wise constant, thus allowing us to control the amount of flow in each element. The objective is to place the solid in such a way as to minimize the total dissipated power - defined by an energy functional. The resulting problem is a constrained optimization problem, which will be solved using the Method of Moving Asymptotes ([17]). We will look at several different numerical examples from [3], and investigate how well they are solved by the finite element methods from part 2.

For an explanation of the notation used in the paper, see Appendix A. The rest of the appendix contains reviews of several areas of mathematics that are relevant for the paper, such as Sobolev spaces, the calculus of variations and measure theory.

Chapter 2
The Stokes equations

2.1 A brief review of fluid mechanics

2.1.1 Notation and operators

This section will give a brief introduction to fluid mechanics following [1], and the Stokes equations will be derived as a special case of the incompressible Navier-Stokes equations. The aim is to give the reader an idea of the physical intuition behind the Stokes equations, not mathematical rigor.

We will work in a Cartesian coordinate system in either two or three dimensions. The position vector of an infinitesimal fluid element will be denoted by $\boldsymbol{x}$, and the fluid's velocity at this point will be denoted by $\boldsymbol{u}$. In three dimensions we have

$$\boldsymbol{x} = (x_1, x_2, x_3)$$
$$\boldsymbol{u}(\boldsymbol{x}) = (u_1(\boldsymbol{x}), u_2(\boldsymbol{x}), u_3(\boldsymbol{x}))$$

A fluid can be thought of as an aggregate body of many small 'fluid elements'. In this case each fluid element will be defined as a cube with side length l traveling at a speed s. The fluid has a constant density ρ and constant viscosity μ. The ratio of the two is called kinematic viscosity, $\nu = \mu/\rho$. There are three main forces acting on the fluid: body forces, inertial force and viscous force.

- **Body/volume force**: An external force acting on the fluid. This could for instance be an applied force like a push, or a natural force like gravity. The gravitational force is assumed to be proportional to the mass of fluid times the gravitational acceleration g,

  $$F_g \propto \rho l^3 g.$$

  More generally we can represent an element of force due to a source $\mathbf{V}$ of volume force as

  $$\delta\mathbf{F} = \mathbf{V}\delta U,$$

  where $U \subset \mathbb{R}^3$ denotes the subset of Euclidean space occupied by the fluid element, and the δ operator denotes a small variation that can be interpreted as

  $$\delta\mathbf{F} = \lim_{\tau \to 0} \frac{\mathbf{F}(\boldsymbol{x} + \tau\boldsymbol{y}) - \mathbf{F}(\boldsymbol{x})}{\tau}, \quad \boldsymbol{x}, \boldsymbol{y} \in U.$$

- **Inertia**: The inertial force is proportional to the fluid mass times it's change in velocity over time. The force required to bring a given mass of fluid to rest over a distance L is

$$F_I \propto \rho l^3 \frac{\Delta v}{\Delta t} = \rho l^3 \frac{s}{l/s} = \rho l^2 s^2$$

- **Viscous**: The viscous force is proportional to the dynamic viscosity of the fluid times the rate of change of velocity with distance, times the surface area of the fluid element. For example consider a sheer flow acting on one face of a cubic fluid element with area $l \times l$. If the velocity goes from 0 to s from the bottom to the top of the face, then the force acting on the face will be

$$F_V \propto \mu \frac{\Delta \boldsymbol{u}}{\Delta x_3} l^2 = \mu l s.$$

The inertial and viscous forces are short-range forces of direct molecular origin and are known as *surface* forces. They decrease very rapidly when distances grow larger than the molecular scale. In an analogous way to the representation of shear forces in solid mechanics, fluid surface forces can be represented by means of a stress tensor. Let $\Gamma = \partial U$ denote the boundary of U. Then

$$\delta \mathbf{F} = \mathbf{T} \cdot \boldsymbol{n} \delta \Gamma,$$

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

which is here shown in its 3-dimensional form. So an infinitesimal change in the surface force is proportional to the outward pointing component of the stress tensor times an infinitesimal change in the (size of the) boundary. Using an argument from continuum mechanics, it can be shown that the stress tensor is symmetric, i.e. that $T_{ij} = T_{ji}$ [1].

The ratio between the inertial and viscous forces are known as the Reynold's number of the fluid, denoted by

$$Re = \frac{\text{inertial}}{\text{viscous}} = \frac{\rho s l}{\mu} = \frac{s l}{\nu}. \tag{2.1}$$

In this paper we work with Stokes flow that has a low Reynold's number, i.e. the viscous forces dominate the inertial forces.

For a review of important operators from vector calculus such as the divergence, curl and Laplacian, see appendix A.

2.1.2 Conservation laws

We begin by choosing a material volume of fluid $U(t)$, that is a volume which changes over time in a way in which it always contains the same group of fluid particles. Conservation of mass requires that the rate of change of the mass within this volume is zero:

$$\frac{d}{dt} \int_U \rho \, \mathrm{d}x = 0. \tag{2.2}$$

Conservation of momentum (Newton's second law) states that the rate of change of fluid momentum (mass times velocity) must be balanced by the total external force applied to the fluid. As discussed above, there are two main forces acting on the fluid: gravity and viscous friction. In this

paper gravity and other external forces will be neglected for the sake of simplicity. Conservation of momentum takes the following form:

$$\frac{d}{dt}\int_U \rho \boldsymbol{u}\,\mathrm{d}x = \int_\Gamma \mathbf{T}\cdot\boldsymbol{n}\,\mathrm{d}S + \int_U \boldsymbol{f}\,\mathrm{d}x, \tag{2.3}$$

where $\Gamma = \partial U$ denotes the boundary of U and $\boldsymbol{f}$ denotes a body force. To move the time derivative inside the integral we need the following theorem [20], which can be seen as a generalization of the Leibniz integration rule [1] to three dimensions.

Theorem 2.1 (Reynold's Transport Theorem)**.** *Suppose U is a region in Euclidean space with boundary ∂U, and let $\boldsymbol{n}(\boldsymbol{x},t)$ be the outward unit normal to the boundary at time t. Let $\boldsymbol{x}(t)$ be the positions of points in the region, $\boldsymbol{u}(\boldsymbol{x},t) \in C^{2,1}(U)$ the velocity field in the region, and let $\boldsymbol{f}(\boldsymbol{x},t) \in C^{2,1}$ be a vector field in the region (it may also be a scalar field). Then*

$$\frac{d}{dt}\int_U \boldsymbol{f}\,\mathrm{d}x = \int_U \frac{\partial \boldsymbol{f}}{\partial t}\,\mathrm{d}x + \int_{\partial U}(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{f}\,\mathrm{d}S. \tag{2.4}$$

Proof. See [20]. □

Applying Reynold's transport theorem to the mass conservation equation (2.2) gives

$$\int_U \frac{\partial \rho}{\partial t} + \rho\,\mathrm{div}\,(\boldsymbol{u})\;\mathrm{d}x = 0 \Rightarrow \frac{\partial \rho}{\partial t} + \rho\,\mathrm{div}\,(\boldsymbol{u}) = 0,$$

because the choice of fluid volume was arbitrary, and thus conservation of mass must also hold point-wise. Since we assume that fluid density is constant we obtain

$$\rho\,\mathrm{div}\,(\boldsymbol{u}) = 0 \Rightarrow \mathrm{div}\,(\boldsymbol{u}) = 0.$$

This is known as the incompressibility condition. It follows from the combination of constant fluid density and conservation of mass.

The transport theorem applied to conservation of momentum, equation (2.3), results in

$$\rho\int_U \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot D)\boldsymbol{u}\,\mathrm{d}x = \int_\Gamma \mathbf{T}\cdot\boldsymbol{n}\,\mathrm{d}S + \int_U \boldsymbol{f}\,\mathrm{d}x.$$

Applying the divergence theorem to the tensor term converts the surface integral to a volume integral:

$$\begin{aligned} \rho\int_U \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot D)\boldsymbol{u}\,\mathrm{d}x &= \int_U \mathrm{div}\,(\mathbf{T})\;\mathrm{d}x + \int_U \boldsymbol{f}\,\mathrm{d}x \\ \Rightarrow \quad \rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot D)\boldsymbol{u}\right) &= \mathrm{div}\,(\mathbf{T}) + \boldsymbol{f} \end{aligned} \tag{2.5}$$

Combining (2.5) with (2.1.2) we obtain the incompressible Navier-Stokes equations, here shown with Dirichlet velocity boundary conditions. Assume $\boldsymbol{u}(\boldsymbol{x},t), \boldsymbol{g}(\boldsymbol{x},t)$ are $C^{2,1}(U)$ vector fields. Then

$$\begin{aligned} \rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot D)\boldsymbol{u}\right) &= \mathrm{div}\,(\mathbf{T}) + \boldsymbol{f} \qquad \text{in } U \\ \mathrm{div}\,(\boldsymbol{u}) &= 0 \qquad \text{in } U \\ \boldsymbol{u} &= \boldsymbol{g} \qquad \text{on } \partial U \\ \int_{\partial U} \boldsymbol{n}\cdot\boldsymbol{g}\,\mathrm{d}x &= 0, \end{aligned} \tag{2.6}$$

[1] Given an integral of the form $\int_c^d f(x,y)\,\mathrm{d}y$, then for $x \in (a,b)$ the derivative of this integral may be expressed as $\frac{d}{dx}\int_c^d f(x,y)\,\mathrm{d}y = \int_c^d \frac{\partial}{\partial x} f(x,y)\,\mathrm{d}y$ provided that f and $\frac{\partial f}{\partial x}$ are continuous in $[a,b]\times[c,d]$.

where $\boldsymbol{g}$ is the prescribed velocity on ∂U. The last equation ensures conservation of mass in U, as well as incompressibility because

$$\int_U \operatorname{div}(\boldsymbol{u})\, \mathrm{d}x = \int_{\partial U} \boldsymbol{n} \cdot \boldsymbol{u}\, \mathrm{d}x = \int_{\partial U} \boldsymbol{n} \cdot \boldsymbol{g}\, \mathrm{d}x = 0.$$

If we assume that the fluid is Newtonian with constant kinematic viscosity ν, we can express the Cauchy stress tensor $\mathbf{T}$ in terms of the pressure $p(\boldsymbol{x}, t) \in C^{1,1}(U)$, and the flow velocity $\boldsymbol{u}$:

$$\mathbf{T} = -p\mathbf{I} + 2\nu\mathbf{D}(\boldsymbol{u}) \tag{2.7}$$

where the rate of strain tensor $\mathbf{D}(\boldsymbol{u})$ is given by

$$\mathbf{D}(\boldsymbol{u}) = \frac{1}{2}\left(D\boldsymbol{u} + (D\boldsymbol{u})^T\right). \tag{2.8}$$

Plugging these expressions into the first of (2.6) gives

$$\begin{aligned}\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot D)\boldsymbol{u}\right) &= \operatorname{div}(2\nu\mathbf{D}(\boldsymbol{u})) - Dp + \boldsymbol{f} \\ &= \nu\Delta\boldsymbol{u} - Dp + \boldsymbol{f}\end{aligned}$$

For slow and steady flow the time derivative vanishes, and the convection term becomes insignificant and is therefore neglected. This leads to the Stokes equations:

$$\begin{cases} -\nu\,\Delta\boldsymbol{u} + Dp = \boldsymbol{f} & \text{in } U \\ \operatorname{div}(\boldsymbol{u}) = 0 & \text{in } U \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \partial U \\ \int_{\partial U} \boldsymbol{n} \cdot \boldsymbol{g}\, \mathrm{d}x = 0. \end{cases} \tag{2.9}$$

2.1.3 Weak formulation of the Stokes equations

Let the domain $U \subset \mathbb{R}^n$ be bounded, open and connected. We begin by assuming that the pair $(\boldsymbol{u}, p) \in C^2(U) \times C^1(U)$ is a classical solution to the Stokes problem (2.9). Since our objective is to solve (2.9) using the finite element method, the requirement that $\boldsymbol{u} \in C^2(U)$ and $p \in C^1(U)$ is problematic. This is because we wish to approximate $(\boldsymbol{u}, p)$ with functions $(\boldsymbol{u}_h, p_h)$ that are piecewise continuous polynomials, and therefore $(\boldsymbol{u}_h, p_h) \notin C^2(U) \times C^1(U)$. To get around this issue we look for 'weak' solutions to the Stokes problem that place less severe restrictions on the smoothness of $(\boldsymbol{u}, p)$, and therefore on our approximations $(\boldsymbol{u}_h, p_h)$. We wish to express the Stokes problem in weak form, and interpret the derivatives in the resulting expression as weak derivatives. The weak solutions can then be found in so-called Sobolev spaces. See appendix B for a review of weak derivatives and Sobolev spaces.

This is not the only reason for seeking weak solutions, however. As we shall see, when the Stokes problem is formulated in weak form, we can use results from functional analysis to prove the existence and uniqueness of solutions, a task that would be more difficult to accomplish for classical solutions to (2.9).

To formulate the Stokes problem in weak form we first of all distinguish between trial functions and test functions. Let $\boldsymbol{v} \in H_0^1(U)$ be a test function that vanishes on the boundary of U, and let $q \in L^2(U)$. The weak formulation of the Stokes equations (2.9) is obtained by multiplying the first

equation of (2.9) by $\boldsymbol{v}$, multiplying the second equation by q, and integrating over U. We then ask that the resulting equations hold for all test functions $(\boldsymbol{v}, q) \in H_0^1(U) \times L^2(U)$:

$$\begin{aligned} -\nu \int_U \Delta \boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}x + \int_U Dp \cdot \boldsymbol{v} \, \mathrm{d}x &= \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x \\ \Rightarrow \nu \int_U D\boldsymbol{u} : D\boldsymbol{v} \, \mathrm{d}x - \int_U p \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x &= \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x, \qquad \forall \boldsymbol{v} \in H_0^1(U) \\ \int_U \operatorname{div}(\boldsymbol{u}) \, q \, \mathrm{d}x &= 0, \qquad \forall q \in L^2(U), \end{aligned} \tag{2.10}$$

where

$$D\boldsymbol{u} : D\boldsymbol{v} = \sum_{i,j=1}^{n} (D\boldsymbol{u})_{ij} (D\boldsymbol{v})_{ij} = \sum_{i,j=1}^{n} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j}.$$

Partial integration has been applied to the first equation, and the resulting boundary term vanishes since $\boldsymbol{v}$ vanishes on ∂U. Note that the pressure function $p \in L^2(U)$ in the term

$$-\int_U p \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x$$

can be interpreted as a Lagrange multiplier for the incompressibility condition $\operatorname{div}(\boldsymbol{u}) = 0$ (see Appendix G for an explanation of the Lagrange multiplier method). We will use this fact in the following section to prove the existence of solutions to the PDE system (2.10).

2.2 Existence and uniqueness of solutions to the Stokes problem

Let the domain $U \subset \mathbb{R}^n$ be bounded, open and connected. We begin by defining the following subspace of $H_0^1(U)$:

$$\mathcal{A} = \left\{ \boldsymbol{v} \in H_0^1(U) \mid \operatorname{div}(\boldsymbol{v}) = 0) \right\}$$

where div is the divergence operator

$$\operatorname{div} : \boldsymbol{v} \in H_0^1(U) \to \operatorname{div}(\boldsymbol{v}) \in L^2(U).$$

So $\mathcal{A}$ is the kernel of div. Equip $\mathcal{A}$ with the inner product

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{H_0^1(U)} = \int_U D\boldsymbol{u} : D\boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}x,$$

and corresponding norm

$$\|\boldsymbol{u}\|_{H_0^1(U)} := \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle_{H_0^1(U)}} = \left(\int_U |D\boldsymbol{u}|^2 + |\boldsymbol{u}|^2 \, \mathrm{d}x \right)^{1/2}.$$

The continuity of the div operator follows from the following estimate, which holds for all $\boldsymbol{v} \in H_0^1(U)$:

$$\|\operatorname{div}(\boldsymbol{v})\|_{L^2(U)}^2 = \sum_{i=1}^{n} \int_U \left| \frac{\partial v_i}{\partial x_i} \right|^2 \mathrm{d}x \leq \int_U |D\boldsymbol{v}|^2 \, \mathrm{d}x \leq \|\boldsymbol{v}\|_{H_0^1(U)}^2.$$

Hence, $\mathcal{A}$ is a closed subspace of $H_0^1(U)$ and $\mathcal{A}$ is a Hilbert space.

We shall need the following result from functional analysis.

Theorem 2.2 (Lax-Milgram). *Assume that H is a Hilbert space and that $a : H \to H$ is a bilinear mapping for which there exist $\alpha, \beta > 0$ such that*

(i) $|a(u, v)| \leq \alpha \|u\| \|v\| \quad \forall u, v \in H,$

(ii) $\beta \|u\|^2 \leq a(u,u) \quad \forall u \in H.$

Let $f : H \to \mathbb{R}$ *be a bounded linear functional on* H*. Then there exists a unique* $u \in H$ *such that*

$$a(u,v) = f(v) \quad \forall v \in H. \tag{2.11}$$

Proof. See appendix B.3. □

Requirement (i) guarantees that the bilinear functional $a(u,v)$ is continuous in both arguments. The second requirement (ii) means that $a(u,v)$ is *coercive*. Note that if $a(u,v)$ is symmetric, that is if $a(u,v) = a(v,u)$, $u, v \in H$, then $\langle u, v\rangle := a(u,v)$ defines an inner product on H, and then the Riesz Representation Theorem (see appendix F) can be applied to show the existence and uniqueness of u. So the significance of Theorem 2.2 is that it does not require $a(u,v)$ to be symmetric.

The variational formulation of the Stokes problem can now be stated as the following theorem [13].

Theorem 2.3 (Stokes' problem)**.** *(i) For every* $\boldsymbol{f} \in L^2(U)$ *there exists a unique* $\boldsymbol{u} \in \mathcal{A}$ *which satisfies*

$$\nu \int_U D\boldsymbol{u} : D\boldsymbol{v} \, \mathrm{d}x = \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x \quad \forall v \in \mathcal{A}. \tag{2.12}$$

(ii) Let $\boldsymbol{u}$ *be the solution of* (2.12)*. Then the relation* (2.12) *determines a unique pressure function* $p \in L^2(U)$ *(up to an additive constant) such that the couple* $(\boldsymbol{u}, p) \in \mathcal{A} \times L^2(U)$ *satisfies*

$$a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = l(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in H_0^1(U), \tag{2.13}$$

where

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) &= \nu \int_U D\boldsymbol{u} \cdot D\boldsymbol{v} \, \mathrm{d}x \\ b(\boldsymbol{v}, p) &= -\int_U p \, \mathrm{div}\,(\boldsymbol{v}) \, \mathrm{d}x \\ l(\boldsymbol{v}) &= \langle \boldsymbol{f}, \boldsymbol{v} \rangle_{L^2(U)} = \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x. \end{aligned} \tag{2.14}$$

(iii) The pair $(\boldsymbol{u}, p)$ *is a weak solution of* (2.9)*.*

Proof. Set $\nu = 1$ in the following. We will start by showing the existence of a solution $\boldsymbol{u} \in \mathcal{A}$ to (2.12), and for that purpose we use the Lax-Milgram Theorem 2.2. We set $H = \mathcal{A}$ in (2.11) and see that the bilinear form $a : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ and the linear form $f : \mathcal{A} \to \mathbb{R}$ are continuous because, using the Cauchy-Schwartz inequality,

$$|a(\boldsymbol{u}, \boldsymbol{v})| = \left| \langle D\boldsymbol{u}, D\boldsymbol{v} \rangle_{L^2(U)} \right| \leq \|D\boldsymbol{u}\|_{L^2(U)} \|D\boldsymbol{v}\|_{L^2(U)} \leq \|\boldsymbol{u}\|_{H_0^1(U)} \|\boldsymbol{v}\|_{H_0^1(U)},$$

and

$$|l(\boldsymbol{v})| = \left| \langle \boldsymbol{f}, \boldsymbol{v} \rangle_{L^2(U)} \right| \leq \|\boldsymbol{f}\|_{L^2(U)} \|\boldsymbol{v}\|_{L^2(U)} \leq \|\boldsymbol{f}\|_{L^2(U)} \|\boldsymbol{v}\|_{H_0^1(U)}.$$

Condition (i) in Theorem 2.2 is thus satisfied. To see that the coercivity condition (ii) is also satisfied we apply the Poincaré-Friedrichs inequality (B.6). It states that there exists a constant c_* such that

$$\|\boldsymbol{u}\|_{L^2(U)} \leq \sqrt{c_*} \|D\boldsymbol{u}\|_{L^2(U)}. \tag{2.15}$$

We have to show that there exists a constant β such that

$$a(\boldsymbol{u}, \boldsymbol{u}) \geq \beta \|\boldsymbol{u}\|^2_{H_0^1(U)}.$$

Given the constant c_* from (2.15) we set $\beta = c_*/(1+c_*)$, and then $c_* = \dfrac{\beta}{1-\beta}$. Inserting this in (2.15) we see that

$$\begin{aligned}
&\|D\boldsymbol{u}\|^2_{L^2(U)} \geq \frac{\beta}{1-\beta}\|\boldsymbol{u}\|^2_{L^2(U)} \\
\Rightarrow &(1-\beta)\|D\boldsymbol{u}\|^2_{L^2(U)} \geq \beta\|\boldsymbol{u}\|^2_{L^2(U)} \\
\Rightarrow &\|D\boldsymbol{u}\|^2_{L^2(U)} \geq \beta\left(\|\boldsymbol{u}\|^2_{L^2(U)} + \|D\boldsymbol{u}\|^2_{L^2(U)}\right) = \beta\|\boldsymbol{u}\|^2_{H^1_0(U)} \\
\Rightarrow &a(\boldsymbol{u},\boldsymbol{u}) \geq \beta\|\boldsymbol{u}\|^2_{H^1_0(U)},
\end{aligned}$$

because $a(\boldsymbol{u},\boldsymbol{u}) = \|D\boldsymbol{u}\|^2_{L^2(U)}$. This shows that condition (ii) in Theorem 2.2 is also satisfied, which implies that there exists a unique solution $\boldsymbol{u}$ to (2.12).

The existence of p is more difficult to show, and we will need the following theorem from [8], which assumes that $U \subset \mathbb{R}^3$.

Theorem 2.4 (Pressure as a Lagrange multiplier)**.** *Let the function $\boldsymbol{u}$ belong to the 'admissible set' $\mathcal{A}$ and $\boldsymbol{f} \in L^2(U;\mathbb{R}^3)$ be given. Then there exists a scalar function $p \in L^2_{loc}(U)$ such that*

$$\int_U D\boldsymbol{u} : D\boldsymbol{v}\,\mathrm{d}x - \int_U p\,\mathrm{div}\,(\boldsymbol{v})\,\mathrm{d}x = \int_U \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}x \tag{2.16}$$

for all $\boldsymbol{v} \in H^1(U;\mathbb{R}^3)$ with compact support within U.

Proof. See appendix D.4. □

The above theorem proofs the existence of $(\boldsymbol{u}, p)$ as claimed in (2.13). Thus we have proven (ii). To show (iii) we refer to the derivation of (2.10) from (2.9). This is precisely the content of (iii). □

For future reference we restate the variational formulation of the Stokes problem as:

$$\begin{cases}
\text{find } (\boldsymbol{u},p) \in \mathcal{A}\times L^2(U) \text{ such that} & \\
a(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{v},p) = l(\boldsymbol{v}) & \forall \boldsymbol{v} \in H^1_0(U) \\
b(\boldsymbol{u},q) = 0 & \forall q \in L^2(U)
\end{cases} \tag{2.17}$$

where

$$\begin{aligned}
a(\boldsymbol{u},\boldsymbol{v}) &= \nu\int_U D\boldsymbol{u}\cdot D\boldsymbol{v}\,\mathrm{d}x \\
b(\boldsymbol{v},p) &= -\int_U p\,\mathrm{div}\,(\boldsymbol{v})\,\mathrm{d}x \\
l(\boldsymbol{v}) &= \langle \boldsymbol{f},\boldsymbol{v}\rangle_{L^2(U)} = \int_U f\cdot\boldsymbol{v}\,\mathrm{d}x.
\end{aligned} \tag{2.18}$$

2.3 Boundary conditions in Sobolev spaces

When we impose the boundary conditions

$$\begin{cases}
\boldsymbol{u} = \boldsymbol{g} & \text{on } \partial U \\
\displaystyle\int_{\partial U} \boldsymbol{n}\cdot\boldsymbol{g}\,\mathrm{d}x = 0 &
\end{cases} \tag{2.19}$$

from (2.9) on the variational problem (2.17), we have to be careful, because the boundary ∂U can no longer be interpreted in the usual point-wise sense. That is $\boldsymbol{u}|_{\partial U}$ is not defined point-wise, but rather as an L^2-function. To get around this we shall assume that U is a Lipschitz domain with a Lipschitz boundary (see Section C.2 in Appendix C or [4] for more details):

Definition 2.5 (Lipschitz boundary)**.** We say U has a Lipschitz boundary ∂U provided there exists a collection of open sets O_i, a positive parameter ϵ, an integer N, and a finite number M, such that

(i) for all $x \in \partial U$, there is an open set O_i such that $\mathrm{B}(x, \epsilon) \subset O_i$,

(ii) no more than N of the sets O_i intersect non-trivially,

(iii) and each domain $O_i \cap U = O_i \cap U_i$, where U_i is a domain whose boundary is a graph of a Lipschitz function ϕ_i. That is

$$U_i = \left\{ (x, y)\mathbb{R}^{n-1} \times \mathbb{R} \mid y < \phi_i(x) \right\}$$

satisfying

$$\|\phi_i\|_{\mathrm{Lip}(\mathbb{R}^{n-1})} \leq M.$$

We will then interpret the boundary conditions 2.19 in the 'trace sense'. This means that we interpret $\boldsymbol{u}|_{\partial U}$ as the image of a mapping

$$T : W^{1,p}(U) \to L^p(\partial U)$$

such that

$$Tu = u|_{\partial U} \quad \text{if} \quad u \in W^{1,p}(U).$$

We will not go into the details of the existence of the 'trace operator' T, but instead consider the following theorem from [4], that tells us how to interpret $\boldsymbol{u}|_{\partial U}$.

Theorem 2.6 (Trace theorem for Lipschitz domain)**.** *Suppose that U has a Lipschitz boundary, and that p is a real number in the range $1 \leq p \leq \infty$. Then there exists a constant C such that*

$$\|v\|_{L^p(\partial U)} \leq C \|v\|_{L^p(U)}^{1-1/p} \|v\|_{W^{1,p}}^{1/p} \quad \forall v \in W^{1,p}(U).$$

So as long as U has a Lipschitz boundary, the restriction of $\boldsymbol{u}$ to the boundary, $\boldsymbol{u}|_{\partial U}$, can be interpreted as an L^p-function. This has some curious implications, however, such as the possibility of $\boldsymbol{u}|_{\partial U}$ being unbounded at a dense set of points. See Section C.2 in Appendix C for further details.

Chapter 3
Solving the Stokes equations using finite elements

In this chapter the mixed formulation and LS-FEM method are introduced and compared, and numerical results are presented.

3.1 Galerkin schemes

The finite element method belongs to a class of numerical methods known as Galerkin methods. The idea is to approximate the solution $u \in V$ to a problem in the infinite dimensional space V, by using a finite dimensional subspace of V. This is called Galerkin approximation. From [13] we have the following definition.

Definition 3.1. Let V be a Banach space. A Galerkin approximation scheme is a sequence $\{V_i\}_{i=1}^{\infty}$ of finite dimensional subspaces of V such that for all $v \in V$, there exists some sequence $\{v_i\}_{i=1}^{\infty}$ with $v_i \in V_i$ for all $i \in \mathbb{N}$, and the sequence $\{v_i\}_{i=1}^{\infty}$ norm-converges to v in V, that is

$$\lim_{i \to \infty} \inf_{w_i \in V_i} \|v - w_i\|_V = 0.$$

The following proposition tells us how to go about constructing a Galerkin scheme.

Proposition 3.2. *Let V be a separable Banach space. Then one can construct a Galerkin approximation scheme $\{V_i\}_{i=1}^{\infty}$ by the following method:*

(i) take $\{u_i\}_{i=1}^{\infty}$ a countable dense subset of V

(ii) and let $V_i = span\,\{u_1, u_2, ..., u_i\}$.

Then the subspaces $\{V_i\}_{i=1}^{\infty}$ satisfy

$$V_1 \subset V_2 \subset \cdots \subset V_i \subset \cdots, \quad \bigcup_{i=1}^{\infty} V_i = V,$$

and $\{V_i\}_{i=1}^{\infty}$ is a Galerkin scheme.

Proof. See [13] p.74. □

Suppose that V is a separable Hilbert space and $a : V \times V \to \mathbb{R}$ is a bilinear, continuous, coercive form, and $l : V \to \mathbb{R}$ is a linear, continuous form. Our goal is to use a Galerkin scheme to solve variational problems in the form

$$\begin{cases} \text{find } u \in V \text{ such that} \\ a(u, v) = l(v) \quad \forall v \in V. \end{cases} \tag{3.1}$$

Since V is separable, proposition 3.2 tells us that there exists a Galerkin scheme $\{V_i\}_{i=1}^{\infty}$, that leads to the following approximated problem.

$$\begin{cases} \text{find } u_i \in V_i \text{ such that} \\ a(u_i, v_i) = l(v_i) \quad \forall v_i \in V_i. \end{cases} \tag{3.2}$$

To show that solutions $u_i \in V_i$ to this approximated problem actually converge to the true solution $u \in V$, we need to show that u_i norm-converges to u under suitable assumptions on the forms $a(\cdot, \cdot)$ and $l(\cdot)$. The requirements of the Lax-Milgram Theorem 2.2 turn out to be sufficient, which motivates the following lemma.

Lemma 3.3 (Céa's lemma)**.** *Let V be a separable Hilbert space and $\{V_i\}_{i=1}^{\infty}$ a Galerkin scheme. Suppose*

$$a(u_i, v_i) = l(v_i) \quad \forall v_i \in V_i, \quad u_i \in V_i$$

and

$$a(u, v) = l(v) \quad \forall v \in V, \quad u \in V,$$

where $a(\cdot, \cdot)$ and $l(\cdot)$ satisfy the assumptions of the Lax-Milgram theorem 2.2 with constants α, β. Then

$$\|u_i - u\| \leq \frac{\alpha}{\beta} \inf_{w \in V_i} \|u - w\| .$$

Proof. See [13] p.76. □

Thus we have proved the validity of the Galerkin scheme for variational problems of the form 3.1 that satisfy the assumptions of the Lax-Milgram theorem. We can therefore construct a Galerkin scheme to solve our main problem of interest, the Stokes problem 2.17.

3.2 The finite element method

The concept of a Galerkin scheme presented so far is quite general, and does not reveal how we intend to calculate an approximate solution to a PDE problem. To do this we will introduce the finite element method, which is a Galerkin approximation scheme in which the finite dimensional subspaces V_i consist of a finite number of 'elements', each equipped with a basis consisting of piecewise polynomials. To make this more clear, we first need a few definitions (following [18]).

Definition 3.4 (Finite element)**.** Let us suppose that

(i) $K \subset \mathbb{R}^n$ is a simply connected bounded open set with piecewise smooth boundary (the element domain);

(ii) $\mathcal{P}$ is a finite-dimensional space of functions defined on K (the space of shape functions);

(iii) $\mathcal{N} = \{N_1, ..., N_k\}$ is a basis for $\mathcal{P}'$ (the set of nodal variables), where $\mathcal{P}'$ denotes the algebraic dual of the linear space $\mathcal{P}$. Then $(K, \mathcal{P}, \mathcal{N})$ is called a finite element.

Definition 3.5 (Nodal basis)**.** Let $(K, \mathcal{P}, \mathcal{N})$ be a finite element, and let $\{\psi_1, \psi_2, ..., \psi_k\}$ be a basis for $\mathcal{P}$, dual to $\mathcal{N}$; namely

$$N_i(\psi_j) = \delta_{ij}, \quad 1 \leq i, j \leq k.$$

Such a basis is called a nodal basis for $\mathcal{P}$.

Now we give an equivalent characterization of condition (iii) in Definition 3.4.

Lemma 3.6. *Let $\mathcal{P}$ be a k-dimensional linear space of functions on $\mathbb{R}^n$, and suppose that $\{N_1, N_2, ..., N_k\}$ is a linearly independent subset of the dual space $\mathcal{P}'$. Then the following two statements are equivalent:*

(i) $\{N_1, N_2, ..., N_k\}$ is a basis for $\mathcal{P}'$;

(ii) Given that $v \in \mathcal{P}$ and $N_i(v) = 0$ for $i = 1, ..., k$, then $v = 0$.

Proof. See [18] p.66. □

This result motivates the following definition.

Definition 3.7. We say that $\mathcal{N}$ determines $\mathcal{P}$ if $\psi \in \mathcal{P}$ with $N(\psi) = 0$ for all $N \in \mathcal{N}$ implies that $\psi = 0$. Furthermore, if $\mathcal{N}$ determines $\mathcal{P}$ on an element domain K, then $(K, \mathcal{P}, \mathcal{N})$ is a valid finite element.

As an example of a finite element we first consider the linear Lagrange element in $\mathbb{R}^2$. Let

$$\mathcal{P}_k := \{\text{all polynomials of degree } \leq k \text{ in two variables}\},$$

let K be a triangle, set $k = 1$. Take $\mathcal{P} = \mathcal{P}_1, \mathcal{N} = \mathcal{N}_1 = \{N_1, N_2, N_3\}$ (and therefore the dimension of $\mathcal{P}_1$ is 3), where $N_i(v) = v(z_i)$ and z_1, z_2, z_3 are the vertices of the triangle K, as shown in figure 3.1(a). Then it can be shown (see [18]) that $\mathcal{N}_1$ determines $\mathcal{P}_1$, and therefore the linear Lagrange element is a valid finite element.

As a second example we consider the quadratic Lagrange element in $\mathbb{R}^2$. See figure 3.1(b). Take $k = 2$ and take $\mathcal{P} = \mathcal{P}_2, \mathcal{N} = \mathcal{N}_2 = \{N_1, N_2, ..., N_6\}$ (and therefore the dimension of $\mathcal{P}_2$ is 6), where

$$N_i(v) = \begin{cases} v(\text{at the ith vertex of the triangle}), & i = 1, 2, 3 \\ v(\text{at the midpoint of the (i-3)rd edge}), & i = 4, 5, 6. \end{cases}$$

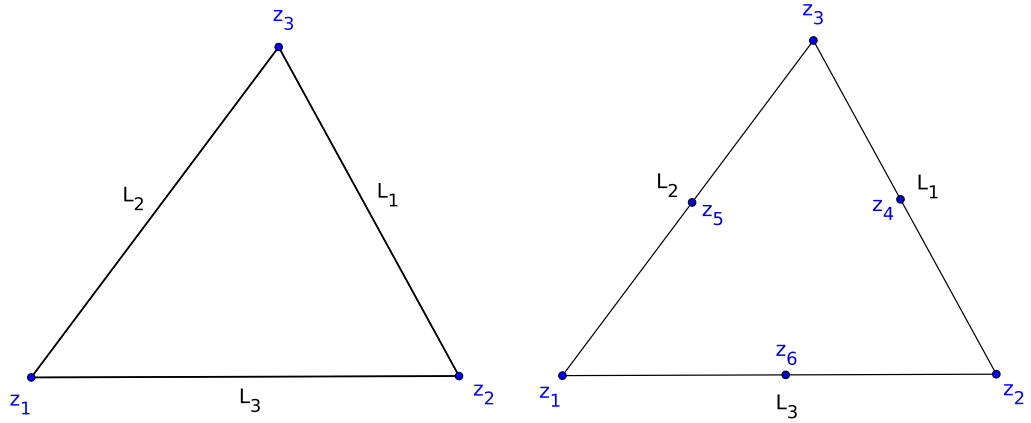
It can again be shown that $\mathcal{N}_2$ determines $\mathcal{P}_2$, and therefore the quadratic Lagrange element is a valid finite element.

Having described two important finite elements, we now wish to piece them together to construct finite-dimensional subspaces of Sobolev spaces.

