Applications of the Discrete Adjoint Method in Computational Fluid Dynamics

by

René Schneider

Submitted in accordance with the requirements for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

April 2006

The candidate confirms that the work submitted is his own and that the appropriate credit has been given where reference has been made to the work of others.

Abstract

The discrete adjoint method allows efficient evaluation of the derivative of a function $I(s)$ with respect to parameters s in situations where I depends on s indirectly, via an intermediate variable $\omega(s)$, which is computationally expensive to evaluate. In this thesis two applications of this method in the context of computational fluid dynamics are considered. The first is shape optimisation, where the discrete adjoint approach is employed to compute the derivatives with respect to shape parameters for a performance functional depending on the solution of a mathematical flow model which has the form of a discretised system of partial differential equations. In this context particular emphasis is given to efficient solution strategies for the linear systems arising in the discretisation of the flow models. Numerical results for two example problems are presented, demonstrating the utility of the approach.

The second application, in adaptive mesh design, allows efficient evaluation of the derivatives of an *a posteriori* error estimate with respect to the positions of the nodes in a finite element mesh. This novel approach makes additional information available which may be utilised to guide the automatic design of adaptive meshes. Special emphasis is given to problems with anisotropic solution features, for which adaptive anisotropic mesh refinement can deliver significant performance improvements over existing adaptive *h*-refinement approaches. Two adaptive solution algorithms are presented and compared to existing approaches by applying them to a reaction-diffusion model problem.

Acknowledgements

Firstly, I would like to thank the sponsors which made this work possible by providing the required financial support: the University of Leeds, Advantage CFD (CASE award) and the Engineering and Physical Sciences Research Council (fees only award). I am also grateful to Chemnitz University of Technology for allowing me to finish this thesis under their employment.

I'd like to thank my supervisor, Professor Peter Jimack, for enabling me to undertake these studies, for providing me with an interesting and challenging topic, for countless pointers to relevant literature and interesting problems, and for his enormous patience with regard to my English language skills.

Special thanks go to Advantage CFD for their support in the development of the finite volume discretisation used in this work, and the inspiration they gave to me by allowing me to see some real, challenging engineering applications of PDE-based simulation.

I would also like to thank all my colleagues from the School of Computing and all my friends in Leeds who made these three years of living in a foreign country a happy and unforgettable experience.

Further, many thanks to my parents and to my future parents-in-law, especially for their support during January–March 2006, when I was struggling to care for my ill girlfriend and our eight month old baby, whilst teaching at Chemnitz University of Technology and trying to finish this thesis.

Finally, and most of all, I would like to thank my beloved Verena. Without her this time in Leeds would never have been such an enjoyable experience. I thank her for bearing with me when I was too deep in thought regarding this work or one of those little other projects, and especially I thank her for giving us the most wonderful souvenir, our lovely son Finley who was born in Leeds in June 2005.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

[75] **R. Schneider and P.K. Jimack**, Efficient preconditioning of the discrete adjoint equations for the incompressible Navier-Stokes equations, *International Journal for Numerical Methods in Fluids*, 47:1277–1283, 2005.

[76] **R. Schneider and P.K. Jimack**, Toward anisotropic mesh adaption based upon sensitivity of a posteriori estimates, *School of Computing Research Report Series 2005.03,* University of Leeds, 2005. Available at http://www.comp.leeds.ac.uk/research/pubs/reports/2005/2005_03.pdf.

Contents

List of Figures

List of Tables

Notation

d :	spatial dimension ($d \in \{1,2,3\}$)
$\mathbf{u} := [u_i]_{i=1}^{d}$:	vector valued function
ω :	vector representation of the discrete solution of a PDE
$\underline{u} := [u_i]_{i=1}^{N}$:	coefficient vector of a finite element solution u
$\mathbf{H}_g^1(\Omega)$:	$\{\mathbf{v} \in (\mathbb{H}^1(\Omega))^d : \mathbf{v} = \mathbf{g} \quad \text{on} \quad \partial\Omega\}$
$L_0^2(\Omega)$:	$\{v \in L^2(\Omega) : \int_\Omega v \, \mathrm{d}\Omega = 0\}$
$\|\|u\|\|_1$:	$\mathbb{H}^1(\Omega) - \text{norm}$
$\|\|u\|\|_0$:	$L^2(\Omega) - \text{norm}$
$\|\|u\|\|_\infty$:	maximum-norm

Chapter 1

Introduction

The discrete adjoint method is a technique allowing efficient evaluation of the sensitivities (derivatives) of a functional depending on the solution of a discretised partial differential equation (PDE). In this thesis we consider two applications of this technique in the context of computational fluid dynamics, a field which is concerned with computer methods for approximating the solutions to mathematical flow models. Typically these mathematical flow models are PDEs, whose solutions characterise the behaviour of the fluid in the flow domain.

The two applications considered here are shape optimisation and adaptive mesh design. Both are fundamental in computational fluid dynamics, although on different levels. The desire to control fluid flow, i.e. to influence how a fluid (e.g. water or air) behaves in a flow domain, is the driving force behind all attempts to analyse and describe fluid flow. Traditionally computational methods have been viewed as approaches to analyse fluid behaviour (e.g. [40]), although recently work concentrates more and more on computational treatment of the control problems (e.g. [41]). The shape of objects in the flow domain, or surrounding the flow domain, is one means of controlling the flow. This type of control is applied in all sorts of situations, ranging from the building of irrigation systems in ancient times (shape=form of the duct), through water taps found in every modern house (shape=opening/closing the valve), to sophisticated aircraft wings designed specifically to provide the best possible performance in certain flight conditions. If one is interested in finding the shape for such an object that delivers the best possible performance of the fluid flow system in a quantitatively measurable sense, this leads to a shape optimisation

problem.

At the other end of the scale most computational methods for analysing flow rely on the discretisation of PDEs. Such discretisations are defined based on a mesh, which denotes a partitioning of the geometrical flow domain into small elements of simple geometry (e.g. triangles, quadrangles, tetrahedra, hexahedra). The size of the elements influences the quality of the approximation to the solution of the PDE. In essence, the smaller the elements, the more accurately the solution can be approximated. However, making the elements smaller generally means that their number grows, implying that the computer resources required to define and solve the resulting discrete problems grow as well. Naturally such resources available to solve a specific problem are limited, and thus the number of elements which may be used is limited. Adaptive mesh design essentially seeks to find meshes which allow for the best possible approximation of the PDE solution with a given maximum number of mesh elements, or to achieve a desired accuracy with the smallest possible number of elements. Thus this is one way to improve efficiency of computer methods for fluid flow analysis[1].

The outline of this thesis is now described, see Figure 1.1. In the following section the discrete adjoint method is introduced in a general form, allowing application in many possible scenarios. As the applications considered in this thesis are fundamentally different in character, they are discussed (almost) independently, each forming the topic for one of the two major chapters of this thesis, Chapter 2 for shape optimisation and Chapter 3 for adaptive mesh design. Both of these chapters contain an introduction to their respective area and define the discretisations on which the considerations are based. This is followed by a discussion of the application of the discrete adjoint method to the respective problem. The chapters are closed with numerical examples illustrating the utility of the approaches under consideration. Finally, in Chapter 4, conclusions regarding both applications are drawn, also highlighting how the techniques developed in this thesis may fit together in order to construct efficient algorithms suited to deal with real engineering problems.

1.1 The Discrete Adjoint Method

The derivation in this section is deliberately kept as concise as possible in order to give emphasis to the simplicity and generality of the basic ideas behind the discrete adjoint

[1] Improving accuracy by reducing the size of the triangulation elements is called h-refinement. An alternative approach is p-refinement, where the order of the approximation method is increased to improve accuracy. Whether h- or p-refinement yields the more cost efficient accuracy improvement depends on the smoothness of the PDE solution. Thus, the most efficient approximation can be achieved by combining both approaches suitably, giving rise to hp-refinement. However, p-refinement is not considered in this thesis.

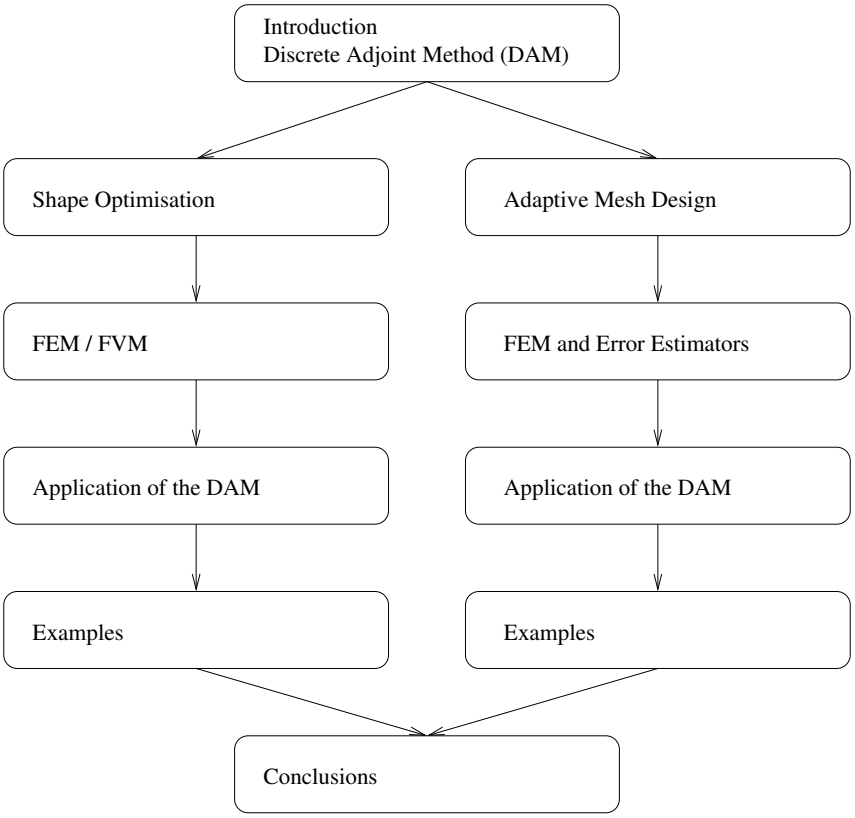


Figure 1.1: Guide to this thesis

method. A more detailed discussion with references to origins and related approaches can, for example, be found in [36].