Definition 3.8 (Local interpolant)**.** Let $(K, \mathcal{P}, \mathcal{N})$ be a finite element and let the set $\{\phi_i : i = 1, ..., k\}$ be a basis for $\mathcal{P}$ dual to $\mathcal{N}$. Given that v is a function for which all $N_i \in \mathcal{N}$, $i = 1, ..., k$, are defined, we introduce the local interpolant $\mathcal{I}_K v$ by

$$\mathcal{I}_K v = \sum_{i=1}^{k} N_i(v)\psi_i.$$

The key properties of the local interpolant are here summarized.



(a) First order (b) Second order

Figure 3.1: Lagrange elements

Lemma 3.9. *The local interpolant has the following properties:*

1. *The mapping $v \mapsto \mathcal{I}_K v$ is linear.*
2. $N_i(\mathcal{I}_K v) = N_i(v),\ i = 1, ..., k.$
3. *$\mathcal{I}_K v = v$ for $v \in \mathcal{P}$; consequently $\mathcal{I}_K$ is idempotent on $\mathcal{P}$, that is, $\mathcal{I}_K^2 = \mathcal{I}_K$.*

We now glue together the element domains to obtain a subdivision of the computational domain U, and merge the local interpolants to obtain a global interpolant.

Definition 3.10 (Subdivision)**.** A subdivision of the computational domain U is a finite collection of open sets $\mathcal{T} = \{K_i\}$ such that

(i) $K_i \cap K_j = \emptyset$ if $i \neq j$, and

(ii) $\bigcup_i \bar{K}_i = \bar{U}$.

Definition 3.11 (Global interpolant)**.** Suppose that U is a bounded open set in $\mathbb{R}^n$ with subdivision $\mathcal{T}$. Assume that each element domain $K \in \mathcal{T}$ is equipped with some type of shape functions $\mathcal{P}$ and nodal variables $\mathcal{N}$ such that $(K, \mathcal{P}, \mathcal{N})$ forms a valid finite element. Let m be the order of the highest partial derivative involved in the nodal variables. For $v \in C^m(\bar{U})$ the global interpolant $\mathcal{I}_h v$ is defined on $\bar{U}$ by

$$\mathcal{I}_h v|_{K_i} = \mathcal{I}_{K_i} v \quad \forall K_i \in \mathcal{T}.$$

To ensure that the global interpolant is continuous on U, the subdivision $\mathcal{T}$ must have certain properties.

Definition 3.12 (Triangulation)**.** A triangulation of a polygonal domain U is a subdivision of U consisting of triangles which have the property that

(iii) No vertex of any triangle lies in the interior of an edge of another triangle.

This is also known as a regular triangulation. We will also assume that the triangulation maximizes the minimum angle of all the angles of the triangles in the triangulation; which is

known as a Delaunay triangulation. We will from now on use the word triangulation to describe a subdivision of a domain $U \subset \mathbb{R}^n$ even if $n > 2$. In that case the term generalizes to tetrahedra (and then higher dimensional polytopes for $n > 3$).

When Lagrange elements are used on a regular triangulation, the resulting finite dimensional space is a subspace of H^1. The final ingredient needed to produce a Galerkin scheme is to ensure that there exists a sequence $v_{h_i} \in V_{h_i}$, $i = 1, 2, ...$, $h_i \to 0$ as $i \to \infty$, such that

$$\left\| v - v_{h_i} \right\|_V \to 0$$

where $v_{h_i} := \mathcal{I}_{h_i}(v)$ is the local interpolant of the true solution $v \in V$, V_{h_i} is a subspace of V for all $i \in \mathbb{N}$, and h is the characteristic size of the largest element in the triangulation of U. To this end we apply Theorem (4.4.20) from [4] (setting $m = 2$ and $s = 1$). Assume that $\left\{ \mathcal{T}^h \right\}$ is a non-degenerate family of subdivisions of a polyhedral domain $U \subset \mathbb{R}^n$. Assume furthermore that $v \in H^2(U) \cap H_0^1(U)$, then

$$\sum_{K \in \mathcal{T}^h} \|v - v_h\|_{H_0^1(U)} \leq Ch \, |v|_{H^2(U)} \, . \tag{3.3}$$

We see that the interpolation error goes to zero, as we refine the mesh, i.e. as $h \to 0$.

The requirement that $v \in H^2(U) \cap H_0^1(U)$ may be restrictive in some cases; we would prefer that $v \in H_0^1(U)$. This restriction may be circumvented by noting that H^2 is dense in $H_0^1(U)$. So for all $v \in H_0^1(U)$ there exists $v_\epsilon \in H^2 \cap H_0^1(U)$ such that

$$\|v - v_\epsilon\|_{H_0^1(U)} < \epsilon,$$

for an arbitrary $\epsilon > 0$. So we can approximate $v \in H^2(U) \cap H_0^1(U)$ arbitrarily well using an element $v_\epsilon \in H_0^1(U)$. The result in (3.3) can thus be extended to functions in $H_0^1(U)$. We conclude that it is possible to create a Galerkin scheme using finite elements.

3.3 Finite element setup for the Stokes problem

To solve the variational problem (2.17) using the finite element method, we take the following steps:

1. Subdivide the computational domain into triangular elements.

2. Define a finite element type for the velocity $\boldsymbol{u}$ and the pressure p, and resulting finite element spaces V_h and P_h.

3. Consider the variational problem (2.17) in the subspace from step 2, seeking an approximate solution $(\boldsymbol{u}_h, p_h) \in (V_h, P_h)$ to the true solution $(\boldsymbol{u}, p) \in V \times L^2(U)$.

4. Impose boundary conditions (and possibly a forcing term $\boldsymbol{f}$) on the finite element approximation.

5. Assemble a linear system of equations based on steps 1-4.

6. Solve the linear system for the unknowns $(\boldsymbol{u}_h, p_h)$.

3.3.1 A mixed finite element formulation for the Stokes equation

Let $\overline{U}_h$ denote a triangulation of the computational domain $\overline{U}$, where h denotes the characteristic size of the largest triangle in the subdivision. We will approximate the velocity $\boldsymbol{u}$ using quadratic Lagrange elements, and the pressure p using linear Lagrange elements. Denoting the approximate solutions by $(\boldsymbol{u}_h, p_h)$, we write

$$(\boldsymbol{u}_h, p_h) \in V_h \times P_h,$$

where V_h is a finite dimensional subspace of V and P_h is a finite dimensional subspace of P. The resulting mixed finite element space $W = V_h \times P_h$ is known as the Taylor-Hood elements, and is a stable standard element pair for the Stokes equations [15].

Consider now the following approximation of (2.17):

$$\begin{cases} \text{find } (\boldsymbol{u}_h, p_h) \in V_h \times P_h \text{ such that} \\ a(\boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) + b(\boldsymbol{u}_h, q_h) = l(\boldsymbol{v}_h) \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times P_h. \end{cases} \tag{3.4}$$

This is a mixed finite element formulation of the Stokes equations, which we will call the *mixed formulation* from now on.

3.4 The least-squares finite element method

The mixed formulation of the Stokes equations has the disadvantage that it leads to a linear system of equations that is symmetric, but indefinite. It has thus lost an important feature; a feature that is associated with what is known as the Rayleigh-Ritz setting. When solutions to PDE problems can be characterized as (unconstrained) minimizers of convex, quadratic functionals on Hilbert spaces, then methods that approximate such solutions based on a Galerkin scheme, such as the finite element method, are said to belong to the Rayleigh-Ritz setting. Finite element methods in the Rayleigh-Ritz setting have several advantageous features. In the words of [10]:

(i) general regions and boundary conditions are relatively easy to treat in a systematic manner;

(ii) the conformity of the finite element spaces suffices to guarantee the stability and optimal accuracy of the approximate solutions;

(iii) all variables can be approximated using a single type of finite element space, e.g., the same degree piecewise polynomials defined with respect to the same grid;

(iv) the resulting linear systems are

   a. sparse

   b. symmetric

   c. positive definite.

Since solutions to the Stokes problem solve a *constrained* minimization problem, the Stokes problem does not fit in the Rayleigh-Ritz setting. Solution methods such as the mixed formulation thus lose some of the features listed above; this particular method feature (iii) and (iv.c). Least-squares finite element methods (LS-FEMs) can be viewed as an attempt to recover some of these features. In fact, they offer the theoretical prospect of retaining all the advantages. The drawback is that not all LS-FEM formulations are practical, and the ones that are, might suffer from other drawbacks.

In section 3.5 we will present an LS-FEM method from [5]. In section 3.6 we present two alternative formulations from [2] that belong to the class of *quasi-norm-equivalent* methods. Finally, in section 3.8 we will test the performance of the LS-FEM methods against each other and the mixed formulation.

3.5 Solving the Stokes equations using LS-FEM

In this section, let $U \subset \mathbb{R}^n$ be an open, simply connected subset of $\mathbb{R}^n$, $n \in \mathbb{N}$. The first step in formulating a least-squares finite element method is to express the PDE system to be solved as a first order system. To do this for the Stokes system (2.9), we introduce the new variable

$$\boldsymbol{\omega} = \operatorname{curl}(\boldsymbol{u})$$

which is known as the vorticity. When $\operatorname{div}(\boldsymbol{u}) = 0$ we have the identity $\Delta \boldsymbol{u} = \operatorname{curl}(\boldsymbol{\omega})$, which leads to the first order system

$$\begin{cases} \nu \operatorname{curl}(\boldsymbol{\omega}) + Dp = \boldsymbol{f} & \text{in } U \\ \operatorname{curl}(\boldsymbol{u}) = \boldsymbol{\omega} & \text{in } U \\ \operatorname{div}(\boldsymbol{u}) = 0 & \text{in } U \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \partial U \\ \displaystyle\int_{\partial U} \boldsymbol{n} \cdot \boldsymbol{g} \, \mathrm{d}x = 0. \end{cases} \tag{3.5}$$

This is known as the velocity-vorticity-pressure (VVP) formulation of the Stokes equations. The second step is to let each equation in (3.5) be equal to zero, and call the resulting expressions the "residuals" of the equations. We define the "residual functional"

$$\mathcal{R}(\boldsymbol{u}, \boldsymbol{\omega}, p) = \|\nu \operatorname{curl}(\boldsymbol{\omega}) + Dp - \boldsymbol{f}\|^2_{L^2(U)} + \|\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}\|^2_{L^2(U)} + \|\operatorname{div}(\boldsymbol{u})\|^2_{L^2(U)}$$

as the sum of squares of the residuals, measured in the L^2-norm [1]. The goal is then to minimize $\mathcal{R}$ over the function space [2]

$$V := \left\{ (\boldsymbol{v}, \boldsymbol{\phi}, q) \in \left[H^1(U)\right]^{3n-2} \mid \boldsymbol{v} = 0 \quad \text{on} \quad \partial U \quad \text{and} \quad \int_U q \, \mathrm{d}x = 0 \right\}. \tag{3.6}$$

That is,

$$\text{find} \quad (\boldsymbol{u}, \boldsymbol{\omega}, p) = \operatorname*{argmin}_{(\boldsymbol{v}, \boldsymbol{\phi}, q) \in V} \mathcal{R}(\boldsymbol{v}, \boldsymbol{\phi}, q). \tag{3.7}$$

This explains the name of the method: We minimize the residual sum of squares of the PDE system, measured in the L^2-norm. The minimization problem is well-posed due to the existence of a unique weak solution $(\boldsymbol{u}, p) \in H^2(U) \times H^1(U)$ to (2.17), and thus a solution $(\boldsymbol{u}, \boldsymbol{\omega}, p)$ to (3.7). Note that $\boldsymbol{u} \in H^2(U) \Rightarrow \boldsymbol{\omega} \in H^1(U)$.

From appendix B we know that V is a Hilbert space with respect to the usual inner product

$$\langle (\boldsymbol{u}, \boldsymbol{\omega}, p), (\boldsymbol{v}, \boldsymbol{\phi}, q) \rangle_{H^1(U)} := \sum_{|\alpha| \leq 1} \langle D^\alpha \boldsymbol{u}, D^\alpha \boldsymbol{v} \rangle_{L^2(U)} + \sum_{|\alpha| \leq 1} \langle D^\alpha \boldsymbol{\omega}, D^\alpha \boldsymbol{\phi} \rangle_{L^2(U)} + \sum_{|\alpha| \leq 1} \langle D^\alpha p, D^\alpha q \rangle_{L^2(U)},$$

[1] Other norms could be used, but we will exclusively use the L^2-norm in this paper.

[2] The vorticity functions $\boldsymbol{\omega}, \boldsymbol{\phi}$ are written as vectors, but are scalar functions when $U \subset \mathbb{R}^2$.

and associated norm $\|\cdot\|_{H^1(U)}$. Furthermore, we consider a bilinear form on V given by

$$\begin{aligned}\langle(\boldsymbol{u},\boldsymbol{\omega},p),(\boldsymbol{v},\boldsymbol{\phi},q)\rangle_{V(U)} := \int_U &(\nu\,\mathrm{curl}\,(\boldsymbol{\omega}) + Dp)\cdot(\nu\,\mathrm{curl}\,(\boldsymbol{\phi}) + Dq)\\ &+\nu^2(\mathrm{curl}\,(\boldsymbol{u}) - \boldsymbol{\omega})\cdot(\mathrm{curl}\,(\boldsymbol{v}) - \boldsymbol{\phi})\\ &+\nu^2\,\mathrm{div}\,(\boldsymbol{u})\,\mathrm{div}\,(\boldsymbol{v})\;\mathrm{d}x,\end{aligned}$$

with corresponding semi-norm

$$\begin{aligned}\|(\boldsymbol{u},\boldsymbol{\omega},p)\|^2_{V(U)} :=& \langle(\boldsymbol{u},\boldsymbol{\omega},p),(\boldsymbol{u},\boldsymbol{\omega},p)\rangle_{V(U)}\\ =& \|\nu\,\mathrm{curl}\,(\boldsymbol{\omega}) + Dp\|^2_{L^2(U)} + \nu^2\,\|\mathrm{curl}\,(\boldsymbol{u}) - \boldsymbol{\omega}\|^2_{L^2(U)} + \nu^2\,\|\mathrm{div}\,(\boldsymbol{u})\|^2_{L^2(U)}.\end{aligned}$$

As we saw in section 2.2, the Stokes problem (2.9) has a unique solution in V for each given function $\boldsymbol{f} \in L^2(U)$. Therefore the bilinear form $\langle\cdot,\cdot\rangle_{V(U)}$ and the form $\|\cdot\|_{V(U)}$ define an inner product and a norm on V. Note that we have added the 'weights' ν^2 to the last two terms to be consistent with the source of the formulation [5].

The least-squares formulation can then be stated as the following variational problem:

$$\begin{cases}\text{find } (\boldsymbol{u},\boldsymbol{\omega},p) \in V \quad \text{such that}\\ B((\boldsymbol{u},\boldsymbol{\omega},p),(\boldsymbol{v},\boldsymbol{\phi},q)) = F((\boldsymbol{v},\boldsymbol{\phi},q)) \quad \forall(\boldsymbol{v},\boldsymbol{\phi},q) \in V\end{cases} \tag{3.8}$$

where $B : V \times V \to \mathbb{R}$ and $F : V \to \mathbb{R}$ are defined by

$$\begin{aligned}B((\boldsymbol{u},\boldsymbol{\omega},p),(\boldsymbol{v},\boldsymbol{\phi},q)) &= \langle(\boldsymbol{u},\boldsymbol{\omega},p),(\boldsymbol{v},\boldsymbol{\phi},q)\rangle_{V(U)}\\ F((\boldsymbol{v},\boldsymbol{\phi},q)) &= \int_U \boldsymbol{f}\cdot(\nu\,\mathrm{curl}\,(\boldsymbol{\phi}) + Dq)\,\mathrm{d}x, \quad \boldsymbol{f} \in L^2(U).\end{aligned}$$

3.6 Quasi-norm-equivalent LS-FEM methods

The LS-FEM formulation (3.8) is an example of a norm-equivalent method. Bochev & Gunzburger (2009, [2]) define a norm-equivalent method as having a residual functional $\mathcal{R}$ satisfying

$$C_1\,\|v\|_S \le \mathcal{R}(v) \le C_2\,\|v\|_S \quad \forall v \in S,$$

where C_1, C_2 are constants and S is the solution space in which we search for v (i.e. V in the previous section). There exists a larger class of methods called quasi-norm-equivalent (QNE) methods, for which the constants C_1 and C_2 depend on the mesh size h. I.e.

$$C_1(h)\,\|v\|_S \le \mathcal{R}(v) \le C_2(h)\,\|v\|_S \quad \forall v \in S.$$

This dependence on h requires QNE methods to explicitly contain h as a parameter or 'weight'. Consider the following example of a QNE formulation for the Stokes equations from [2] p.259.

$$\begin{aligned}&\int_U h^2(\mathrm{curl}\,(\boldsymbol{\omega}_h) + Dp_h)\cdot(\mathrm{curl}\,(\boldsymbol{\phi}_h) + Dq_h)\\ &+(\mathrm{curl}\,(\boldsymbol{u}_h) - \boldsymbol{\omega}_h)\cdot(\mathrm{curl}\,(\boldsymbol{v}_h) - \boldsymbol{\phi}_h)\\ &+\mathrm{div}\,(\boldsymbol{u}_h)\,\mathrm{div}\,(\boldsymbol{v}_h)\;\mathrm{d}x\\ =\; &h^2\int_U \boldsymbol{f}_h\cdot(\mathrm{curl}\,(\boldsymbol{\phi}_h) + Dq_h)\,\mathrm{d}x\end{aligned} \tag{3.9}$$

where $(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in [\mathcal{P}_{r+1}]^d \times [\mathcal{P}_r]^{d(d-1)/2} \times \mathcal{P}_r$. Alternatively we may use the same order polynomials to approximate all three variables, and use h instead of h^2 as a weight for the first residual. This leads to the method

$$\begin{aligned} &\int_U h(\operatorname{curl}(\boldsymbol{\omega}_h) + Dp_h) \cdot (\operatorname{curl}(\boldsymbol{\phi}_h) + Dq_h) \\ &+ (\operatorname{curl}(\boldsymbol{u}_h) - \boldsymbol{\omega}_h) \cdot (\operatorname{curl}(\boldsymbol{v}_h) - \boldsymbol{\phi}_h) \\ &+ \operatorname{div}(\boldsymbol{u}_h) \operatorname{div}(\boldsymbol{v}_h) \, \mathrm{d}x \\ = h &\int_U \boldsymbol{f}_h \cdot (\operatorname{curl}(\boldsymbol{\phi}_h) + Dq_h) \, \mathrm{d}x \end{aligned} \tag{3.10}$$

where $(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in [\mathcal{P}_r]^d \times [\mathcal{P}_r]^{d(d-1)/2} \times \mathcal{P}_r$.

The weights h and h^2 come about as an approximation of the H^{-1} norm. It is defined as

$$\|g\|_{H^{-1}(U)} = \sup_{u \in H^1(U)} \frac{\int_U gu \, \mathrm{d}x}{\|u\|_{H^1}}$$

and can be approximated by

$$\|g\|_{H^{-1}(U)} = h^k \|g\|_{L^2(U)}$$

where k is the polynomial order used to approximate g (see [2] p. 581). The methods (3.9) and (3.10) measure the residual of the first of (3.5),

$$\nu \operatorname{curl}(\boldsymbol{\omega}) + Dp = \boldsymbol{f} \qquad \text{in } U,$$

in the H^{-1} norm, which is why only this term is weighted by h^2 in (3.9) and h in (3.10) (as well as the corresponding right-hand-side term containing f).

3.7 Error analysis of the finite element method

In this section we shall use the following notation:

$$\|\cdot\|_k := \|\cdot\|_{W^{k,2}(U)}.$$

In particular we have that

$$\begin{aligned} \|\cdot\|_0 &:= \|\cdot\|_{L^2(U)} \\ \|\cdot\|_1 &:= \|\cdot\|_{H^1(U)}. \end{aligned}$$

Based on the Bramble-Hilbert lemma, see appendix B.5, we have the following optimal error bound in the H^1-norm ([18]).

Proposition 3.13 (Optimal error bound)**.** *Let $\mathcal{T}$ denote a triangulation of $U \subset \mathbb{R}^n$ using the finite element $(K, \mathcal{P}, \mathcal{N})$ satisfying the following conditions:*

1. *K is star-shaped with respect to some ball contained in K (for instance a triangle/polyhedron);*
2. *$\mathcal{P}_{m-1} \subset \mathcal{P} \subset W^{m,\infty}(K)$;*
3. *$\mathcal{N} \subset \left(C^l(\bar{K})\right)'$ (i.e. the nodal variables in $\mathcal{N}$ involve derivatives up to order l, and each element of the set $\mathcal{N}$ is a bounded linear functional on $C^l(\bar{K})$).*

Furthermore, set

$$V_h = \mathcal{I}_h \left(H^m(U) \cap H_0^1(U) \right)$$

and suppose $u \in H^m(U) \cap H_0^1(U)$*. Then the finite element approximation* u_h *of* u *satisfies the error bound*

$$\|u - u_h\|_{H^1(U)} \leq Ch^{m-1} |u|_{H^m(U)}$$

for some constant C*.*

Proof. See [18]. □

For example, if we use continuous piecewise polynomials of degree $m-1$ on a regular triangulation of U, then

(i) $m = 2$ corresponds to linear basis functions, giving an error of $\mathcal{O}(h)$ in the H^1-norm, provided that $u \in H^2(U) \cap H_0^1(U)$;

(ii) $m = 3$ corresponds to quadratic basis functions, giving an error of $\mathcal{O}(h^2)$ in the H^1-norm, provided that $u \in H^3(U) \cap H_0^1(U)$;

where h as usual denotes the characteristic size of the largest element in the triangulation of U. In general, the smallest possible error that can be obtained for a given m in the H^1-norm is of size $\mathcal{O}(h^{m-1})$.

3.7.1 Error bounds for least-squares FEMs

Error bounds for the LS-FEM (3.8) are given by theorem 5.2 in [5] (which also establishes stability and convergence):

Theorem 3.14 (Error bound for (3.8))**.** *Define* V *as in* (3.6)*. The LS-FEM scheme* (3.8) *is stable and convergent in the following sense:*

$$\nu \|\boldsymbol{u}_h\|_1 + \nu \|\boldsymbol{\omega}_h\|_0 + \|p_h\|_0 \leq C \|\boldsymbol{f}\|_0,$$

and

$$\lim_{h \to 0} (\nu \|\boldsymbol{u} - \boldsymbol{u}_h\|_1 + \nu \|\boldsymbol{\omega} - \boldsymbol{\omega}_h\|_0 + \|p - p_h\|_0) = 0.$$

Moreover, if the exact solution $(\boldsymbol{u}, \boldsymbol{\omega}, p) \in V \cap \left[H^{r+1}(U)\right]^{3d-2}$*, then there exists a positive constant* C *independent of* ν *and* h *such that*

$$\nu \|\boldsymbol{u} - \boldsymbol{u}_h\|_1 + \nu \|\boldsymbol{\omega} - \boldsymbol{\omega}_h\|_0 + \|p - p_h\|_0 \leq Ch^r (\nu \|\boldsymbol{u}\|_{r+1} + \nu \|\boldsymbol{\omega}\|_{r+1} + \|p\|_{r+1}).$$

Proof. See [5]. □

We will state another theorem before we comment on the implications. Error bounds for the quasi-norm-equivalent methods (3.9) and (3.10) are given by the following (modification of) theorem (Theorem 7.14 from [2] p.262).

Theorem 3.15. *Define the two function spaces*

$$V_h^r = [\mathcal{P}_r]^d \cap \left[H_0^1(U)\right]^d \times [\mathcal{P}_r]^{d(d-1)/2} \times \mathcal{P}_r \quad V_h^{r^+} = [\mathcal{P}_{r+1}]^d \cap \left[H_0^1(U)\right]^d \times [\mathcal{P}_r]^{d(d-1)/2} \times \mathcal{P}_r \tag{3.11}$$

and let

(i) $(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in V_h$, $r \geq 1$, *denote a solution of the Stokes equations* (2.9) *using the h-weighted quasi-norm-equivalent LS-FEM formulation* (3.10).

(ii) $(\boldsymbol{u}_h^+, \boldsymbol{\omega}_h^+, p_h^+) \in V_h^{r^+}$, $r \geq 1$, *denote a solution of the Stokes equations* (2.9) *using the* h^2*-weighted quasi-norm-equivalent LS-FEM formulation* (3.9).

Assume that the exact solution $(\boldsymbol{u}, \boldsymbol{\omega}, p)$ *of the VVP system* (3.5) *in the domain* U*, belongs to the space*

$$V_q = \left[H^{q+2}(U)\right]^d \cap \left[H_0^1(U)\right]^d \times \left[H^{q+1}(U)\right]^{d(d-1)/2} \times H^{q+1}(U) \cap L_0^2(U)$$

for some $q \geq 0$*. Then there exists a positive constant* C *such that*

1.
$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_1 + \|\boldsymbol{\omega} - \boldsymbol{\omega}_h\|_0 + \|p - p_h\|_0 \leq Ch^{\tilde{r}} \left(\|\boldsymbol{u}\|_{\tilde{r}+1} + \|\boldsymbol{\omega}\|_{\tilde{r}} + \|p\|_{\tilde{r}}\right)$$

2.
$$\left\|\boldsymbol{u} - \boldsymbol{u}_h^+\right\|_1 + \left\|\boldsymbol{\omega} - \boldsymbol{\omega}_h^+\right\|_0 + \left\|p - p_h^+\right\|_0 \leq Ch^{\tilde{r}+1} \left(\|\boldsymbol{u}\|_{\tilde{r}+2} + \|\boldsymbol{\omega}\|_{\tilde{r}+1} + \|p\|_{\tilde{r}+1}\right)$$

and

(i)
$$\|\boldsymbol{\omega} - \boldsymbol{\omega}_h\|_1 + \|p - p_h\|_1 \leq Ch^{\tilde{r}-1} \left(\|\boldsymbol{u}\|_{\tilde{r}+1} + \|\boldsymbol{\omega}\|_{\tilde{r}} + \|p\|_{\tilde{r}}\right),$$

(ii)
$$\left\|\boldsymbol{\omega} - \boldsymbol{\omega}_h^+\right\|_1 + \left\|p - p_h^+\right\|_1 \leq Ch^{\tilde{r}} \left(\|\boldsymbol{u}\|_{\tilde{r}+2} + \|\boldsymbol{\omega}\|_{\tilde{r}+1} + \|p\|_{\tilde{r}+1}\right),$$

where $\tilde{r} = \min(r, q)$.

Proof. The theorem is stated without proof in [2]. □

Theorem 3.15 states that the accuracy of the numerical solution depends both on the smoothness of the exact solution, and the order of the polynomials we use to approximate the exact solution. So assuming that $q \geq 1$, r becomes the limiting factor. Setting $r = 1$ we expect that:

- The h-weighted first order QNE method will give a velocity H^1 error of $\mathcal{O}(h)$, and vorticity and pressure L^2 errors of $\mathcal{O}(h)$ as well.
- The h^2-weighted second order QNE method will give a velocity H^1 error of $\mathcal{O}(h^2)$, and vorticity and pressure L^2 errors of $\mathcal{O}(h^2)$ as well. The H^1 errors of the vorticity and pressure should both be $\mathcal{O}(h)$.

Similarly, for the un-weighted method (3.8) we expect that the velocity H^1 error is of $\mathcal{O}(h)$, and the vorticity and pressure L^2 errors are of $\mathcal{O}(h)$ as well (using piecewise linear functions to approximate all three variables).

3.8 Convergence tests

We measure the *convergence rate* $\beta \in \mathbb{R}$ of a given finite element method as

$$\|ERR(h_i)\| = \alpha h_i^{\beta} \Rightarrow \log(\|ERR(h_i)\|) = \log(\alpha) + \beta \log(h_i) \quad i = 1, ..., N,$$

where h_i denotes the grid size, $ERR(h_i)$ denotes the difference between the numerical solution and the true solution of a variable using a grid of size h_i, and α is a constant. For velocity, as an example, we would have $ERR(h_i) = \boldsymbol{u} - \boldsymbol{u}_{h_i}$. Based on N different values of $ERR(h_i)$ and h_i, the parameters α and β can be estimated using linear regression. We will call the absolute value of the resulting estimate $\hat{\beta}$ of β the 'convergence rate' of the method, which of course depends on what norm $\|\cdot\|$ we use to measure the error. Note that $\hat{\beta}$ is negative for a convergent method.

To test the convergence properties of the various formulations of the Stokes problem, we will use three different model problems. Two of them are from [5]; one is in 2D, and the other is in 3D. We will also use another 2D model problem from [2], to see if the nature of the model problem has an influence on the performance of the numerical methods.

3.8.1 2D model problem A: Polynomials

Let $U = [0,1] \times [0,1]$ and define

$$\begin{pmatrix} u_1 \\ u_2 \\ \omega \\ p \end{pmatrix} = \begin{pmatrix} x^2(1-x)^2(2y - 6y^2 + 4y^3) \\ y^2(1-y)^2(-2x + 6x^2 - 4x^3) \\ x^2(1-x)^2(-2 + 12y - 12y^2) + y^2(1-y)^2(-2 + 12x - 12x^2) \\ x^2 + y^2 - \frac{20}{3}xy + x + y \end{pmatrix} \tag{3.12}$$

where $\boldsymbol{u} = 0$ on ∂U, $\operatorname{div}(\boldsymbol{u}) = 0$ in U, and $\int_U p \, \mathrm{d}x = 0$. The domain is uniformly partitioned into $1/h^2$ squares, each of which is divided into two triangles. To obtain a unique solution we 'pin the pressure' at $p(0,0) = 0$, instead of enforcing $\int_U p \, \mathrm{d}x = 0$. This is because the latter constraint is more costly from a numerical perspective, because it ruins the sparsity of the matrix A in the linear system $Ax = b$, that we solve to obtain our solution x. The sparsity is ruined because a single full row is introduced in A.

The Stokes equations were solved in FEniCS using the PETSc LU decomposition solver.

Several different methods were tested. We set the dynamic viscosity coefficient $\nu = 1$ and suppress the h subscripts for the functions in the formulations for brevity. We stress that all four formulations solve the same problem, namely (2.17).

(i) Mixed formulation

$$\int_U D\boldsymbol{u} : D\boldsymbol{v} - \operatorname{div}(\boldsymbol{v})\, p - q \operatorname{div}(\boldsymbol{u}) \, \mathrm{d}x = \int_U \boldsymbol{u} \cdot \boldsymbol{f} \, \mathrm{d}x. \tag{3.13}$$

(ii) CY: LS-FEM formulation from Chang & Yang (2002) [5].

$$\begin{aligned} &\int_U (\operatorname{curl}(\boldsymbol{\omega}) + Dp) \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) + (\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}) \cdot (\operatorname{curl}(\boldsymbol{v}) - \boldsymbol{\phi}) + \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x \\ &= \int_U \boldsymbol{f} \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) \, \mathrm{d}x \end{aligned} \tag{3.14}$$

(iii) QNE: Quasi-norm-equivalent LS-FEM formulation (3.9).

$$\int_U h^2(\operatorname{curl}(\boldsymbol{\omega}) + Dp) \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) + (\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}) \cdot (\operatorname{curl}(\boldsymbol{v}) - \boldsymbol{\phi}) + \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x$$
$$= h^2 \int_U \boldsymbol{f} \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) \, \mathrm{d}x \tag{3.15}$$

(iv) QNEh: Modification of (iii) using the weight h instead h^2 (3.10).

$$\int_U h(\operatorname{curl}(\boldsymbol{\omega}) + Dp) \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) + (\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}) \cdot (\operatorname{curl}(\boldsymbol{v}) - \boldsymbol{\phi}) + \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x$$
$$= h \int_U \boldsymbol{f} \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq) \, \mathrm{d}x \tag{3.16}$$

For the CY, QNE, and QNEh formulations, we will try using both first and second order Lagrange elements to approximate the velocity. We shall write CY1 to denote first order elements, and CY2 to denote second order elements. Similar notation is used for the other methods. We shall use the following grid dimensions:

$$1/h = \{10, 20, 40, 80, 160\}.$$

Tables 3.1-3.2 show the results of the numerical experiments. Convergence rates are measured using velocity, unless otherwise specified. We see that

- the results for the QNE methods confirm the results from theorem 3.15.
- the mixed formulation dominates in terms of accuracy, while being competitive with the other second order methods in terms of solver time.
- the QNE1 method shows a convergence rate of 0.19 in the L^2-norm, which owes to the fact that it's misspecified (it uses linear Lagrange elements for velocity, not quadratic, as it should).
- CY1 dominates CY2, QNE2 dominates QNE1, and QNE1h dominates QNE2h when we take solver time into account (QNE2h shows slightly higher convergence rates than QNE1h, but is far slower).
- the convergence rate of the mixed formulation at 1.99 in the H^1-norm is as expected. The higher convergence rate of 3.98 in the L^2 norm may have to do with the fact that we are using polynomials to approximate a true solution which is itself a polynomial (in this case we use second order polynomials to approximate a fourth order polynomial in the variables x and y). In the following section we try a different model problem that consists of exponential and trigonometric functions, to see if this affects the convergence rate (the convergence rate falls to 3.55, which is still quite high).

Define the 'power relative error' by

$$PRE = 100 \frac{J(\boldsymbol{u}_h) - J(\boldsymbol{u})}{J(\boldsymbol{u})}, \tag{3.17}$$

where $\boldsymbol{u}$ is the true solution, $\boldsymbol{u}_h$ is the numerical solution, and

$$J(\boldsymbol{v}) = \frac{1}{2} \int_U D\boldsymbol{v} : D\boldsymbol{v} - \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x. \tag{3.18}$$

In chapter 4 we will use a slight modification of this functional (we add a penalty term) for the purpose of topology optimization. Since the optimization process is based on minimizing this

Method	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$	$\|\omega-\omega_h\|_{L^2}$	$\|p-p_h\|_{L^2}$	PRE (%)	L^2-conv	H^1-conv	SolTime
MF	1.037e-10		6.819e-09	-5.58e-07	3.98	1.99	36.3
CY1	5.841e-04	6.262e-03	6.752e-02	81.1	1.25	1.18	8.1
CY2	5.852e-04	6.270e-03	6.760e-02	81.3	1.28	1.21	35.9
QNE1	8.825e-04	6.497e-03	2.244e-03	-1.33	0.19	0.382	8.3
QNE2	1.991e-06	1.317e-05	6.825e-04	0.00899	2.31	2.34	35.6
QNE1h	5.176e-05	3.945e-04	4.239e-03	0.604	1.51	1.2	11.0
QNE2h	4.204e-05	3.279e-04	4.298e-03	0.664	1.63	1.63	48.8

Table 3.1: Results for 2D model problem A (3.12). Errors and other data are reported for grid size $1/h = 160$. PRE (%) = power relative error (3.17), L^2-conv = velocity convergence rate in L^2-norm, H^1-conv = velocity convergence rate in H^1-norm, SolTime = solver time in seconds.

	Velocity		Vorticity		Pressure		PRE
Method	L^2	H^1	L^2	H^1	L^2	H^1	
MF	3.98	1.99			3.54	1	4.35
CY1	1.25	1.18	1.21	0.987	0.936	0.984	1.12
CY2	1.28	1.21	1.22	0.997	0.945	0.993	1.16
QNE1	0.19	0.382	0.221	0.381	1.47	0.865	1.54
QNE2	2.31	2.34	2.48	1.16	1.9	1.02	3.56
QNE1h	1.51	1.2	1.61	1.25	1.4	1.08	2.33
QNE2h	1.63	1.63	1.7	1.27	1.42	1.09	2.37

Table 3.2: Results for 2D model problem A. Convergence rates for velocity, vorticity, pressure and PRE in the L^2 and H^1-norms. PRE = power relative error.

functional, it is particularly important that it is calculated accurately. In table 3.1 we see that, for $1/h = 160$, PRE is very low for the mixed formulation, at $-5.58e-07\%$, but strikingly high for the CY1 formulation at 81.1%. Considering how small h is, this error is enormous, and suggests that, when it comes to calculating the power functional $J(\boldsymbol{v})$, the CY1 formulation is essentially useless. This suggests that the CY1 formulation cannot solve the topology optimization problem considered in chapter 4. We finally note that the QNE2 and QNE1h formulations do quite well with respect to PRE; they come out at 0.00899% and 0.604%, respectively.