Consider a scalar-valued function, I, of an independent vector variable s, such that

$$I(s) := \widetilde{I}(\omega(s), s), \tag{1.1.1}$$

where the vector $\omega(s)$ is defined implicitly by the (possibly nonlinear) system

$$0 = R(\omega(s), s). \tag{1.1.2}$$

Consider the effect of small perturbations δs of s in (1.1.1) and (1.1.2). Discarding higher order derivative terms, such a perturbation results in a perturbation δI in I,

$$\delta I = \frac{\partial \widetilde{I}}{\partial \omega}\delta\omega + \frac{\partial \widetilde{I}}{\partial s}\delta s, \tag{1.1.3}$$

where $\delta\omega$ is defined by the perturbation of (1.1.2) which has to be zero, i.e.

$$0 = \delta R = \frac{\partial R}{\partial \omega}\delta\omega + \frac{\partial R}{\partial s}\delta s. \tag{1.1.4}$$

As $\delta R = 0$, we can multiply it by an arbitrary term Ψ^T and subtract it from the right-hand side of (1.1.3), giving

$$\begin{aligned}\delta I &= \frac{\partial \widetilde{I}}{\partial \omega}\delta\omega + \frac{\partial \widetilde{I}}{\partial s}\delta s - \Psi^T\left(\frac{\partial R}{\partial \omega}\delta\omega + \frac{\partial R}{\partial s}\delta s\right) \\ &= \left(\frac{\partial \widetilde{I}}{\partial \omega} - \Psi^T\frac{\partial R}{\partial \omega}\right)\delta\omega + \left(\frac{\partial \widetilde{I}}{\partial s} - \Psi^T\frac{\partial R}{\partial s}\right)\delta s.\end{aligned} \tag{1.1.5}$$

This implies that δI may be evaluated without calculating $\delta\omega$ provided

$$\left[\frac{\partial R}{\partial \omega}\right]^T \Psi = \left[\frac{\partial \widetilde{I}}{\partial \omega}\right]^T, \tag{1.1.6}$$

and if $\omega(s)$ is well-defined by (1.1.2) then Equation (1.1.6) uniquely defines Ψ which is of the same dimension as ω. Equation (1.1.6) is known as the adjoint equation and Ψ as the adjoint solution. With this choice of Ψ the perturbation δI is

$$\delta I = \left(\frac{\partial \widetilde{I}}{\partial s} - \Psi^T\frac{\partial R}{\partial s}\right)\delta s,$$

revealing the representation of the total derivative

$$\frac{\mathrm{D}I}{\mathrm{D}s} = \frac{\partial \widetilde{I}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}. \tag{1.1.7}$$

The importance of this representation is that, once the original equation (1.1.2) is solved and $I(s)$ evaluated from (1.1.1), $\mathrm{D}I/\mathrm{D}s$ may be evaluated for little more than the cost of a single solve of the linear system (1.1.6) and a single matrix-vector product in (1.1.7), regardless of the dimension of s. This is compared to other methods of evaluating $\mathrm{D}I/\mathrm{D}s$ which typically require the solution of (1.1.2) (or a linearised version) per component of s.

Evaluating the derivative by means of (1.1.6) and (1.1.7) is called the discrete adjoint method if $R(.,.)$ is the discretisation of a PDE and ω the discrete representation of the solution of the PDE. The designation as discrete adjoint is used in order to distinguish this approach from a related one, which applies the same ideas on the PDE level, resulting in an adjoint PDE as an analogue to (1.1.6), defining an adjoint solution which is a function defined on the PDE domain. Since the adjoint technique is applied on the continuous level, this approach is denoted the continuous adjoint method, see [36] and references therein for an introduction.

This relatively simple idea of the discrete adjoint method gives rise to powerful applications, two of which will be discussed in the following two chapters.

Chapter 2

Shape Optimisation and Computational Fluid Dynamics

This chapter is concerned with an application for which the discrete adjoint method is particularly well suited: shape optimisation. After a brief introduction, defining the general form of a shape optimisation problem in the context of this thesis and introducing the mathematical models for fluid flow, different numerical approaches to shape optimisation are discussed in Section 2.2. As we will see, the discrete adjoint method forms one possible component for a class of such approaches, gradient based optimisation.

Two discretisation methods for the mathematical flow models, finite elements and finite volumes, are introduced in sections 2.3 and 2.4. In the context of shape optimisation the resulting equation systems have to be solved repeatedly for variations of the domain geometry. Thus, efficient solution strategies for these systems are of special importance. Section 2.5 discusses two such solution strategies and the length of the section reflects its importance, as well as the degree of difficulty this issue poses.

Application of the discrete adjoint method for the two discretisations is discussed in Section 2.6 and the chapter is closed by numerical examples in Section 2.7 illustrating the utility of the approaches presented in this chapter.

2.1 Introduction

In Subsection 2.1.1 a general shape optimisation problem is defined, which forms the object of consideration of this whole chapter. This definition contains a general PDE subproblem, examples of which are given in Subsection 2.1.2 in the context of fluid dynamics. Amongst these is the primary PDE model for this chapter, the stationary incompressible Navier-Stokes equations.

2.1.1 Shape optimisation, a problem statement

Virtually any part of any machine that mankind has ever produced has had its shape designed in order to fulfil certain functions or criteria. In most situations simply obtaining a shape that ensures that the object fulfils its function is satisfactory, but in some situations the shape of a part is also of importance to the overall performance of a device (i.e. how well it performs). For example, the thickness distribution of a beam may influence the weight and durability of a construction. Naturally, in such situations one is interested in finding a shape that gives the best performance for the object (typically, ensuring that it satisfies a constraint such as being capable of bearing a prescribed load). This is the area of shape optimisation in its broadest sense.

In this work we restrict the meaning of the term shape optimisation to problems where the mathematical modelling of the problem yields a performance criterion which is a functional of the solution of a partial differential equation (PDE) and the shape of the object defines the geometrical domain for the PDE. So the general form of a shape optimisation problem in this thesis is

$$\min_{\mathscr{F}\in\mathscr{D}} I(u,\mathscr{F}) \tag{2.1.1a}$$

$$\text{subject to} \quad \mathscr{L}(u) = f \quad \text{in} \quad \Omega(\mathscr{F}) \tag{2.1.1b}$$

$$\text{and} \quad \mathscr{B}(u) = g \quad \text{on} \quad \partial\Omega(\mathscr{F}), \tag{2.1.1c}$$

where $\mathscr{F}$ denotes the shape, $\mathscr{D}$ the set of admissible shapes (which may contain some geometrical and other simple constraints), $\mathscr{L}$ the differential operator of the PDE, $\Omega(\mathscr{F})$ the PDE domain as function of the shape, $\mathscr{B}$ an operator defining the boundary conditions and f and g are given functions.

According to [66, Section 2.6] consideration of such problems goes as far back as the year 1910 when Hadamard, in the context of the calculus of variations, published a formula for a shape derivative of the Green's function of the Laplace operator. Pironneau's

work [66] itself represents a significant contribution to this area and contains some further historical background information, see [66, Section 2.6].

An abstract theory for problems of this kind can be found in, for example, [44], answering questions regarding the existence of solutions for a broad class of problems as well as discussing the possibilities for numerical analysis. The sufficient conditions for the existence of solutions, as given in [44], contain two key ingredients:

- continuity of the mapping $u(\mathscr{F}) : \mathscr{D} \longrightarrow \mathbf{V}$, defining the solution u of the PDE (2.1.1b), (2.1.1c) as a function of the shape $\mathscr{F}$ ($\mathbf{V}$ denotes the appropriate function space for u), and
- compactness of the set of admissible shapes $\mathscr{D}$.

The proofs for the continuity of the mapping $u(\mathscr{F})$ are usually rather technical and depend on the properties of the PDE, of course. The importance of the compactness of the set of admissible shapes $\mathscr{D}$ is demonstrated by a counter example at the end of Chapter 1 in [44][1].

In the area of computational fluid dynamics (CFD) the PDE part of (2.1.1) is a set of equations describing the motion of the fluid in the domain $\Omega(\mathscr{F})$. Before we continue consideration of problem (2.1.1) we discuss a few selected models of fluid flow and specify which one will be used for the remainder of this work.

2.1.2 Mathematical flow models

Preliminaries. The dynamics of fluid motion are generally described by interaction of the dependent variables

velocity	$\mathbf{u} = [u_1, \ldots, u_d]^T$,	
temperature	T,	
pressure	p,	
density	ρ	and
energy	E.	

[1] This counter example problem can be summarised as a heat transfer problem, where in a domain of constant volume a positive constant heat production density is assumed. If one seeks to minimise the integral of the square of the temperature in the domain by optimising the shape of a part of the boundary where homogeneous Dirichlet boundary conditions are applied, one finds that oscillating boundaries result in low values of the performance function. It turns out that the shorter the wavelength of the oscillations the lower the value of the performance function. Yet, no boundary shape with oscillations of infinitely short wavelength exists.

All of these are considered to be functions of the independent variables of space $\mathbf{x} = [x_1, \ldots, x_d]^T$ and time t, where $d \in \{1,2,3\}$ denotes the spatial dimension. A given force density $\mathbf{f} = [f_1, \ldots, f_d]^T$ may act on the fluid from outside, such as gravity for example. Detailed derivation from first principles for the models presented in this section can be found in most introductory books on fluid dynamics, such as [1, 65] for example. Here we only present a very brief overview of four of the most important models, highlighting the different underlying assumptions that are made in their derivations.

Navier-Stokes Equations. The Navier-Stokes Equations are considered the most general model for fluid flow. They comprise of a momentum conservation equation (2.1.2a), a mass conservation equation (2.1.2b) and a constitutive law for the fluid behaviour: for example (2.1.2c) for perfect gases under normal conditions.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \sum_{i=1}^{d} u_i \frac{\partial \mathbf{u}}{\partial x_i}\right) - \mu\Delta\mathbf{u} - (3\lambda+\mu)\nabla\nabla.\mathbf{u} + \nabla p = \mathbf{f}, \tag{2.1.2a}$$

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho\mathbf{u}) = 0, \tag{2.1.2b}$$

$$k\rho^{\gamma} = p. \tag{2.1.2c}$$

The form given above can be found in [93, Section 1] for example. Note that (2.1.2c) imposes strong assumptions on the particular fluid being a gas in a narrow temperature range. It can be replaced by other thermodynamic models for more general situations (which may also be PDEs). Also note that (2.1.2a) assumes the fluid to be Newtonian, so stress is assumed to be proportional to the velocity gradient. The constants λ, μ, k and γ are model parameters.

For many flow situations specialised flow models can be derived from the general Navier-Stokes equations. The following three models all fall into this category.

Compressible Euler Equations. Here the main assumption is that viscous effects are negligible, for example because gases generally have a very low viscosity. In this situation one can set $\mu = \lambda = 0$ in (2.1.2), which gives

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \sum_{i=1}^{3} u_i \frac{\partial \mathbf{u}}{\partial x_i}\right) + \nabla p = \mathbf{f}, \tag{2.1.3a}$$

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho\mathbf{u}) = 0 \tag{2.1.3b}$$

$$k\rho^{\gamma} = p. \tag{2.1.3c}$$

The Euler equations (2.1.3) are considered to provide a good model for fluids with low viscosity in high speed flows (e.g. air around a wing at high speeds, if viscous effects in the boundary layer are not of interest). Again we emphasise that (2.1.3c) imposes strong assumptions on the particular fluid being a gas in a narrow temperature range and can be replaced by other thermodynamic models if necessary.

Incompressible Navier-Stokes Equations. For liquids in general and for gases in slow flows (typically Mach $<\sim 0.3$) incompressibility of the fluid may be assumed, leading to $\rho = \text{const}$ as replacement for the constitutive law (2.1.2c). This simplifies the Navier-Stokes equations (2.1.2) to

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\,\Delta\mathbf{u} + \mathbf{u}.\nabla\mathbf{u} + \nabla p = \mathbf{f}, \tag{2.1.4a}$$

$$\nabla.\mathbf{u} = 0. \tag{2.1.4b}$$

Again we see a momentum equation (2.1.4a) and an equation of mass conservation (2.1.4b). The constant ν is the kinematic viscosity, defined as $\nu = \mu/\rho$, where μ is the viscosity parameter which appeared in (2.1.2). The constant ν is tightly related to the Reynolds number $Re = V\ell\rho/\mu$, a dimensionless constant characterising the flow. If the characteristic length ℓ and characteristic velocity V of the flow are constant, ν is proportional to $1/Re$.

Very often a slightly different form of the incompressible Navier-Stokes equations can be found in the literature, which uses a nonlinear convection term $\nabla.(\mathbf{u}\mathbf{u}^T)$ that is derived in a different way,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla p + \nabla.(\mathbf{u}\mathbf{u}^T) = \nu\,\Delta\mathbf{u} + \mathbf{f}, \tag{2.1.5a}$$

$$\nabla.\mathbf{u} = 0. \tag{2.1.5b}$$

These two versions of the incompressible Navier-Stokes equations are mathematically equivalent due to $\mathbf{u}$ being divergence free (2.1.4b). The form (2.1.5) is more commonly used in the Finite Volume literature as it is more suitable for the derivation of this type of discretisation, while in the Finite Element literature the form (2.1.4) is more frequently found.

Stationary states of the fluid system are characterised by $\frac{\partial \mathbf{u}}{\partial t} = 0$ and $\frac{\partial p}{\partial t} = 0$, i.e. they do not vary with time[2]. If the solutions of (2.1.5) are known to converge to a station-

[2]Note that for incompressible flow $\frac{\partial p}{\partial t} = 0$ is implied by $\frac{\partial \mathbf{u}}{\partial t} = 0$ and $\frac{\partial \mathbf{f}}{\partial t} = 0$.

ary state for $t \to \infty$ and one is only interested in the long term behaviour of the system rather than possible initial temporal effects, then it makes sense to drop the time derivative term from (2.1.5) (or (2.1.4) respectively) and only analyse the resulting stationary incompressible Navier-Stokes equations.

Stokes Equations. The Stokes equations deal with the opposite extreme to the Euler equations. Here one assumes that viscous effects dominate to such an extend that inertia effects are negligible. In such regimes compressible effects can usually be neglected as well, so it is appropriate to further simplify the incompressible models by neglecting the inertia terms $\mathbf{u}.\nabla\mathbf{u}$ in (2.1.4) (or $\nabla.(\mathbf{u}\mathbf{u}^T)$ in (2.1.5)), leading to

$$\rho\frac{\partial \mathbf{u}}{\partial t} - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \tag{2.1.6a}$$

$$\nabla.\mathbf{u} = 0. \tag{2.1.6b}$$

Note that, of the fluid flow models presented in this section, the Stokes equations (2.1.6) are the only one featuring a fully linear differential operator; all other models are nonlinear PDEs.

Concluding remarks. A vast majority of industrially relevant flows can be modelled by the incompressible Navier-Stokes (INS) equations. For this reason we select this model for consideration in the remainder of this work.

However, even in the range where INS provides a good model for the flow, limitations in its usefulness for numerical simulations arise from the properties of solutions. For high[3] Reynolds number flows the solutions of the INS do not usually tend to a stable steady state, but oscillations at various length and timescales remain. This has to be accounted for, for example by using time accurate simulation (rather than steady state computation), even if one is only interested in long term averaged effects. Further, the difference of the length and timescales for some effects in this regime would require extremely fine spatial and temporal discretisations, which makes accurate numerical simulation impossible due to constraints of computational power available for all but a few simple problems, see e.g. [105, Section 1.13].

To overcome this obstacle the concepts of time averaged Navier-Stokes can be con-

[3]We use the terms high, moderate and low Reynolds number here without specifying value ranges. The reason behind this is that the Reynolds numbers at which certain effects set in vary greatly with the geometry of the domains under consideration. If $Re = 1000$ still yields a stable steady state solution for one problem (e.g. driven cavity), the flow may become oscillatory at about $Re = 10$ for another (e.g. flow around a cylinder). Thus specifying value ranges is problem dependent.

sidered, which attempt to model the effects of sub-grid time and length scale effects by analytical as well as empirical models. This is the area of turbulence modelling. Various approaches exist, some of which claim to be successful for a wide variety of engineering applications, see [105, Section 1.13] for a list of references. However, for the purpose of this work we will not utilise such turbulence models but restrict ourselves to low or moderate Reynolds number regimes, where these additional models are not necessary, and where stable steady state solutions exist.

2.2 Numerical methods in shape optimisation

This section gives a systematic overview of numerical approaches to shape optimisation in general, without claiming to be complete, but with emphasis on how well the individual approaches are suited to the problem of interest as specified in Section 2.1. The goal is to give the reader sufficient background on the possible choices as well as to highlight which choices are taken in this present work and why.

2.2.1 Preliminaries

Discretisation of problem (2.1.1) generally requires two distinct sub-problems to be considered:

- discretisation of the shape optimisation sub-problem (2.1.1a) and
- discretisation of the PDE sub-problem (2.1.1b) and (2.1.1c).

For each part of the discretisation various possibilities arise, resulting in different approaches with varying degrees of interaction between the discretisations. In this section we consider a selection of possible approaches and highlight some of their advantages and disadvantages.

The need for discretisation of the shape may appear surprising at first, so we briefly illustrate this point. The shape of a domain is generally defined by its boundary. For a two dimensional domain this boundary is a curve and for three dimensional domains it is a surface (not necessarily simply connected). Sufficient smoothness assumed, parts of the boundary can be considered functions of the form $\mathbb{R}^{d-1} \mapsto \mathbb{R}^d$ fulfilling certain smoothness criteria. As spaces of such functions are generally infinite dimensional, it becomes clear that some kind of discretisation is required in order to allow numerical calculations in finite dimensional spaces.

If the discretisation of the shape is treated separately from the discretisation of the PDE, parametric domain or parametric shape definitions result, leading to parametric PDE problems. In many engineering situations parametric shape definitions may be the most natural choice anyway, as manufacturing capabilites restrict the variety of possible shapes from the outset, i.e. the parameterisation may be given by the manufacturing process. In the general case one can define a family of shape parameterisations $\mathscr{F}_\varkappa$, with $\varkappa > 0$ denoting a discretisation parameter, such that approximation properties improve as $\varkappa \to +0$. This approach is for example considered in [44], from which a key result regarding the convergence of the discrete solutions to solutions of the continuous problem is reproduced

in Appendix A. In essence, if the continuous and discrete PDE problems are well-posed for all feasible domains, the PDE discretisation is compatible with the shape discretisation, the discrete shapes are asymptotically dense in the space of feasible shapes, the set of feasible shapes is compact and the performance criterion is continuous, then any sequence of optimal discrete shapes $\{\mathscr{F}_\varkappa\}, \quad \varkappa \to +0$, contains a subsequence which converges to a locally optimal solution of (2.1.1) and any accumulation point of such a sequence is a locally optimal solution of (2.1.1).

From here on the shape will be understood as a parametric shape, unless stated otherwise. The terms shape and shape parameters will be used as synonyms and the symbol $\mathscr{F}$ will denote a shape parameter vector.

A systematic overview of some of the most important choices in designing a shape optimisation algorithm is given in Figure 2.1 in the form of a decision tree. The following subsections discuss each of the possible choices in Figure 2.1 in some more detail and define the approach taken in this work. In part we follow a similar discussion to that of [41, Chapter 2], while we aim to give a short presentation highlighting properties of the approaches from the viewpoint required for the work described in this thesis.

Note that most of the choices are based on assumptions concerning the typical properties of problem (2.1.1). Thus, for a given specific problem a different choice may be more favourable if the properties of this specific problem do not coincide with assumptions of a typical problem made in this work. Such assumptions will be stated and justified where they are used.