3.8.2 2D model problem B: Exponential and trigonometric functions

The model problem in (3.12) specifies the variables in terms of polynomials. This may lead to over-optimistic results, because we then use polynomials to approximate polynomials, which is more accurate than, for instance, approximating trigonometric functions with polynomials. So to test the validity of (3.12) we use the following alternative problem with a true solution involving exponential and trigonometric functions (from [2] p.268).

Let $U = [0,1] \times [0,1]$ and define

$$\begin{pmatrix} u_1 \\ u_2 \\ \omega \\ p \end{pmatrix} = \begin{pmatrix} e^x \cos(y) + \sin(y) \\ -e^x \sin(y) + 1 - x^3 \\ \operatorname{curl}(\boldsymbol{u}) \\ \sin(y)\cos(x) + xy^2 - \frac{1}{6} - \sin(1)(1-\cos(1)) \end{pmatrix} \tag{3.19}$$

where the "true" solution is assigned to ∂U, $\operatorname{div}(\boldsymbol{u}) = 0$ in U, and $\int_U p \, \mathrm{d}x = 0$. The results are shown in tables 3.3-3.4. They are similar to model problem A; the main differences are that

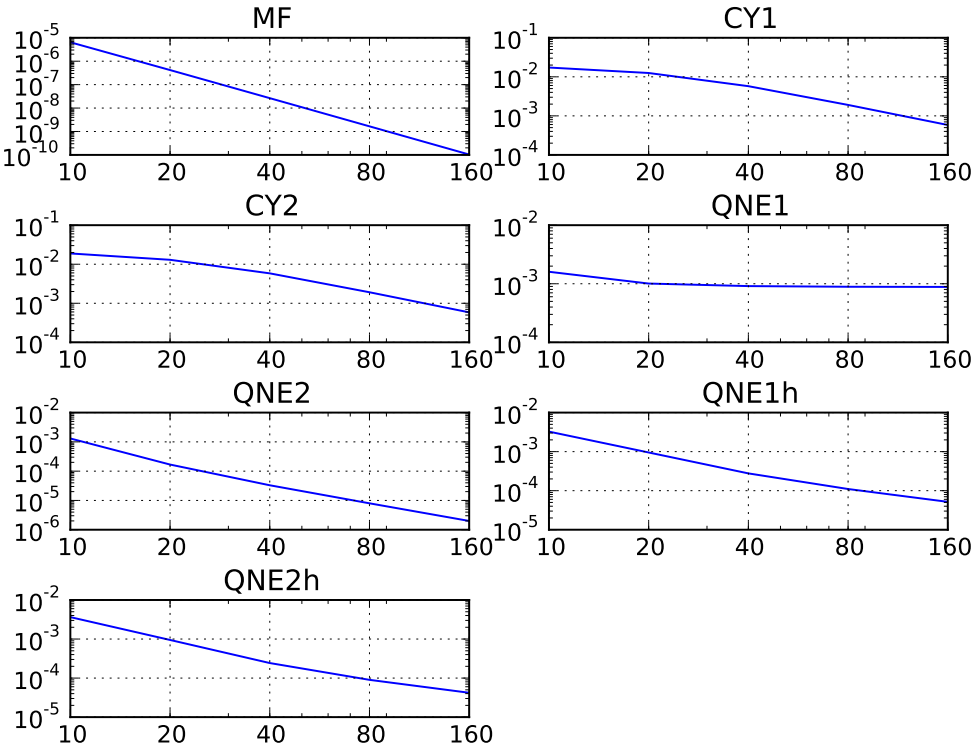


Figure 3.2: Model problem A velocity L^2-error: $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{L^2}$. The x-axis shows the grid size $1/h$; the y-axis shows the error.

- the H^1 convergence rates are close to their theoretical optimal levels. The methods MF and QNE2 that use second order polynomials to approximate the velocity, show second order convergence rates for velocity, and first order rates for vorticity and pressure. The correctly specified methods that use first order polynomials to approximate the velocity, namely CY1 and QNE1h, show at least first order convergence for all three variables.

- the velocity L^2 convergence rate is lower for MF at 3.55, but higher for QNE2 at 3.19. So the performance of the methods seems to depend on the nature of the true solution (in this case polynomials vs. exponential and trigonometric functions).

- the PRE is now calculated with significantly more accuracy by all the methods; especially QNE2 shows a big improvement. So PRE seems to be quite sensitive to the form of the true solution and/or the boundary conditions.

3.8.3 3D model problem

We now investigate the comparative performance of the different formulations in three dimensions. For this purpose we use the 3D model problem from [5]. The true solution is expressed as polynomials of up to fourth order, as in 2D model problem A. Let $U = [0,1]^3 \subset \mathbb{R}^3$ and define the

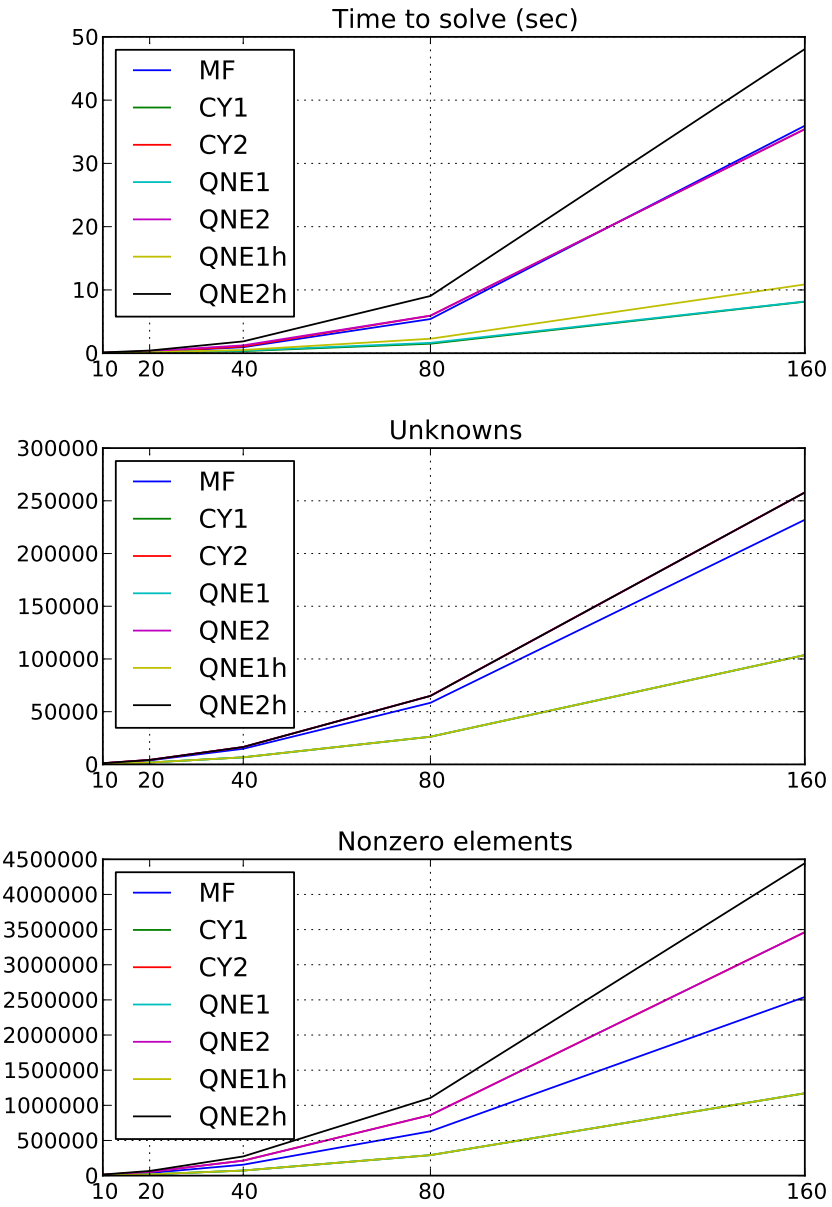


Figure 3.3: 2D model problem A diagnostics.

Method	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$	$\|\omega-\omega_h\|_{L^2}$	$\|p-p_h\|_{L^2}$	PRE (%)	L^2-conv	H^1-conv	SolTime
MF	3.631e-10		5.249e-06	-1.67e-09	3.55	2	36.1
CY1	1.469e-04	1.557e-03	1.953e-02	0.00184	1.33	1.01	8.2
CY2	1.344e-04	1.449e-03	2.028e-02	0.00022	1.25	1.19	35.8
QNE1	1.497e-03	1.425e-02	4.201e-02	0.00384	0.546	0.872	8.3
QNE2	4.707e-08	1.683e-05	2.062e-04	9.63e-09	3.19	2.05	36.1
QNE1h	2.025e-04	2.189e-03	5.122e-03	0.0046	1.66	1.05	11.1
QNE2h	1.586e-04	1.731e-03	3.597e-03	0.00283	1.69	1.5	48.7

Table 3.3: Results for 2D model problem B (3.19). Errors and other data are reported for grid size $1/h = 160$. PRE (%) = power relative error, L^2-conv = velocity convergence rate in L^2-norm, H^1-conv = velocity convergence rate in H^1-norm, SolTime = solver time in seconds.

	Velocity		Vorticity		Pressure		PRE
Method	L^2	H^1	L^2	H^1	L^2	H^1	
MF	3.55	2			2.06	1.02	4.06
CY1	1.33	1.01	1.29	0.995	0.779	1	1.89
CY2	1.25	1.19	1.24	0.963	0.85	0.965	1.76
QNE1	0.546	0.872	0.505	0.366	0.468	0.617	1.7
QNE2	3.19	2.05	2.06	1.01	1.88	1.08	4.05
QNE1h	1.66	1.05	1.53	1.02	1.35	1.02	1.84
QNE2h	1.69	1.5	1.54	1.03	1.43	1.04	1.81

Table 3.4: Results for 2D model problem B. Convergence rates for velocity, vorticity, pressure and PRE in the L^2 and H^1-norms. PRE = power relative error.

vector potential $\boldsymbol{\Phi}$ by

$$\boldsymbol{\Phi} = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{pmatrix} = \begin{pmatrix} x(1-x)y^2(1-y)^2z^2(1-z)^2 \\ y^2(1-y)^2(-2x+6x^2-4x^3) \\ x^2(1-x)^2y(1-y)z^2(1-z)^2 \\ x^2(1-x)^2y^2(1-y)^2z(1-z) \end{pmatrix}$$

and the velocity as

$$\boldsymbol{u} = \operatorname{curl}(\boldsymbol{\Phi})$$

Then we have $\operatorname{div}(\boldsymbol{u}) = 0$ and $\boldsymbol{u} = 0$ on ∂U. Also define the vorticity $\boldsymbol{\omega}$ and pressure p of the exact solution as

$$\begin{aligned} \boldsymbol{\omega} &= \operatorname{curl}(\boldsymbol{u}), \\ p &= -2xyz + x^2 + y^2 + z^2 + xy + xz + yz - x - y - z. \end{aligned}$$

We again 'pin the pressure' at $p_h(0,0,0) = 0$ to obtain a unique solution. The domain U is subdivided into 2*$1/h^3$ tetrahedrals ($1/h^3$ cubes each divided into two tetrahedrals), where $h \in \mathbb{N}$, denotes the length of a single tetrahedral in all three space dimensions. The Stokes equations are again solved in FEniCS using a direct solver (LU decomposition) for small grids, and iterative solvers for $1/h \geq 12$. The iterative solvers are

- For the mixed formulation: Transpose-free quasi minimal residual (TFQMR) solver and an algebraic multigrid preconditioner as suggested in [16]. This is because the finite element discretization of the formulation leads to a symmetric, but indefinite linear system. In

addition, we use the following expression as a preconditioner:

$$\int_U D\boldsymbol{u} : D\boldsymbol{v} - pq \, \mathrm{d}x = \int_U \boldsymbol{u} \cdot \boldsymbol{f} \, \mathrm{d}x,$$

again by recommendation from [16]. See also [15].

- For the LS-FEM formulations: Conjugate gradient solver and Jacobi preconditioner, because the LS-FEM method leads to a symmetric, positive definite linear system.

Since the results for the 2D problem showed that the CY2, QNE1 and QNE2h formulations are dominated by the other formulations, i.e. it is always better to use CY1 rather than CY2, etc. So now we only consider the remaining four methods: MF, CY1, QNE2 and QNE1h. We will use the grid sizes

$$1/h = \{4, 8, 16, 20\}.$$

Larger grids were not used due to memory constraints. The results are shown in table 3.5 and figures 3.4,3.5. We make the following observations:

- The mixed formulation again dominates in terms of accuracy, while exhibiting a convergence rate of 3.47 in the H^1-norm, well above the theoretical optimal convergence rate of 2. The convergence rate is this high because the gradient of the power functional $J(\boldsymbol{u})$ for the true solution is approximated numerically. It wasn't possible to use the exact expression due to memory limitations (for grids larger than $1/h = 12$). So the result looks better than it really is.

- The power relative error is again far lower for the mixed formulation than the rest. But now, the mixed formulation takes significantly longer to solve, since we are solving a symmetric indefinite linear system, instead of a symmetric positive definite linear system. The latter type of linear system allows us to use the conjugate gradient (CG) algorithm, which is significantly faster than the TFQMR algorithm. On a $1/h = 20$ grid the TFQMR algorithm uses 39 iterations, compared to 462 iterations for the CG algorithm using the QNE2 formulation. But each iteration takes significantly longer, and the solution process takes 215.1 seconds for MF compared to 125.6 seconds for QNE2. So now, the LS-FEM method has an advantage in terms of CPU time, but there is still a significant accuracy gap. Some interesting questions are:

  - If the grid used for the QNE2 formulation is refined to the point where the solution time is the same as for the TFQMR formulation, how does the accuracy of the solution (measured as $\|\boldsymbol{u} - \boldsymbol{u}_h\|$ or PRE) then compare?
  - More generally: How does the ratio of accuracy to solver time evolve for the various formulations, as the grid is refined further.

- A drawback of the QNE2 formulation is that its condition number is $\mathcal{O}h^{-4}$, so the solver time is likely to rise dramatically as the grid is refined beyond $1/h = 20$, at least in the absence of a good preconditioner.

Method	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$	$\|\omega-\omega_h\|_{L^2}$	$\|p-p_h\|_{L^2}$	PRE (%)	L^2-conv	H^1-conv	SolTime
MF	1.753e-07	0.000e+00	1.696e-03	0.000128	3.76	3.47	216.7
CY1	2.872e-03	3.092e-02	4.399e-01	-88.4	0.117	0.0206	51.8
QNE2	7.293e-05	9.152e-04	5.465e-02	-1.94	2.34	2.15	127.4
QNE1h	3.685e-04	4.264e-03	1.009e-01	-20.2	0.813	0.844	84.5

Table 3.5: Results using 3D model problem. Errors and other data are reported for grid size $1/h = 20$. PRE (%) = power relative error, L^2-conv = velocity convergence rate in L^2-norm, H^1-conv = velocity convergence rate in H^1-norm, SolTime = solver time in seconds.

	Velocity		Vorticity		Pressure		PRE
Method	L^2	H^1	L^2	H^1	L^2	H^1	
MF	3.76	3.47			1.47	1.01	5.91
CY1	0.117	0.0206	0.223	0.176	0.0984	0.14	0.404
QNE2	2.34	2.15	2.25	1.42	1.33	1.25	2.33
QNE1h	0.813	0.844	1.01	0.755	0.75	0.802	1.22

Table 3.6: Results for 3D model problem. Convergence rates for velocity, vorticity, pressure and PRE in the L^2 and H^1-norms. PRE = power relative error.

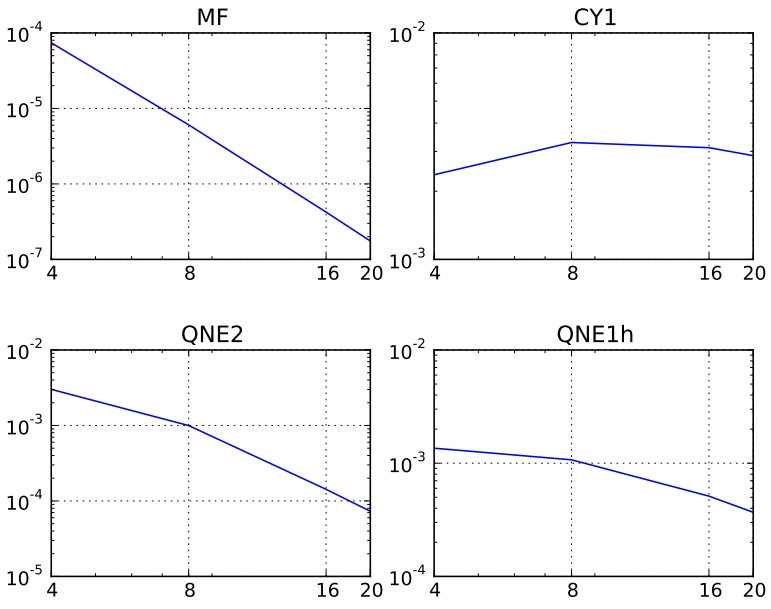


Figure 3.4: L^2-error of the velocity: $\|\boldsymbol{u}-\boldsymbol{u}_h\|_{L^2}$. The x-axis shows the grid size $1/h$; the y-axis shows the error.

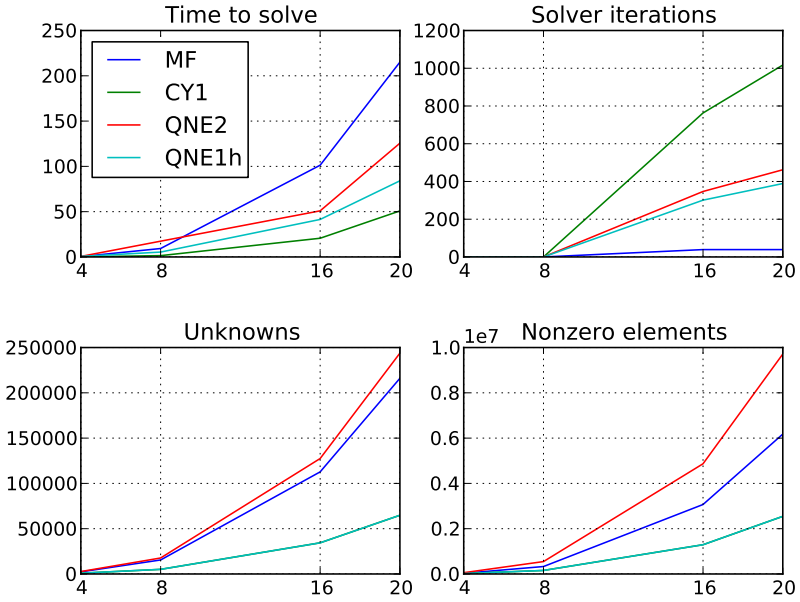


Figure 3.5: 3D model problem diagnostics.

Chapter 4
Topology optimization

4.1 Problem formulation

The general purpose of optimization is to find a parameter of a problem in a way that maximizes or minimizes a given quantity connected to the problem. In this thesis the problem is Stokes flow, and the quantity to be minimized is the power dissipation, following the methodology of [3]. This power dissipation of the flow can be expressed as a functional of the velocity vector field. So minimizing the power dissipation amounts to minimizing this functional. Let the domain U be an open, simply connected subset of $\mathbb{R}^2$. We define the total power dissipation functional is defined as

$$J(\boldsymbol{u}) = \frac{\nu}{2}\int_U D\boldsymbol{u}\cdot D\boldsymbol{u} - \int_U \boldsymbol{f}\cdot\boldsymbol{u}. \tag{4.1}$$

In order to control the fluid flow (i.e. the fluid's velocity vector field) in the domain U we will add a 'penalty term' to the Stokes equations. The penalty term will be a function of the coordinate vector $\boldsymbol{x}$, and can be used to control where in the domain we wish to have, or not have, fluid flow. We will think of areas with zero flow as areas filled with an impenetrable material (like concrete, for example). We define the penalty term as

$$\alpha(\rho(\boldsymbol{x})), \quad \boldsymbol{x}\in U,\ \rho\in[0,1]\,,\ \alpha\in[\underline{\alpha},\overline{\alpha}]\,, \tag{4.2}$$

where $0\approx\underline{\alpha}$, $\overline{\alpha} >> 0$ are constants.

Think of $\rho(\boldsymbol{x})$ as an indicator function that determines how freely the fluid can flow in a given point $\boldsymbol{x}\in U$: $\rho = 1$ corresponds to free flow, and $\rho = 0$ corresponds to no flow (i.e. the presence of material). We allow ρ to vary within the interval $[0,1]$, but we prefer a solution with ρ values close to either extreme, to have a clear distinction between areas with or without flow. A value of $\rho = 0.5$ would correspond to fluid flow with reduced velocity (i.e. the presence of some semi-permeable material, for example a filter or concrete riddled with many small holes).
The penalty term $\alpha(\rho)$ varies inversely with ρ, so that

$$\begin{cases}\alpha(0)=\overline{\alpha}\Rightarrow & \text{no flow}\\ \alpha(1)=\underline{\alpha}\Rightarrow & \text{free flow}\end{cases} \tag{4.3}$$

With the addition of the penalty term, the Stokes equations with Dirichlet boundary conditions become

$$\begin{aligned}\alpha(\rho)\boldsymbol{u}-\nu\Delta\boldsymbol{u}+Dp &= \boldsymbol{f} && \text{in } U\\ \operatorname{div}(\boldsymbol{u}) &= 0 && \text{in } U\\ \boldsymbol{u} &= \boldsymbol{g} && \text{on } \partial U,\end{aligned} \tag{4.4}$$

where $\boldsymbol{g}$ is a piecewise continuous function that prescribes the velocity on the boundaries. The addition of the $\alpha(\rho)$ term is reasonable from a physical perspective, because it acts as an external force controlling the fluid flow, where ρ is the control. From an optimal design perspective we would call ρ the design function.

Define the following subspace of $H^1(U)$:

$$W := \left\{ v \in H^1(U) \mid \boldsymbol{v} = \boldsymbol{g} \text{ on } \partial U \right\}.$$

This is the 'trial' space in which we will search for $\boldsymbol{u}$. We will also need the following two 'test' spaces:

$$\begin{aligned} V &:= H_0^1(U) = \left\{ \boldsymbol{v} \in H^1(U) \mid \boldsymbol{v} = 0 \text{ on } \partial U \right\} \\ P &:= L^2(U). \end{aligned}$$

The trial space for p is also P. For a given $\alpha(\rho)$, the weak formulation of (4.4) can then be expressed as:

$$\begin{cases} \text{find } (\boldsymbol{u}, p) \in W \times P \text{ such that} \\ \displaystyle\int_U \alpha(\rho)\boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}x + \nu \int_U D\boldsymbol{u} \cdot D\boldsymbol{v} \, \mathrm{d}x - \int_U p \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x = \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x & \forall \boldsymbol{v} \in V \\ \displaystyle\int_U \operatorname{div}(\boldsymbol{u}) \, q \, \mathrm{d}x = 0 & \forall q \in P. \end{cases}$$

Expressed in terms of functionals:

$$\begin{cases} \text{find } (\boldsymbol{u}, p) \in W \times P \text{ such that} \\ a_\alpha(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = l(\boldsymbol{v}) & \forall \boldsymbol{v} \in V \\ b(\boldsymbol{u}, q) = 0 & \forall q \in P, \end{cases}$$

where

$$\begin{aligned} a_\alpha(\boldsymbol{u}, \boldsymbol{v}) &= \int_U \alpha(\rho)\boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}x + \nu \int_U D\boldsymbol{u} \cdot D\boldsymbol{v} \, \mathrm{d}x \\ b(\boldsymbol{v}, p) &= -\int_U p \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x \\ l(\boldsymbol{v}) &= \int_U f \cdot \boldsymbol{v} \, \mathrm{d}x. \end{aligned} \tag{4.5}$$

We define the power functional $J_\rho : H^1(U) \to \mathbb{R}$ by [1]

$$J_\rho(\boldsymbol{v}) = \frac{1}{2} a_\alpha(\boldsymbol{v}, \boldsymbol{v}) - l(\boldsymbol{v}), \tag{4.6}$$

and the closed subset W_{div} of W:

$$W_{\text{div}} = \{ \boldsymbol{v} \in W \mid \operatorname{div}(\boldsymbol{v}) = 0 \}.$$

Then the velocity vector field $\boldsymbol{u} \in W_{\text{div}}$ that solves (4.5) can be found by minimizing the power functional J_ρ in W_{div}:

$$J_\rho(\boldsymbol{u}) = \min_{v \in U_{\text{div}}} J_\rho(\boldsymbol{v}).$$

[1] Borrvall & Petersson ([3]) call this the total potential power.

This is a convex minimization problem with a stationarity condition that is exactly (4.5). Interpreting the pressure p as a Lagrange multiplier (as in theorem 2.4), we relax the condition $\operatorname{div}(\boldsymbol{v}) = 0$ and form the Lagrangian

$$\mathcal{L}_\alpha(\boldsymbol{v}, q) = J_\rho(\boldsymbol{v}) + b(\boldsymbol{v}, q)$$

and look for a saddle point over all $(\boldsymbol{v}, q) \in W \times P$. The saddle point $(\boldsymbol{u}, p) \in W \times P$ then satisfies (4.5).

This motivates the following re-statement of the variational problem (4.5):

$$\begin{cases} \text{find } (\boldsymbol{u}, p) \in W \times P \text{ such that} \\ a_\alpha(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) + b(\boldsymbol{u}, q) = l(\boldsymbol{v}) \quad \forall (\boldsymbol{v}, q) \in V \times P. \end{cases} \tag{4.7}$$

The incompressibility condition has now been added as a Lagrange multiplier, and the problem has been rewritten as a single equation.

So far we haven't placed any restrictions on the function $\alpha(\rho)$. As mentioned earlier, $\rho(\boldsymbol{x})$ is an indicator function that determines where we wish to have fluid flow, and where we wish to have no flow (i.e. material). We control the flow by adding $\alpha(\rho)$ to equation (4.5) as a penalty term. In general, a topology optimization problem will prescribe a certain amount of material on the domain U. As an example we could say that half of the domain should be filled with material. This condition can be expressed as

$$\frac{1}{|U|} \int_U \rho(x) = \gamma, \tag{4.8}$$

where $|U|$ denotes the area (Lebesgue measure) of U, and $\gamma = 0.5$. Adding the 4.8 condition to the problem (4.7), we obtain the following **topology optimization problem**:

Solve (4.7) subject to the condition that a fraction $\gamma \in [0, 1]$ of the domain U must be filled by fluid, i.e.

$$\frac{1}{|U|} \int_U \rho(x) = \gamma, \tag{4.9}$$

and

(i) the boundary condition $\boldsymbol{u} = \boldsymbol{g}$ on ∂U is fulfilled,

(ii) the total dissipated power of the fluid

$$J_\rho(\boldsymbol{v}) = \frac{1}{2} a_\alpha(\boldsymbol{v}, \boldsymbol{v}) - \langle \boldsymbol{f}, \boldsymbol{v} \rangle \tag{4.10}$$

is minimized over all $\boldsymbol{v} \in W$.

Proposition 4.1. *The constrained optimization problem* (4.9)-(4.10) *is well-posed.*

Proof. See [3]. □

The process of solving a topology optimization problem is best illustrated with an example.

4.1.1 Model problem

Let $U := [0,1] \times [0,1] \subset \mathbb{R}^2$ and consider the Stokes problem with the following boundary conditions:

$$\begin{cases} \boldsymbol{u}(x,y) = (0,0) & \text{for } y = 0 \text{ or } y = 1 \\ \boldsymbol{u} = (1-(2y-1)^2, 0) & \text{for } x = 0 \\ \boldsymbol{u} = (3[1-(6y-3)^2], 0) & \text{for } x = 1 \text{ and } y \in [1/3, 2/3] \\ \boldsymbol{u} = (0,0) & \text{for } x = 1 \text{ and } y \notin [1/3, 2/3]. \end{cases} \tag{4.11}$$

This creates a 'parabolic' inflow on the left, and a parabolic outflow on the right, with the same amount of fluid entering and exiting the domain. The boundary conditions at the top and bottom of the domain are known as 'no-slip' boundary conditions. Figure 4.1 shows the resulting numerical solution $(\boldsymbol{u}_h, p_h)$ on a 100×100 grid ($h = 1/100$). The fluid behaves as expected, speeding up at the outflow on the right, where the pressure decreases as the fluid escapes.

Consider now the problem: Solve (4.7) subject to the conditions that

(i) half of the domain U must be filled by fluid, i.e.

$$\frac{1}{|U|} \int_U \rho(x) = \gamma, \tag{4.12}$$

where $\gamma = 0.5$;

(ii) the boundary conditions (4.11) are fulfilled;

(iii) the total dissipated power of the fluid

$$J_\rho(\boldsymbol{v}) = \frac{1}{2} a_\alpha(\boldsymbol{v}, \boldsymbol{v}) - \langle \boldsymbol{f}, \boldsymbol{v} \rangle \tag{4.13}$$

is minimized over all $\boldsymbol{v} \in W$.

If we had to guess what such a solution would look like, we would think of where to place the material in order to disturb the fluid flow as little as possible, thus minimizing the total dissipated power. We would guess that the material will be clustered in parts of the domain where the fluid flow - arising from the boundary conditions - is as slow as possible. I.e. at the top and bottom of the domain, with more material toward the corners on the right. In figure 4.1(a) we see that the flow is strongest at the hole on the right, and therefore there will probably not be much material there, and as little material as possible along the left boundary where the fluid comes in.

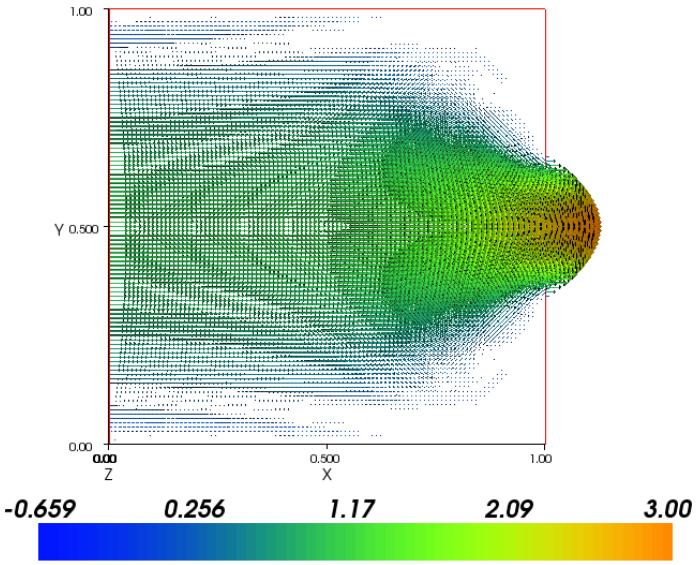
Another way to guess at the solution is to consider what would happen if at first the entire domain was filled with a light, but impenetrable material, such as sand, and the fluid flow is then allowed to 'wash away' the sand until only half of it is left. Then we would arrive at an answer similar to the one above. Below we will see that this is a pretty good answer.

We now turn to the question of how to obtain the numerical solution.

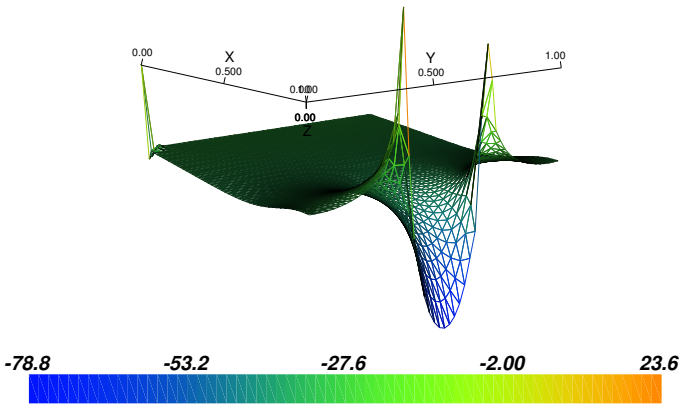
4.2 Finite element formulation

Consider now the following approximation of (4.7):

$$\begin{cases} \text{find } (\boldsymbol{u}_h, p_h) \in V_h \times P_h \text{ such that} \\ a_\alpha(\boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) + b(\boldsymbol{u}_h, q_h) = l(\boldsymbol{v}_h) \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times P_h. \end{cases} \tag{4.14}$$



(a) Velocity vector field



(b) Pressure function

Figure 4.1: Model problem before optimization, i.e. without the $\alpha(\rho)\boldsymbol{u}$ penalty term.

As we saw in section 2.1.3, the pressure function $p \in L^2(U)$ is unique up to an additive constant. To obtain a unique numerical solution, we must place a restriction on p. Two common restrictions are

(i) Force the pressure to integrate to zero:

$$\int_U p \, \mathrm{d}x = 0.$$

(ii) 'Pin' the pressure at a single point:

$$p(0,0) = 0.$$

The latter option is more attractive from a numerical point of view, because it requires less computation than option (i). Option (i) introduces a full row of ones into the system matrix A of the discretized problem. Thus any sparsity properties of A are lost. We will therefore use option (ii).

We can now make a few more comments to figure 4.1. It can be seen that the pressure has been pinned at $p(0,0) = 0$, to ensure that the numerical solution converges. We see that the pressure 'spikes' at the points $(0, 1/3, 2/3)$ on the right, where the boundary isn't smooth. The reason for the spike is that we are trying to approximate a non-smooth function with an L^2 function that doesn't converge point-wise, but rather in the L^2-norm. This may lead to strong oscillations at points of discontinuity, resembling the Gibbs phenomenon.

4.3 Least-Squares FEM formulation

To formulate the problem (4.4) in least-squares form we simply add the penalty term $\alpha \boldsymbol{u}$ to the formulation (3.8). The resulting problem is

$$\begin{aligned} &\text{find } (\boldsymbol{u}, \boldsymbol{\omega}, p) \in V \text{ such that} \\ &B_\alpha((\boldsymbol{u}, \boldsymbol{\omega}, p), (\boldsymbol{v}, \boldsymbol{\phi}, q)) = F((\boldsymbol{v}, \boldsymbol{\phi}, q)) \quad \forall (\boldsymbol{v}, \boldsymbol{\phi}, q) \in V, \end{aligned} \tag{4.15}$$

where $B_\alpha : V \times V \to \mathbb{R}$ and $F : V \to \mathbb{R}$ are defined by

$$\begin{aligned} B_\alpha\left((\boldsymbol{u}, \boldsymbol{\omega}, p), (\boldsymbol{v}, \boldsymbol{\phi}, q)\right)_{V(U)} := \int_U &(\alpha \boldsymbol{u} + \nu \operatorname{curl}(\boldsymbol{\omega}) + Dp) \cdot (\alpha \boldsymbol{v} + \nu \operatorname{curl}(\boldsymbol{\phi}) + Dq) \\ &+ \nu^2 (\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}) \cdot (\operatorname{curl}(\boldsymbol{v}) - \boldsymbol{\phi}) \\ &+ \nu^2 \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v}) \, \mathrm{d}x \\ F((\boldsymbol{v}, \boldsymbol{\phi}, q)) = \int_U &\boldsymbol{f} \cdot (\nu \operatorname{curl}(\boldsymbol{\phi}) + Dq) \, \mathrm{d}x, \quad \boldsymbol{f} \in L^2(U). \end{aligned}$$

We will here summarize the different formulations as in (3.13)-(3.10), but with the penalty term included. We again set $\nu = 1$.

1. Mixed formulation. $(\boldsymbol{u}_h, p_h) \in \mathcal{P}_2 \times \mathcal{P}_1$.

$$\int_U \alpha \boldsymbol{u} \cdot \boldsymbol{v} + \boldsymbol{u} : D\boldsymbol{v} - \operatorname{div}(\boldsymbol{v}) \, p - q \operatorname{div}(\boldsymbol{u}) \, \mathrm{d}x = \int_U \boldsymbol{u} \cdot f \, \mathrm{d}x. \tag{4.16}$$

2. CY: LS-FEM formulation from Chang & Yang (2002) [5]. See (4.15).

$$(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in \mathcal{P}_1 \times \mathcal{P}_1 \times \mathcal{P}_1.$$

3. QNE: Quasi-norm equivalent LS-FEM formulation from Bochev & Gunzburger (2009) [2] p.259. $(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in \mathcal{P}_2 \times \mathcal{P}_1 \times \mathcal{P}_1$.

$$\begin{aligned}&\int_U h^2(\alpha\boldsymbol{u} + \operatorname{curl}(\boldsymbol{\omega}) + Dp) \cdot (\alpha\boldsymbol{v} + \operatorname{curl}(\boldsymbol{\phi}) + Dq) \\ &+ (\operatorname{curl}(\boldsymbol{u}) - \boldsymbol{\omega}) \cdot (\operatorname{curl}(\boldsymbol{v}) - \boldsymbol{\phi}) + \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v})\, \mathrm{d}x \\ &= h^2 \int_U \boldsymbol{f} \cdot (\operatorname{curl}(\boldsymbol{\phi}) + Dq)\, \mathrm{d}x. \end{aligned} \tag{4.17}$$

4. QNEh: Modification of (4.17) using the weight h instead h^2. $(\boldsymbol{u}_h, \boldsymbol{\omega}_h, p_h) \in \mathcal{P}_1 \times \mathcal{P}_1 \times \mathcal{P}_1$.