2.2.2 Gradient free vs. gradient based methods

As the title suggests this decision concerns whether or not the optimisation algorithm uses the first derivatives (with respect to $\mathscr{F}$) of the problem variables I and u. There are a number of arguments for either of the approaches. Firstly, in order to use such derivative information, it has to be available. By that we do not only mean that the derivatives have to exist and be continuous, but also that their values should be computable with reasonable expense. The latter of these requirements may present a substantial obstacle in some cases, for example if proprietary software tools are to be used for the solution of the PDE model (2.1.1b) and (2.1.1c). Such software tools may not provide the required information and it may be impossible to extend them to do so. Finite difference approximations of the derivatives may or may not be possible in such a case depending on whether the discrete function, implemented by the software is differentiable.

Examples for gradient free optimisation techniques are direct search methods like the

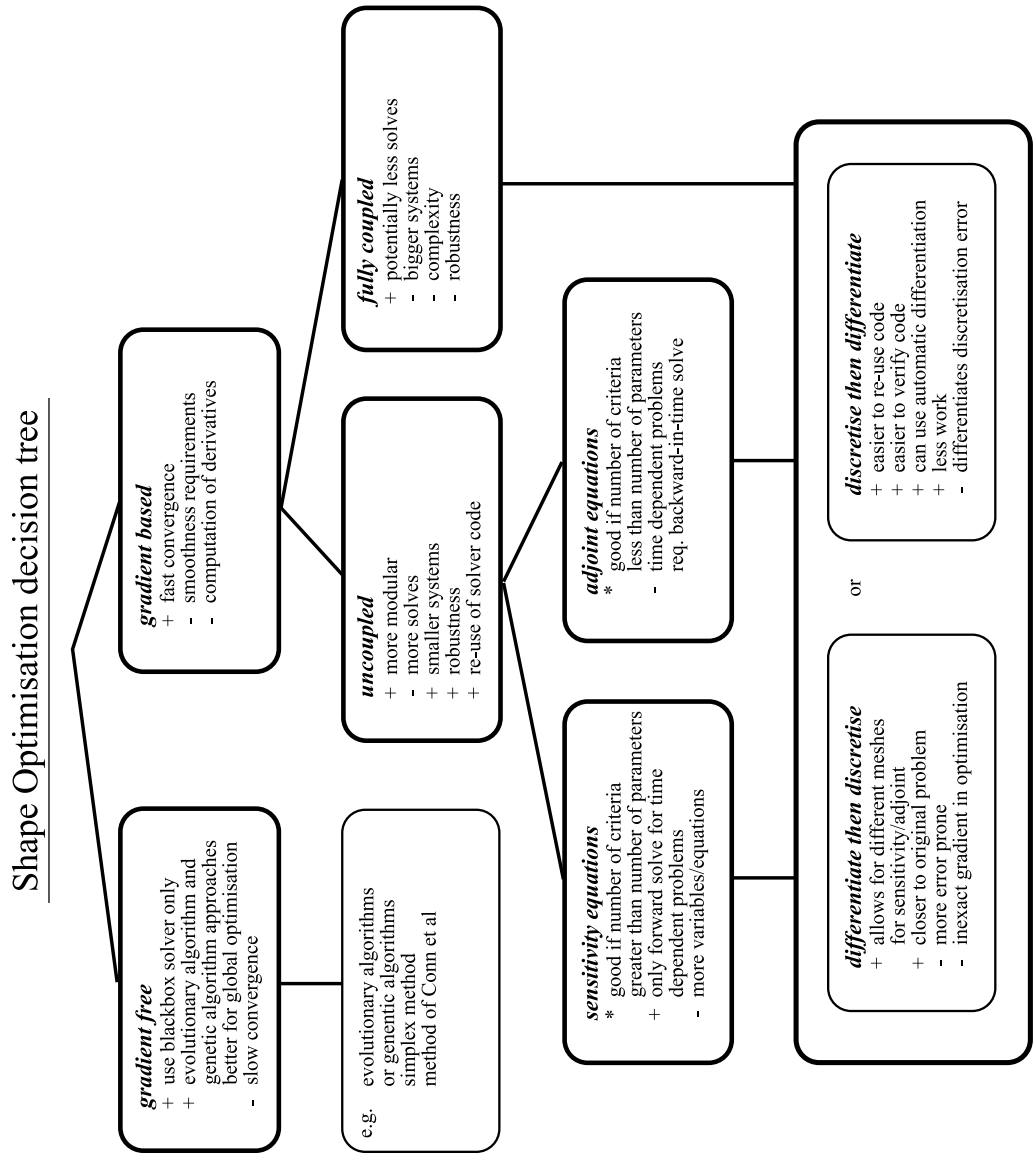


Figure 2.1: Shape optimisation decision tree

simplex method [59], genetic algorithms (GA) or evolutionary algorithms (EA) which hold a population of competing solution candidates at each step (e.g. [25] for an introduction), or methods based on continuously updated low order polynomial approximations of the cost function (e.g. [24, 67]). All of these methods have in common that they use stored data of objective function evaluations for different parameter values in order to determine new candidate solutions by some mechanism. In a sense the local behaviour of the function is deduced from evaluated function values at different points in parameter space. Thus, in high dimensional parameter spaces these methods require a rather large number of function evaluations to even identify a downhill direction, let alone get close to an extremum.

An advantage generally attributed to these methods (e.g. [24, 25]) is that they tend to be harder to distract by local extrema and more likely to find solutions that are good in a global sense, while gradient based methods tend to get attracted to a local extremum close to the initial guess.

In contrast, gradient based optimisation techniques utilise the gradient information to identify search directions. Further, Newton methods utilise second derivative information (or approximations thereof in quasi-Newton methods) to approximate optimal step lengths along the parameter space directions. As a result these methods usually require far fewer steps to obtain an accurate approximation of a locally optimal solution. For a thorough overview of such techniques, and the underlying theory, we refer to the textbook [61], as the details of this are beyond the scope of this work. However, we briefly comment on two techniques which are of special importance to the optimisation techniques used in this work. Both of these methods are described in [61] together with a summary of their origins and an analysis of them.

One of the most important ingredients in the optimisation technique employed in this work is the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update formula. This allows one to compute a symmetric positive definite approximation to the Hessian matrix (matrix of second derivatives) of a cost function, using only the gradients at each step in an iterative optimisation solver. This technique allows one to obtain fast super-linear convergence [61, Chapter 8], without the need for the (often prohibitively expensive) computation of second derivatives which are required for quadratically convergent Newton methods for optimisation. For this approach to develop its full potential, a relatively high regularity of the cost function is required, i.e. the function is required to be three times continuously differentiable. This, however, does not represent a disadvantage for the problems considered in this work.

The second technique of special importance to this work, and which therefore war-

rants further comment here, is the method of sequential quadratic programming (SQP). This method forms the basis for the most successful general optimisation codes which can handle general nonlinear objective functions and nonlinear equality as well as nonlinear inequality constraints [61, Chapter 18]. In SQP methods an auxiliary optimisation problem, with a quadratic objective function and linear constraints, is solved at each step of the iterative process, defining a search direction for a line search, which in turn defines the new iterate. These auxiliary quadratic optimisation problems are defined by the Hessian of the objective function (or the BFGS approximation of it) and the first order derivatives of the objective function and constraints. Thus, the problem of solving a general nonlinear optimisation problem is reduced to solving a sequence of quadratic optimisation problems which are well understood and for which various solution methods exist [61].

The fast convergence of the gradient-based quasi-Newton methods makes them a favourable choice, especially when the gradients of the cost functional can be evaluated efficiently. It is demonstrated as part of the work of this chapter that this is indeed the case when the discrete adjoint method is applied, see sections 2.6.2, 2.6.3 and 2.7.2.2.

2.2.3 Uncoupled vs. fully coupled approaches

There are two fundamental ways of understanding problem (2.1.1):

- as an optimisation problem with equality constraints (2.1.1b) and (2.1.1c), or
- as an optimisation problem where the solution operator for the PDE (2.1.1b) and (2.1.1c) is part of the objective function.

Further constraints resulting from the set of admissible shapes $\mathscr{D}$ are set aside for now, to concentrate on the way in which the PDE constraints are incorporated into the problem. This classification applies to both, the continuous problem (2.1.1) with continuous shape $\mathscr{F}$ and PDE constraints, as well as the analogous discrete problem with discrete shape $\mathscr{F}_\varkappa$ and discretised PDEs as constraints. Depending on which understanding of problem (2.1.1) is adopted different approaches to solve the problem emerge.

The second category in the classification above can be understood as eliminating the equality constraint from the optimisation problem by redefining the objective function as combination of the solution operator and the original objective function. This approach has long been known in the optimisation community [61], with both advantages and disadvantages associated with it. The main advantage is that it reduces the dimension of the optimisation problem, since the dependent variables u are eliminated from it. However,

the original structure of the problem is lost and ill-conditioning can be introduced into the problem [61, Section 15.2]. In our context we call this the *uncoupled* approach, whereas we call the treatment as an equality constrained optimisation problem the *fully coupled* approach.

In the context of shape optimisation problems the advantages and disadvantages of both approaches are more pronounced than for moderately dimensional general problems. A major argument in favour of the fully coupled approach is that it may potentially require fewer solves of linearised equations than the uncoupled approach. This is because each iteration of the optimisation solver requires essentially only one such solve, while in the uncoupled approach the potentially nonlinear PDE (or its discretisation) has to be solved to a relatively high accuracy, usually requiring multiple solves of linear systems per step of the optimisation solver. However, the nesting of nonlinear solves in the uncoupled case does have significant advantages as well. First of all this approach is more modular, which simplifies code development. Further, the linear systems that have to be solved are usually much smaller due to the un-coupling.

The modularity can be a huge advantage. For example it may be possible to re-use existing simulation codes inside "off the shelf" optimisation software, e.g. `DONLP2` [84]. Thus one can rely on robust existing codes. In contrast, the potential advantage of the fully coupled approach, of requiring fewer solves, may fade away quickly if the coupled solve converges slowly or does not even converge at all. The convergence properties of this coupled system may be fundamentally different to those of the individual systems.

Overall we conclude that the modularity and simplicity of the uncoupled approach outweighs the drawbacks of this approach, although this is clearly a subjective judgement. Thus, the fully coupled approach is not considered further in this work. However, we remark that the interested reader may find material on the fully coupled approach in [41] and the individual articles in [16], for example.

2.2.4 Sensitivity vs. adjoint equations

This subsection covers the choice of the method by which the gradient of the cost functional is determined. On first sight the two approaches proposed in the subsection title may not seem to be the only possible choices, but the other approaches worth mentioning, such as automatic differentiation or finite differences, can also fit into these categories as we will see. First let us explain what the names of the categories mean in the context of this work.

Sensitivity equation methods compute the sensitivity of the solution u of the PDE with

respect to the shape parameters $\mathscr{F}$ in some way. The gradient of the performance criterion I is then evaluated by applying the chain rule of differentiation, using the sensitivity $\partial u/\partial \mathscr{F}$, i.e.

$$\frac{\mathrm{D}I(u(\mathscr{F}),\mathscr{F})}{\mathrm{D}\mathscr{F}} = \frac{\partial I}{\partial u}\frac{\partial u}{\partial \mathscr{F}} + \frac{\partial I}{\partial \mathscr{F}}.$$

A detailed discussion of this approach can be found for example in [86], which also answers theoretical questions concerning the differentiability of u with respect to the parameters.

In contrast to the sensitivity equation approach, adjoint equation methods omit the sensitivities of u and instead compute an adjoint solution Ψ as explained in Section 1.1. The relationship (1.1.7)

$$\frac{\mathrm{D}I(u(\mathscr{F}),\mathscr{F})}{\mathrm{D}\mathscr{F}} = \frac{\partial I}{\partial \mathscr{F}} - \Psi^T \frac{\partial R}{\partial \mathscr{F}}$$

suggests that the adjoint solution can be thought of as a sensitivity measure of the performance criterion I with respect to changes in the residuals of the (possibly discretised) PDE. Computation of the gradient is then completed by computing the derivatives of the (discretised) PDE residual and weighting them with Ψ.

Let us assume for one moment that the gradients of more than one, say m, criteria I_i be required in the optimisation algorithm. This may for example be the case if there are multiple objectives for the optimisation (see [58] for an introduction), or if the derivatives of constraints which depend on the PDE solution u have to be evaluated as well as those of the performance criterion.

The important difference between the two approaches in this subsection is that the adjoint solution Ψ is specific to the performance criterion, i.e. one Ψ is computed per I_i and used in the computation of all components of the gradient $\mathrm{D}I_i/\mathrm{D}\mathscr{F}$, while the sensitivities of u are specific to the parameter components, i.e. one sensitivity $\partial u/\partial \mathscr{F}_j$ is computed for each component of the parameter vector $\mathscr{F}$ and can be used to calculate the derivatives of any performance criterion $\mathrm{D}I_i/\mathrm{D}\mathscr{F}_j$. The costs for solving the adjoint or sensitivity equations are comparable, and are the most significant costs for either method. Thus, it is advantageous to use the sensitivity equation approach if m is greater than the number of parameters $\mathscr{F}_j$, while it is more efficient to use the adjoint approach if the number of parameters $\mathscr{F}_j$ is greater than m.

The case of multi objective optimisation can be reverted to a series of single objective optimisation problems by means of the weighting method [58], so this will not be considered further in this work. Moreover, the general form of the shape optimisation problem considered in this thesis, (2.1.1), contains only one performance criterion I and no con-

straints depending on the PDE solution u. Thus, $m = 1$ for the problems of interest. At the same time there can be a large number of shape parameters resulting from the shape discretisation. Therefore, for this type of problem the adjoint approach is usually the more efficient choice.

However, it has to be mentioned that for time dependent problems the adjoint solution has to be calculated backward in time, starting from the final time of the simulation. For nonlinear problems the primal solution u will be required to formulate the linearised problems at each time-step. This means that either the solution has to be stored during the forward solve, resulting in enormous storage requirements, or has to be recomputed by additional forward solves during the adjoint solve stage, resulting in prohibitive computational costs. This can be a major drawback, rendering the adjoint approach less attractive for time dependent problems. A good approach to limit the resulting problems is to store the solution at appropriately selected points in time and recompute it over small time intervals. This way the storage, as well as the computational costs, can be bounded to moderate values [46].

Finally we comment on how finite difference and automatic differentiation fit into this classification. If the derivatives $\mathrm{D}I/\mathrm{D}\mathscr{F}$ are to be approximated by finite differences, then the PDE has to be solved for perturbations of each parameter, $\widetilde{\mathscr{F}_j} = \mathscr{F}_j \pm h$, resulting in perturbed values of the solution u and performance criterion I. Even though approximations to $\partial u/\partial \mathscr{F}_j$ may not be computed explicitly, this can be classed as a sensitivity equation approach, because an equation is solved for each parameter $\mathscr{F}_j$ and the perturbed solutions are an analogue to the sensitivity $\partial u/\partial \mathscr{F}_j$.

Automatic differentiation, i.e. the differentiation of computer programs by software techniques, can be used to implement either of these classes depending on the method that is used. In automatic differentiation the so-called forward and reverse modes exist (e.g. [39]), where the forward mode corresponds to the sensitivity equation approach and the reverse mode to the adjoint equation approach.

2.2.5 Combining differentiation and discretisation

There are two fundamentally different ways of combining differentiation and discretisation: to *differentiate then discretise* (diff-disc) or to *discretise then differentiate* (disc-diff). As Gunzburger comments in [41, Section 2.9], the choice of either approach is a philosophical question, since both approaches have clear advantages and disadvantages.

First, let us define these approaches. In the *differentiate then discretise* approach the performance functional I and the PDE system (2.1.1b) and (2.1.1c) are differentiated at

the continuous level, using either of the approaches in Subsection 2.2.4. This results in a set of PDEs whose solution is used to determine the derivatives $\mathrm{D}I/\mathrm{D}\mathscr{F}$. These PDEs are then discretised and solved to obtain approximations to $\mathrm{D}I/\mathrm{D}\mathscr{F}$.

In contrast, in the *discretise then differentiate* approach the PDE system (2.1.1b) and (2.1.1c) is discretised first, resulting in a set of algebraic equations and algebraic expressions for I_h, which serves as an approximation for I. These algebraic equations, or even a computer program that solves them, are then differentiated using one of the approaches from Subsection 2.2.4. This way a set of equations is obtained which determines an approximation of $\mathrm{D}I/\mathrm{D}\mathscr{F}$.

One apparent drawback of the disc-diff variant is that not only the performance functional I but also the discretisation error $(I_h - I)$ is differentiated. Thus if the sole interest is to approximate $\mathrm{D}I/\mathrm{D}\mathscr{F}$ this approach is not necessarily the best one. However, as the resulting approximation is the derivative of I_h, this approach results in a derivative which is consistent with I_h, allowing good performance for optimisation algorithms for I_h. In contrast, if the diff-disc variant is employed, then the approximation $(\mathrm{D}I/\mathrm{D}\mathscr{F})_h$ is not necessarily the derivative of I_h. Thus, if this approximation of the gradient is used, the optimisation algorithm is likely to run into difficulties because the descent direction indicated by $(\mathrm{D}I/\mathrm{D}\mathscr{F})_h$ is not really a descent direction for I_h. Typically this leads to a slow down in progress and ultimately to a breakdown of the optimisation algorithm.

If an adjoint approach is employed, the diff-disc approach may present an advantage, because in this approach the mesh for the adjoint problem may be different to the mesh used for the forward simulation. This allows one to use different adaptive meshes to obtain good approximations for I as well as $\mathrm{D}I/\mathrm{D}\mathscr{F}$.

From a software development point of view the disc-diff approach possesses several major advantages over the diff-disc variant. A logical first step in attempting shape optimisation for a problem is to develop software that allows one to approximate I for a given geometry, which is usually done by discretising the PDE problem. Once this step is mastered one is presented with the choice of diff-disc or disc-diff. The diff-disc approach requires one to differentiate the PDE, which results in a new set of PDEs. The structure of these PDEs may be different to that of the original PDE, thus requiring one to develop a new discretisation scheme that is suitable for them. As a result one may end up writing an essentially new code for the differentiated PDEs.

In the case of the disc-diff variant it is easier to build upon the existing code. The solution strategy that is employed for the primal simulation can be adapted in a straightforward manner to solve the differentiated problem. Also, the definition of the differentiated problem builds upon the existing software, as all that is required of the additional parts is to

deliver derivatives of the functionalities found in the existing code. This situation is ideal for application of automatic differentiation software. Even if automatic differentiation can not be applied, the correspondence to the original code is an enormous advantage, as it allows verification of the developed new routines by comparing the computed derivative values to those obtained by applying finite differences to the original routines. Thus every step in the development can be verified independently.

This, along with the above observations concerning the consistency of the gradients, are the reasons that we prefer the disc-diff approach and therefore choose this approach for the remainder of this work.

Finally we remark that both the *automatic differentiation* and *finite differences* approaches from Subsection 2.2.4 are inherently of the disc-diff type.

2.2.6 Mesh deformation

All numerical approaches to shape optimisation have to deal with the different shapes of the domains in consecutive evaluations of the PDE solution and the performance criterion. As the PDE discretisations usually require a triangulation, a mesh, to describe the domain, it is necessary to provide such a mesh for every shape for which the performance criterion I is to be evaluated.

Mesh generation for general domains can be a computationally expensive task, especially for complicated three-dimensional geometries. Even worse, automatic mesh generation tools such as [77, 80] produce meshes which are not continuously dependent upon the geometry data provided, as they may for example produce meshes with differing connectivity. This discontinuity of the meshes with respect to the shape parameters translates into a discontinuity of the discrete performance function, unless this is prevented.

One approach which does not suffer from these problems is to use parametric meshes. These meshes define a fixed connectivity but variable node positions and the node positions are smooth functions of the shape parameters. Immediate drawbacks of this approach are that it is only feasible for moderately complex geometries and that it requires the user to define a suitable mesh connectivity. With such a parametric mesh approach the geometries which can be treated often have to be restricted to prevent degradation of the mesh quality, as in Section 2.7.1.1 for example.