4.4 The optimization process

The topology optimization problem (4.9)-(4.10) is a constrained optimization problem, and we will use the Method of Moving Asymptotes (MMA) to solve it (see appendix G.2 for details and references on MMA). The MMA algorithm solves constrained optimization problems in the following form:
Let $x \in \mathbb{R}^N$ be an N-dimensional vector and let $f_0(x)$ be the objective function to be minimized subject to $M > 0$ constraints.

$$\begin{aligned} \text{minimize} \quad & f_0(x) + z + 0.05z^2 + \sum_{i=1}^{M} c_i y_i + 0.5 y_i^2 \\ \text{subject to} \quad & f_i(x) - a_i z - y_i \leq f_i^{MAX}, \quad i = 1, ..., M \\ & x_j^{MIN} \leq x_j \leq x_j^{MAX}, \quad j = 1, ..., N \\ & y_i \geq 0, \qquad y = 1, ..., M \\ & z \geq 0, \end{aligned} \tag{4.18}$$

where M, N are strictly positive integers, $x_j^{MIN} \leq x_j^{MAX},\ j = 1, ..., N$, and $f_i^{MAX},\ i = 1, ..., M$ are constants. The additional variables a_i, c_i, y_i and z are included to give the algorithm the flexibility to handle many types of problems; they were not used for the purposes of this thesis.

For the topology optimization problem we wish to solve, we set $f_0 := J_\rho(\boldsymbol{u}_h)$, $x := \rho$, etc. At each iteration the MMA algorithm takes $f0$, f_i, $i = 1, ..., M$, and an x-vector x_{in} as input, and then returns an updated x-vector x_{out} as output. If all goes well then $f_0(x_{out}) < f_0(x_i n)$, while all constraints are still satisfied.

The optimization process consists of the following general steps:

(0) Make an initial guess for the design function ρ.

(1) Solve Stokes equations in some domain U using ρ.

(2) Compute the power functional $J_\rho(\boldsymbol{u})$.

(3) Compute the variation δJ_ρ of the power functional. This will be explained below.

(4) Update ρ using the MMA algorithm.

(5) Compute some measure of the quality of the solution based on (2).

(6) If the solution is still not good enough, go to step 1. Otherwise stop.

4.5 Numerical results

In the numerical examples below we will set $U = [0,1] \times [0,1]$ and use the following parameter values:

$$q = 0.1,\ \nu = 1,\ \underline{\alpha} = 2.5 \cdot 10^4,\ \overline{\alpha} = 2.5 \cdot 10^{-4}.$$

We also set the forcing term $\boldsymbol{f} = \mathbf{0}$. To specify velocity boundary conditions with nice parabolic inflows and outflows, we will use the following formula

$$g(t) = \bar{g}\left[1 - \left(\frac{2t - (a+b)}{b-a}\right)^2\right], \quad t \in [a,b].$$

The graph of g is a parabola going from a to b with maximum $\bar{g}$ at $t = (b-a)/2$.

4.5.1 The interpolation function $\alpha(\rho)$

Since one of the goals of the optimization process is to have a clear distinction between fluid and non-fluid areas, we wish to choose $\alpha(\rho)$ such that ρ only takes values close to 0 and 1. $\rho(\boldsymbol{x})$ will be constant on each element, so if $\alpha(\rho)$ is a linear function, it will impose a too severe restriction on the design (see [3] p.96). We therefore choose the following convex, q-parametrized function [3]:

$$\alpha_q(\rho) := \overline{\alpha} + (\underline{\alpha} - \overline{\alpha})\rho\frac{1+q}{\rho+q} \quad , \tag{4.19}$$

where the parameter $q > 0$ is used to control how 'sudden' the transition from fluid to non-fluid areas in the domain should be. Figure 4.2 shows $\alpha_q(\rho)$ for different values of q, with $\underline{\alpha} = 2.5 \cdot 10^{-4}$ and $\overline{\alpha} = 2.5 \cdot 10^4$.

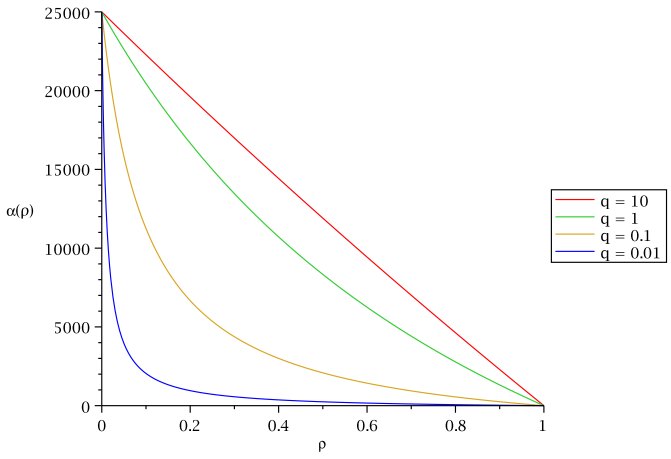


Figure 4.2: The function $\alpha_q(\rho)$.

4.5.2 Calculating the gradient of $J_{\alpha(\rho)}$

The MMA algorithm takes the gradient $D_\rho J_{\alpha(\rho)}(\boldsymbol{u})$ of $J_{\alpha(\rho)}(\boldsymbol{u})$ (see (4.6)) as input. $D_\rho J_{\alpha(\rho)}$ is the variation in $J_{\alpha(\rho)}$ as we change ρ. Say that we have solved (4.5) for $(\boldsymbol{u}, p)$, that is, we have found $\boldsymbol{u} = \underset{v \in W}{\operatorname{argmin}} J_\rho(\boldsymbol{v})$. Then using the interpolation function (4.19) we see that

$$D_\rho J_{\alpha(\rho)}(\boldsymbol{u}) = D_\rho \left[\frac{1}{2} \int_U \alpha(\rho) \, |\boldsymbol{u}|^2 + |D\boldsymbol{u}|^2 \, \mathrm{d}x - \int_U f \cdot \boldsymbol{u} \, \mathrm{d}x \right] \tag{4.20}$$

$$= \frac{1}{2} \int_U \frac{\partial}{\partial \rho} \alpha(\rho) \, |\boldsymbol{u}|^2 \, \mathrm{d}x = \frac{1}{2} \int_U \alpha'(\rho) \, |\boldsymbol{u}|^2 \, \mathrm{d}x \tag{4.21}$$

where

$$\alpha'(\rho) = (\underline{\alpha} - \overline{\alpha}) \frac{(1+q)q}{(\rho+q)^2}.$$

Note that the terms in (4.20) that don't contain $\alpha(\rho)$ vanish, even though $\boldsymbol{u}$ in a sense 'depends on' ρ. This is because the minimizing vector

$$\boldsymbol{u}(\rho) = \underset{v \in W}{\operatorname{argmin}} J_\rho(\boldsymbol{v})$$

satisfies the optimality condition

$$D_{\boldsymbol{u}} J_\alpha(\boldsymbol{u}) = 0 \quad \text{for} \quad \boldsymbol{u} = \boldsymbol{u}(\rho).$$

As mentioned earlier, ρ is constant on each element in a triangulation of U. So the above calculations show us that given a vector $\boldsymbol{\rho}$, specifying $\rho(\boldsymbol{x})$ in each element, the power gradient $D_\rho J_\rho$ is easy to compute.

Four numerical examples will be shown. All four are from [3], and figure 4.3 shows blueprints for the boundary conditions of each example (from [3]).

4.5.3 Example 1: Optimal diffusion (model problem continued)

We solve (4.7) using the boundary conditions (4.11) and the mixed formulation (4.16). The MMA algorithm is run until

(i) the absolute value of the relative change in the objective function $J_\rho(\boldsymbol{u}_h)$ is less than 0.05%

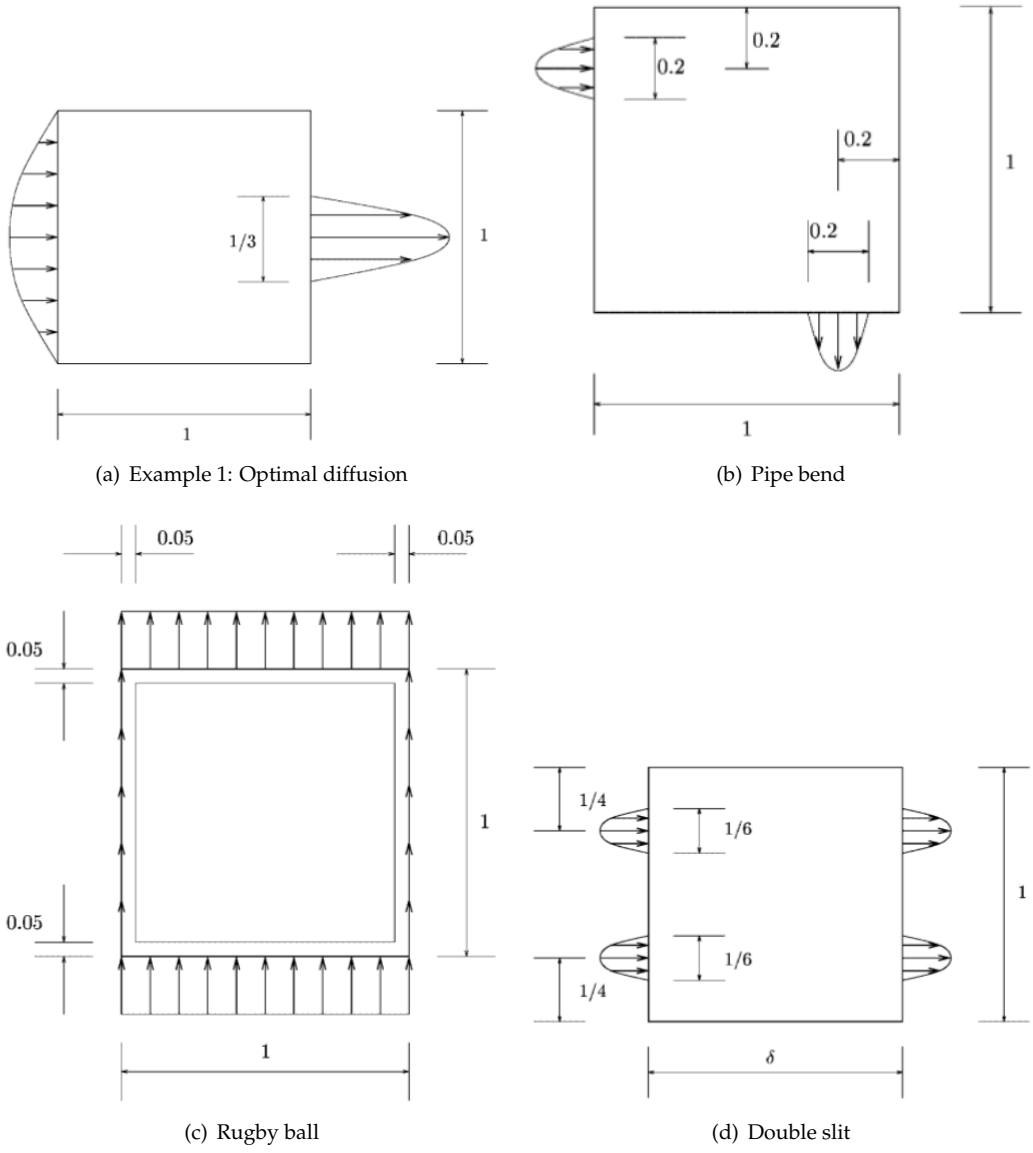
$$100 \frac{J_\rho^{i+1}(\boldsymbol{u}_h) - J_\rho^i(\boldsymbol{u}_h)}{J_\rho^i(\boldsymbol{u}_h)} < 0.05,$$

(ii) the constraint (4.12) is violated by at most 0.05%;

(iii) 50 iterations are done without signs of convergence.

The value 0.05%, though quite arbitrary, has proven to be a good 'tolerance level'.

The result is shown in figure 4.4 and confirms our guess from earlier: the material is clustered towards the top and bottom of the domain, skewed towards the right. The value of the power functional before and after optimization is shown in table 4.1; it is 12.3 without any material, 673.5 with $\rho(\boldsymbol{x}) = 0.5$ for all $\boldsymbol{x} \in U$, and 30.61 after the optimization process, so clearly the optimization process has lowered the value of the objective function significantly. Borrvall & Petersson (2003, [3]) obtain a value of 30.46 after 33 iterations, so their result has essentially been replicated.



(a) Example 1: Optimal diffusion

(b) Pipe bend

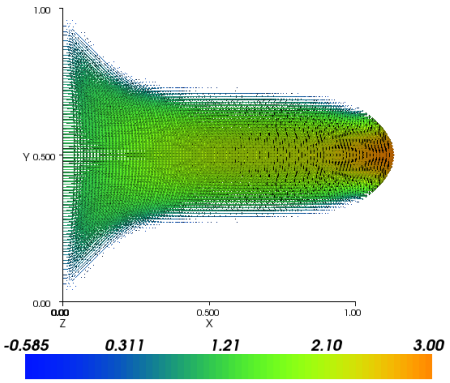
(c) Rugby ball

(d) Double slit

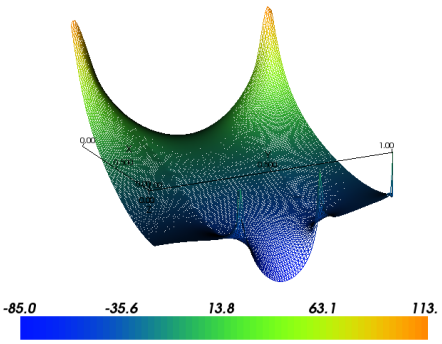
Figure 4.3: Blueprints of boundary conditions for example problems from [3].

Figure 4.5 shows the result of an attempt to solve the optimal diffusion problem with the least-squares formulation (4.15) (CY1). It clearly does not converge to the 'correct' solution, as in figure 4.4 using the mixed formulation. The problem is that the power functional with penalty term 4.10 is not calculated with sufficient accuracy. In section 3.8.1 it was shown that the CY1 formulation calculates the functional $J(v_h)$ in equation 3.18 with an error of 81.1% on a 160×160 grid. On a 100×100 grid the error is even higher at $PRE(CY1) = 176.2\%$, which is enormous compared to the mixed formulation: $PRE(MF) = -1.30 \cdot 10^{-5}$. It is therefore no surprise that the CY1 formulation cannot solve the topology optimization problem (4.9)-(4.10). The CY1 formulation shows a decent convergence rate for velocity in both the L^2 and H^1 norms (1.25 and 1.18, respectively), but it's not nearly accurate enough for the purpose of calculating the power functional $J(v)$.

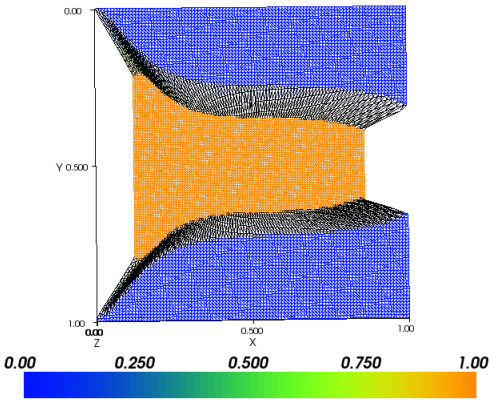
The QNE1h formulation does better than the CY1 formulation in the sense that it comes closer to the 'topologically correct' solution, but it is still far off from the result obtained using the mixed formulation. See figure H.1 in Appendix H. The objective function becomes 13.64 after 50 iterations



(a) Velocity vector field



(b) Pressure function

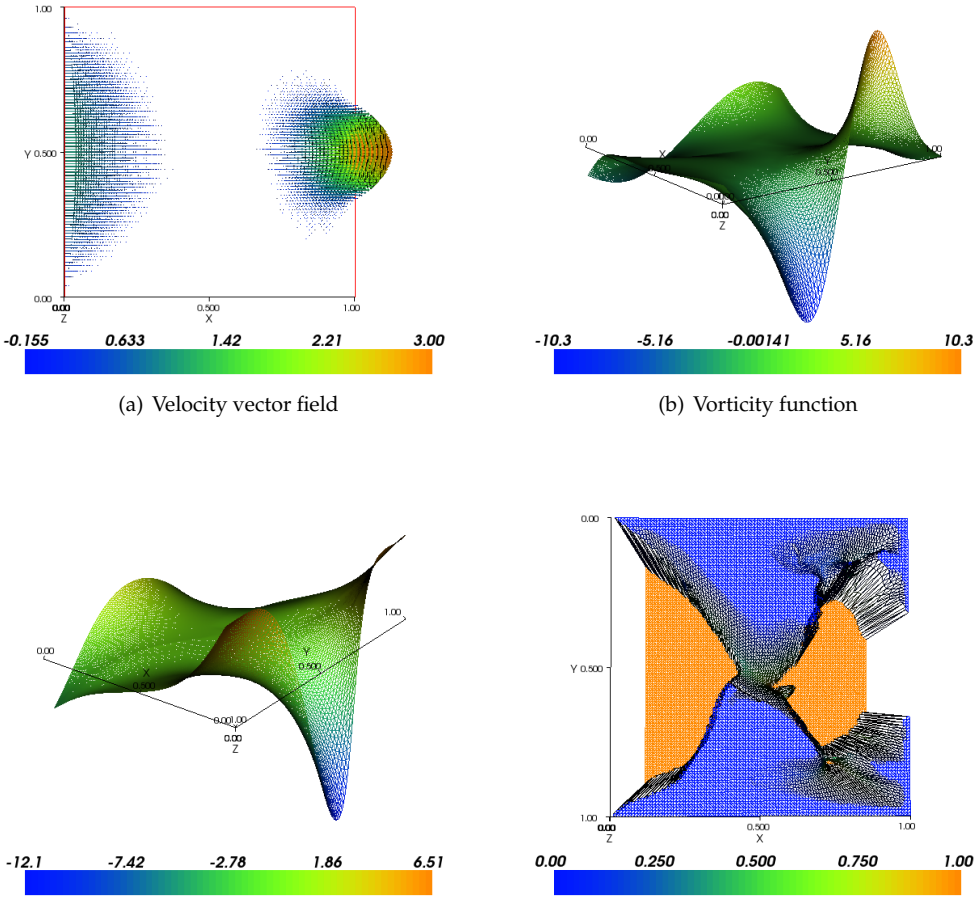


(c) Material distribution

Figure 4.4: Model problem (optimal diffusion) after optimization. Solved using mixed formulation with grid size $1/h = 100$, material ratio $\gamma = 0.5$, and $(\boldsymbol{u}_h, p_h) \in V_h \times P_h = \mathcal{P}_2 \times \mathcal{P}_1$.

compared to 30.61 for the mixed formulation.

the QNE2 formulation does better still, but the objective function, at 28.82, is still some way from 30.61. See figure H.2 in Appendix H. Furthermore, the resulting plot of the velocity vector field is asymmetrical across the line $y = 0.5$, which is obviously wrong, since the boundary conditions are perfectly symmetrical. So even on a large grid, the QNE2 formulation does not succeed in generating an exact solution (using the mixed formulation as the benchmark). The asymmetry of the numerical solution can probably be explained by the introduction of the vorticity into the formulation of the PDE. Since the definition $\boldsymbol{\omega} = \text{curl}\,(\boldsymbol{u})$ is enforced as a residual to be minimized, if this residual is greater than zero (to machine precision), an error is introduced into the computation. The propagation of such an error can then cause the asymmetry seen in figure H.2.



(a) Velocity vector field

(b) Vorticity function

(c) Pressure function

(d) Material distribution

Figure 4.5: Model problem (optimal diffusion) after optimization. Solved using LS-FEM formulation (4.15) with grid size $1/h = 100$, material ratio $\gamma = 0.5$, $(\boldsymbol{u}_h, p_h) \in V_h \times P_h = \mathcal{P}_1 \times \mathcal{P}_1$.

4.5.4 Example 2: Pipe bend

We now use the boundary conditions

$$\begin{cases} \boldsymbol{u}(x,y) = (0,0) & \text{for } x = 1 \text{ or } y = 1 \\ \boldsymbol{u} = (1-(10y-1)^2, 0) & \text{for } x = 0 \text{ and } y \in [0.7, 0.9] \\ \boldsymbol{u} = (-1+(10y-1)^2, 0) & \text{for } x \in [0.7, 0.9] \text{ and } y = 0 \\ \boldsymbol{u} = (0,0) & \text{for } x = 0 \text{ and } y \notin [0.7, 0.9] \\ \boldsymbol{u} = (0,0) & \text{for } y = 0 \text{ and } x \notin [0.7, 0.9]. \end{cases} \tag{4.22}$$

The result is shown in figure 4.6 using $\gamma = 0.08\pi$. This particular value of γ was chosen because it corresponds to the area of a quarter torus, which is known to be the optimal design in the case of an ideal fluid [3]. In the case of Stokes flow we see that the optimal design changes to an almost straight pipe between the two holes in the domain. A viscous fluid flowing through the domain loses power at the bottom of the pipe, whereas an ideal fluid loses power at the top of the pipe

Problem	method	γ	f0_ini	f0_fin	iters.	c.v.	conv.?	$1/h$	unkn.
BP1 plain	MF	1		12.3		0%		100	91003
BP1	B&P (2003)	0.5		30.46	33	0%		100	
	MF	0.5	673.5	30.61	16	-0.007511%	Yes	100	91003
	CY1	0.5	188.2	8.159	17	-0.02344%	No	100	40804
	QNE1h	0.5	238.5	13.64	50	-0.05741%	No	100	40804
	QNE1h	0.5	422.7	16.22	50	-0.077%	No	160	103684
	QNE2	0.5	658.4	28.82	48	-0.04437%	Close	100	101204
BP2 plain	MF	1		2.6953		0%			91003
BP2	B&P (2003)	0.2513		9.76	85	0%		100	
	MF	0.2513	111.9	9.7466	22	-0.02679%	Yes	100	91003
	CY1	0.2513	27.66	1.492	30	-0.004671%	No	100	40804
	QNE1h	0.2513	22.46	1.899	50	-0.04704%	No	100	40804
	QNE2	0.2513	90.71	4.702	50	-1.987%	No	100	101204
BP3 plain	MF	1				0%			91003
BP3	B&P (2003)	0.8		31.75	47	0%			
	MF	0.8	219.7	31.71	22	-0.03244%	Yes	100	91003
	CY1	0.8	77.52	8.453	39	-0.007647%	No	100	40804
	QNE1h	0.8	204.2	5.624	50	-0.1303%	No	100	40804
	QNE2	0.8	219.7	28.74	50	-0.1762%	No	100	101204

Table 4.1: Results from numerical example 1-3. f0_ini = initial value (before optimization) of $J_\rho(\boldsymbol{u}_h)$; f0_final = final value of $J_\rho(\boldsymbol{u}_h)$; C.v = constraint violation (allowed fluid volume γ); conv. = convergence.

as well. Therefore we see the steep gradient in the optimal design - it minimizes the dissipated power due to friction at the bottom of the pipe.

As shown in table 4.1, the power dissipation function is 2.70 in the absence of material, 111.9 when $\rho(\boldsymbol{x}) = 0.5$ for all $\boldsymbol{x} \in U$, and 9.75 after 22 iterations of the optimization process (using the mixed formulation). Borrvall & Petersson (2003) report a value of 9.76 after 85 iterations.

Neither of the LS-FEM methods converge towards the 'correct' solution obtained by the mixed formulation. See figure H.3 in Appendix H for a plot of the solution obtained using the CY1 formulation.

4.5.5 Example 3: Rugby ball

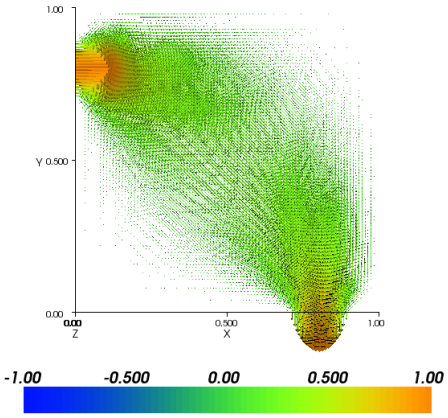
We now use the boundary conditions

$$\begin{cases} \boldsymbol{u}(x,y) = (0,1) & \text{for } x = 0 \text{ or } x = 1 \\ \boldsymbol{u}(x,y) = (0,1) & \text{for } y = 0 \text{ or } y = 1 \end{cases} \tag{4.23}$$

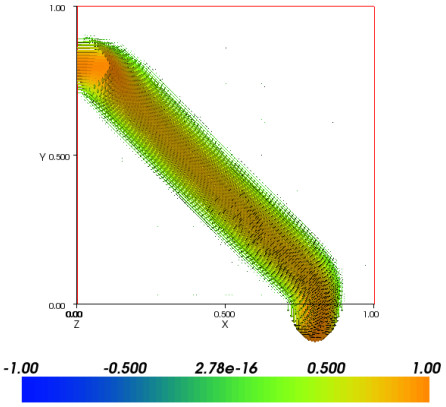
When $\alpha(\rho) = 0$ in all of U, the result is a uniform upward flow of magnitude 1. The pressure is also uniform in this case. In addition to the velocity boundary conditions, we prescribe fluid along the boundary of the domain:

$$\rho(\boldsymbol{x}) = 1 \quad \text{for } x \in [0, 0.05] \cup [0.95, 1] \text{ and } y \in [0, 0.05] \cup [0.95, 1]. \tag{4.24}$$

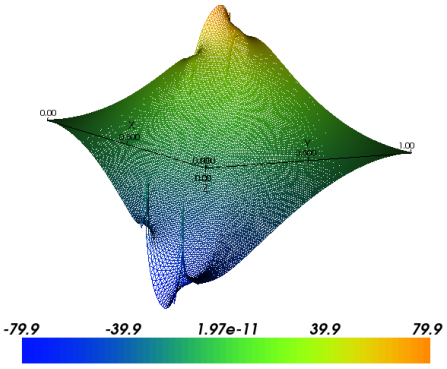
The result of topology optimization is shown in figure 4.7 for $\gamma = 0.8$.



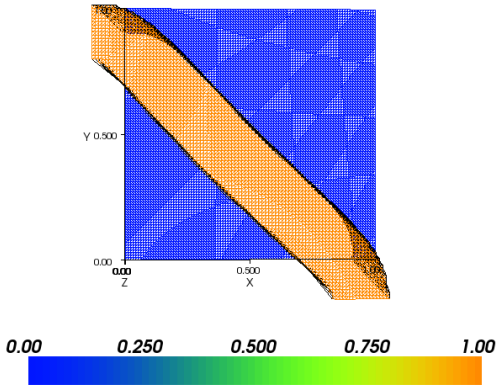
(a) Velocity (before)



(b) Velocity (after)



(c) Pressure function



(d) Material distribution

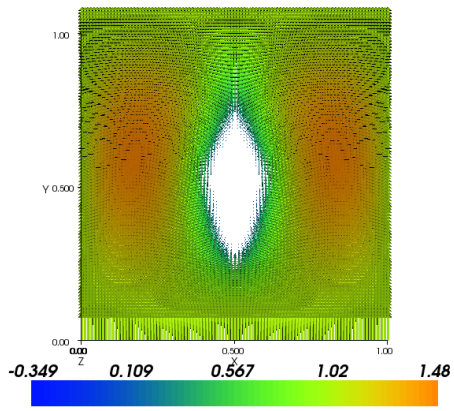
Figure 4.6: Pipe bend problem before and after optimization. Solved using mixed formulation with grid size $1/h = 100$, material ratio $\gamma = 0.08\pi$, and $(\boldsymbol{u}_h, p_h) \in V_h \times P_h = \mathcal{P}_2 \times \mathcal{P}_1$.

4.5.6 Example 4: Double slit

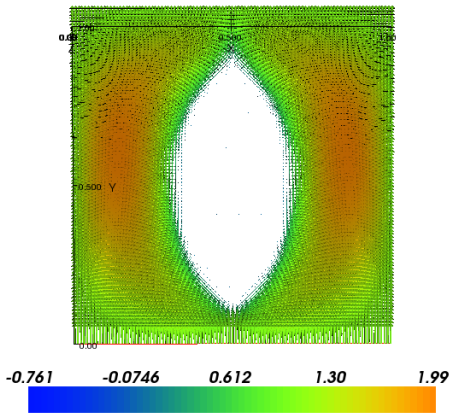
Let $\delta > 0$ be a constant and set $U = [0, \delta] \times [0, 1]$. We now use the boundary conditions

$$\begin{cases} \boldsymbol{u}(x,y) = (0,0) & \text{for } y = 0 \text{ or } y = 1 \\ \boldsymbol{u} = (1-(12y-9)^2, 0) & \text{for } x = 0 \text{ and } y \in [2/3, 5/6] \\ \boldsymbol{u} = (1-(12y-3)^2, 0) & \text{for } x = 0 \text{ and } y \in [1/6, 1/3] \\ \boldsymbol{u} = (1-(12y-9)^2, 0) & \text{for } x = \delta \text{ and } y \in [2/3, 5/6] \\ \boldsymbol{u} = (1-(12y-3)^2, 0) & \text{for } x = \delta \text{ and } y \in [1/6, 1/3] \\ \boldsymbol{u} = (0,0) & \text{for } x = 0 \text{ and } y \notin [1/6, 1/3] \cup [2/3, 5/6] \\ \boldsymbol{u} = (0,0) & \text{for } x = \delta \text{ and } y \notin [1/6, 1/3] \cup [2/3, 5/6]. \end{cases} \tag{4.25}$$

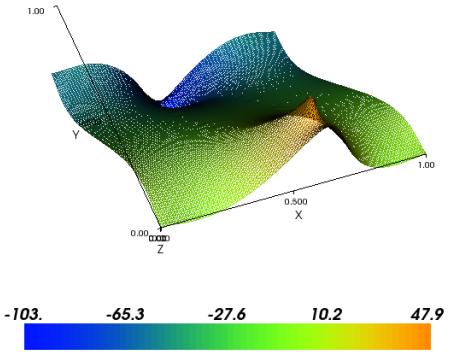
The results are reported in table 4.2. Figure 4.8 shows the resulting flow for $\delta = 1$, with and without material, with $\gamma = 1/3$. The optimal design is two straight pipes going from side to side.



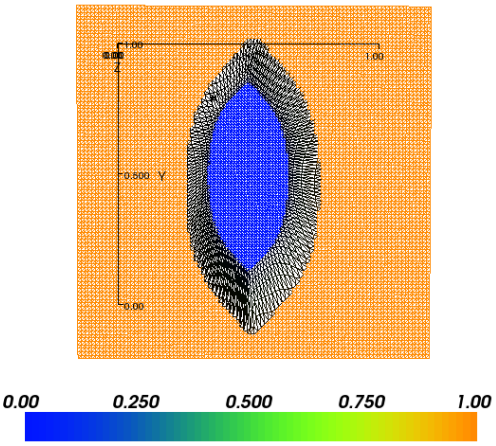
(a) Velocity ($\gamma = 0.95$)



(b) Velocity vector field ($\gamma = 0.8$)



(c) Pressure function ($\gamma = 0.8$)



(d) Material distribution ($\gamma = 0.8$)

Figure 4.7: Rugby ball problem solved using mixed formulation with grid size $1/h = 100$ and $(\boldsymbol{u}_h, p_h) \in V_h \times P_h = \mathcal{P}_2 \times \mathcal{P}_1$.

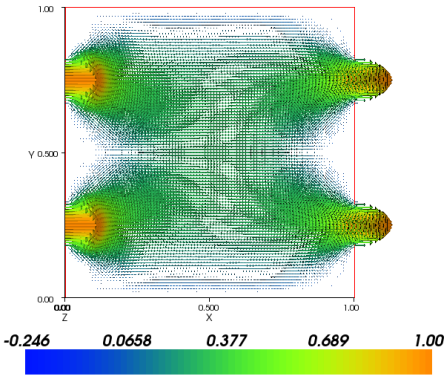
If we increase δ enough, say to $\delta = 1.5$, the topology of the optimal design changes to a single pipe in the middle, connected to the holes on both sides of the domain; see figure 4.9. The total dissipated power is 21.97 for $\delta = 1$ and 22.1 for $\delta = 1.5$. Note that the two values are very similar even though the domain is 50% larger for $\delta = 1.5$.

In the case of $\delta = 1.5$, using a q-value of 0.1 resulted in two independent channels, as for the case of $\delta = 1$. So to obtain the 'topologically correct' solution with $\delta = 1.5$, it was necessary to initially lower the value of q (see (4.19)) to 0.01, and then gradually increase q until the objective function fulfilled the stopping criteria (see also the discussion in [3]). This amounts to 'preconditioning' the problem with a low $q - value$, and then 'refining' the solution using gradually higher values of q. The plots on the left of figure 4.9 show the result of 20 iterations using $q = 0.01$. While the result is a single channel in the middle, and thus 'topologically correct', it is clear that the distinction between regions with or without fluid has become less clear. This is why we subsequently increase

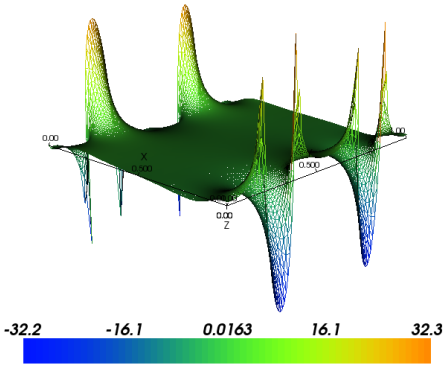
Problem	BP4 plain					
Method	MF	MF				
δ	1	1.5				
f0_fin	4.9562	5.1094				
Unknowns	91003	136253				
Problem	BP4					
Method	B&P (2003)	B&P (2003)	MF	MF	MF	MF
δ	1	1.5	1	1.5	1.5	1.5
q	0.1	0.01→0.1	0.1	0.1	0.01	0.01→0.05
f0_ini			129.1	29.36	29.36	29.36
f0_fin	25.67	27.64	21.9721	29.25	24.26	22.1
Iterations	61	236	19	26	20	63
C.v.			-0.028%	-0.01056%	-9.933%	-0.01618%
Success			Almost	No	No	Yes
Grid dims.	100x100	150x100	100x100	150x100	150x100	150x100
Unknowns			91003	136253	136253	136253

Table 4.2: Results from numerical example 4: 'double slit'. f0_ini = initial value (before optimization) of $J_\rho(\boldsymbol{u}_h)$; f0_final = final value of $J_\rho(\boldsymbol{u}_h)$; C.v = constraint violation (allowed fluid volume γ).

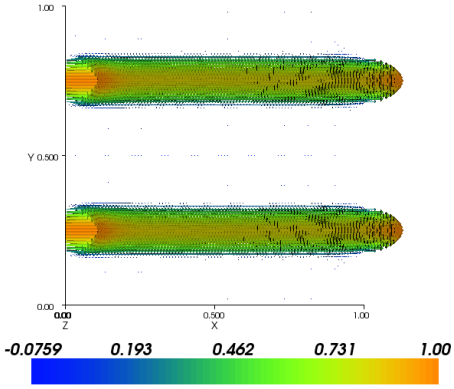
q to 'refine' the solution.



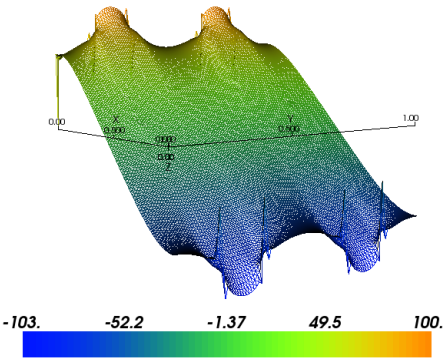
(a) Velocity (before)



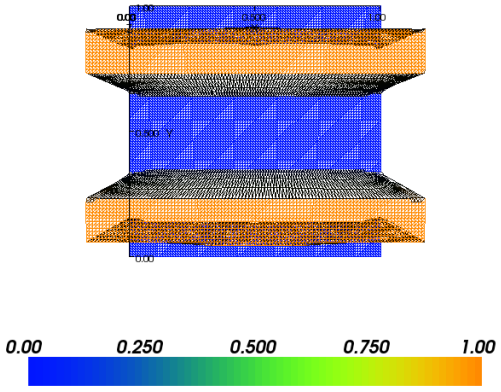
(b) Pressure (before)



(c) Velocity (after)

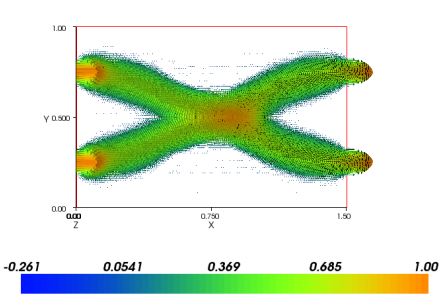


(d) Pressure (after)

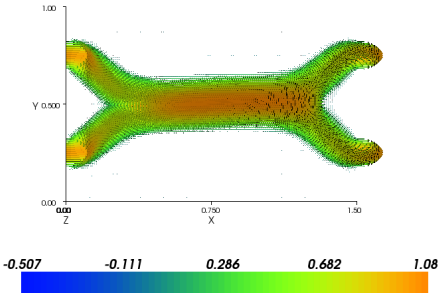


(e) Material distribution

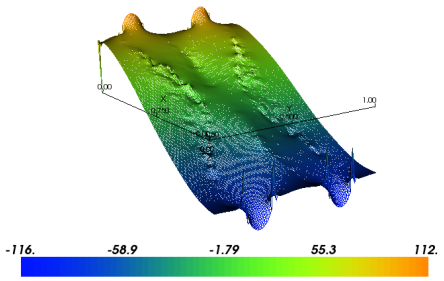
Figure 4.8: Double slit problem solved using mixed formulation with $\delta = 1$, $\gamma = 1/3$, grid size $1/h_x \times 1/h_y = 100 \times 100$ and $(\boldsymbol{u}_h, p_h) \in \mathcal{P}_2 \times \mathcal{P}_1$.



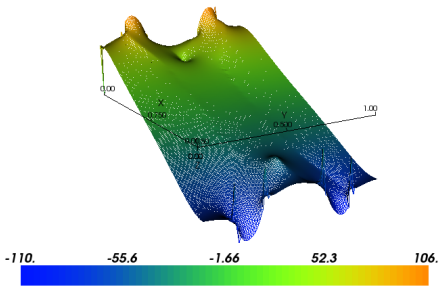
(a) Velocity; $q = 0.01$



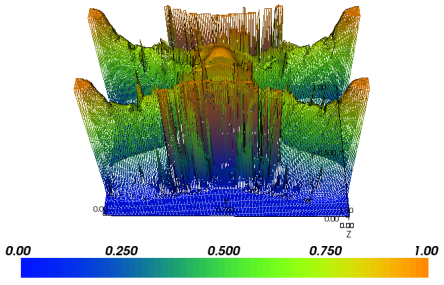
(b) Velocity; $q = 0.05$



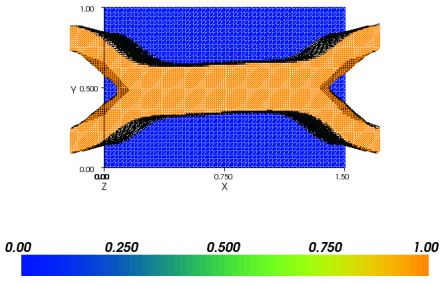
(c) Pressure; $q = 0.01$



(d) Pressure; $q = 0.05$



(e) Material; $q = 0.01$



(f) Material; $q = 0.05$

Figure 4.9: Double slit problem solved using mixed formulation with $\delta = 1.5$, $\gamma = 1/3$, grid size $1/h_x \times 1/h_y = 150 \times 100$ and $(\boldsymbol{u}_h, p_h) \in \mathcal{P}_2 \times \mathcal{P}_1$. The figures on the left show the result of 20 iterations with $q = 0.01$; the ones on the right show the result of 63 iterations with q being gradually increased from $q = 0.01$ to $q = 0.05$.

Chapter 5
Conclusion

5.1 Summary of findings

In this section we will summarize the findings that have been made in the report.
In part 2 we found that:

- The finite element method based on a mixed formulation (MF) of the Stokes problem outperforms the least-squares methods (LS-FEM) in 2D, in terms of accuracy and speed. The LS-FEM from Chang and Yang (2002, [5]) performed particularly poorly, and was dominated by another equal-order (same polynomial space for each variable) method, the h-weighted quasi-norm-equivalent (QNE) method. The CY method was particularly inaccurate for calculating the total potential power functional. The MF showed higher convergence rates than the comparable h^2-weighted QNE LS-FEM, while taking similar time to run.

- The convergence rates in [5] were confirmed in 2D, but could not be confirmed for the 3D model problem.

- The advantages of the LS-FEM, particularly the fact that they lead to symmetric, positive definite systems of linear equations, became relevant in three dimensions. For very large problems, the ratio of accuracy to speed moved in favor of the LS-FEM. Due to memory limitations, grids larger than $1/h^3 = 20^3$ were not investigated, however.

In part 3 we found that:

- It was possible to replicate the numerical results (example 1-4) from Borrvall and Petersson (2003, [3]) using a combination of a finite element method based on the mixed formulation of the Stokes problem and the MMA algorithm.

- The LS-FEM formulation from Chang & Yang (2002, [5]), CY1, proved to be too inaccurate to solve topology optimization problems, also on very large grids. It is conjectured that this is due to the failure of the method to calculate power functionals with sufficient accuracy.

- It was to some extent possible to solve topology optimization problems with QNE LS-FEMs. The first order h-weighted method, QNE1h, is not very accurate (compared to the mixed formulation), and does not produce satisfactory results, but does far better than the CY1 formulation, the only difference between the two methods being the weight h. This agrees with the statement in [2] on p.269 that "the weights are critical for the accuracy of

quasi-normal-equivalent DLSPs"[1]. The second order h^2-weighted method, QNE2, produces somewhat more accurate solutions, but the results are still far from satisfactory, compared to the mixed formulation. We conclude that topology optimization problems in two dimensions based on the Stokes equations should be solved using the mixed formulation.

5.2 Suggestions for future research

The above conclusions suggest several paths for future research.

- Convergence tests
  - Do 3D convergence tests in parallel on a cluster to compare the different methods on larger grids. This could in particular shed light on whether least-squares methods have an advantage over the mixed formulation for very large problems. The FEniCS software is designed to be able to run in parallel, so this should be possible without altering the code written for this paper significantly.
- Topology optimization in 2D
  - To look at problems with a forcing term, such as numerical example five in [3].
  - To investigate more complex domains and/or boundary conditions, and see how the various FEM formulations perform for such problems.
- Topology optimization in 3D
  - Do topology optimization in 3D. Essentially all the 2D examples in Chapter 4 generalize to 3D. The 'pipe bend' and 'rugby ball' would be particularly interesting to do in three dimensions.
  - Compare the performance of quasi-norm-equivalent methods to the mixed formulation for very large 3D problems.
- Analysis
  - Investigate why the weighted LS-FEM methods outperform the unweighted methods so significantly. And why the unweighted methods (CY) calculate power functionals so particularly inaccurately, despite of their reasonable velocity convergence rates (in both the L^2 and H^1 norms).
  - Investigate alternative topology optimization methods, and compare their performance to the methods investigated in this paper, using the example problems from Chapter 4.

[1] DLSP = discrete least-squares principle, the term Bochev and Gunzburger use for the class of discrete LS-FEM formulations.

List of Appendices

A. Notation

A.1. General notation

Adapted from [8].

- Let U and V be open subsets of R^n. We write

$$V \subset\subset U$$

  if $V \subset \overline{V} \subset U$ and $\overline{V}$ is compact, and say V is *compactly contained* in U.

- $L^p(U) = \left\{ u : U \to \mathbb{R} \mid u \text{ is Lebesgue measurable, } \|u\|_{L^p(U)} < \infty \right\}$, where

$$\|u\|_{L^p(U)} = \left(\int_U |u|^p \, \mathrm{d}x \right)^{1/p}, \quad 1 \leq p < \infty.$$

- $L^\infty(U) = \left\{ u : U \to \mathbb{R} \mid u \text{ is Lebesgue measurable, } \|u\|_{L^\infty(U)} < \infty \right\}$, where

$$\|u\|_{L^\infty(U)} = \operatorname{ess\,sup}_U |u| .$$

- $L^p_{\mathrm{loc}}(U) = \{u : U \to \mathbb{R} \mid v \in L^p(V) \text{ for each } V \subset\subset U\}$. We say that $u \in L^p_{\mathrm{loc}}(U)$ is locally summable.

- Let A, B be nonempty subsets of a metric space M equipped with the metric $\mathrm{d}(\cdot, \cdot)$. Then

$$\operatorname{dist}(A, B) := \inf_{x \in A, y \in B} \mathrm{d}(x, y).$$

- $B^0(x, r) = \{y \in \mathbb{R}^n \mid |x - y| < r\}$ = open ball in $\mathbb{R}^n$ with center x and radius $r > 0$.

- $B(x, r)$ = closed ball with center x, radius $r > 0$.

- $\alpha(n)$ = volume of unit ball $B(0, 1)$ in $\mathbb{R}^n = \dfrac{\pi^{n/2}}{\Gamma(n/2 + 1)}$.

- The average of f over the ball $B(x, r)$ is denoted by

$$\fint_{B(x,r)} f \, \mathrm{d}y := \frac{1}{\alpha(n) r^n} \int_{B(x,r)} f \, \mathrm{d}y.$$

A.2. Multi-index notation

The following definitions are from [18]. Let $\mathbb{N}$ denote the non-negative integers. An n-tuple

$$\alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{N}^n$$

is called a multi-index. The non-negative integer $|\alpha| := \alpha_1 + \cdots + \alpha_n$ is referred to as the length of the multi-index α. Let

$$D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_n}\right)^{\alpha_n} = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}} = \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n}.$$

Assume $u : U \to \mathbb{R}$, $\boldsymbol{x} \in U \subset \mathbb{R}^n$, and let k be a non-negative integer. Then

$$D^k u(\boldsymbol{x}) := \{D^\alpha u(\boldsymbol{x}) \,|\, |\alpha| = k\},$$

the set of all partial derivatives of order k. There are a few important special cases.

(i) If $k = 1$ we consider the elements of Du as being arranged in a vector called the gradient vector:

$$Du = \text{grad}\,(u) \left(\frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_n}\right).$$

(i) If $k = 2$ we regard the elements of $D^2 u$ as being arranged in a matrix called the Hessian matrix:

$$D^2 u = \begin{bmatrix} \dfrac{\partial^2 u}{\partial x_1^2} & \cdots & \dfrac{\partial^2 u}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 u}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 u}{\partial x_n^2} \end{bmatrix}$$

The trace of the Hessian matrix is known as the Laplacian of u:

$$\text{tr}(D^2 u) = \Delta u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}.$$

We have the additional rules:

$$\alpha! = \prod_{j=1}^{n} \alpha_j!$$

$$x^\alpha = \prod_{j=1}^{n} x_j^{\alpha_j}, \quad x \in \mathbb{R}^n.$$

The product rule for differentiation in terms of a multi-index α is given by

$$\partial^\alpha (fg) = \sum_{\beta+\gamma=\alpha} \frac{\alpha!}{\beta!\gamma!} (\partial^\beta f)(\partial^\gamma g),$$

where β and γ are multi-indices and $f \in C^{|\beta|}$, $g \in C^{|\gamma|}$.

Example A.1. Suppose that $n = 3$, and $\alpha = (\alpha_1, \alpha_2, \alpha_3)$, $\alpha_j \in \mathbb{N}$, $j = 1, 2, 3$. Then for u, a function of three variables x_1, x_2, x_3,

$$\begin{aligned} \sum_{|\alpha|=3} D^\alpha u = &\frac{\partial^3 u}{\partial x_1^3} + \frac{\partial^3 u}{\partial x_1^2 \partial x_2} + \frac{\partial^3 u}{\partial x_1^3} \\ &+ \frac{\partial^3 u}{\partial x_1 \partial x_2^2} + \frac{\partial^3 u}{\partial x_1 \partial x_3^2} + \frac{\partial^3 u}{\partial x_2^3} \\ &+ \frac{\partial^3 u}{\partial x_1 \partial x_2 \partial x_3} + \frac{\partial^3 u}{\partial x_2^2 \partial x_3} + \frac{\partial^3 u}{\partial x_2 \partial x_3^2} + \frac{\partial^3 u}{\partial x_3^3}. \end{aligned}$$

A.3. Vector-valued functions

From [8] with some additions.
Let U and V be open subsets of R^n.

- If $m > 1$ and $\boldsymbol{u} : U \to \mathbb{R}^m$, $\boldsymbol{u} = (u_1, ..., u_m)$, we define

$$D^\alpha \boldsymbol{u} = (D^\alpha u_1, ..., D^\alpha u_m)$$

for each multi-index α. Then

$$D^k \boldsymbol{u} = \{D^\alpha \boldsymbol{u} \mid |\alpha| = k\}$$

and

$$\left|D^k \boldsymbol{u}\right| = \left(\sum_{|\alpha|=k} |D^\alpha \boldsymbol{u}|^2\right)^{1/2}.$$

- In the special case $k = 1$ we write

$$D\boldsymbol{u} = \text{grad}\,(u) = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_m}{\partial x_1} & \cdots & \frac{\partial u_m}{\partial x_n} \end{bmatrix}$$

- If $m = n$, we have

$$\text{div}\,(\boldsymbol{u}) = \text{tr}(D\boldsymbol{u}) = \sum_{i=1}^{n} \frac{\partial u_i}{\partial x_i}.$$

- The Laplacian is defined as

$$\Delta \boldsymbol{u} = \sum_{i=1}^{n} \frac{\partial^2 u_i}{\partial x_i^2}.$$

- For two differentiable functions $\boldsymbol{u}, \boldsymbol{v} : U \to \mathbb{R}^n$ we write

$$D\boldsymbol{u} : D\boldsymbol{v} = \sum_{i,j=1}^{n} (D\boldsymbol{u})_{ij}(D\boldsymbol{v})_{ij} = \sum_{i,j=1}^{n} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j}.$$

- The spaces $C(U; \mathbb{R}^m)$, $L^p(U; \mathbb{R}^m)$, etc. consist of the functions $\boldsymbol{u} : U \to \mathbb{R}^m$, $\boldsymbol{u} = (u_1, ..., u_m)$, with $u_i \in C(U)$, $L^p(U)$, etc., $i = 1, ..., m$.

A.4. The curl operator

Let ϕ be a scalar function, $\boldsymbol{u} \in \mathbb{R}^3$, $\boldsymbol{v} \in \mathbb{R}^2$ be vectors, and define

$$\epsilon_{ijk} = \begin{cases} 0 & \text{if any of } i, j, k \text{ is the same} \\ 1 & \text{if } i, j, k \text{ is an even permutation of } 1, 2, 3 \\ -1 & \text{if } i, j, k \text{ is an odd permutation of } 1, 2, 3 \end{cases}$$

There are three even and three odd permutations of the elements in the set $\{1, 2, 3\}$. An even permutation results from making zero or two swaps (a swap is to interchange two elements), and an odd permutation consists of making one or three swaps.

The curl operator is defined as

- $\operatorname{curl}(\phi) = \left(\frac{\partial \phi}{\partial x_2}, -\frac{\partial \phi}{\partial x_1}\right)$
- $\operatorname{curl}(\boldsymbol{v}) = \frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{\partial x_2}$
- $\operatorname{curl}(\boldsymbol{u}) = \left(\epsilon_{ijk}\frac{\partial u_k}{\partial x_j}\right)_{i=1}^{3} = \left(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}, \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}, \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}\right)$.

B. Sobolev spaces and weak solutions to PDEs

B.1. Function spaces and weak derivatives

We follow [18]. Let U be an open set in R^n, let $k \in \mathbb{N}$, and let α be a multi-index (see A.2). We denote by $C^k(U)$ the set of all continuous real-valued functions defined on U such that $D^\alpha u$ is continuous on U for all $\alpha = (\alpha_1, ..., \alpha_n)$ with $|\alpha| \leq k$. Assuming that U is a bounded open set, $C^k(\overline{U})$ will denote the set of all u in $C^k(\overline{U})$ such that $D^\alpha u$ can be extended from U to a continuous function on $\overline{U}$, the closure of the set U, for all $\alpha = (\alpha_1, ..., \alpha_n)$, $|\alpha| \leq k$. $C^k(\overline{U})$ can be equipped with the norm

$$\|u\|_{C^k(\overline{U})} := \sum_{|\alpha| \leq k} \sup |D^\alpha u(x)|. \tag{B.1}$$

In particular when $k = 0$ we shall write $C(\overline{U})$ to denote the set of all continuous functions defined on $\overline{U}$.

The **support** of a continuous function u defined on an open set $U \subset \mathbb{R}^n$ is defined as the closure in U of the set $\{x \in U : u(x) \neq 0\}$. We shall write $\mathrm{spt}(u)$ for the support of u. Thus, $\mathrm{spt}(u)$ is the smallest closed subset of U such that $u = 0$ in $U \setminus \mathrm{spt}(u)$.
We denote by $C_0^k(U)$ the set of all u contained in $C^k(U)$ whose support is a bounded subset of U. Note that this implies that u and its first k derivatives, $D^i u$, $i \leq k$ vanish on the boundary of U, denoted by ∂U. So

$$C_0^k(U) = \left\{ u \in C^k(U) \mid \mathrm{spt}(u) \subseteq U,\ u = 0,\ D^k u = 0 \text{ on } \partial U \right\}. \tag{B.2}$$

We also define

$$C_0^\infty(U) = \bigcap_{k \geq 0} C_0^k(U). \tag{B.3}$$

Suppose that u is a smooth function, say $u \in C^k(U)$, with U an open subset of R^n, and let $v \in C_0^\infty(U)$; then the following integration by parts formula holds:

$$\int_U D^\alpha u(x) \cdot v(x)\, \mathrm{d}x = (-1)^{|\alpha|} \int_U u(x) \cdot D^\alpha v(x)\, \mathrm{d}x, \quad |\alpha| \leq k,\ \forall v \in C_0^\infty(U). \tag{B.4}$$

Note that all terms involving the boundary of U vanish because v and all of its derivatives are identically zero on ∂U.

Now suppose that u is a locally integrable function defined on U, i.e. $u \in L^1_{\mathrm{loc}}(U)$. Suppose also that there exists a function $w_\alpha \in L^1_{\mathrm{loc}}(U)$ such that

$$\int_U w_\alpha(x) \cdot v(x)\, \mathrm{d}x = (-1)^{|\alpha|} \int_U u(x) \cdot D^\alpha v(x),\ \forall v \in C_0^\infty(U); \tag{B.5}$$

then we say that w_α is a **weak derivative** of the function u of order α, and we write $w_\alpha = D^\alpha u$. The uniqueness of w_α is a consequence of the following lemma.

Lemma B.1 (DuBois Reymond's). *Suppose that $w \in L^1_{loc}(U)$, $U \subset \mathbb{R}^n$. If*

$$\int_U w(x)v(x)\,\mathrm{d}x = 0 \quad \forall v \in C_0^\infty(U)$$

then $w(x) = 0$ for almost every[1] $x \in U$.

Note that if u is sufficiently smooth, say $u \in C^k(U)$, then its weak derivative $D^\alpha u$ of order $|\alpha| \leq k$ coincides with the corresponding partial derivative in the classical pointwise sense. As an example, the function

$$u(x) = (1 - |x|)_+, \quad x \in U = \mathbb{R},$$

is not differentiable at the points 0 and ± 1, and thus $u \notin C(U)$. However, it can easily be verified that u has a weak derivative in U given by

$$w(x) = \begin{cases} 0, & x < -1, \\ 1, & x \in (-1, 0), \\ -1, & x \in (0, 1), \\ 0, & x > 1. \end{cases}$$

B.2. Sobolev spaces

Sobolev spaces are subspaces of the L^p spaces in which the weak derivatives of functions are required to have finite L^p norm. More precisely, let k be a nonnegative integer and suppose that $p \in [1, \infty]$. We define (with D^α denoting a weak derivative of order $|\alpha|$)

$$W^{k,p}(U) = \{u \in L^p(U) \mid D^\alpha u \in L^p(U), 1 \leq |\alpha| \leq k\}.$$

$W^{k,p}(U)$ is called a Sobolev space of order k; it is equipped with the (Sobolev) norm

$$\|u\|_{W^{k,p}(U)} := \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|^p_{L^p(U)} \right)^{1/p} \quad \text{when } 1 \leq p \leq \infty$$

and

$$\|u\|_{W^{k,\infty}(U)} := \sum_{|\alpha| \leq k} \|D^\alpha u\|_{L^\infty(U)} \quad \text{when } p = \infty.$$

Letting

$$|u|_{W^{k,p}(U)} := \left(\sum_{|\alpha| = k} \|D^\alpha u\|^p_{L^p(U)} \right)^{1/p},$$

for $p \in [1, \infty)$, we can write

$$\|u\|_{W^{k,p}(U)} := \left(\sum_{j=0}^{k} |u|^p_{W^{j,p}(U)} \right)^{1/p}.$$

Similarly, letting

$$|u|_{W^{k,\infty}(U)} := \sum_{|\alpha| = k} \|D^\alpha u\|_{L^\infty(U)},$$

[1] I.e. the set of x-values in U for which $w(x) \neq 0$, has Lebesgue measure zero, see Appendix E.

we have that

$$\|u\|_{W^{k,\infty}(U)} := \sum_{j=0}^{k} |u|_{W^{j,\infty}(U)}.$$

When $k \geq 1$, $|\cdot|_{W^{k,p}(U)}$ is called the Sobolev semi-norm [2] on $W^{k,p}(U)$.

An important special case corresponds to taking $p = 2$; the resulting space $W^{k,2}(U)$ is then a Hilbert space with the inner product

$$(u, v)_{W^{k,2}(U)} := \sum_{|\alpha| \leq k} (D^\alpha u, D^\alpha v).$$

For this reason it is typically written as $H^k(U)$ instead of $W^{k,2}(U)$. In this paper we will frequently refer to the Hilbertian Sobolev spaces $H^1(U)$ and $H^2(U)$. Our definitions of $W^{k,p}(U)$ and its norm and semi-norm, for $p = 2$ and $k = 1$, give:

$$\begin{aligned} H^1(U) &= \left\{ u \in L^2(U) \mid Du \in L^2(U) \right\}, \\ \|u\|_{H^1(U)} &= \left\{ \|u\|^2_{L^2(U)} + \|Du\|^2_{L^2(U)} \right\}^{1/2}, \\ |u|_{H^1(U)} &= \left\{ \|Du\|^2_{L^2(U)} \right\}^{1/2}. \end{aligned}$$

Similarly, for $p = 2$ and $k = 2$,

$$\begin{aligned} H^2(U) &= \left\{ u \in L^2(U) \mid D^\alpha u \in L^2(U),\ 1 \leq |\alpha| \leq 2 \right\}, \\ \|u\|_{H^2(U)} &= \left\{ \|u\|^2_{L^2(U)} + \sum_{1 \leq |\alpha| \leq 2} \|D^\alpha u\|^2_{L^2(U)} \right\}^{1/2}, \\ |u|_{H^2(U)} &= \left\{ \sum_{|\alpha| = 2} \|D^\alpha u\|^2_{L^2(U)} \right\}^{1/2}. \end{aligned}$$

Finally, we define the Sobolev space $H^1_0(U)$ as the closure of $C^\infty_0(U)$ in the norm of $\|\cdot\|_{H^1(U)}$; in other words, $H^1_0(U)$ is the set of all $u \in H^1(U)$ such that u is the limit in $H^1(U)$ of a sequence $\{u_m\}_{m=1}^\infty \in C^\infty_0(U)$. It can be shown (assuming that ∂U is sufficiently smooth) that

$$H^1_0(U) = \left\{ u \in H^1(U) \mid u = 0 \text{ on } \partial U \text{ in the trace sense}^3 \right\};$$

i.e. $H^1_0(U)$ is, in fact, the set of all functions u in $H^1(U)$ such that $u = 0$ on ∂U, the boundary of the set U.

The following result gives a useful connection between the L^2-norm of a function in $H^1_0(U)$ and the norm of its gradient.

Lemma B.2 (Poincaré-Friedrichs inequality)**.** *Suppose that U is a bounded open set in $\mathbb{R}^n$ (with a sufficiently smooth boundary[4] ∂U) and let $u \in H^1_0(U)$; then there exists a constant $c_*(U)$, independent of u, such that*

$$\int_U |u(x)|^2 \, \mathrm{d}x \leq c_* \sum_{i=1}^{n} \int_U \left| \frac{\partial u}{\partial x_i}(x) \right|^2 \mathrm{d}x. \tag{B.6}$$

[2] From [18]: When $k \geq 1$, $|\cdot|_{W^{k,p}(U)}$ is only a semi-norm rather than a norm because if $|\cdot|_{W^{k,p}(U)} = 0$ for $u \in W^{k,p}(U)$ it does not necessarily follow that $u(x) = 0$ for almost every x in U (all that is known is that $D^\alpha u(x) = 0$ for almost every $x \in U$, $|\alpha| = k$), so $|\cdot|_{W^{k,p}(U)}$ does not satisfy the first axiom of norm (which says that $\|u\| \geq 0$, $\forall u \in U$, and $\|u(x)\| = 0 \Leftrightarrow u(x) = 0$ a.e. in U [6])

[3] See appendix C.2.

[4] It is sufficient that U is a polygonal domain in $\mathbb{R}^2$ or a polyhedron in $\mathbb{R}^3$.

B.3. Weak solutions to variational problems

The following theorem asserts the existence of solutions to certain variational problems.

Theorem B.3 (Lax-Milgram). *Assume that H is a Hilbert space and that $a : H \to H$ is a bilinear mapping for which there exist $\alpha, \beta > 0$ such that*

(i) $|a(u,v)| \leq \alpha \|u\| \|v\| \quad \forall u, v \in H,$

(ii) $\beta \|u\|^2 \leq a(u,u) \quad \forall u \in H.$

Let $f : H \to \mathbb{R}$ be a bounded linear functional on H. Then there exists a unique $u \in H$ such that

$$a(u,v) = f(v) \quad \forall v \in H. \tag{B.7}$$

Proof. 1. For each $u \in H$, the mapping $v \mapsto a(u,v)$ is a bounded linear functional on the Hilbert space H (equipped with the inner product $(\cdot,\cdot)$); whence Riesz' Representation Theorem (see Appendix F) asserts the existence of a unique $w \in H$ such that

$$a(u,v) = (w,v) \quad \forall v \in H. \tag{B.8}$$

Let us write $Au = w$ whenever (B.8) holds; so that

$$a(u,v) = (Au,v) \quad \forall u, v \in H. \tag{B.9}$$

2. We first claim that $A : H \to H$ is a bounded linear operator. Indeed if $\lambda_1, \lambda_2 \in \mathbb{R}$ and $u_1, u_2 \in H$ we see for each $v \in H$ that

$$\begin{aligned}(A(\lambda_1 u_1 + \lambda_2 u_2), v) &= a(\lambda_1 u_1 + \lambda_2 u_2, v) \quad \text{by (B.9)} \\ &= \lambda_1 a(u_1, v) + \lambda_2 a(u_2, v) \\ &= \lambda_1 (Au_1, v) + \lambda_2 (Au_2, v) \quad \text{by (B.9) again} \\ &= (\lambda_1 Au_1 + \lambda_2 Au_2, v).\end{aligned}$$

This holds for each $v \in H$, so A is linear. Furthermore

$$\|Au\|^2 = (Au, Au) = a(u, Au) \leq \alpha \|u\| \|Au\| \quad \text{by (i).}$$

Consequently $\|Au\| \leq \alpha \|u\| \ \forall u \in H$, and so A is bounded.

3. Next we assert

$$\begin{cases} A \text{ is one-to-one}^5\text{(injective)} \\ R(A), \text{ the range of } A, \text{ is closed in } H. \end{cases} \tag{B.10}$$

To prove this, we compute

$$\beta \|u\|^2 \leq a(u,u) = (Au,u) \leq \|Au\| \|u\|.$$

Hence

$$\beta \|u\| \leq \|Au\|. \tag{B.11}$$

Now choose $u, v \in H$, $u \neq v$, for which $Au = Av$. Then $Au - Av = A(u-v) = 0$, so $\|A(u-v)\| = 0$. But this implies that $\beta \|u - v\| = 0 \Rightarrow u = v$ (due to the first axiom of norm:

[5] If $Au = Av$ then $u = v$, and if $u \neq v$ then $Au \neq Av$.

$\|u\| = 0 \Rightarrow u = 0$), so A is injective.
Now, $R(A)$ is closed in H if every Cauchy sequence $\{v_i\}_{i=1}^{\infty} \subset R(A)$ has limit point $v = \lim_{i\to\infty} v_i \in R(A)$. To show this, we consider an arbitrary Cauchy sequence $\{v_i\}_{i=1}^{\infty} \subset R(A)$ such that $v_i = Au_i \in R(A)$ for a sequence $\{u_i\}_{i=1}^{\infty} \subset H$. We must show that $\{u_i\}_{i=1}^{\infty}$ is also a Cauchy sequence with limit $u = \lim_{i\to\infty} \in H$. From (B.11) we see that

$$\beta \|u_i - u_j\| \leq \|A(u_i - u_j)\| = \|v_i - v_j\|, \quad \forall i, j \in \mathbb{N},$$

which shows that $\{u_i\}_{i=1}^{\infty}$ is indeed a Cauchy sequence. Furthermore, by the continuity of A, we have that

$$v = \lim_{i\to\infty} Au_i = A \lim_{i\to\infty} u_i = Au \in R(A).$$

Since $\{v_i\}_{i=1}^{\infty}$ was arbitrary, we have shown that $R(A)$ is closed.

4. We now demonstrate that

$$R(A) = H. \tag{B.12}$$

For if not, then, since $R(A)$ is closed, there would exist (by the Projection Theorem, see F) a nonzero element $w \in H$ with $w \in R(A)^{\perp}$ (because $H = R(A) \oplus R(A)^{\perp}$). But this fact in turn implies the contradiction $\beta \|w\|^2 \leq a(w, w) = (Aw, w) = 0$, where the last equality is due to $Aw \in R(A)$ and $w \in R(A)^{\perp}$.

5. Next, we observe once more from Riesz' Representation Theorem that

$$f(v) = (w, v) \quad \forall v \in H$$

for some element $w \in H$. We then utilize (B.10) and (B.12) to find $u \in H$ satisfying $Au = w$. Then $a(u, v) = (Au, v) = (w, v) = f(v)$, and this is (B.7).

6. Finally, we show that there is at most one element $u \in H$ verifying (B.7). For if both $a(u, v) = f(v)$ and $a(\tilde{u}, v) = f(v)$, then $a(u - \tilde{u}, v) = 0$ for all $v \in H$. We set $v = u - \tilde{u}$ to find $\beta \|u - \tilde{u}\|^2 \leq a(u - \tilde{u}, u - \tilde{u}) = 0$. This implies that $u = \tilde{u}$, and therefore, since $u, \tilde{u} \in H$ were arbitrary, u is unique. This completes the proof.

□

B.4. Galerkin orthogonality and Céa's lemma

Consider the elliptic boundary value problem

$$-\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x), \quad x \in U, \tag{B.13}$$

$$u = 0 \qquad \text{on } \partial U, \tag{B.14}$$

where U is a bounded open set in $\mathbb{R}^n$, $a_{ij} \in L^{\infty}(U)$, $i, j = 1, ..., n$; $b_i \in W_1^{\infty}(U)$, $i = 1, ..., n$, $c \in L^{\infty}(U)$, $f \in L^2(U)$, and assume that there exists a positive constant $\tilde{c}$ such that

$$\sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \geq \tilde{c}\sum_{i=1}^{n} \xi_i^2 \quad \forall \xi = (\xi_1, ..., \xi_n) \in \mathbb{R}^n, \ \forall x \in \overline{U}.$$

The weak formulation of (B.13)-(B.14) is

$$\text{find } u \in H_0^1(U) \text{ such that } a(u, v) = l(v) \quad \forall v \in H_0^1(U), \tag{B.15}$$

where the bilinear functional $a(\cdot,\cdot)$ and the linear functional $l(\cdot)$ are defined by

$$a(u,v) = \sum_{i,j=1}^{n} \int_U \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j}\,\mathrm{d}x + \sum_{i=1}^{n}\int_U b_i(x)\frac{\partial u}{\partial x_i}\,\mathrm{d}x + \int_U c(x)uv\,\mathrm{d}x,$$
$$l(v) = \int_U f(x)v(x)\,\mathrm{d}x.$$

Applying the Lax-Milgram theorem B.3 to (B.15) with $H = H_0^1(U)$ leads to the following result,[18]

Theorem B.4. *Suppose that $a_{ij} \in L^\infty(U)$, $i,j = 1,...,n$, $b_i \in W_1^\infty(U)$, $i = 1,...,n$, $c \in L^\infty(U)$, $f \in L^2(U)$ and assume that the following conditions are fulfilled:*

$$\sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \geq \tilde{c}\sum_{i=1}^{n}\xi_i^2 \quad \forall \xi = (\xi_1,...,\xi_n) \in \mathbb{R}^n,\ \forall x \in \overline{U}.$$

and

$$c(x) - \frac{1}{2}\sum_{i=1}^{n}\frac{\partial b_i}{\partial x_i} \geq 0, \quad x \in \overline{U}.$$

Then the boundary value problem (B.13)-(B.14) *possesses a unique weak solution $u \in H_0^1(U)$. In addition,*

$$\|u\|_{H^1(U)} \leq \frac{1}{c_0}\|f\|_{L^2(U)}. \tag{B.16}$$

Proof. See [18] p.20. □

Now suppose that V_h is a finite-dimensional subspace of H_0^1, without making further precise assumptions on the nature of V_h (although we implicitly assume that V_h consists of continuous piecewise polynomials defined on a subdivision of "fineness" h of the computational domain U). The finite element approximation of (B.15) is

$$\text{find } u_h \in V_h \text{ such that } a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h. \tag{B.17}$$

As, by hypothesis, V_h is contained in $H_0^1(U)$ it follows from the Lax-Milgram theorem that (B.17) has a unique solution u_h in V_h. Moreover, (B.15) holds for any $v = v_h \in V_h$; namely,

$$a(u, v_h) = l(v_h) \quad \forall v_h \in V_h.$$

Subtracting (B.17) from this identity we deduce that

$$a(u - u_h, v_h) = 0 \quad \forall v_h \in V_h, \tag{B.18}$$

which is referred to as **Galerkin orthogonality**.

Condition (ii) in B.3 can be expressed as

$$a(v,v) \geq c_0\|v\| = c_0\left(\int_U |v|^2\,\mathrm{d}x + \sum_{i=1}^{n}\int_U \left|\frac{\partial v}{\partial x_i}\right|^2\,\mathrm{d}x\right),$$

where $c_0 > 0$ is a constant. Setting $v = u - u_h \in H_0^1(U)$ we have that

$$\|u - u_h\|_{H_0^1(U)}^2 \leq \frac{1}{c_0}a(u - u_h, u - u_h),$$

and it follows from (B.18) that

$$\|u - u_h\|_{H_0^1(U)}^2 \leq \frac{1}{c_0}a(u - u_h, u - u_h). \tag{B.19}$$

Condition (i) in B.3 with $H = H_0^1(U)$ is given by

$$|a(w,v)| \leq c_1 \|w\|_{H^1(U)} \|v\|_{H^1(U)}.$$

Setting $w = u - v_h$ we obtain the inequality

$$a(u-u_h, u-v_h) \leq c_1 \|u-u_h\|_{H^1(U)} \|u-v_h\|_{H^1(U)}. \tag{B.20}$$

Combining (B.19) with (B.20) we deduce that

$$\|u-u_h\|_{H^1(U)} \leq \frac{c_1}{c_0} \|u-v_h\|_{H^1(U)} \quad \forall v_h \in V_h. \tag{B.21}$$

Thus we have proved the following result.