The advantages of automatic mesh generation and parametric meshes may be combined by using an automatically generated mesh for a base geometry, and then adjusting this mesh to new geometries by deforming it. This deformation can for example be defined by solutions of the Lamé equations of linear elasticity, or modified versions thereof,

as discussed in [87] for example. Unfortunately, such deformations may decrease the mesh quality, e.g. interior angles of triangles may increase with the deformation and get close to π. For strong deformations cells may collapse or even change orientation. Even though the application of the equations of linear elasticity does not prevent such degradation of mesh quality, it guarantees that for sufficiently small changes of the boundary geometry, the degradation in the interior is small as well. Thus a neighbourhood of the base geometry may be implicitly defined in which the quality changes of the mesh are tolerable. Furthermore, to compute such a deformation may be significantly faster than generating a whole new mesh for each geometry.

A practical shape optimisation algorithm may use a hybrid approach, using mesh deformation so long as it produces meshes of tolerable quality, switching to a new base mesh as and when the deformations become too strong. This way it is guaranteed that for any given discrete shape $\mathscr{F}_\varkappa$ there exists a neighbourhood in which the discrete performance function $I_h(\mathscr{F}_\varkappa)$ is smooth. Discontinuities only occur when the base geometry changes. Whether these discontinuities present a problem for the optimisation algorithm or not depends on the local behaviour of the underlying smooth function $I(\mathscr{F}_\varkappa)$ and the size of the jump that is caused by the discontinuity.

To illustrate this, the top part of Figure 2.2 contains a hypothetical example of a smooth relationship between a parameter and the performance criterion $I(\mathscr{F}_\varkappa)$ (smooth function), a discontinuous discrete approximation $I_h(\mathscr{F}_\varkappa)$ which may result from discontinuous dependency of the mesh on the parameter (non-smooth-approx), and a piecewise smoothed approximation $I_h(\mathscr{F}_\varkappa)$ which may result from the hybrid approach as described above (deformation-smoothed). An attempted step of an optimisation algorithm using the hybrid approach may then successfully pass one of the remaining discontinuities if it yields a reduction of the piecewise smoothed performance criterion $I_h(\mathscr{F}_\varkappa)$ as in the top part of Figure 2.2, or it may fail to pass the discontinuity because the approximated performance at the new candidate solution is increased, even though for the underlying smooth performance criterion the parameter change would result in a reduction of the performance criterion as in the bottom part of Figure 2.2. A situation similar to that depicted in the bottom part of Figure 2.2 may even cause a break-down of the optimisation algorithm, if even arbitrarily small steps in the downhill direction indicated by the gradient of the smoothed function pass a jump discontinuity which increases the approximated performance function. Thus, the hybrid approach does not resolve the problem completely, but represents an improvement compared to using automatic mesh generation alone. Note that the accuracy of the results of optimising the discrete performance function is always limited by the accuracy of this discrete approximation to the continuous performance

function. The better the accuracy of the latter approximation, the less difficulties of this kind can arise.

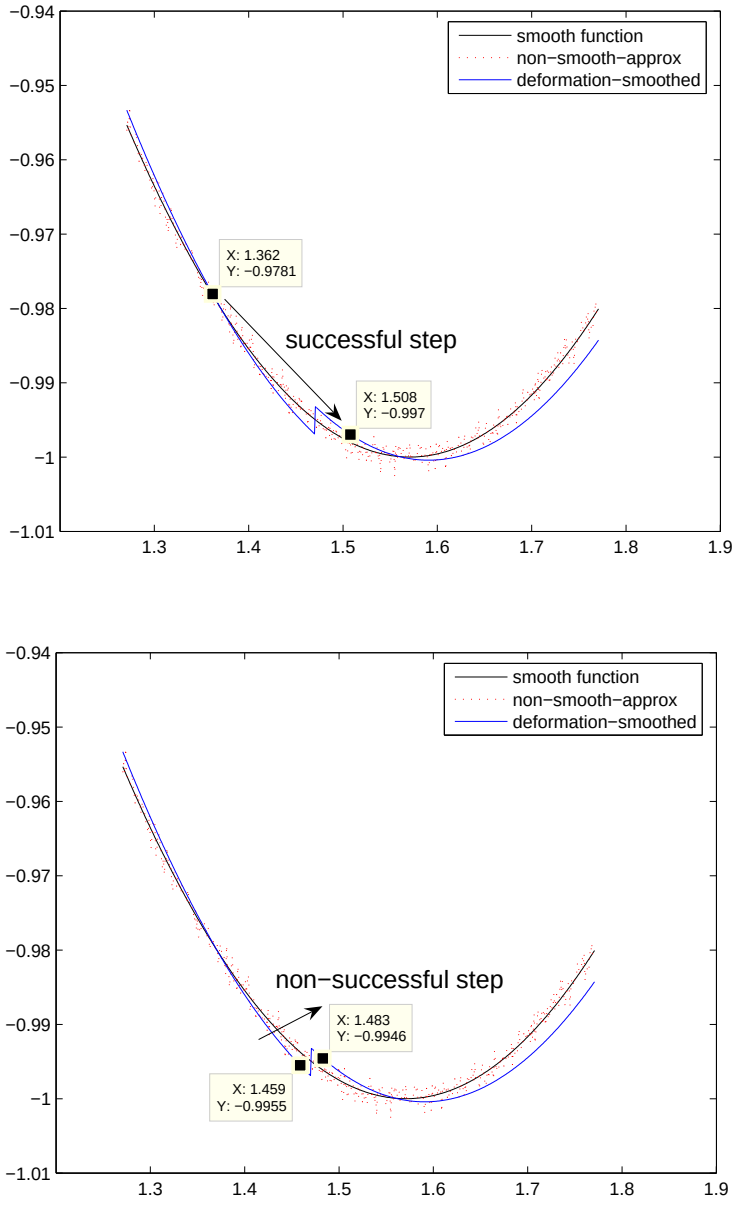


Figure 2.2: Illustration of mesh-generation induced discontinuity, a piecewise smoothed version generated by mesh-deformation and how the remaining discontinuities due to switching the base mesh may affect an optimisation algorithm

2.3 Finite elements and the incompressible Navier-Stokes equations

2.3.1 Finite element discretisation

For the purpose of this section we assume the reader to have basic knowledge of the finite element method (FEM) and the associated theory. If required, introductions can be found in many classic text books, e.g. [18, 21, 51]. Here we will give a brief introduction how finite elements (FE) may be applied to the incompressible Navier-Stokes equations, highlighting issues which are not present in simpler applications of FE, such as the Poisson equation for example. The presentation here is based upon [40] and we refer to this work for a thorough discussion of the topic.

We start the exposition by introducing a weak form of the stationary incompressible Navier-Stokes equations (2.1.4), find $(\mathbf{u}, p) \in \mathbf{H}^1_g(\Omega) \times L^2_0(\Omega)$ such that

$$a(\mathbf{u},\mathbf{v}) + c(\mathbf{u},\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) = 0 \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{2.3.1a}$$

$$b(\mathbf{u},q) = 0 \qquad \forall q \in L^2_0(\Omega). \tag{2.3.1b}$$

For the d dimensional domain Ω the bilinear forms $a(.,.)$ and $b(.,.)$ are defined as

$$a(\mathbf{u},\mathbf{v}) := \int_\Omega \nu \ \mathrm{grad}\,\mathbf{u} : \mathrm{grad}\,\mathbf{v} \,\mathrm{d}\Omega \tag{2.3.2}$$

$$:= \int_\Omega \nu \sum_{i,j=1}^{d} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j} \,\mathrm{d}\Omega, \tag{2.3.3}$$

$$b(\mathbf{u},q) := \int_\Omega q \ \mathrm{div}\,\mathbf{u} \,\mathrm{d}\Omega, \tag{2.3.4}$$

and the tri-linear form $c(.,.,.)$ as

$$c(\mathbf{w},\mathbf{u},\mathbf{v}) := \int_\Omega \mathbf{w}\cdot\mathrm{grad}\,\mathbf{u}\cdot\mathbf{v} \,\mathrm{d}\Omega \tag{2.3.5}$$

$$:= \int_\Omega \sum_{i,j=1}^{d} w_j \frac{\partial u_i}{\partial x_j} v_i \,\mathrm{d}\Omega. \tag{2.3.6}$$

A discretisation of (2.3.1) is defined in the usual way, i.e. by replacing the infinite dimensional function spaces $\mathbf{H}^1_g(\Omega)$ and $L^2_0(\Omega)$ by finite dimensional subspaces, $\mathbf{V}^h_g \subset \mathbf{H}^1_g(\Omega)$ and $S^h_0 \subset L^2_0(\Omega)$ respectively. Thus, the discrete problem is: find $(\mathbf{u}^h, p^h) \in \mathbf{V}^h_g \times S^h_0$ such

that

$$a(\mathbf{u}^h,\mathbf{v}^h)+c(\mathbf{u}^h,\mathbf{u}^h,\mathbf{v}^h)+b(\mathbf{v}^h,p^h)=0 \qquad \forall \mathbf{v}^h\in\mathbf{V}_0^h, \tag{2.3.7a}$$

$$b(\mathbf{u}^h,q^h)=0 \qquad \forall q^h\in S_0^h. \tag{2.3.7b}$$

Bases may be chosen for these finite dimensional function spaces,

$$\begin{aligned}\mathbf{V}^h &= \text{span}(\Phi^h),\\ \Phi^h &:= \left[\varphi_1^1,\dots\varphi_N^1,\varphi_1^2,\dots\varphi_N^2,\dots\varphi_N^d\right]^T, \qquad (2.3.8a)\\ S^h &= \text{span}(\Theta^h),\\ \Theta^h &:= \left[\theta_1,\dots\theta_M\right]^T, \qquad (2.3.8b)\end{aligned}$$

which, using the expressions $\mathbf{u}^h = \underline{u}^T\,\Phi^h$ and $p^h = \underline{p}^T\,\Theta^h$, allows to express (2.3.1) as a set of algebraic equations in coefficient vectors $\underline{u}$ and $\underline{p}$. The velocity basis functions in (2.3.8a) are defined such that φ_i^j denotes the i-th basis function for the j-th velocity component, i.e. a vector valued function of which all components are zero, apart from the j-th component which is a piecewise polynomial. The pressure basis functions are denoted by a different symbol, θ_i, to emphasise that a different set of basis functions may be used than for the velocity components.