Lemma B.5 (Céa's Lemma)**.** *The finite element approximation u_h to $u \in H_0^1(U)$, the weak solution to the problem* (B.13)-(B.14)*, is the near-best fit to u in the norm $\|\cdot\|_{H^1(U)}$; i.e.,*

$$\|u-u_h\|_{H^1(U)} \leq \frac{c_1}{c_0} \min_{v_h \in V_h} \|u-v_h\|_{H^1(U)}.$$

B.5. The Bramble-Hilbert lemma

The following result is a key tool for error analysis of finite element methods. See [18] p.75.

Lemma B.6 (Bramble-Hilbert lemma)**.** *Suppose that $U \subset \mathbb{R}^n$ is a bounded open set in $\mathbb{R}^n$ and assume that U is star-shaped with respect to every point in a set $B \subset U$ of positive measure contained in U (i.e. for all $x \in U$ the closed convex hull of $\{x\} \cup B$ is a subset of U). Let l be a bounded linear functional on the Sobolev space $W^{m,p}(U)$, $m \geq 1$, $1 < p < \infty$, such that $l(Q) = 0$ for any polynomial Q of degree $\leq m-1$. Then there exists a constant $C_1 > 0$ such that*

$$|l(v)| \leq C_1 |v|_{W^{m,p}(U)} \quad \forall v \in W^{m,p}(U).$$

Proof. See [18] p.75. □

Lemma B.6 leads to the *optimal error bound* in Proposition 3.13.

C. Calculus

C.1. Smoothing using mollifiers

In this section we describe a tool called a 'mollifier', that allows us to build smooth approximations to a given function (see [8] p.629). Notation: If $U \subset \mathbb{R}^n$ is open, $\epsilon > 0$, write

$$U_\epsilon := \{x \in U \mid \operatorname{dist}(x, \partial U) > \epsilon\}.$$

Definition C.1. (i) Define $\eta \in C^\infty(R^n)$ by

$$\eta(x) := \begin{cases} C \exp\left(\dfrac{1}{|x|^2 - 1}\right) & \text{if } |x| < 1 \\ 0 & \text{if } |x| \geq 1, \end{cases}$$

the constant $C > 0$ selected so that $\displaystyle\int_{R^n} \eta \, dx = 1$.

(ii) For each $\epsilon > 0$, set

$$\eta_\epsilon(x) := \frac{1}{\epsilon^n} \eta(\frac{x}{\epsilon}).$$

We call η the *standard mollifier*. The functions η_ϵ are C^∞ and satisfy

$$\int_{R^n} \eta_\epsilon \, dx = 1, \ \operatorname{spt}(\eta_\epsilon) \subset \mathrm{B}(0, \epsilon).$$

Definition C.2. If $f : U \to \mathbb{R}$ is locally integrable, define its mollification

$$f^\epsilon := \eta_\epsilon * f \quad \text{in } U_\epsilon.$$

That is,

$$f^\epsilon = \int_U \eta_\epsilon(x - y) f(y) \, dy = \int_{\mathrm{B}(0,\epsilon)} \eta_\epsilon(y) f(x - y) \, dy$$

for $x \in U_\epsilon$.

Theorem C.3 (Properties of mollifiers)**.** *(i)* $f^\epsilon \in C^\infty(U_\epsilon)$.

(ii) $f^\epsilon \to f$ *a.e. as* $\epsilon \to 0$.

(iii) If $f \in C(U)$, *then* $f^\epsilon \to f$ *uniformly on compact subsets of* U.

(iv) If $1 \leq p < \infty$ *and* $f \in L^p_{loc}(U)$, *then* $f^\epsilon \to f$ *in* $L^p_{loc}(U)$.

Proof. See [8] p.630. □

C.2. Traces

The following theorem is from [8] section 5.5.
Take $1 \leq p \leq \infty$.

Theorem C.4. *(Trace theorem for domain with C^1 boundary)*
Assume U is bounded and ∂U is C^1. Then there exists a bounded linear operator

$$T : W^{1,p}(U) \to L^p(\partial U)$$

such that

(i)

$$Tu = u|_{\partial U} \quad \text{if} \quad u \in W^{1,p}(U) \cap C(\bar{U}), \quad \text{and}$$

(ii)

$$\|Tu\|_{L^p(\partial U)} \leq C \|u\|_{W^{1,p}(U)},$$

for each $u \in W^{1,p}(U)$, with the constant C depending only on p and U.

Definition C.5. We call Tu the trace of u on ∂U.

Proof. See [8] p.258. □

The following theorem describes what it means for a function to have zero trace when ∂U is C^1.

Theorem C.6. *(Trace-zero functions in $W^{1,p}$)*
Assume U is bounded and ∂U is C^1. Suppose furthermore that $u \in W^{1,p}(U)$. Then

$$u \in W_0^{1,p}(U) \;\Leftrightarrow\; Tu = 0 \qquad \text{on } \partial U.$$

Proof. See [8] p.259. □

It is possible to use domains that do not have a C^1 boundary, but then we need to make other assumptions about the smoothness of ∂U. The boundary of some simple domains are not C^1, for example a box in $\mathbb{R}^2$, due to the discontinuous derivatives at the corners. The boundary can be interpreted as a Lipschitz continuous function, however. The box in R^2 for instance has a Lipschitz continuous boundary with Lipschitz constant 1. This can be seen by centering the coordinate system at one of the corners, and re-orienting it, such that the corner of the box is locally equal to the graph of $f(x) = |x|$. Since the numerical value of the derivative of $|x|$ is one, the Lipschitz constant is 1.

Definition C.7 (Lipschitz continuous function)**.** A function $f : X \to Y$ between two normed spaces X, Y is Lipschitz continuous if there exists a constant $K \geq 0$ such that

$$\frac{\|f(x_1) - f(x_2)\|_Y}{\|x_1 - x_2\|_X} \leq K \quad \forall x_1, x_2 \in X.$$

In the case that $f : \mathbb{R} \to \mathbb{R}$ we can define the norm

Definition C.8 (Lipschitz norm)**.**

$$\|f\|_{Lip(U)} = \|f\|_{L^\infty(U)} + \sup\left\{\frac{|f(x) - f(y)|}{|x - y|} : x, y \in U; x \neq y\right\}.$$

and the corresponding space of Lipschitz functions

$$\mathrm{Lip}(U) = \left\{ f \in L^\infty(U) \mid \|f\|_{Lip(U)} < \infty \right\}.$$

Definition C.9 (Lipschitz domain [19])**.** Let $n \in \mathbb{N}$ and $U \subset \mathbb{R}^n$ be an open, bounded subset of $\mathbb{R}^n$. U is a Lipschitz domain with Lipschitz boundary ∂U if

$$\text{for all } p \in \partial U \text{ there exists } r > 0 \text{ and a map } h_p : \mathrm{B}(p,r) \to \mathrm{B}(0,1) \text{ such that}$$

- h_p is a bijection
- h_p and h_p^{-1} are both Lipschitz continuous functions
- $h_p(\partial U \cap \mathrm{B}(p,r)) = \{(x_1, ..., x_n) \in \mathrm{B}(0,1) \mid x_n = 0\}$
- $h_p(U \cap \mathrm{B}(p,r)) = \{(x_1, ..., x_n) \in \mathrm{B}(0,1) \mid x_n > 0\}$.

The rest of this section is from [4].

Definition C.10 (Lipschitz boundary)**.** We say U has a Lipschitz boundary ∂U provided there exists a collection of open sets O_i, a positive parameter ϵ, an integer N, and a finite number M, such that

(i) for all $x \in \partial U$, there is an open set O_i such that $\mathrm{B}(x, \epsilon) \subset O_i$,

(ii) no more than N of the sets O_i intersect non-trivially,

(iii) and each domain $O_i \cap U = O_i \cap U_i$, where U_i is a domain whose boundary is a graph of a Lipschitz function ϕ_i. That is

$$U_i = \left\{ (x,y)\mathbb{R}^{n-1} \times \mathbb{R} \mid y < \phi_i(x) \right\}$$

satisfying

$$\|\phi_i\|_{\mathrm{Lip}(\mathbb{R}^{n-1})} \leq M.$$

One consequence of this definition is that we can now relate Sobolev spaces on a given domain to those on all of $\mathbb{R}^n$.

Theorem C.11 (Extention mapping)**.** *Suppose that U has a Lipschitz boundary. Then there is an extension mapping $E : W^{k,p}(U) \to W^{k,p}(\mathbb{R}^n)$ defined for all nonnegative integers k and real numbers p in the range $1 \leq p \leq \infty$ satisfying $Ev|_U = v$ for all $v \in W^{k,p}(U)$ and*

$$\|Ev\|_{W^{k,p}(\mathbb{R}^n)} \leq C \|v\|_{W^{k,p}(U)}$$

where C is independent of v.

The following proposition tells us how to interpret ∂U in the special case when $U = \mathrm{B}(0,1)$, which is a Lipschitz domain.

Proposition C.12. *Let U denote the unit disk in $\mathbb{R}^2$. For all $u \in H^1(U)$, the restriction $u|_{\partial U}$ may be interpreted as a function in $L^2(\partial U)$ satisfying*

$$\|u\|_{L^2(\partial U)} \leq 8^{1/4} \|u\|^{1/2}_{L^2(U)} \|u\|^{1/2}_{H^1(U)}.$$

Proof. See [4] p.37. □

Remark. ([4] p.39) Note that this proposition does not assert that pointwise values of u on ∂U make sense, only that $u|_{\partial U}$ is square integrable on ∂U. This leaves open the possibility that u could be infinite at a dense set of points on ∂U. For smooth functions, the trace defined here is the same as the ordinary pointwise restriction to the boundary.

For Lipschitz domains in general we have the following trace theorem that we will state without proof ([4]).

Theorem C.13 (Trace theorem for Lipschitz domain)**.** *Suppose that U has a Lipschitz boundary, and that p is a real number in the range $1 \leq p \leq \infty$. Then there exists a constant C such that*

$$\|v\|_{L^p(\partial U)} \leq C \|v\|_{L^p(U)}^{1-1/p} \|v\|_{W^{1,p}}^{1/p} \quad \forall v \in W^{1,p}(U).$$

So when U is a Lipschitz domain, we may interpret $u|_{\partial U}$ as an L^p function.

D. Calculus of variations

D.1. The Euler-Lagrange equation

The material in this section is from [8] chapter 8.

We will consider PDE problems that can be stated in the form

$$A[u] = 0 \tag{D.1}$$

where $A[\cdot]$ is a (possibly) nonlinear partial differential operator, and u is the unknown. $A[\cdot]$ is the "derivative" of an appropriate "energy" functional $A[\cdot] = I'[\cdot]$. The problem can then be stated as

$$I'[u] = 0.$$

The point is that while it may be difficult to solve (D.1) directly, it may be easier to find critical points of the functional $I[\cdot]$.

Suppose $U \subset \mathbb{R}^n$ is a bounded, open set with smooth boundary ∂U, and we are given a smooth function

$$L : \mathbb{R}^n \times \mathbb{R} \times \bar{U} \to \mathbb{R}.$$

We call L the Lagrangian. We will write

$$L = L(p, z, x) = L(p_1, ..., p_n, z, x_1, ..., x_n)$$

for $p \in \mathbb{R}^n$, $z \in \mathbb{R}$, and $x \in U$. We also write

$$\begin{aligned} D_p L &= (L_{p_1}, ..., L_{p_n}) \\ D_z L &= L_z \\ D_x L &= (L_{x_1}, ..., L_{x_n}) \end{aligned}$$

where subscripts denote differentiation with respect to the subscripted variable. We assume $I[\cdot]$ to have the explicit form

$$I[w] := \int_U L(Dw(x), w(x), x)\, \mathrm{d}x,$$

for smooth functions $w : \bar{U} \to \mathbb{R}$ satisfying, say, the boundary condition

$$w = g \qquad \text{on } \partial U. \tag{D.2}$$

Suppose now that a particular smooth function u satisfying the boundary condition $u = g$ on ∂U happens to be a minimizer of $I[\cdot]$ among all functions w satisfying (D.2). Then u solves the Euler-Lagrange equation

$$-\sum_{i=1}^{n} \left(L_{p_i}(Du, u, x)\right)_{x_i} + L_z(Du, u, x) = 0 \qquad \text{in } U. \tag{D.3}$$

To demonstrate this, consider any smooth function $v \in C_c^\infty(U)$, and the real-valued function

$$i(\tau) := I[u + \tau v] \quad (\tau \in \mathbb{R}) = \int_U L(Du + \tau Dv, u + \tau v, x)\,\mathrm{d}x,$$

which has a minimum at $\tau = 0$. Differentiating $i(\tau)$ we obtain what is known as the *first variation* of $I[\cdot]$:

$$0 = i'(0) = \int_U \left[\sum_{i=1}^n \left(L_{p_i}(Du, u, x) \right)_{x_i} + L_z(Du, u, x) \right] v\,\mathrm{d}x,$$

which holds for all test functions $v \in C_c^\infty(U)$, and thus u solves (D.3).

The *second variation* of $I[\cdot]$ is obtained by observing that since u is a minimum of $I[\cdot]$, we must have $i''(0) \geq 0$.

$$\begin{aligned} 0 \leq i''(0) = \int_U & \sum_{i,j=1}^n L_{p_i p_j}(Du, u, x) v_{x_i} v_{x_j} \\ & + 2 \sum_{i=1}^n L_{p_i z}(Du, u, x) v_{x_i} v + L_{zz}(Du, u, x) v^2\,\mathrm{d}x, \quad \forall v \in C_c^\infty(U). \end{aligned} \tag{D.4}$$

It can be shown that (D.4) leads to the estimate

$$\sum_{i,j=1}^n L_{p_i p_j}(Du, u, x) \xi_i \xi_j \geq 0 \quad (\xi \in \mathbb{R}^n, x \in U), \tag{D.5}$$

which gives a clue to the basic convexity assumptions needed for the existence of solutions to the Euler-Lagrange equation.

D.2. Coercivity, convexity, lower semi-continuity

Assume that $1 < q < \infty$ is fixed, and suppose

$$\begin{cases} \text{there exist constants } \alpha > 0, \beta \geq 0 \text{ such that} \\ L(p, z, x) \geq \alpha\,|p|^q - \beta \\ \text{for all } p \in \mathbb{R}^n,\ z \in \mathbb{R},\ x \in U. \end{cases} \tag{D.6}$$

This is equivalent to the following "coercivity condition":

$$I[w] \geq \delta\,\|Dw\|_{L^q(U)}^q - \gamma \tag{D.7}$$

for $\gamma := \beta\,|U| = \beta m(U)$, $m(\cdot)$ denoting the Lebesgue measure, and some constant $\delta > 0$. Thus $I[w] \to \infty$ as $\|Dw\|_{L^q} \to \infty$. We denote by

$$\mathcal{A} := \left\{ w \in W^{1,q}(U) \mid w = g \qquad \text{on } \partial U \text{ in the trace sence} \right\}$$

the set of *admissible* functions w.

Definition D.1 (Weak lower semi-continuity)**.** We say that a function $I[\cdot]$ is (sequentially) weakly lower semi-continuous on $W^{1,q}(U)$, provided

$$I[u] \leq \liminf_{k \to \infty} I[u_k]$$

whenever

$$u_k \rightharpoonup u \quad \text{weakly in} \quad W^{1,q}(U).$$

From (D.5) we know that the functional $L(p, z, x)$ is convex in its first argument, which motivates the following theorem.

Theorem D.2. *Assume that L is smooth, bounded below, and in addition*

$$\text{the mapping } p \mapsto L(p, z, x) \text{ is convex,}$$

for each $z \in \mathbb{R}$, $x \in U$. Then

$$I[\cdot] \text{ is weakly lower semi-continuous on } W^{1,q}(U).$$

We can now establish that $I[\cdot]$ has a minimizer among the functions in $\mathcal{A}$.

Theorem D.3 (Existence of minimizer)**.** *Assume that L satisfies the coercivity inequality* (D.6) *and is convex in the variable p. Suppose also the admissible set $\mathcal{A}$ is nonempty.*

Then there exists at least one function $u \in \mathcal{A}$ solving

$$I[\cdot] = \min_{w \in \mathcal{A}} I[w].$$

In order to attain uniqueness of the minimizer u, we need to make some additional assumptions. Suppose that

$$L = L(p, x) \text{ does not depend on } z, \tag{D.8}$$

and

$$\begin{cases} \text{there exists } \theta > 0 \text{ such that} \\ \sum_{i,j=1}^{n} L_{p_i p_j}(p, x)\xi_i \xi_j \geq \theta \left|\xi\right|^2 \quad (p, \xi \in \mathbb{R}^n; x \in U). \end{cases} \tag{D.9}$$

Condition (D.9) says the mapping $p \mapsto L(p, x)$ is uniformly convex for each x.

Theorem D.4 (Uniqueness of minimizer)**.** *Suppose* (D.8), (D.9) *hold. Then a minimizer $u \in \mathcal{A}$ of $I[\cdot]$ is unique.*

D.3. Weak solutions of Euler-Lagrange equation

In order to be more precise about when a minimizer $u \in \mathcal{A}$ of $I[\cdot]$ solves the Euler-Lagrange equation, we need to place some growth conditions on L and its derivatives. Suppose that

$$\left|L(p, z, x)\right| \leq C\left(|p|^q + |z|^q + 1\right), \tag{D.10}$$

and also

$$\begin{cases} \left|D_p L(p, z, x)\right| \leq C\left(|p|^{q-1} + |z|^{q-1} + 1\right) \\ \left|D_z L(p, z, x)\right| \leq C\left(|p|^{q-1} + |z|^{q-1} + 1\right) \end{cases} \tag{D.11}$$

for some constant C and all $p \in \mathbb{R}^n$, $z \in \mathbb{R}$, $x \in U$. These conditions ensure that the integral in the following definition is well-defined (see [8] p. 451 for details).

Definition D.5 (Weak solution of Euler-Lagrange equation)**.** We say $u \in \mathcal{A}$ is a weak solution of the boundary-value problem for Euler-Lagrange equation

$$\begin{cases} -\sum_{i=1}^{n} \left(L_{p_i}(Du, u, x)\right)_{x_i} + L_z\left(Du, u, x\right) = 0 & \text{in } U \\ u = g & \text{on } \partial U. \end{cases} \tag{D.12}$$

provided

$$\int_U \sum_{i=1}^{n} L_{p_i}(Du, u, x) v_{x_i} + L_z(Du, u, x) v \, \mathrm{d}x = 0$$

for all $v \in W_0^{1,q}(U)$.

Theorem D.6 (Solution of Euler-Lagrange equation)**.** *Assume L verifies the growth conditions* (D.10),(D.11)*, and $u \in \mathcal{A}$ satisfies*

$$I[u] = \min_{w \in \mathcal{A}} I[w].$$

Then u is a weak solution of (D.12)*.*

D.4. Constraints: Pressure as a Lagrange multiplier

Suppose $U \subset \mathbb{R}^3$ is open, bounded, simply connected, and set

$$I[\boldsymbol{w}] := \int_U \frac{1}{2} |D\boldsymbol{w}|^2 - \boldsymbol{f} \cdot \boldsymbol{w} \, \mathrm{d}x,$$

for $\boldsymbol{w}$ belonging to

$$\mathcal{A} := \left\{ \boldsymbol{w} \in H_0^1(U; \mathbb{R}^3) \mid \operatorname{div}(\boldsymbol{w}) = 0 \text{ in } U \right\},$$

where $\boldsymbol{f} \in L^2(U; \mathbb{R}^3)$ is given. The existence of a unique minimizer $\boldsymbol{u} \in \mathcal{A}$ is given by the Lax-Milgram Theorem 2.2.

The following theorem (and proof) is from [8] p.472.

Theorem D.7 (Pressure as a Lagrange multiplier)**.** *Let the function $\boldsymbol{u}$ belong to the 'admissible set' $\mathcal{A}$ and let $\boldsymbol{f} \in L^2(U; \mathbb{R}^3)$ be given. Then there exists a scalar function $p \in L^2_{loc}(U)$ such that*

$$\int_U D\boldsymbol{u} : D\boldsymbol{v} \, \mathrm{d}x = \int_U p \operatorname{div}(\boldsymbol{v}) + \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x \tag{D.13}$$

for all $\boldsymbol{v} \in H^1(U; \mathbb{R}^3)$ with compact support within U.

Remark. We interpret (D.13) as saying that $(\boldsymbol{u}, p)$ is a weak solution to the Stokes problem (2.9). The function p is the pressure and arises as a Lagrange multiplier corresponding to the incompressibility condition $\operatorname{div}(\boldsymbol{u}) = 0$.

Proof. 1. Assume first $\boldsymbol{v} \in \mathcal{A}$. Then for each $\tau \in \mathbb{R}$, $\boldsymbol{u} + \tau\boldsymbol{v} \in \mathcal{A}$. Thus

$$i(\tau) := I[u + \tau v] = \int_U \frac{1}{2} |D\boldsymbol{u} + \tau D\boldsymbol{v}|^2 - \boldsymbol{f} \cdot (\boldsymbol{u} + \tau\boldsymbol{v}) \, \mathrm{d}x.$$

Differentiating with respect to τ we get

$$\begin{aligned} i'(\tau) &= \int_U (D\boldsymbol{u} + \tau D\boldsymbol{v}) : D\boldsymbol{v} - \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x \\ \Rightarrow 0 = i'(0) &= \int_U D\boldsymbol{u} : D\boldsymbol{v} - \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x, \end{aligned} \tag{D.14}$$

because $I[u] = \min_{w \in \mathcal{A}} I[w] \Rightarrow i'(0) = 0$.

2. Fix now $V \subset\subset U$, V smooth and simply connected, and select $w \in H_0^1(V; \mathbb{R}^3)$ with $\operatorname{div}(w) = 0$. Choose $0 < \epsilon < \operatorname{dist}(V, \partial U)$ and set $\boldsymbol{v} = \boldsymbol{v}^\epsilon := \eta_\epsilon * w$ in (D.14), η_ϵ denoting the usual mollifier and w defined to be zero in $U - V$. Then

$$0 = \int_U D\boldsymbol{u} : D\boldsymbol{v}^\epsilon - \boldsymbol{f} \cdot \boldsymbol{v}^\epsilon \, \mathrm{d}x = \int_U D\boldsymbol{u}^\epsilon : Dw - \boldsymbol{f}^\epsilon \cdot w \, \mathrm{d}x \tag{D.15}$$

for

$$\boldsymbol{u}^\epsilon := \eta_\epsilon * \boldsymbol{u}, \quad \boldsymbol{f}^\epsilon := \eta_\epsilon * \boldsymbol{f}. \tag{D.16}$$

As $\boldsymbol{u}^\epsilon$ is smooth, (D.15) implies

$$\int_V (-\Delta \boldsymbol{u}^\epsilon - \boldsymbol{f}^\epsilon) \cdot w \, \mathrm{d}x = 0 \tag{D.17}$$

for each $w \in H_0^1(V; \mathbb{R}^3)$ with $\operatorname{div}(\boldsymbol{w}) = 0$.

3. Fix any smooth vector field $\xi \in C_c^\infty(V; \mathbb{R}^3)$ and put $w = \operatorname{curl}(\xi)$ in (D.17). This is legitimate since $\operatorname{div}(\boldsymbol{w}) = \operatorname{div}(\operatorname{curl}(\xi)) = 0$. Then, temporarily writing $h := \Delta \boldsymbol{u}^\epsilon + \boldsymbol{f}^\epsilon$, we find

$$0 = \int_V h \cdot \operatorname{curl}(\xi) \, \mathrm{d}x = \int_V h^1(\xi^3_{x_2} - xi^2_{x_3}) + h^2(\xi^1_{x_1} - xi^3_{x_1}) + h^3(\xi^2_{x_1} - xi^1_{x_2}) \, \mathrm{d}x.$$

As $\xi^1, \xi^2, \xi^3 \in C_c^\infty(V)$ are arbitrary, we deduce $\operatorname{curl}(h) = 0$ in V. Since V is simply connected, there consequently exists a smooth function p^ϵ in V such that

$$Dp^\epsilon = h = \Delta \boldsymbol{u}^\epsilon + \boldsymbol{f}^\epsilon \quad \text{in } V. \tag{D.18}$$

4. If necessary we can add a constant to p^ϵ to ensure $\int_V p^\epsilon \, \mathrm{d}x = 0$.

In view of this normalization, there exists a smooth vector field $\boldsymbol{v}^\epsilon : V \to \mathbb{R}^3$ solving

$$\begin{cases} \operatorname{div}(\boldsymbol{v}^\epsilon) = p^\epsilon & \text{in } V \\ \boldsymbol{v}^\epsilon = 0 & \text{on } \partial V. \end{cases} \tag{D.19}$$

In addition we have the estimate

$$\|\boldsymbol{v}^\epsilon\|_{H^1(V;R^3)} \leq C \|p^\epsilon\|_{L^2(V)}, \tag{D.20}$$

the constant C depending only on V[1].

Now compute

$$\begin{aligned} \int_V (p^\epsilon)^2 \, \mathrm{d}x &= \int_V \operatorname{div}(\boldsymbol{v}^\epsilon) \, \mathrm{d}x \quad \text{by (D.19)} \\ &= -\int_V Dp^\epsilon \cdot \boldsymbol{v}^\epsilon \, \mathrm{d}x \\ &= \int_V (-\Delta \boldsymbol{u}^\epsilon - \boldsymbol{f}^\epsilon) \cdot \boldsymbol{v}^\epsilon \, \mathrm{d}x \quad \text{by (D.18)} \\ &= \int_V D\boldsymbol{u}^\epsilon : D\boldsymbol{v}^\epsilon - \boldsymbol{f}^\epsilon \cdot \boldsymbol{v}^\epsilon \, \mathrm{d}x \\ &\leq \|\boldsymbol{v}^\epsilon\|_{H^1(V;\mathbb{R}^3)} \left(\|\boldsymbol{u}^\epsilon\|_{H^1(V)} + \|\boldsymbol{f}^\epsilon\|_{L^2(V)} \right) \\ &\leq C \|p^\epsilon\|_{L^2(U)} \left(\|\boldsymbol{u}\|_{H_0^1(U)} + \|\boldsymbol{f}\|_{L^2(U)} \right) \quad \text{by (D.20).} \end{aligned}$$

Thus

$$\|p^\epsilon\|_{L^2(V)} \leq C \left(\|\boldsymbol{u}\|_{H^1(V)} + \|\boldsymbol{f}\|_{L^2(V)} \right). \tag{D.21}$$

5. In view of estimate (D.21) there exists a subsequence $\epsilon_j \to 0$ so that

$$p^{\epsilon_j} \rightharpoonup p \quad \text{weakly in } L^2(V) \tag{D.22}$$

[1] The proof of the existence of the vector field $\boldsymbol{v}^\epsilon$ is omitted in [8], and he refers to [7].

for some $p \in L^2(V)$. Now (D.18) implies

$$\int_V D\boldsymbol{u}^\epsilon : D\boldsymbol{v}\,\mathrm{d}x = \int_V p^\epsilon \operatorname{div}(\boldsymbol{v}) + \boldsymbol{f}^\epsilon \cdot \boldsymbol{v}\,\mathrm{d}x$$

for all $\boldsymbol{v} \in H_0^1(V;\mathbb{R}^3)$. Sending $\epsilon = \epsilon_j \to 0$ we find

$$\int_V D\boldsymbol{u} : D\boldsymbol{v}\,\mathrm{d}x = \int_V p \operatorname{div}(\boldsymbol{v}) + \boldsymbol{f} \cdot \boldsymbol{v}\,\mathrm{d}x \tag{D.23}$$

as well.

6. Finally choose a sequence of sets $V_k \subset\subset U$ $(k = 1, ...)$ as above, with $V_1 \subset V_2 \subset \cdots$ and $U = \cup_{k=1}^\infty V_k$. Utilizing steps 2-5 we find $p_k \in L^2(V_k)$ $(k = 1, ...)$ so that

$$\int_{V_k} D\boldsymbol{u} : D\boldsymbol{v}\,\mathrm{d}x = \int_{V_k} p_k \operatorname{div}(\boldsymbol{v}) + \boldsymbol{f} \cdot \boldsymbol{v}\,\mathrm{d}x \tag{D.24}$$

for each $\boldsymbol{v} \in H_0^1(V_k;\mathbb{R}^3)$. Adding constant as necessary to each p_k, we deduce from (D.24) that if $1 \le l \le k$, then $p_k = p_l$ on V_l. We finally define $p = p_k$ on V_k $(k = 1, ...)$.

□

E. Measure theory

The material in this section is primarily from [11] with a few additions from [8].

We will here give a review of measure theory, with the goal of defining a Lebesgue measurable function, an integrable function, and some important convergence theorems for integrals.

E.1. σ-algebras and measures

Given an arbitrary set $\mathcal{X}$, which we shall call a 'universe', an arbitrary collection of subsets of $\mathcal{X}$ is called a **paving**. $\mathcal{X}$ could for instance be $\{1,2,3\}$, or the real line $\mathbb{R}$. An example of a paving is the power set $\mathbb{P}(\mathcal{X})$, the collection of all subsets of $\mathcal{X}$. We then have

$$\mathbb{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$

Since $\mathbb{R}$ is an infinite (and also uncountable) set, its power set is of course unimaginably large. A more useful paving, with the properties that we need for measurability, is the σ-algebra.

Definition E.1 (σ-algebra)**.** A paving $\mathbb{E}$ on a set $\mathcal{X}$ is called a σ-algebra if

1. $\mathcal{X} \in \mathbb{E}$,
2. $A \in \mathbb{E} \Rightarrow A^c \in \mathbb{E}$,
3. $A_1, A_2, ... \in \mathbb{E} \Rightarrow \bigcup_{n=1}^{\infty} \in \mathbb{E}$.

The above definition is highly 'compressed'; σ-algebras contain almost every imaginable collection of subsets of $\mathcal{X}$, and they are stable under almost any set operation, such as countable union, intersection, etc. A **measurable space** is a pair $(\mathcal{X}, \mathbb{E})$, consisting of a set $\mathcal{X}$ and a σ-algebra $\mathbb{E}$ on $\mathcal{X}$. We say that a subset $A \subset \mathcal{X}$ is $\mathbb{E}$-measurable if $A \in \mathbb{E}$.

Definition E.2 (Generated σ-algebra)**.** The σ-algebra $\sigma(\mathbb{D})$ generated by a paving $\mathbb{D}$ on a set $\mathcal{X}$, is defined as the intersection of all σ-algebras containing $\mathbb{D}$. Symbolically,

$$\sigma(\mathbb{D}) = \bigcap_{\substack{\mathbb{E} \text{ is a } \sigma\text{-algebra on } \mathcal{X}, \\ \mathbb{D} \subset \mathbb{E}}} \mathbb{E}.$$

The most important σ-algebra in Euclidean space is the Borel σ-algebra.

Definition E.3 (Borel σ-algebra)**.** The Borel algebra $\mathbb{B}$ on $\mathbb{R}$ is the σ-algebra generated by the open sets. Symbolically we write $\mathbb{B} = \sigma(\mathbb{O})$.

Theorem E.4. *The system of open intervals $\mathbb{I}$ is a generator for $\mathbb{B}$. Symbolically we have that $\mathbb{B} = \sigma(\mathbb{I})$.*

Proof. See [11] p.17. □

A similar notion holds in R^k.

Definition E.5. The Borel algebra $\mathbb{B}_k$ on $\mathbb{R}^k$ is the σ-algebra generated by the open sets. Symbolically we write $\mathbb{B}_k = \sigma(\mathbb{O}_k)$.

Theorem E.6. *The system of open boxes $\mathbb{I}^k$ is a generator for $\mathbb{B}_k$. Symbolically we have that $\mathbb{B}_k = \sigma(\mathbb{I}^k)$.*

Proof. See [11] p.17. □

Now we will equip our measurable space $(\mathcal{X}, \mathbb{E})$ with a measure. But first some more definitions. Let $\mathbb{E}$ be a paving on $\mathcal{X}$. A **non-negative set function** μ is a map $\mu : \mathbb{E} \to [0, \infty]$.

Definition E.7 (σ-additivity)**.** Let $\mathbb{E}$ be a σ-algebra on a set $\mathcal{X}$. A non-negative set function μ defined on $\mathbb{E}$ is σ**-additive** if it for any sequence $A_1, A_2, ...$ of $\mathbb{E}$-sets holds that

$$\mu\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mu(A_n) \quad \text{if} \quad A_i \cap A_j = \emptyset \; for i \neq j.$$

Definition E.8 (Measure)**.** Let $(\mathcal{X}, \mathbb{E})$ be a measurable space. A non-negative set function μ on $\mathbb{E}$ is a **measure** if it is σ-additive and satisfies that $\mu(\emptyset) = 0$.

When we equip a measurable space $(\mathcal{X}, \mathbb{E})$ with a measure μ, we obtain the triple $(\mathcal{X}, \mathbb{E}, \mu)$ which is called a measure space.

E.2. The Lebesgue measure

Definition E.9 (Lebesgue measure)**.** The **Lebesgue measure** m on $(\mathbb{R}, \mathbb{B})$ is the measure satisfying for $a, b \in \mathbb{R}$

$$m((a, b)) = b - a \quad \text{for } a < b.$$

This definition generalizes to higher dimensions as follows.

Definition E.10. The **Lebesgue measure** m_k on $(\mathbb{R}^k, \mathbb{B}_k)$ is the measure satisfying

$$m_k((a_1, b_1) \times (a_2, b_2) \times \cdots \times (a_k, b_k)) = \prod_{i=1}^{k} (b_i - a_i), \tag{E.1}$$

for $a_1, a_2, ..., a_k, b_1, b_2, ..., b_k \in \mathbb{R}$, $a_1 < b_1, a_2 < b_2, ..., a_k < b_k$.

The following results are useful for doing calculations with measures. See [11] for proofs.

Lemma E.11 (Monotonicity)**.** *If $(\mathcal{X}, \mathbb{E}, \mu)$ is a measure space, and if $A, B \in \mathbb{E}$ satisfy $A \subset B$, then*

$$\mu(A) \leq \mu(B).$$

Lemma E.12 (Upward continuity)**.** *If $(\mathcal{X}, \mathbb{E}, \mu)$ is a measure space. For any* ***increasing*** *sequence $A_1 \subset A_2 \subset ...$ of $\mathbb{E}$-sets, it holds that*

$$\mu(A_n) \to \mu\left(\bigcup_{i=1}^{\infty} A_i\right) \quad \textit{for } n \to \infty.$$

Lemma E.13 (Downward continuity)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. Consider a* ***decreasing*** *sequence $A_1 \supset A_2 \supset ...$ of $\mathbb{E}$-sets. If $\mu(A_n) < \infty$ for at least one $n \in N$ it holds that*

$$\mu(A_n) \to \mu\left(\bigcap_{i=1}^{\infty} A_i\right) \quad \text{for } n \to \infty.$$

Lemma E.14 (Boole's inequality)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. For any sequence $A_1, A_2, ...$ of $\mathbb{E}$-sets it holds that*

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mu(A_n).$$

Lemma E.15. *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. If $A, B \in \mathbb{E}$ satisfy $A \subset B$, and if $\mu(A) < \infty$, then*

$$\mu(B \setminus A) = \mu(B) - \mu(A).$$

As an example of how these results can be applied, we consider a hyperplane in R^k of the form

$$H = \left\{ x \in \mathbb{R}^k \mid \langle x, \boldsymbol{v} \rangle = c \right\},$$

where $v \neq 0$ is a k-vector and c is a real number, and $\langle \cdot, \cdot \rangle$ denotes the canonical inner product in $\mathbb{R}^k$. If $c = 0$ the hyperplane is in fact a subspace in $\mathbb{R}^k$ of dimension $k - 1$; if $c \neq 0$ it is a translated subspace. As $x \mapsto \langle x, v \rangle$ is continuous, it follows that H is closed, and thus $\mathbb{B}_k$-measurable.

If the vector v is one of the canonical basis vectors, we say that the hyperplane is **parallel to the coordinate system**. It will thus be of the form

$$H = \left\{ (x_1, ..., x_k) \in \mathbb{R}^k \mid x_m = c \right\}, \tag{E.2}$$

for some $m = 1, 2, ..., k$. Note that each face of the bounded boxes occurring in (E.1) is a subset of a hyperplane parallel to the coordinate system.

Lemma E.16. *Let H be a hyperplane in R^k parallel to the coordinate system. Then H has Lebesgue measure zero, that is*

$$m_k(H) = 0.$$

Proof. (from [11] p.44)
Consider the hyperplane in (E.2). For $N, n \in \mathbb{N}$ we let

$$H_{N,n} = \left\{ (x_1, ..., x_k) \in \mathbb{R}^k \mid |x_m - c| < \frac{1}{n}, |x_i| < N \quad \forall i \neq m \right\}.$$

This is an open box with $k - 1$ sides of length $2N$ and one side of length $\frac{2}{n}$. For fixed N, the boxes $H_{N,n}$ form a decreasing sequence with intersection

$$H_N = \left\{ (x_1, ..., x_k) \in \mathbb{R}^k \mid |x_m - c| = 0, |x_i| < N \quad \forall i \neq m \right\}.$$

As

$$m_k(H_{N,n}) = \frac{(2N)^{k-1}2}{n} \to 0 \quad \text{for } n \to \infty,$$

we conclude from lemma E.13 that $m_k(H_N) = 0$. But $H_1, H_2, ...$ form an increasing sequence of sets, with union equal to H. So lemma E.12 ensures that $m_k(H) = 0$. □

Lemma E.16 can be extended to the following result.

Theorem E.17. *Let $U \subset \mathbb{R}^k$ be a bounded, simply connected subset of $\mathbb{R}^k$ with C^1 boundary ∂U. Then ∂U has k-dimensional Lebesgue measure zero, i.e.*

$$m_k(\partial U) = 0.$$

E.3. Uniqueness of measures

First we introduce some new concepts.

Definition E.18 (Nullset)**.** Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. A subset $A \subset \mathcal{X}$ is a μ-**nullset** if there exists an $\mathbb{E}$-set B such that

$$A \subset B, \quad \mu(B) = 0.$$

If a nullset itself is measurable, it must have measure 0. But we explicitly allow nullsets to be non-measurable, to be able to handle pathologies like the Cantor set. The complement of a μ-nullset is a set that occurs μ-**almost surely** or μ-**almost everywhere**.

Definition E.19 (σ-finite)**.** The measure space $(\mathcal{X}, \mathbb{E}, \mu)$ is σ-finite if there is an increasing sequence $K_1 \subset K_2 \subset ...$ of $\mathbb{E}$-sets, such that

1. $\mu(K_n) < \infty \quad \forall n \in \mathbb{N}$,
2. $\bigcup_{n=1}^{\infty} K_n = \mathcal{X}$.

We easily see that $(\mathbb{R}, \mathbb{B}, m)$ is σ-finite by setting $K_n = (-n, n)$ for $n = 1, 2, ...$ Similarly, $(\mathbb{R}^k, \mathbb{B}_k, m_k)$ is σ-finite for every k, which we can see by setting $K_n = (-n, n) \times \cdots \times (-n, n)$ for $n = 1, 2, ...$ According to [11] σ-finite measures is a 'large' class of measures, and the uniqueness of the Lebesgue measure follows from the fact that it belongs to this class and the following theorem.

Theorem E.20 (Uniqueness theorem for σ-finite measures)**.** *Let μ and ν be two arbitrary measures on a measurable space $(\mathcal{X}, \mathbb{E})$. Assume that $\mathbb{E} = \sigma(\mathbb{D})$ and that*

$$\mu(D) = \nu(D) \quad \forall D \in \mathbb{D}.$$

If $\mathbb{D}$ is stable under intersections[1] *and if there is an increasing sequence $D_1 \subset D_2 \subset ...$ of $\mathbb{D}$-sets with $\mu(D_n) < \infty$ for all n, such that $\bigcup_{n=1}^{\infty} D_n = \mathcal{X}$, it holds that $\mu = \nu$.*

Corollary E.21 (Uniqueness of the Lebesgue measure)**.** *There is at most one Lebesgue measure on $(\mathbb{R}^k, \mathbb{B}_k)$.*

Proof. ([11] p.64.) The claim is that if m and m' both satisfy (E.1) they must be equal. But the paving $\mathcal{I}_k$ of open boxes is a generator for $\mathbb{B}_k$, stable under intersections. The condition (E.1) ensures that m and m' agree on $\mathcal{I}_k$-sets, and that each of these $\mathcal{I}_k$-sets has finite measure. To be able to use theorem E.20 what remains is to construct an increasing sequence $I_1 \subset I_2 \subset ...$ of open boxes so that $\bigcup_{n=1}^{\infty} I_n = \mathbb{R}^k$. A sensible choice is

$$I_n = (-n, n) \times (-n, n) \times \cdots \times (-n, n).$$

□

[1] The intersection of a countable number of sets in $\mathbb{D}$ also belongs to $\mathbb{D}$.

E.4. Measurable functions

From [11] p.69:

"Let $(\mathcal{X}, \mathbb{E})$ and $(\mathcal{Y}, \mathbb{K})$ be two measurable spaces, and let $f : \mathcal{X} \to \mathcal{Y}$ be a map. If we have a measure μ on $(\mathcal{X}, \mathbb{E})$, we can discuss the 'size' of certain subsets of $\mathcal{X}$. But in many cases the natural questions are related to the size of subsets of $\mathcal{Y}$. A way to assign a size to a subset $B \subset \mathcal{Y}$ is to pull it back to $\mathcal{X}$ via f, giving us the pre-image $f^{-1}(B)$. The μ-size of the pre-image can be seen as an expression of the size of B."

Definition E.22 (Measurable map)**.** Let $(\mathcal{X}, \mathbb{E})$ and $(\mathcal{Y}, \mathbb{K})$ be two measurable spaces, and let $f : \mathcal{X} \to \mathcal{Y}$ be a map. We say that f is **measurable** if

$$f^{-1}(B) \in \mathbb{E} \quad \forall B \in \mathbb{K}.$$

Remark. If we need to be explicit about the σ-algebras used by the map f, we say that f is $\mathbb{E} - \mathbb{K}$-measurable.

The following lemma gives a useful connection between the measurability of a map, and the generator of the σ-algebra of the map's image.

Lemma E.23. *Let $(\mathcal{X}, \mathbb{E})$ and $(\mathcal{Y}, \mathbb{K})$ be two measurable spaces, and let $f : \mathcal{X} \to \mathcal{Y}$ be a map. Let $\mathbb{D}$ be a paving on $\mathcal{Y}$ and assume that $\mathbb{K} = \sigma(\mathbb{D})$. If*

$$f^{-1}(D) \in \mathbb{E} \quad \forall D \in \mathbb{D},$$

then f is $E - K$-measurable.

In the case of Euclidean spaces measurability follows from continuity.

Lemma E.24. *Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be a map between two Euclidean spaces. If f is continuous, it is $\mathbb{B}_n - \mathbb{B}_k$-measurable.*

Proof. ([11] p.74.)
The Borel algebra $\mathbb{B}_k$ is generated by the open sets $\mathbb{O}_k$ of $\mathbb{R}^k$. As f is continuous, it holds that

$$f^{-1}(G) \in \mathbb{O}_n \quad \forall G \in \mathbb{O}_k.$$

Hence we see that

$$f^{-1}(G) \in \mathbb{B}_n \quad \forall G \in \mathbb{O}_k,$$

and so the desired result follows from lemma E.23. □

Definition E.25. Let $(\mathcal{X}, \mathbb{E})$ be a measurable space. By $\mathcal{M} = \mathcal{M}(\mathcal{X}, \mathbb{E})$ we denote the space of **measurable real-valued functions** on $\mathcal{X}$, that is the collection of $\mathbb{E} - \mathbb{B}$-measurable functions from $\mathcal{X}$ to $\mathbb{R}$.

We denote by $\mathbb{B}^+$ the space of non-negative measurable functions.

Definition E.26 (Equality almost everywhere)**.** Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. Two functions $f, g \in \mathcal{M}$ are said to be μ-almost surely equal or equal μ-almost everywhere, abbreviated

$$f = g \quad \mu\text{-a.s.} \quad \text{or} \quad f = g \quad \mu\text{-a.e.},$$

if the set of disagreement of the two functions is a μ-nullset. That is, if

$$\mu(\{x \mid f(x) \neq g(x)\}) = 0.$$

E.5. Integration of non-negative functions

A **simple function** $s : \mathcal{X} \to \mathbb{R}$ is an $\mathbb{E} - \mathbb{B}$-measurable function that can be written as a linear combination of indicator functions

$$s(x) = \sum_{i=1}^{n} c_i 1_{A_i}(x), \quad x \in \mathcal{X},$$

where each $c_i \in \mathbb{R}$ and $A_i \in \mathbb{B}$. A non-negative simple function is a simple function that satisfies $s(x) \geq 0$ for all $x \in \mathcal{X}$. Let $\mathcal{S}^+$ denote the space of such functions.

Definition E.27 (Pre-integral)**.** Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space and let $s \in \mathcal{S}^+$ be a non-negative measurable simple function. If

$$s(x) = \sum_{i=1}^{n} c_i 1_{A_i}(x), \quad x \in \mathcal{X},$$

is the canonical representation, we define the pre-integral of s as

$$I(s) = \sum_{i=1}^{n} c_i \mu(A_i).$$

Definition E.28 (Lebesgue integral of non-negative function)**.** Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. For a function $f \in \mathcal{M}^+$ we define the **Lebesgue integral** of f with respect to μ as

$$\int f \, \mathrm{d}\mu = \sup \left\{ I(s) \mid s \in \mathcal{S}^+, s \leq f \right\}.$$

Lemma E.29 (Fatou's lemma)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. For any sequence $f_1, f_2, \ldots$ of $\mathcal{M}^+$-functions it holds that*

$$\liminf_{n \to \infty} \int f_n \, \mathrm{d}\mu \geq \int \liminf_{n \to \infty} f_n \, \mathrm{d}\mu.$$

E.6. Integration of real-valued functions

Definition E.30 (Lebesgue integral of real-valued function)**.** Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. A function $f \in \mathcal{M}(\mathcal{X}, \mathbb{E})$ is **integrable** if

$$\int f^+ \, \mathrm{d}\mu < \infty, \quad \int f^- \, \mathrm{d}\mu < \infty.$$

If f is integrable, we define the **Lebesgue integral** of f as

$$\int f \, \mathrm{d}\mu = \int f^+ \, \mathrm{d}\mu - \int f^- \, \mathrm{d}\mu.$$

The class of all integrable real-valued functions is denoted by $\mathcal{L} = \mathcal{L}(\mathcal{X}, \mathbb{E}, \mu)$, and we write $f \in \mathcal{L}$ if f is integrable.

We note that [8] distinguishes between summable and integrable functions, where summable functions have finite integrals, and integrable functions may integrate to $\pm\infty$.

Lemma E.31. *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space, and let $f \in \mathcal{M}$. It holds that f is integrable if and only if*

$$\int |f| \, \mathrm{d}\mu < \infty.$$

Definition E.32. Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. For a function $f \in \mathcal{M}$ and a subset $A \in \mathbb{E}$ we define the Lebesgue integral of f over the domain A as

$$\int_A f \, \mathrm{d}\mu = \int 1_A f \, \mathrm{d}\mu.$$

Lemma E.33 (Vanishing almost surely)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space and let $f \in \mathcal{M}$ be a measurable function. It holds that*

$$\int |f| \, \mathrm{d}\mu = 0 \Leftrightarrow f = 0 \quad \mu\text{-a.s.}$$

Theorem E.34. *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space. Suppose $f, g \in \mathcal{M}$ are μ-almost surely equal. If f is integrable, then g is also integrable, and it holds that*

$$\int f \, \mathrm{d}\mu = \int g \, \mathrm{d}\mu.$$

The following convergence theorems for integrals are the center-piece of Lebesgue's integration theory.

Theorem E.35 (Monotone Convergence Theorem)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space and let*

$$f_1 \leq f_2 \leq \cdots \leq f_k \leq f_{k+1} \leq \cdots$$

be an increasing sequence of $\mathcal{M}$-functions, $f_i : \mathcal{X} \to \mathbb{R}$, $i = 1, \ldots$. Then

$$\int \lim_{k\to\infty} \mathrm{d}\mu = \lim_{k\to\infty} \int f_k \, \mathrm{d}\mu.$$

Theorem E.36 (Dominated Convergence Theorem)**.** *Let $(\mathcal{X}, \mathbb{E}, \mu)$ be a measure space and let $f_1, f_2, \ldots : \mathcal{X} \to \mathbb{R}$ be a sequence of $\mathcal{M}$-functions. Assume that there is a measurable limit function $f \in \mathcal{M}$, $f : \mathcal{X} \to \mathbb{R}$ such that*

$$f_n(x) \to f(x) \quad \text{for } n \to \infty \ \mu - \text{almost surely.}$$

If there is an integrable upper bound g for the sequence $|f_1|, |f_2|, \ldots$, then f as well as $f_1, f_2, \ldots$ are integrable, and it holds that

$$\int f_n \, \mathrm{d}\mu \to \int f \, \mathrm{d}\mu \quad \text{for } n \to \infty.$$

E.7. Differentiation

The following theorem is from [8] Appendix E. See A for the definition of $⨍$.

Theorem E.37 (Lebesgue's differentiation theorem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally summable (see A).*

(i) Then for a.e. point $x_0 \in \mathbb{R}^n$,

$$⨍_{B(x_0,r)} f \, \mathrm{d}x \to f(x_0) \quad \text{as } r \to 0.$$

(ii) In fact, for a.e. point $x_0 \in \mathbb{R}^n$,

$$⨍_{B(x_0,r)} |f(x) - f(x_0)| \, \mathrm{d}x \to f(x_0) \quad \text{as } r \to 0. \tag{E.3}$$

A point x_0 at which (E.3) holds is called a Lebesgue point of f.

Remark. More generally, if $f \in L^p_{\mathrm{loc}}(\mathbb{R}^n)$ for some $1 \leq p < \infty$, then for a.e. point $x_0 \in \mathbb{R}^n$ we have

$$⨍_{B(x_0,r)} |f(x) - f(x_0)|^p \, \mathrm{d}x \to f(x_0) \quad \text{as } r \to 0.$$

F. Functional analysis

The material in this section is from [12].

Definition F.1 (Bounded linear functional (operator))**.** A linear operator $T : V \to W$ between normed vector spaces $(V, \|\cdot\|_V)$ and $(W, \|\cdot\|_W)$ is called a *bounded* linear operator if it maps the closed unit ball $C(V)$ in V into a bounded set (a ball) in W.

Lemma F.2. *Let $T : v \to W$ be a bounded linear operator between normed vector spaces $(V, \|\cdot\|_V)$ and $(W, \|\cdot\|_W)$. Then*

$$\|T(X)\|_W \leq \|T\| \, \|x\|_V \quad \forall x \in V.$$

Definition F.3 (Orthogonal sum)**.** Let M and N be closed linear subspaces of a Hilbert space H with $M \perp N$. Then we define the *orthogonal sum* $M \oplus N$ of M and N by

$$M \oplus N = \{z \in H \mid z = x + y, \, x \in M, \, y \in N\}.$$

Lemma F.4. *For the orthogonal sum $M \oplus N$ of two closed, orthogonal linear subspaces M and N of a Hilbert space H, it holds that*

(a) The representation $z = x + y$ in $M \oplus N$ is unique.

(b) $M \oplus N$ is a closed linear subspace of H.

Theorem F.5 (The Projection Theorem)**.** *If M is a closed linear subspace of the Hilbert Space H, then $H = M \oplus M^{\perp}$.*

Theorem F.6 (Riesz' Representation Theorem)**.** *Let $\phi : H \to \mathbb{C}$ be a continuous linear functional on a Hilbert space H equipped with an inner product $(\cdot, \cdot)$. Then there is a unique vector $z \in H$ such that $\phi(x) = (x, z)$ for all $x \in H$.*

G. Optimization

G.1. The Lagrange multiplier method

This section is based on [9] and [14].
The Lagrange multiplier method allows us to maximize a function $f(x_1, ..., x_n)$, $n \in \mathbb{N}$, subject to the constraint $g(x_1, ..., x_n) = c$, by constructing the auxiliary function

$$H(x_1, ..., x_n, \lambda) = f(x_1, ..., x_n) - \lambda \left[g(x_1, ..., x_n) - c \right]$$

and then finding values $x_1, ..., x_n, \lambda$ for which the partial derivatives of H are all zero. The scalar λ is called a Lagrange multiplier, and quantifies the objective function's sensitivity to a given constraint. The method can be generalized to multiple constraints by simply adding more Lagrange multipliers:

$$H(x_1, ..., x_n, \lambda_1, ..., \lambda_m) = f(x_1, ..., x_n) - \lambda_1 \left[g(x_1, ..., x_n) - c_1 \right] - \cdots - \lambda_m \left[g(x_1, ..., x_m) - c_m \right],$$

where $m \in \mathbb{N}$.

The method of Lagrange multipliers can also be applied to normed spaces. But in order to do so, we must first define an appropriate way to differentiate functionals (see [14]).

Definition G.1 (Fréchet differential)**.** Let T be a functional defined on an open domain D in a normed space X and having range in a normed space Y. If for fixed $x \in D$ and each $h \in X$ there exists $\delta T(x; h) \in Y$ which is linear and continuous with respect to h such that

$$\lim_{\|h\| \to 0} \frac{\|T(x + h) - T(x) - \delta T(x; h)\|}{\|h\|} = 0,$$

then T is said to be Fréchet differentiable at x and $\delta T(x; h)$ is said to be the Fréchet differential of T at x with increment h.

Now suppose that the we wish to optimize a functional f subject to n nonlinear constraints given in the implicit form:

$$g_i(x) = 0, \quad i = 1, ..., n. \tag{G.1}$$

These n equations define a region U in the space X within which the optimal vector x_0 is constrained to lie. We assume that f, g_i, $i = 1, ..., n$, are continuous and Fréchet differentiable on the normed space X.

Definition G.2 (Regular point)**.** A point x_0 satisfying the constraints (G.1) is said to be a regular point of these constraints if the n linear functionals $g_1'(x_0), g_2'(x_0), ..., g_n'(x_0)$ are linearly independent.

The following theorem gives the necessary conditions satisfied by the solution of the constrained extremum problem.

Theorem G.3. *If x_0 is an extremum of the functional f subject to the constraints $g_i(x) = 0$, $i = 1, 2, ..., n$; and if x_0 is a regular point of these constraints, then*

$$\delta f(x_0; h) = 0$$

for all h satisfying $\delta g_i(x_0; h) = 0$, $i = 1, 2, ..., n$.

From theorem G.3 it is easy to derive a finite-dimensional version of the Lagrange multiplier method by using the following lemma.

Lemma G.4. *Let $f_0, f_1, ..., f_n$ be linear functionals on a vector space X and suppose that $f_0(x) = 0$ for every $x \in X$ satisfying $f_i(x) = 0$ for $i = 1, 2, ..., n$. Then there are constants $\lambda_1, \lambda_2, ..., \lambda_n$ such that*

$$f_0 + \lambda_1 f_1 + \lambda_2 f_2 + \cdots + \lambda_n f_n = 0.$$

Proof. This can be proved using the Hahn-Banach theorem. See [14]. □

Theorem G.5. *If x_0 is an extremum of the functional f subject to the constraints*

$$g_i(x) = 0, \quad i = 1, 2, ..., n,$$

and x_0 is a regular point of these constraints, then there are n scalars, λ_i, $i = 1, 2, ..., n$, that render the functional

$$f(x) + \sum_{i=1}^{n} \lambda_i g_i(x)$$

stationary at x_0.

Proof. By Theorem G.3 the differential $\delta f(x_0; h)$ is zero whenever each of the differentials $\delta g_i(x_0; h)$ is zero. The result then follows immediately from Lemma G.4. □

G.2. The Method of Moving Asymptotes

The Method of Moving Asymptotes (MMA) algorithm is a constrained optimization algorithm developed by Krister Svanberg [17].

The optimization problem is assumed to be in the following form:
Let $x \in \mathbb{R}^N$ be an N-dimensional vector and let $f_0(x)$ be the objective function to be minimized subject to $M > 0$ constraints[1].

$$\begin{aligned} \text{minimize} \quad & f_0(x) + z + 0.05z^2 + \sum_{i=1}^{M} c_i y_i + 0.5 y_i^2 \\ \text{subject to} \quad & f_i(x) - a_i z - y_i \leq f_i^{MAX}, \quad i = 1, ..., M \\ & x_j^{MIN} \leq x_j \leq x_j^{MAX}, \quad j = 1, ..., N \\ & y_i \geq 0, \qquad y = 1, ..., M \\ & z \geq 0, \end{aligned} \tag{G.2}$$

[1] It is possible to do unconstrained optimization by setting $M = 1$ and then specifying a dummy constraint such as $\sum_j x_j \leq S$, where $S := 1 + \sum_j x_j^{MAX}$.

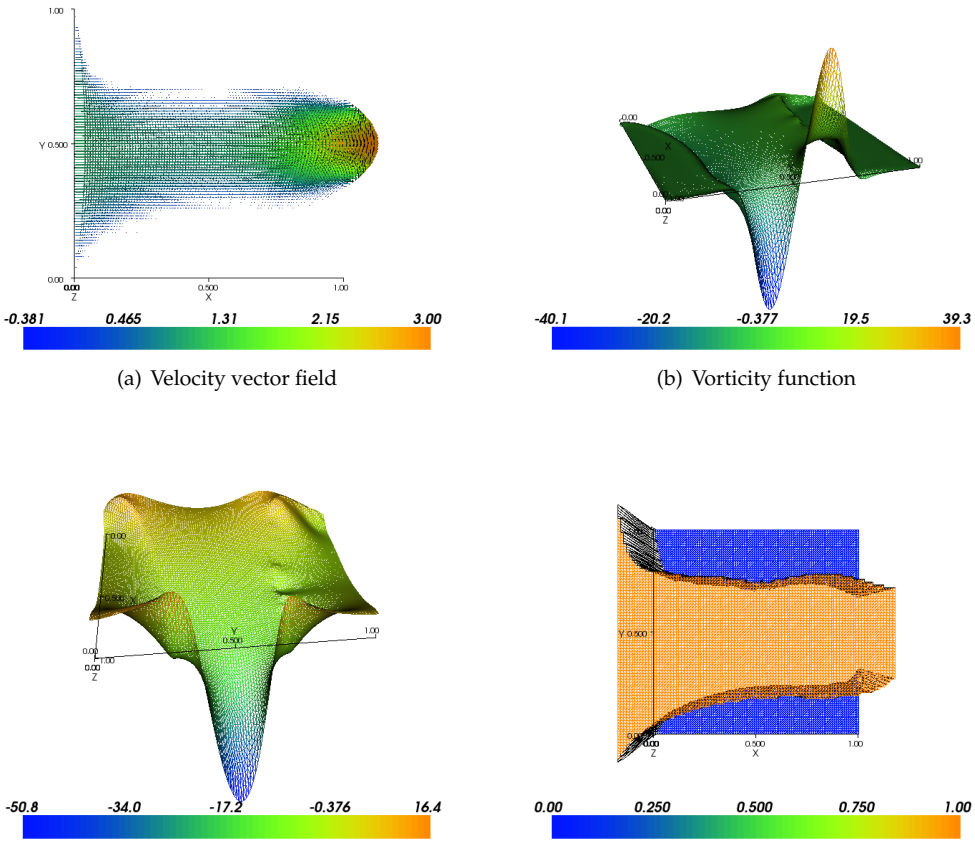
where M, N are strictly positive integers, $x_j^{MIN} \leq x_j^{MAX},\ j = 1, ..., N$, and $f_i^{MAX},\ i = 1, ..., M$ are constants. The additional variables a_i, c_i, y_i and z are included to give the algorithm the flexibility to handle many types of problems; they were not used for the purposes of this paper. The implementation of the MMA algorithm used by the author was written in Fortran77 (by Krister Svanberg), compiled in Linux and called from Python.

H. Plots

H.1. Topology optimization example 1: Optimal diffusion

Figure H.1 shows the solution of the optimal diffusion problem on a 100×100 grid using the QNE1h formulation.

Figure H.2 shows the solution of the optimal diffusion problem on a 100×100 grid using the QNE2 formulation.



(a) Velocity vector field

(b) Vorticity function

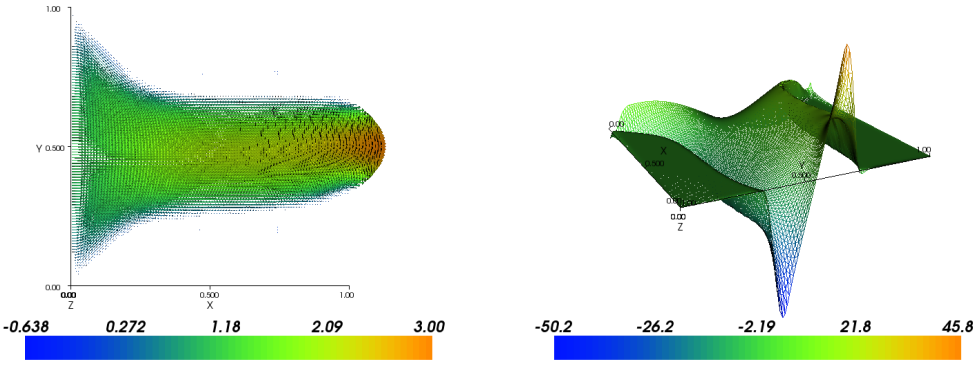
(c) Pressure function

(d) Material distribution

Figure H.1: Optimal diffusion problem with $1/h = 100$ using the QNE1h formulation.

H.2. Topology optimization example 2: Pipe bend

Figure H.3 shows the solution of the optimal diffusion problem on a 100×100 grid using the CY1 formulation.



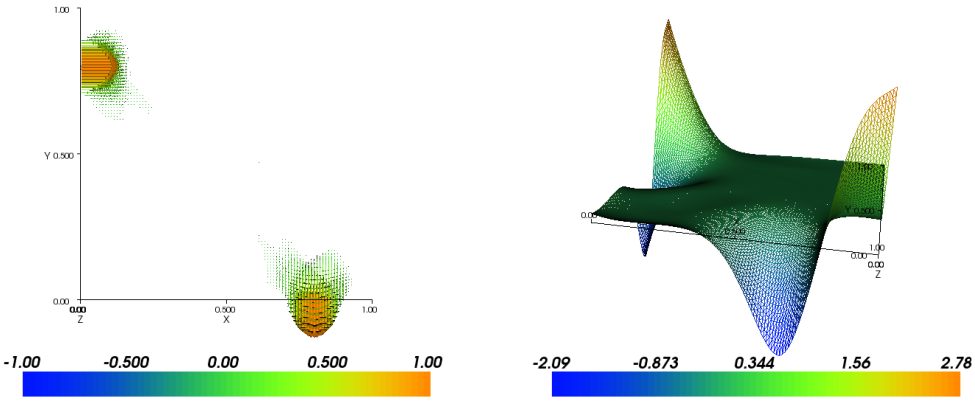
(a) Velocity vector field

(b) Vorticity function

(c) Pressure function

(d) Material distribution

Figure H.2: Optimal diffusion problem with $1/h = 100$ using the QNE2 formulation.



(a) Velocity vector field (b) Vorticity function

(c) Pressure function (d) Material distribution

Figure H.3: Optimal diffusion problem with $1/h = 100$ using the CY1 formulation.

I. Computer code

I.1. FEniCS

FEniCS is an open source software library for automated scientific computing, with a particular focus on automated solution of differential equations by finite element methods. The main interface of FEniCS is DOLFIN, a C++/Python library that wraps the functionality of other FEniCS components and external software, and handles the communication between these components. See `www.fenicsproject.org/about/`. The code written for this thesis can be found at:

`https://github.com/jenschristiansen/MMC-thesis-code.git`

I.2. Convergence analysis in FEniCS

The Python code setup for the 2D convergence analysis in Section 3.8 is described here (code for the 3D analysis can be found using the link above).

The 2D convergence analysis was done using three scripts:

- `CY1conv.py` is the main script where the desired methods and grid dimensions are chosen. The results are saved to disk when done.
- `CY1solver.py` contains the actual FEniCS solver and is imported by `CY1conv.py`. This script imports the `DOLFIN` library.
- `CY1plot.py` loads the results saved by `CY1conv.py` and writes output to screen / plots the results.

Listing I.1: CY1conv.py

```
## Chang & Yang numerical example 1 convergence analysis ##

# File: CY1conv.py

# Created on April 18, 2012
# Last revised on May 30, 2012
# Author: Jens V. Christiansen
#-----------------------------------------#

import CY1solver
import numpy as np
import matplotlib.pyplot as plt

from time import time

# xx1 = first order velocity
# xx2 = second order velocity
```

```
dims = [10,20,40,80,160]
methods = ['MF','CY1','CY2','QNE1','QNE2','QNE1h','QNE2h']

# output is 'results', a 3-dim array: dims x vars x methods, vars = output variables
t1 = time()
results = CY1solver.driver(dims,methods)
t2 = time()
print 'Time elapsed: %0.3f seconds.' % (t2-t1)

DictOut = {}
DictOut['dims']=dims
DictOut['methods']=methods
DictOut['results']=results
DictOut['model']=CY1solver.model

import pickle
output = open('CY1_methods.pkl', 'wb') # set output file name here

output.close()
```