Before we continue to define the element spaces used in this work, we remark that the discrete spaces $\mathbf{V}_g^h$ and S_0^h have to fulfil the so called div-stability condition or LBB condition [40],

$$\exists\gamma>0,\ \text{independent of } h: \qquad \inf_{\substack{q^h\in S_0^h\\ q^h\neq 0}}\ \sup_{\substack{\mathbf{v}^h\in\mathbf{V}_0^h\\ \mathbf{v}^h\neq 0}}\left(\frac{b(\mathbf{v}^h,q^h)}{|\mathbf{v}^h|_1\|q^h\|_0}\right)\geq\gamma,$$

to allow a stable discretisation. A number of pairs of spaces $\mathbf{V}_g^h$ and S_0^h are known to produce unstable discretisations, like for example the combination of piecewise linear velocity and piecewise linear pressure approximations on the same mesh. A further discussion of this matter, including techniques for proving div-stability, error estimates, an overview of known stable and unstable element pairs, stabilisation methods for unstable discretisations and references to the original research papers can for example be found in [40].

For the purpose of this work the div-stable Taylor-Hood element pair is employed, i.e. continuous piecewise quadratic velocity and continuous piecewise linear pressure

approximation on triangular elements. The difference in the polynomial degree reflects the differing smoothness properties of the components of the continuous solutions. Further, the P_2 approximation of the velocities gives a reasonable compromise between the favourable convergence properties of higher order discretisations and the increased costs these imply (higher bandwidth of matrices, need for higher order integration formulae).

It is well known (e.g. [51, Section 9.5] for a convection-diffusion model problem), that if the viscosity constant ν is small, the Galerkin FE solution may exhibit oscillatory behaviour uncharacteristic for the continuous solution. This numerical instability occurs if the flow is convection dominated and the mesh is not sufficiently refined to capture the short length scale diffusion effects. The problem has been well studied using convection-diffusion model problems and can be characterised by the mesh Peclet number (e.g. [68]),

$$Pe := \frac{h||u||_\infty}{2\nu},$$

i.e. if $Pe > 1$ the Galerkin FE discretisation may be unstable and oscillations may occur. For problems where viscosity is very small it may be impractical or even impossible to refine the mesh far enough to guarantee $Pe \leq 1$. To overcome this problem stabilised discretisations have been introduced, e.g. Streamline-Upwind Petrov Galerkin (SUPG) and Sub-Grid Stabilisation (SGS) techniques have been proposed as a remedy, see e.g. [47] for an overview of different approaches. These approaches use modified weak forms with improved stability properties to allow somewhat more meaningful approximations even if the mesh can not be sufficiently refined to capture all the involved effects.

The necessity for this kind of stabilisation has been questioned as it is often feasible to get good approximations without stabilisation if the mesh is well chosen, see [90] for example. Critics again (e.g. [71]) argue that this approach is not sufficiently robust as the stability may be very sensitive to the choice of the mesh, and the information necessary for the construction of such meshes (e.g. position of interior layers) may not always be available *a priori*. Therefore a combined approach is probably still a favourable choice.

However, if the viscosity constant ν is very small, or equivalently the Reynolds number is very large, other problems occur as well, i.e. stationary solutions may not be unique (bifurcation phenomena), they may not describe the behaviour well as the physical system does not approach a steady ($\partial/\partial t = 0$) state for $t \to \infty$, but oscillations remain, or indeed, steady solutions may not even exist.

For the purpose of this work we restrict ourselves to problems where sufficient viscosity is present such that the criterion $Pe \leq 1$ can be fulfilled. Thus application of standard Galerkin discretisation is possible and the stabilised discretisations are only used for

coarse grid discretisations in the context of multigrid techniques, see Subsection 2.5.2.2.

Finally, we may observe that the left-hand side of the weak form (2.3.1) is nonlinear in the velocity $\mathbf{u}$, as the term $\mathbf{u}$ appears twice as argument for the tri-linear term $c(.,.,.)$. Thus the discretisation will result in a set of nonlinear algebraic equations. We will discuss techniques to deal with this nonlinearity in the following subsection.

2.3.2 Linearisation

The canonical approach to solving nonlinear systems is to use Newton's method, which applied to (2.3.7) yields

$$\begin{aligned} &a(\mathbf{u}^h_{(k+1)},\mathbf{v}^h)+c(\mathbf{u}^h_{(k)},\mathbf{u}^h_{(k+1)},\mathbf{v}^h) \\ &+c(\mathbf{u}^h_{(k+1)},\mathbf{u}^h_{(k)},\mathbf{v}^h)+b(\mathbf{v}^h,p^h_{(k+1)}) = c(\mathbf{u}^h_{(k)},\mathbf{u}^h_{(k)},\mathbf{v}^h) \qquad \forall \mathbf{v}^h \in \mathbf{V}^h_0, \qquad (2.3.9a) \\ &b(\mathbf{u}^h_{(k+1)},q^h) = 0 \qquad \forall q^h \in S^h_0. \qquad (2.3.9b) \end{aligned}$$

Using an appropriate starting guess $\mathbf{u}^h_{(0)}$ this iterative method may be used to approximate the solution of the nonlinear system of algebraic equations. The well known properties of the Newton method also apply in this case [40, Section 6.1]: close to a solution it is quadratically convergent, and it may fail to converge if the initial guess is too far from a solution.

An obvious but interesting observation is that for the Galerkin FE discretisation of (2.3.1) the approaches "linearise then discretise" and "discretise then linearise" coincide, so long as Newton's method and the same FE function spaces are used. This allows some insight into the properties of the Jacobian as its correspondence to linear differential operators can be used.

Unfortunately, the requirement of a "good" initial guess can be quite problematic and so gives rise to the need for other approaches to solve the nonlinear system. Probably the most widespread such approach is to use Picard iteration,

$$\begin{aligned} &a(\mathbf{u}^h_{(k+1)},\mathbf{v}^h)+c(\mathbf{u}^h_{(k)},\mathbf{u}^h_{(k+1)},\mathbf{v}^h)+b(\mathbf{v}^h,p^h_{(k+1)}) = 0 \qquad \forall \mathbf{v}^h \in \mathbf{V}^h_0, \qquad (2.3.10a) \\ &b(\mathbf{u}^h_{(k+1)},q^h) = 0 \qquad \forall q^h \in S^h_0. \qquad (2.3.10b) \end{aligned}$$

Like the Newton iteration this iterative solution process requires an initial velocity guess $\mathbf{u}^h_{(0)}$ which may well be chosen to be zero everywhere apart from the Dirichlet boundary, where it should satisfy the boundary conditions for consistency. In [40, Section 6.3], where this method is referred to as the *simple iteration method*, it is stated that this method

is linearly convergent for arbitrary initial guess $\mathbf{u}^h_{(0)}$. This method may be regarded as an in-exact Newton method, since it discards some of the terms in (2.3.9), i.e. the reaction terms (zeroth order terms). Another approach to motivate this linearisation can be derived from the time-dependent problem, where it may be argued that convection of the velocities, rather than a reaction, is the natural process underlying the nonlinear convective term. Using a backward Euler time discretisation and taking the time-step size to the infinite limit yields the Picard iteration.

Hybrid solution strategies, combining the global convergence of the Picard iteration to find a "good" initial guess for the faster convergent Newton method, are of course an even better approach to solving the nonlinear system. However, for the Reynolds number range considered here, the approach of first computing as high a *Re* solution as possible on a coarse mesh using Newton linearisation, and then interpolating this solution and using it as an initial guess for a Newton iteration on the finer mesh for higher *Re*, was found to be sufficient to obtain initial guesses for the solution on each respective level. Once the desired Reynolds number can be used on the fine meshes, the interpolated solution on a finer level may even be a good enough approximation such that a single Newton iteration is enough to solve the nonlinear system to the required accuracy, as demonstrated in [22] for example.

Often (e.g. [96, Section 3.3]) Picard iteration is favoured over Newton linearisation because the resulting linear problems are better conditioned, especially for large Reynolds number (i.e. small viscosity constant ν). However, for the range of relatively small Reynolds numbers considered here, the fast convergence of the Newton method and that it is required to formulate the discrete adjoint problem make it the preferable choice, even though the involved linear systems are more demanding, as the results in Section 2.5.2 demonstrate.

Ultimately the application of the Discrete Adjoint Method requires the solution of a system of equations with the transpose of the Jacobian of the discrete system as coefficient matrix. Thus, every effort that is put into solving systems with this Jacobian matrix efficiently pays double, as the techniques can be used for the adjoint equation as well.

2.4 Finite volumes and the incompressible Navier-Stokes equations

2.4.1 Fundamental ideas of the finite volume method

The Finite Volume Method (FVM) is an approach to approximating solutions of conservation equations of the form

$$\frac{\partial \mathbf{q}}{\partial t} + \operatorname{div}(f(\mathbf{q})) = 0 \qquad \text{in} \quad [0,T] \times \Omega \tag{2.4.1}$$

numerically, where $\mathbf{q}$ is a vector of dependent variables, $f(\mathbf{q})$ a matrix valued function called the flux, which describes the quantities which are to be conserved, $[0,T]$ is the time domain and Ω is the spatial domain. The first step in deriving a Finite Volume (FV) discretisation is to integrate (2.4.1) over a so called control volume $V \subset \Omega$ and to apply the divergence theorem (requiring $f(\mathbf{q})$ to be continuously differentiable on V),

$$\begin{aligned} 0 &= \int_V \frac{\partial \mathbf{q}}{\partial t} \, \mathrm{d}V + \int_V \operatorname{div}(f(\mathbf{q})) \, \mathrm{d}V \\ &= \frac{\partial}{\partial t} \int_V \mathbf{q} \, \mathrm{d}V + \int_{\partial V} f(\mathbf{q}).n \, \mathrm{d}A. \end{aligned} \tag{2.4.2}$$

Here ∂V denotes the surface of the volume V and n the outward normal of this surface. If a sufficiently smooth solution exists, (2.4.1) is equivalent to (2.4.2) holding for all $V \subset \Omega$ and $t \in [0,T]$. The smoothness requirements of the latter formulation are not as strong as those of (2.4.1) and therefore it is called a weak formulation. In most cases from theoretical mechanics (e.g. for the Navier-Stokes equations) the conservation equation (2.4.1) is derived from the formulation (2.4.2) in the first place, so the latter is even a more natural formulation, as it does not increase smoothness requirements.