Listing I.2: CY1solver.py

```

## Chang & Yang numerical example 1 convergence analysis ##

# File: CY1solver.py

# Created on April 18, 2012
# Last revised on May 30, 2012
# Author: Jens V. Christiansen
#-------------------------------------#

from dolfin import *
import numpy as np

cols = 34 # number of output variables

def driver(dims,methods):
        results = np.zeros((len(dims),cols,len(methods)))
        for j,method in enumerate(methods):
                output = np.zeros((len(dims),cols))
                for i,dim in enumerate(dims):
                        output[i,:] = calc(method,dim)
                        #if (i==(len(dims)-1) and j==(len(methods)-1)):
                        #       it = 0
                        #       for key,value in ED.iteritems():
                        #               print it,key,value
                        #               it += 1
                results[:,:,j] = output
        return results

## Shared stuff ##
model = "BG"

# True solution
if model=="CY":
        u_x = "pow(x[0],2)*pow(1.0-x[0],2)*(2.0*x[1]-6.0*pow(x[1],2)+4.0*pow(x[1],3))"
        u_y = "pow(x[1],2)*pow(1.0-x[1],2)*(-2.0*x[0]+6.0*pow(x[0],2)-4.0*pow(x[0],3))"
```

```
37          u_t = Expression((u_x,u_y))
            w_t = Expression("pow(x[0],2)*pow(1.0-x[0],2)*(-2.0+12.0*x[1]-12.0*pow
                (x[1],2))␣\
39  ␣␣␣␣␣␣␣␣␣+pow(x[1],2)*pow(1.0-x[1],2)*(-2.0+12.0*x[0]-12.0*pow(x[0],2))")
            p_t = Expression("pow(x[0],2)+pow(x[1],2) -20.0/3.0*x[0]*x[1]+x[0]+x[1]
                ")
41          f1 = "1.0␣+␣2.0*x[0]␣-␣20.0/3.0*x[1]␣\
  ␣␣␣␣␣␣␣␣-4.0*(1.0-6.0*x[0]+6.0*x[0]*x[0])*pow(-1.0+x[1],2)*x[1]␣\
43  ␣␣␣␣␣␣␣␣-4.0*(1.0-6.0*x[0]+6.0*x[0]*x[0])*(-1.0+x[1])*x[1]*x[1]␣\
  ␣␣␣␣␣␣␣␣-12.0*pow(-1.0+x[0],2)*x[0]*x[0]*(-1.0+2.0*x[1])"
45          f2 = "1.0␣-␣20.0/3.0*x[0]␣+␣2.0*x[1]␣\
  ␣␣␣␣␣␣␣␣+12.0*(-1.0+2.0*x[0])*pow(-1.0+x[1],2)*x[1]*x[1]␣\
47  ␣␣␣␣␣␣␣␣+4.0*(1.0-6.0*x[1]+6.0*x[1]*x[1])*pow(-1.0+x[0],2)*x[0]␣\
  ␣␣␣␣␣␣␣␣+4.0*(1.0-6.0*x[1]+6.0*x[1]*x[1])*(-1.0+x[0])*x[0]*x[0]"
49          f = Expression((f1,f2))
            Du00 = "0.2e1*x[0]*pow(0.1e1-x[0],0.2e1)*(0.2e1*x[1]-0.6e1*x[1]*x
                [1]+0.4e1*pow(x[1],0.3e1))-0.2e1*x[0]*x[0]*(0.1e1-x[0])*(0.2e1*x
                [1]-0.6e1*x[1]*x[1]+0.4e1*pow(x[1],0.3e1))"
51          Du01 = "x[0]*x[0]*pow(0.1e1-x[0],0.2e1)*(0.2e1-0.12e2*x[1]+0.12e2*x
                [1]*x[1])"
            Du10 = "x[1]*x[1]*pow(0.1e1-x[1],0.2e1)*(-0.2e1+0.12e2*x[0]-0.12e2*x
                [0]*x[0])"
53          Du11 = "0.2e1*x[1]*pow(0.1e1-x[1],0.2e1)*(-0.2e1*x[0]+0.6e1*x[0]*x
                [0]-0.4e1*pow(x[0],0.3e1))-0.2e1*x[1]*x[1]*(0.1e1-x[1])*(-0.2e1*x
                [0]+0.6e1*x[0]*x[0]-0.4e1*pow(x[0],0.3e1))"
            Du = Expression(((Du00,Du01),(Du10,Du11))) # true grad(u)
55          Du1 = Expression((Du00,Du01))
            Du2 = Expression((Du10,Du11))
57          Dw = Expression(("0.2e1*x[0]*pow(0.1e1-x[0],0.2e1)*(-0.2e1+0.12e2*x
                [1]-0.12e2*x[1]*x[1])-0.2e1*x[0]*x[0]*(0.1e1-x[0])*(-0.2e1+0.12e2*x
                [1]-0.12e2*x[1]*x[1])+x[1]*x[1]*pow(0.1e1-x[1],0.2e1)*(0.12e2-0.24
                e2*x[0])","x[0]*x[0]*pow(0.1e1-x[0],0.2e1)*(0.12e2-0.24e2*x[1])+0.2
                e1*x[1]*pow(0.1e1-x[1],0.2e1)*(-0.2e1+0.12e2*x[0]-0.12e2*x[0]*x[0])
                -0.2e1*x[1]*x[1]*(0.1e1-x[1])*(-0.2e1+0.12e2*x[0]-0.12e2*x[0]*x[0])
                "))
            Dp = Expression(("0.2e1*x[0]-0.20e2/0.3e1*x[1]+0.1e1","0.2e1*x[1]-0.20
                e2/0.3e1*x[0]+0.1e1"))
59
    elif model=="BG":
61          u_x = "exp(x[0])␣*␣cos(x[1])␣+␣sin(x[1])"
            u_y = "-exp(x[0])␣*␣sin(x[1])␣+␣0.1e1␣-␣pow(x[0],␣0.3e1)"
63          u_t = Expression((u_x,u_y))
            w_t = Expression("-0.3e1␣*␣x[0]␣*␣x[0]␣-␣cos(x[1])")
65          p_t = Expression("sin(x[1])*cos(x[0])+x[0]*x[1]*x[1]-0.1e1/0.6e1-sin
                (0.1e1)*(0.1e1-cos(0.1e1))")
            f1 = "sin(x[1])-sin(x[1])*sin(x[0])+x[1]*x[1]"
67          f2 = "0.6e1*x[0]+cos(x[1])*cos(x[0])+0.2e1*x[0]*x[1]"
            f = Expression((f1,f2))
69          Du00 = "exp(x[0])*cos(x[1])"
            Du01 = "-exp(x[0])*sin(x[1])+cos(x[1])"
71          Du10 = "-exp(x[0])*sin(x[1])-0.3e1*x[0]*x[0]"
            Du11 = "-exp(x[0])*cos(x[1])"
73          Du = Expression(((Du00,Du01),(Du10,Du11))) # true grad(u)
            Du1 = Expression((Du00,Du01))
75          Du2 = Expression((Du10,Du11))
            Dw = Expression(("-0.6e1*x[0]","sin(x[1])"))
77          Dp = Expression(("-sin(x[1])*sin(x[0])+x[1]*x[1]","cos(x[1])*cos(x[0])
                +0.2e1*x[0]*x[1]"))

79  # Boundaries
    def boundary(x, on_boundary):
81          return x[0] < DOLFIN_EPS or x[1] < DOLFIN_EPS or x[0] > 1-DOLFIN_EPS \
```

```
    or x[1] > 1-DOLFIN_EPS
83  def corner(x, on_boundary):
        return x[0] < DOLFIN_EPS and x[1] < DOLFIN_EPS
85  noslip = Constant((0.0, 0.0))

87  def calc(method,dim):
            print 'Solving for method: ',method
89          print 'Mesh size: ',dim
            from time import time
91          t1 = time()

93          ERR = np.zeros(cols)
            # create mesh
95          mesh = UnitSquare(dim, dim) # domain is unit square [0,1]x[0,1]
            h=1.0/dim
97          h2=h*h

99          if method=='MF':
                    V1 = VectorFunctionSpace(mesh, "CG", 2)
101                 Q = FunctionSpace(mesh, "CG", 1)
                    W = V1*Q
103         elif (method=='CY1' or method=='QNE1' or method=='QNE1h'):
                    V1 = VectorFunctionSpace(mesh, "CG", 1)
105                 V2 = FunctionSpace(mesh, "CG", 1)
                    Q = FunctionSpace(mesh, "CG", 1)
107                 W = MixedFunctionSpace([V1,V2,Q])
            elif (method=='CY2' or method=='QNE2' or method=='QNE2h'):
109                 V1 = VectorFunctionSpace(mesh, "CG", 2)
                    V2 = FunctionSpace(mesh, "CG", 1)
111                 Q = FunctionSpace(mesh, "CG", 1)
                    W = MixedFunctionSpace([V1,V2,Q])
113
            if model=="CY":
115                 bc0 = DirichletBC(W.sub(0), noslip, boundary)
            elif model=="BG":
117                 bc0 = DirichletBC(W.sub(0), u_t, boundary)

119         if method=='MF':
                    bc1 = DirichletBC(W.sub(1), p_t, corner, "pointwise")
121         else:
                    bc1 = DirichletBC(W.sub(2), p_t, corner, "pointwise")
123         bcs = [bc0, bc1]

125         if method=='MF':
                    (u, p) = TrialFunctions(W)
127                 (v, q) = TestFunctions(W)
            else:
129                 (u, w, p) = TrialFunctions(W)
                    (v, phi, q) = TestFunctions(W)
131
            if method=='MF':
133                 a = (inner(grad(u),grad(v))-div(v)*p-q*div(u))*dx
                    L = inner(f,v)*dx
135         elif (method=='CY1' or method=='CY2'):
                    a = (inner(curl(w)+grad(p),curl(phi)+grad(q))+inner(curl(u)-w,
                        curl(v)-phi)+inner(div(u),div(v)))*dx
137                 L = inner(f,curl(phi)+grad(q))*dx
            elif (method=='QNE1' or method=='QNE2'):
139                 a = (h2*inner(curl(w)+grad(p),curl(phi)+grad(q))+inner(curl(u)
                        -w,curl(v)-phi) \
    + inner(div(u),div(v)))*dx
141                 L = h2*inner(f,curl(phi)+grad(q))*dx
```

```
        elif (method=='QNE1h' or method=='QNE2h'):
143             h = Circumradius(u.cell())
                a = (inner((curl(w)+grad(p)),h*(curl(phi)+grad(q)))+inner(curl
                    (u)-w,curl(v)-phi) \
145 + inner(div(u),div(v)))*dx  + inner(h*curl(w),h*curl(phi))*dx
                L = inner(f,h*(curl(phi)+grad(q)))*dx
147         U = Function(W)

149         t2=time()
            solve(a==L,U,bcs)
151         t3=time()
            print 'Solver took %0.3f seconds.' % (t3-t2)
153
            if method=='MF':
155             u,p = U.split()
            else:
157             u,w,p = U.split()

159         # Calculate errors
            if dim <= 120:
161             Vp = VectorFunctionSpace(mesh,"CG",5) # memory intensive!
                Tp = TensorFunctionSpace(mesh,"CG",3)
163             Qp = FunctionSpace(mesh,"CG",5) # memory intensive!
            else:
165             Vp = VectorFunctionSpace(mesh,"CG",2)
                Tp = TensorFunctionSpace(mesh,"CG",2)
167             Qp = FunctionSpace(mesh,"CG",2)

169         u1=Expression(u_x)
            u2=Expression(u_y)
171         u_tp = project(u_t,Vp)
            u1_tp = project(u1,Qp)
173         u2_tp = project(u2,Qp)
            w_tp = project(w_t,Qp)
175         p_tp = project(p_t,Qp)
            f_tp = project(f,Vp)
177         Du_tp = project(Du,Tp)
            Du1_tp = project(Du1,Vp)
179         Du2_tp = project(Du2,Vp)
            Dw_tp = project(Dw,Vp)
181         Dp_tp = project(Dp,Vp)
            ED = {} # dictionary
183
            # L2 errors
185         ERR[0]=ED['u1err']= sqrt(abs(assemble(inner(u[0]-u1,u[0]-u1)*dx)))
            ERR[1]=ED['u2err']= sqrt(abs(assemble(inner(u[1]-u2,u[1]-u2)*dx)))
187         ERR[2]=ED['u_err']= sqrt(abs(assemble(inner(u-u_t,u-u_t)*dx)))
            print 'Vel. L2 err: ',ED['u_err']
189         if method<>'MF':
                ERR[3]= ED['w_err'] = sqrt(assemble((w-w_t)**2*dx))
191             print 'Vor. L2 err: ',ED['w_err']
            ERR[4]=ED['p_err']= sqrt(assemble((p-p_t)**2*dx))
193
            # H1 (semi-norm) errors
195         ERR[5]=ED['u1errH1']= sqrt(assemble(inner(grad(u[0])-Du1_tp,grad(u[0])
                -Du1_tp)*dx))
            ERR[6]=ED['u2errH1']= sqrt(assemble(inner(grad(u[1])-Du2_tp,grad(u[1])
                -Du2_tp)*dx))
197         ERR[7]=ED['u_errH1']= sqrt(assemble(inner(grad(u)-Du_tp,grad(u)-Du_tp)
                *dx))
            if method<>'MF':
199             ERR[8]=ED['w_errH1']= sqrt(assemble(inner(grad(w)-Dw_tp,grad(w
```

```
                )-Dw_tp)*dx))
        ERR[9]=ED['p_errH1']= sqrt(assemble(inner(grad(p)-Dp_tp,grad(p)-Dp_tp)
            *dx))
201
        # Energy
203     ERR[10]=ED['E'] = assemble(0.5*inner(grad(u),grad(u))*dx-inner(f,u)*dx
            )
        ERR[11]=ED['E_tr'] = assemble(0.5*inner(Du_tp,Du_tp)*dx-inner(f,u_t)*
            dx)
205     ERR[12]=ED['E_err'] = ED['E']-ED['E_tr']
        ERR[13]=ED['E_re'] = 100*ED['E_err']/ED['E_tr']
207
        ERR[30]=ED['E2'] = ED['E'] + assemble(0.5*inner(u,u)*dx)
209     ERR[31]=ED['E_tr2'] = ED['E_tr'] + assemble(0.5*inner(u_tp,u_tp)*dx)
        ERR[32]=ED['E_err2'] = ED['E2']-ED['E_tr2']
211     ERR[33]=ED['E_re2'] = 100*ED['E_err2']/ED['E_tr2']

213     ERR[14]=ED['F'] = assemble(inner(f_tp,f_tp)*dx) # total force

215     ERR[15]=ED['gradU_tot'] = sqrt(assemble(inner(Du_tp,Du_tp)*dx))
        ERR[16]=ED['gradU_err'] = sqrt(assemble(inner(grad(u)-Du_tp,grad(u)-
            Du_tp)*dx))
217     ERR[17]=ED['gradU_re'] = 100*ED['gradU_err']/ED['gradU_tot']

219     # Residuals
        if method<>'MF':
221             ERR[18]=ED['r1']=assemble(inner(curl(w)+grad(p)-f,curl(w)+grad
                    (p)-f)*dx)
                ERR[19]=ED['r2']=assemble(inner(curl(u)-w,curl(u)-w)*dx)
223             ERR[20]=ED['r3']=assemble(inner(div(u),div(u))*dx)

225     t4 = time()
        print 'Error calculation took %0.3f seconds.' % (t4-t3)
227
        cnt = 0
229     t5 = time()
        A, bb = assemble_system(a, L, bcs)
231     N = A.size(0)
        for i in range(N):
233             row = A.getrow(i)
                row = row[1] # get second array, the one with actual values
235             cnt += sum(abs(row)>DOLFIN_EPS)
        t6 = time()
237
        ERR[21]=ED['tSolve']= t3-t2 # time to solve
239     ERR[22]=ED['tSetupSolve']= t3-t1 # time to setup and solve
        ERR[23]=ED['tErrors']= t4-t3 # time to calc errors
241     ERR[24]=ED['tSparsity']= t6-t5 # time to calc sparsity
        ERR[25]=ED['tTotal']= t6-t1 # total time elapsed
243
        ERR[26]=ED['dim']=dim
245     ERR[27]=ED['unknowns']=len(U.vector().array())
        ERR[28]=ED['nonzero']=cnt
247     ERR[29]=ED['sparsity']=float(cnt)/float(N*N)

249     return ERR
```

Listing I.3: CY1plot.py

```
1 ## Chang & Yang numerical example 1 convergence analysis ##

3 # File: CY1plot.py
```

```
# Created on April 18, 2012
# Last revised on May 30, 2012
# Author: Jens V. Christiansen
#-------------------------------------#

import numpy as np
import matplotlib.pyplot as plt
import pickle,pprint

pkl_file = open('CY1_CY01_4_methods.pkl', 'rb')
DictIn = pickle.load(pkl_file)
pkl_file.close()

dims = DictIn['dims']
methods = DictIn['methods']
results = DictIn['results']
model = DictIn['model']

lm = len(methods)
II = len(dims)-1

print 'Model problem: %s' %(model)
print 'Dimensions: ',dims

# vvp L2 error, tot enrgy rel error, convergence order,
# print velocity L2 error on largest grid
print '\nVelocity L2 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,2,i]
# print vorticity L2 error on largest grid
print '\nVorticity L2 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,3,i]
# print pressure L2 error on largest grid
print '\nPressure L2 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,4,i]
# print velocity H1 error on largest grid
print '\nVelocity H1 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,7,i]
# print vorticity H1 error on largest grid
print '\nVorticity H1 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,8,i]
# print pressure H1 error on largest grid
print '\nPressure H1 error:'
for i,method in enumerate(methods):
        print method,"%.3e" % results[II,9,i]
# print total energy relative error on largest grid
print '\nTotal energy rel. error:'
for i,method in enumerate(methods):
        print method,"%.3g %%" % results[II,13,i]
# print residuals on largest grid
print '\nResiduals:'
for i,method in enumerate(methods):
        print method,"%.3e, %.3e, %.3e" % (results[II,18,i],results[II,19,i],
            results[II,20,i])
# print solver time on largest grid
print '\nSolver time:'
for i,method in enumerate(methods):
```

```
65          print method,"%.1f" % results[II,21,i]
    var = 2 # output variable of interest, 2=Velocity L2 error
67  x = np.array(dims)
    arrConv = np.zeros([lm,2,4]) # arrConv = methods x L2/H1 x velocity/vorticity/
        pressure/TERE
69  # Calc convergence order
    A = np.vstack([np.log(x), np.ones(len(x))]).T
71  print '\nConvergence order for velocity (L^2 norm):'
    for i,method in enumerate(methods):
73          m,c = np.linalg.lstsq(A,np.log(results[:,var,i]))[0]
            print method,"%.3g" % abs(m)
75          arrConv[i,0,0]=abs(m)
    print '\nConvergence order for velocity (H^1 norm):'
77  for i,method in enumerate(methods):
            # calc H1err from H1 semi-norm
79          H1err = np.sqrt(pow(results[:,7,i],2)+pow(results[:,var,i],2))
            m,c = np.linalg.lstsq(A,np.log(H1err))[0]
81          print method,"%.3g" % abs(m)
            arrConv[i,1,0]=abs(m)
83  var = 3
    print '\nConvergence order for vorticity (L^2 norm):'
85  for i,method in enumerate(methods):
            if method<>"MF":
87                  m,c = np.linalg.lstsq(A,np.log(results[:,var,i]))[0]
                    print method,"%.3g" % abs(m)
89                  arrConv[i,0,1]=abs(m)
    print '\nConvergence order for vorticity (H^1 norm):'
91  for i,method in enumerate(methods):
            # calc H1err from H1 semi-norm
93          if method<>"MF":
                    H1err = np.sqrt(pow(results[:,8,i],2)+pow(results[:,var,i],2))
95                  m,c = np.linalg.lstsq(A,np.log(H1err))[0]
                    print method,"%.3g" % abs(m)
97                  arrConv[i,1,1]=abs(m)
    var = 4
99  print '\nConvergence order for pressure (L^2 norm):'
    for i,method in enumerate(methods):
101         m,c = np.linalg.lstsq(A,np.log(results[:,var,i]))[0]
            print method,"%.3g" % abs(m)
103         arrConv[i,0,2]=abs(m)
    print '\nConvergence order for pressure (H^1 norm):'
105 for i,method in enumerate(methods):
            # calc H1err from H1 semi-norm
107         H1err = np.sqrt(pow(results[:,9,i],2)+pow(results[:,var,i],2))
            m,c = np.linalg.lstsq(A,np.log(H1err))[0]
109         print method,"%.3g" % abs(m)
            arrConv[i,1,2]=abs(m)
111 print '\nConvergence order for TERE:'
    for i,method in enumerate(methods):
113         m,c = np.linalg.lstsq(A,np.log(abs(results[:,13,i])))[0]
            print method,"%.3g" % abs(m)
115         arrConv[i,0,3]=abs(m)
    print '\nUnknowns, nonzero entries, sparsity:'
117 for i,method in enumerate(methods):
            print method,"%.3e, %.3e, %.3e" % (results[II,27,i],results[II,28,i],
                results[II,29,i])
119 # print total energy relative error on largest grid
    print '\nTotal energy rel. error:'
121 for i,method in enumerate(methods):
            print method, results[:,13,i]
123 # Write output for Latex table (mixed stuff)
    f = open('CY1_latex1.txt', 'w')
```

```
125 for i,method in enumerate(methods):
            s1 = [method,results[II,2,i],results[II,3,i],results[II,4,i],results[
                II,13,i],\
127 arrConv[i,0,0],arrConv[i,1,0],results[II,21,i]]
            s = "%s & %.3e & %.3e & %.3e & %.3g & %.3g & %.3g & %.1f\\\\\n" % \
129 (s1[0],s1[1],s1[2],s1[3],s1[4],s1[5],s1[6],s1[7])
            f.write(s)
131 f.close()
    # Write output for Latex table (convergence rates)
133 f = open('CY1_latex2.txt', 'w')
    for i,method in enumerate(methods):
135         s1 = [method,arrConv[i,0,0],arrConv[i,1,0],arrConv[i,0,1],arrConv[i
                ,1,1],arrConv[i,0,2],arrConv[i,1,2],arrConv[i,0,3]]
            s = "%s & %.3g & %.3g & %.3g & %.3g & %.3g & %.3g & %.3g \\\\\n" % \
137 (s1[0],s1[1],s1[2],s1[3],s1[4],s1[5],s1[6],s1[7])
            f.write(s)
139 f.close()
    if lm<=2:
141         plotDim = lm*100+10
    elif lm<=4:
143         plotDim = 220
    elif lm<=6:
145         plotDim = 320
    elif lm<=8:
147         plotDim = 420
    elif lm==9:
149         plotDim = 330
    elif lm<=12:
151         plotDim = 430
    hspace = 0.6
153 # Make loglog plot of var
    var = 2 # Velocity L2 error
155 plt.figure(1)
    plt.subplots_adjust(hspace=hspace)
157 a = plotDim
    for i,method in enumerate(methods):
159         a += 1
            plt.subplot(a)
161         plt.loglog(dims, results[:,var,i])
            plt.xticks(dims,dims)
163         plt.grid(True)
            plt.title(method)
165         plt.xlim(dims[0],dims[II])
    plt.savefig('CY1_loglog1.pdf')
167
    # Make loglog plot of var
169 var = 13 # Total energy relative error
    var = 7 # H1 velocity error
171 plt.figure(2)
    plt.subplots_adjust(hspace=hspace)
173 a = plotDim
    for i,method in enumerate(methods):
175         a += 1
            plt.subplot(a)
177         plt.loglog(x, abs(results[:,var,i]))
            plt.xticks(dims,dims)
179         plt.grid(True)
            plt.title(method)
181         plt.xlim(dims[0],dims[II])
    plt.savefig('CY1_loglog2.pdf')
183
    var = 21 # Time to solve
```

```
# make regular plot of var
plt.figure(3, figsize=(7, 11), dpi=80)
plt.subplots_adjust(hspace=0.3)
plt.subplot(311)
plt.title('Time to solve (sec)')
lines = plt.plot(x,results[:,21,:]) # time to solve
plt.xticks(dims,dims)
plt.grid(True)
plt.xlim(dims[0],dims[II])
plt.legend( lines,
        ('MF', 'CY1', 'CY2', 'QNE1', 'QNE2', 'QNE1h', 'QNE2h'),
        'upper left')
plt.subplot(312)
plt.title('Unknowns')
lines = plt.plot(x,results[:,27,:]) # unknowns
plt.xticks(dims,dims)
plt.grid(True)
plt.xlim(dims[0],dims[II])
plt.legend( lines,
        ('MF', 'CY1', 'CY2', 'QNE1', 'QNE2', 'QNE1h', 'QNE2h'),
        'upper left')
plt.subplot(313)
plt.ticklabel_format(axis='y',style='sci')
plt.title('Nonzero elements')
plt.plot(x,results[:,28,:]) # nonzero elements
plt.xticks(dims,dims)
plt.grid(True)
plt.xlim(dims[0],dims[II])
plt.legend( lines,
        ('MF', 'CY1', 'CY2', 'QNE1', 'QNE2', 'QNE1h', 'QNE2h'),
        'upper left')
plt.savefig('CY1_plot1.pdf')
plt.show()
```

I.3. Topology optimization in FEniCS

The topology optimization problems in Chapter 4 were solved using two scripts for each problem. The code for numerical example 1: 'optimal diffusion' is shown here. The rest can be found at GitHub; see the link above.

- `BP1.py` is the main script. It
  - contains the optimization loop;
  - imports `BP1import.py` and calls functions from there;
  - writes results to a log, saves them to disk, and plots them.
- `BP1import.py` has multiple functions:
  - it imports the `DOLFIN` library;
  - it imports the MMA algorithm from `ksmma.so`, which is compiled Fortran code;
  - it contains the main 'solver' function, which is called by `BP1.py` in each iteration of the optimization process.

Listing I.4: BP1.py

```
## Replication of Borrvall & Petersson (2003) numerical examples

# Created:      15/02/12
# Last edited:  20/05/12
# Author:       Jens V. Christiansen

# File: BP1.py

# Solution procedure
if 0: '''
0) Make an initial guess for the design function \alpha(\rho)
1) Solve Stokes equations in some domain using \alpha(\rho)
2) Compute energy functional J_\alpha (u)
3) Find variation of energy functional \delta J
4) Update \alpha(\rho) using some algorithm (MMA)
5) Compute some measure of the quality of the solution
6) If the solution is still not good enough, go to step 1. Otherwise stop.
'''
#-----------------------------------------#
# Numerical example 1: Optimal diffusion #
#-----------------------------------------#

import numpy as np
import ksmma
from BP1import import *
from time import time
t1 = time()

# Set parameters (set mesh dim and solver method in BP1InitDolfin) #
mma_tol = 0.05 # stop MMA when objective function changes less than this (in %)
max_ite = 50
gamma = 0.5 # allowed fluid volume

### Call PDE solver ###
# set initial guess for rho vector
rho = gamma *np.ones(N)
xval = rho
print 'Number of elements: ',N

from time import time
t2 = time()
f0val,fval,df0dx,dfdx,U = SolveStokes(xval,useLS,SSargs)
t3 = time()
f0ini = f0val

# Save initial solution for plot
U0 = Function(W)
U0.vector()[:] = U.vector()[:]

### Call MMA ###
fmax = gamma *np.ones(M)
print fval
print 'Initial guess:'
print 'x-values:'
print xval
print 'Objective function f0=%.4f' % (f0val)
print 'Constraint value: %.4f' % (fval)
print 'Time taken to solve PDE: %.3f seconds.\n' % (t3-t2)

```

```
# optimization iteration
it = 0
cont = 1
f0old = f0val

#for k in range(iterations):
while cont:
        it += 1
        # make MMA step
        t2 = time()
        ksmma.mmasub(it,M,N,GEPS,iyfree,xval,xmma,\
                xmin,xmax,xold1,xold2,xlow,xupp, \
                alfa,beta,a,b,c,y,z,ulam, \
                f0val,fval,fmax,df0dx,dfdx, \
                p,q,p0,q0,uu,gradf,dsrch,hessf)
        ksmma.xupdat(N,it,xmma,xval,xold1,xold2)
        t3 = time()

        # evaluate everything again
        t4 = time()
        f0val,fval,df0dx,dfdx,U = SolveStokes(xval,useLS,SSargs)
        t5 = time()
        f0chg = f0val-f0old
        f0chg_pc = 100*(f0val-f0old)/f0old
        f0old = f0val
        vio    = 100*(fval-gamma)/gamma
        print 'Iteration: %.0f' % (it)
        print 'x-values: ', xval
        print 'Objective function f0=%.4f' % (f0val)
        print 'Change in f0 = %.4f or %.3f %%' % (f0chg,f0chg_pc)
        print 'Constraint value: %.4f' % (fval)
        print 'Constraint violation: %.3f %%' % (vio)
        print 'Time taken to run MMA: %.3f' % (t3-t2)
        print 'Time taken to solve PDE: %.3f\n' % (t5-t4)
        if (it>5 and abs(f0chg_pc)<mma_tol and abs(vio)<mma_tol) or it ==
            max_ite:
                cont = 0

### Write to log ###
from datetime import datetime
now = datetime.now()
f = open('../BP_log.txt', 'a')
s1 = "Run finished at: %s\nProblem: BP1\nMethod: %s\ndim: %.3g\nTime taken: %
    .3g min %.3f sec\nIterations: %.3g\nUnknowns: %.6g\n" % \
(now.strftime("%Y-%m-%d %H:%M:%S"),method,dim,int((t5-t1)/60),(t5-t1) % 60,it,
    unkn)
s2 = "Gamma: %.3g\nmma_tol: %.3g\nmax_ite: %.3g\n" % (gamma,mma_tol,max_ite)
s3 = "Obj. func. f0\n    initial: %.4g\n    final: %.4g\n    change: %.3g %%\
    nConstraint\n    fval: %.4g\n    violation: %.4g %%\n\n" % (f0ini,f0val
    ,100*(f0val-f0ini)/f0ini,fval,vio)
f.write(s1)
f.write(s2)
f.write(s3)
f.close()

### Display results ###
print 'Solution done.\nTotal time taken: %.3g min %.3f sec on a %.3g x %.3g
    grid.\n' \
% (int((t5-t1)/60),(t5-t1) % 60,dim,dim)

# Get sub-functions
if useLS:
```

```
        u, w, p = U.split()
117     u0, w0, p0 = U0.split()
else:
119     u, p = U.split()
        u0, p0 = U0.split()
121 rho = Function(G)
rho.vector()[:] = xval
123
# Save solution in VTK format
125 ufile_pvd = File("results/bp1_velocity.pvd")
ufile_pvd << u
127 pfile_pvd = File("results/bp1_pressure.pvd")
pfile_pvd << p
129 rhofile_pvd = File("results/bp1_rho.pvd")
rhofile_pvd << rho
131 if useLS:
        wfile_pvd = File("results/bp1_vorticity.pvd")
133     wfile_pvd << w

135 # Plot solution
plot(u,mesh = mesh,wireframe = True,axes = True)
137 if useLS:
        plot(w,mesh = mesh,wireframe = True,axes = True)
139 plot(p,mesh = mesh,wireframe = True,axes = True)
plot(rho,mesh = mesh,wireframe = True,axes = True)
141 interactive()
```

Listing I.5: BP1import.py

```
1 # File: BP1import.py

3 ### Dolfin initialization ###

5 from dolfin import *
import numpy as np
7
method = 'QNE1h' # methods=['MF','CY1','QNE2','QNE1h']
9 dim = 100

11 if method<>'MF':
        useLS = 1
13 else:
        useLS = 0
15 usePrBC = 0 # prescribe p=0 at outlet

17 if method<>'MF':
        usePrBC = 0 # LS requires velocity BCs
19
# create mesh
21 mesh = UnitSquare(dim, dim) # domain is unit square [0,1]x[0,1]
h = 1.0/dim
23 h2 = h*h

25 # Define function spaces
if method=='CY1':
27      ww = 1
        V1 = VectorFunctionSpace(mesh, "CG", 1) # velocity
29 elif method=='QNE1h':
        ww = h
31      V1 = VectorFunctionSpace(mesh, "CG", 1) # velocity
elif method=='QNE2':
33      ww = h2
```

```
        V1 = VectorFunctionSpace(mesh, "CG", 2) # velocity
if useLS:
        V2 = FunctionSpace(mesh, "CG", 1) # vorticity
        Q = FunctionSpace(mesh, "CG", 1) # pressure
        W = MixedFunctionSpace([V1,V2,Q]) # see p. 109 in FEniCS manual
else:
        V = VectorFunctionSpace(mesh, "CG", 2)
        Q = FunctionSpace(mesh, "CG", 1)
        W = V * Q

G = FunctionSpace(mesh,"DG",0)

unkn = len(Function(W).vector().array())
print 'Number of unknowns: ',unkn

# Boundaries
def right_hole(x, on_boundary):
        return x[0] > (1.0 - DOLFIN_EPS) and (x[1] > 0.333 and x[1] < 0.666)
def right_wall(x, on_boundary):
        return x[0] > (1.0 - DOLFIN_EPS) and (x[1] <= 0.333 or x[1] >= 0.666)
def left(x, on_boundary): return x[0] < DOLFIN_EPS
def top_bottom(x, on_boundary):
        return x[1] > 1.0 - DOLFIN_EPS or x[1] < DOLFIN_EPS
def corner(x, on_boundary):
        return x[0] > 1-DOLFIN_EPS and x[1] > 1-DOLFIN_EPS

# No-slip boundary condition for velocity
noslip = Constant((0.0, 0.0))
bc11 = DirichletBC(W.sub(0), noslip, top_bottom)
bc12 = DirichletBC(W.sub(0), noslip, right_wall)

g_bar_in = 1 # flow velocity at center of left boundary
g_bar_out = 3 # flow velocity at center of right boundary

# BCs for inflow / outflow
inflow = Expression(("1-(2*x[1]-1)*(2*x[1]-1)", "0.0"))
outflow = Expression(("3*(1-(6*x[1]-3)*(6*x[1]-3))", "0.0"))
bc21 = DirichletBC(W.sub(0), inflow, left)
bc22 = DirichletBC(W.sub(0), outflow, right_hole)

zero = Constant(0)
if usePrBC:
        bc31 = DirichletBC(W.sub(1), zero, right_hole)
else:
        bc31 = DirichletBC(W.sub(1), zero, corner, "pointwise")

# Collect boundary conditions
if usePrBC:
        bcs = [bc11, bc12, bc21, bc31] # pressure + velocity BCs
else:
        bcs = [bc11, bc12, bc21, bc22, bc31] # pin pressure, velocity BCs
        #bcs = [bc11, bc12, bc21, bc22] # plain velocity BCs
if useLS:
        bcs = [bc11, bc12, bc21, bc22, bc31] # pin pressure, velocity BCs

# Initialize piecewise constant coefficient vectors
alpha = Function(G)
grad_alpha = Function(G)
dummy = Function(G)
test = Function(G)
qqq = TestFunction(G)
N = len(alpha.vector().array())
```

```
# Define constants
mu = 1
alpha_low = 2.5*mu/100**2
alpha_high = 2.5*mu/0.01**2

# alpha >= 0 is inverse permeability: alpha close to zero => fluid flow
# rho \in [0,1]. rho = 0 => no flow, rho = 1 => free flow
# q = qval > 0 parameter determines the level of 'grey' in the optimal design
qval = 0.1
def alpha_q(rho):
        N = len(rho)
        alpha = np.zeros(N)
        for i in range(N):
                alpha[i] = alpha_high+(alpha_low-alpha_high)*rho[i]*(1+qval)/(
                    rho[i]+qval)
        return alpha
def grad_alpha_q(rho):
        N = len(rho)
        grad_alpha = np.zeros(N)
        for i in range(N):
                grad_alpha[i] = -(-alpha_low+alpha_high)*(1+qval)*qval/(rho[i
                    ]+qval)**2
        return grad_alpha

# Define variational problem
f = Constant((0.0, 0.0))
if useLS:
        (u, w, p) = TrialFunctions(W) # w = omega
        (v, phi, q) = TestFunctions(W)
        L = ww*inner(f,curl(phi)+grad(q))*dx
        SSargs = [u,p,v,q,w,phi]
else:
        (u, p) = TrialFunctions(W)
        (v, q) = TestFunctions(W)
        L = inner(f,v)*dx
        SSargs = [u,p,v,q]
U = Function(W)

### End of Dolfin initialization ###

### MMA initialization ###

#     The problem is assumed to be on the following form:
#
#     minimize  f_0(x) + z + 0.05*z^2 + sum{c_i*y_i + 0.5*(y_i)^2}
#
#     subject to  f_i(x) - a_i*z - y_i <= fmax_i ,    i=1,..,M
#                          xmin_j <= x_j <= xmax_j ,    j=1,..,N
#                                 y_i >= 0 ,            i=1,..,M
#                                   z >= 0 .

# DEFINE PROBLEM DIMENSIONS & FUNCTIONS
# number of constraints
M = 1
# number of variables
GEPS = 1.0E-07
xmin = 0   *np.ones(N)
xmax = 1  *np.ones(N)

a     = 0.0   *np.ones(M)
c     = 1000.*np.ones(M)
```

```
# ALLOCATE OTHER STORAGE
xold1 = np.zeros(N)
xold2 = np.zeros(N)
xmma  = np.zeros(N)
xlow  = np.zeros(N) # lower bound for x
xupp  = np.zeros(N) # upper bound for x
alfa  = np.zeros(N)
beta  = np.zeros(N)
df0dx = np.zeros(N) # gradient of objective function
p0    = np.zeros(N)
q0    = np.zeros(N)

uu    = np.zeros(M)
ulam  = np.zeros(M)
b     = np.zeros(M)
y     = np.zeros(M)
gradf = np.zeros(M)
dsrch = np.zeros(M)
hessf = np.zeros(M*(M+1)/2)
iyfree= np.zeros(M,dtype=np.int32)

dfdx  = np.zeros(N*M) # Jacobian of constraint function
p     = np.zeros(N*M)
q     = np.zeros(N*M)

z     = 0.

### End of MMA initialization ###

### Begin Stokes solver ###
def SolveStokes(rho,useLS,ssArgs):
        u = SSargs[0]
        p = SSargs[1]
        v = SSargs[2]
        q = SSargs[3]
        if useLS:
                w = SSargs[4]
                phi = SSargs[5]

        # build alpha(rho) coefficient vector
        alpha_vec = alpha_q(rho)
        alpha.vector()[:] = alpha_vec
        grad_alpha_vec = grad_alpha_q(rho)
        grad_alpha.vector()[:] = grad_alpha_vec

        # Update a() bilinear form
        if useLS:
                a=(ww*inner(curl(w)+grad(p)+alpha*u,curl(phi)+grad(q)+alpha*v)
                    \
+inner(curl(u)-w,curl(v)-phi)+inner(div(u),div(v)))*dx
        else:
                a=alpha*inner(u,v)*dx+mu*inner(grad(u),grad(v))*dx-p*div(v)*dx
                    -q*div(u)*dx

        # Solve
        from time import time
        t2=time()
        #solver.solve(U.vector(), bb)
        solve(a==L,U,bcs)
        t3=time()

```

```
        # Get sub-functions
217     if useLS:
            u, w, p = U.split()
219     else:
            u, p = U.split()
221
        # Calculate energy functional and its gradient (vector)
223     energy = 0.5*(alpha*inner(u,u)+mu*inner(grad(u),grad(u)))*dx-inner(f,u
            )*dx
        E = assemble(energy)
225     Lqq = 0.5 * grad_alpha * inner(u,u) * qqq *dx
        grad_E = assemble(Lqq)
227
        # Calculate fluid volume
229     dummy.vector()[:] = rho
        fluid_vol = assemble(dummy*dx)
231     dummy.vector()[:] = np.ones(N)
        dom_vol = assemble(dummy*dx)
233
        # Calculate constraint
235     fval = fluid_vol/dom_vol # = constraint value
        dfdx = np.zeros(N)
237     for c in cells(mesh):
            dfdx[c.index()] = c.volume()
239
        # Calculate residuals
241     if useLS:
            r1 = assemble(ww*inner(curl(w)+grad(p)+alpha*u-f,curl(w)+grad(
                p)+alpha*u-f)*dx)
243         r2 = assemble(inner(curl(u)-w,curl(u)-w)*dx)
            r3 = assemble(inner(div(u),div(u))*dx)
245     print 'Energy gradient: '
        print grad_E.array()
247     if useLS:
            print 'Residuals:',r1,r2,r3
249     f0val = E
        df0dx = grad_E
251     return f0val,fval,df0dx,dfdx,U

253 ### End Stokes solver ###
```

Bibliography

[1] Harry B. Bingham, Poul S. Larsen, and Allan V. Barker. *Computational Fluid Dynamics: Lecture Notes for Course no. 41319*. DTU, 2011.

[2] Pavel B. Bochev and Max D. Gunzburger. *Least-Squares Finite Element Methods*. Springer, 2009.

[3] Thomas Borrvall and Joakim Petersson. Topology optimization of fluids in stokes flow. *Int J. Numer. Meth. Fluids*, 41:77–107, 2003.

[4] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 2008.

[5] Ching L. Chang and Suh-Yuh Yang. Analysis of the l^2 least-squares finite element method for the velocity-vorticity-pressure stokes equations with velocity boundary conditions. *Applied Mathematics and Computation*, 130:121–144, 2002.

[6] Ole Christensen. *Functions, Spaces, and Expansions: Mathematical Tools in Physics and Engineering*. DTU Mathematics, 2010.

[7] B. Dacorogna and J. Moser. On a partial differential equation involving a jacobian determinant. *Ann. Inst. H. Poincaré*, 7:1–26, 1990.

[8] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[9] Jr. George B. Thomas and Ross L. Finney. *Calculus and Analytic Geometry*. Addison-Wesley Publishing Company, fifth edition, 1979.

[10] Max Gunzburger. Least-squares finite element methods. *http://people.sc.fsu.edu/~mgunzburger/fem/ls.html*.

[11] Ernst Hansen. *Measure theory*. Department of Mathematical Sciences, University of Copenhagen, fourth edition, 2009.

[12] Vagn Lundsgaard Hansen. *Functional Analysis: Entering Hilbert Space*. World Scientific, 2006.

[13] Gérard Michaille Hedy Attouch, Giuseppe Buttazzo. *Variational Analysis in Sobolev and BV Spaces: Applications to PDEs and Optimization*. SIAM, 2006.

[14] David G. Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1969.

[15] FEniCS Project. Fenics documentation. *http://fenicsproject.org/documentation/*.

[16] FEniCS Project. Fenics stokes example. *http://fenicsproject.org/documentation/dolfin/dev/python/demo/pde/stokes-iterative/python/documentation.html*.

[17] Krister Svanberg. The method of moving asymptotes - a new method for structural optimization. *Int J. Numer. Meth. Engineering*, 24:359–373, 1987.

[18] Endre Süli. *Finite Element Methods for Partial Differential Equations*. Mathematical Institute, University of Oxford, 2011.

[19] Wikipedia. Lipschitz domain. *http://en.wikipedia.org/wiki/Lipschitz_domain*.

[20] Wikipedia. Reynold's transport theorem. *http://en.wikipedia.org/wiki/Reynolds_transport_theorem*.