In order to discretise the equations, the domain Ω is split into a set of finite, non-overlapping, non-empty control volumes V_i, $\Omega = \bigcup_i \overline{V_i}$, and (2.4.2) is applied to each of these control volumes only, rather than all possible volumes $V \subset \Omega$. A discrete representation of the dependent functions $\mathbf{q}$ and an associated way of evaluating the terms in (2.4.2) are chosen, which complete the definition of a set of algebraic equations forming the discrete system. There are many ways these final steps can be done and different discretisation schemes emerge from the precise definition of the V_i, the data representation and the way the terms in (2.4.2) are evaluated. Naturally the properties of the variants differ, and an appropriate scheme has to be chosen considering the properties and requirements

of the application, otherwise one may easily end up with an unstable discretisation.

It shall not be the subject of this work to give an overview of the different approaches that have been developed, the interested reader may be referred to [31, 105] for example. The main subject of this work is the discrete adjoint method (DAM). Therefore only one particular FV scheme is selected here, and the application of the DAM is investigated for this scheme, in the hope to gain more insight into the DAM in general.

2.4.2 A basic finite volume scheme for incompressible Navier-Stokes

The incompressible Navier-Stokes equations (2.1.5) do not fit directly into the framework of (2.4.1), as they do not contain a temporal derivative of the pressure. This can be overcome by adding an artificial compressibility term,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla p + \operatorname{div}\left(\mathbf{u}\mathbf{u}^T\right) - \nu \operatorname{div}\ \operatorname{grad}\ \mathbf{u} = 0 \tag{2.4.3a}$$

$$\frac{\partial p}{\partial t} + \beta \operatorname{div}(\mathbf{u}) = 0, \tag{2.4.3b}$$

with the constant parameter $\beta > 0$ chosen suitably [6]. Steady state solutions of these equations satisfy the incompressibility condition $\operatorname{div}(\mathbf{u}) = 0$. However, transient solutions do not necessarily possess this property. This presents no obstacle if, as in this work, this scheme is only used to compute steady state solutions.

From this the vector of dependent variables $\mathbf{q}$ and the flux $f(\mathbf{q})$ can be identified as

$$\mathbf{q} := \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix}, \tag{2.4.4}$$

$$f_{(:,i)} := \begin{bmatrix} \delta_{i,1} p + u_1 u_i - \nu \frac{\partial u_1}{\partial x_i} \\ \delta_{i,2} p + u_2 u_i - \nu \frac{\partial u_2}{\partial x_i} \\ \vdots \\ \delta_{i,d} p + u_d u_i - \nu \frac{\partial u_d}{\partial x_i} \\ \beta u_i \end{bmatrix}, \qquad i = 1, \ldots, d, \tag{2.4.5}$$

where $\delta_{i,j}$ is the Kronecker δ. Hence, the normal flux $F(\mathbf{q},n) := f(\mathbf{q}).n$ from (2.4.2) is

$$F(\mathbf{q},n) := f(\mathbf{q}).n = \begin{bmatrix} pn_1 + u_1(\mathbf{u}.n) - \boldsymbol{\nu}\frac{\partial u_1}{\partial n} \\ pn_2 + u_2(\mathbf{u}.n) - \boldsymbol{\nu}\frac{\partial u_2}{\partial n} \\ \vdots \\ pn_d + u_d(\mathbf{u}.n) - \boldsymbol{\nu}\frac{\partial u_d}{\partial n} \\ \beta\mathbf{u}.n \end{bmatrix}. \tag{2.4.6}$$

The FV scheme that has been chosen for the purpose of this work is a first order cell centred finite volume scheme, utilising Roe's flux difference splitting approach to stabilise the discrete formulation, as in [6] and [95, pp. 333–338] for example. For further simplicity considerations are restricted to block Cartesian meshes.

The scheme is called cell centred because the discrete function values of $\mathbf{q}$ are considered to be localised at the centres of mesh cells. The integral over the boundary ∂V_i of mesh cell V_i is approximated as

$$\int_{\partial V_i} f(\mathbf{q}).n \, \mathrm{d}A \approx \sum_{j:A_j \subset \partial V_i} F_j |A_j|, \tag{2.4.7}$$

where the discrete flux F_j is an approximation to the normal flux $F_j \approx F(\mathbf{q},n)$ on the cell interface A_j (edge in two dimensions, face in three). The evaluation of (2.4.2) is performed in a cell interface oriented way. The discrete flux F_j is evaluated for each cell interface, and the contribution $F_j|A_j|$ is added to the residual vector for both adjacent cells with opposite sign, of course, as the outward normal has opposing directions for the two neighbouring cells.

For the purpose of defining the discrete flux F_j, both it and $F(\mathbf{q},n)$ are split into

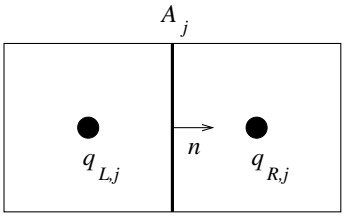


Figure 2.3: Notation for the definition of the discrete flux F_j at interface A_j

viscous and inviscid terms,

$$F(\mathbf{q},n) = F_{\mathrm{v}}(\mathbf{q},n) + F_{\mathrm{i}}(\mathbf{q},n),$$
$$F_j = F_{j,\mathrm{v}} + F_{j,\mathrm{i}}\ ,$$
$$F_{j,\mathrm{v}} \approx F_{\mathrm{v}}(\mathbf{q},n) := \begin{bmatrix} -\boldsymbol{\nu}\frac{\partial u_1}{\partial n} \\ -\boldsymbol{\nu}\frac{\partial u_2}{\partial n} \\ \vdots \\ -\boldsymbol{\nu}\frac{\partial u_d}{\partial n} \\ 0 \end{bmatrix}, \tag{2.4.8}$$
$$F_{j,\mathrm{i}} \approx F_{\mathrm{i}}(\mathbf{q},n) := \begin{bmatrix} pn_1 + u_1(\mathbf{u}.n) \\ pn_2 + u_2(\mathbf{u}.n) \\ \vdots \\ pn_d + u_d(\mathbf{u}.n) \\ \beta\mathbf{u}.n \end{bmatrix}, \tag{2.4.9}$$

which are treated separately due to their fundamentally different character. An important observation is that the normal flux $F(\mathbf{q},n)$ has to be approximated on the cell interface while the state variables $\mathbf{q}$ are considered located at the cell centres. Let $\mathbf{q}_{L,j}$ and $\mathbf{q}_{R,j}$ denote the states in the cells left and right of interface A_j respectively, see Figure 2.3. The normal derivatives of the viscous flux (2.4.8) can be approximated on the interface by using the central difference

$$\frac{\partial \mathbf{u}}{\partial n} \approx \frac{\mathbf{u}_{R,j} - \mathbf{u}_{L,j}}{h_j}, \tag{2.4.10}$$

where h_j denotes the distance of the cell centres of the left and right cells. If the interface is exactly half-way between the centres of the adjacent cells, this results in a second order consistent approximation, otherwise only first order is achieved.

Definition of the discrete inviscid flux is more demanding. A straightforward ap-

proach to approximate $F(\mathbf{q},n)$ at the interface would be to use the arithmetic mean of left and right cell evaluations, $1/2(F(\mathbf{q}_{L,j},n)+F(\mathbf{q}_{L,j},n))$. Unfortunately this would result in an unstable discretisation (in a similar way as central difference schemes yield an unstable discretisation of convection equations, see [31, Section 4.4.2]). Stabilisation may be achieved using Roe's approach, which modifies the arithmetic mean by an upwinding term

$$F_{j,\mathrm{i}} := \frac{1}{2}\left(F(\mathbf{q}_{L,j},n)+F(\mathbf{q}_{R,j},n)\right) - \frac{1}{2}|J|(\mathbf{q}_{R,j}-\mathbf{q}_{L,j}),$$

with

$$|J| := R|\Lambda|L.$$

Here $|\Lambda|$ is the diagonal matrix with the absolute values of the eigenvalues of the Jacobian $\partial F(\mathbf{q},n)/\partial\mathbf{q}$, and R and L are defined by diagonalising this Jacobian,

$$R\,\Lambda\,L = \frac{\partial F(\mathbf{q},n)}{\partial \mathbf{q}}.$$

The authors in [95] provide an interpretation of this spatial discretisation scheme in terms of equivalent differential operators on a given two dimensional grid, i.e. the stabilised (nonlinear) operator becomes

$$\begin{bmatrix} -\boldsymbol{\nu}\Delta+2u\partial_x+v\partial_y & u\partial_y & \partial_x \\ -\frac{h}{2}(|v|\partial_{yy}+\frac{2u^2+\beta}{\sqrt{u^2+\beta}}\partial_{xx}) & -\frac{h}{2}\frac{uv}{v^2+\beta}(2\sqrt{v^2+\beta}-|v|)\partial_{yy} & -\frac{h}{2}u(\frac{1}{\sqrt{u^2+\beta}}\partial_{xx}+\frac{\sqrt{v^2+\beta}-|v|}{v^2+\beta}\partial_{yy}) \\ v\partial_x & -\boldsymbol{\nu}\Delta+u\partial_x+2v\partial_y & \partial_y \\ -\frac{h}{2}\frac{uv}{u^2+\beta}(2\sqrt{u^2+\beta}-|u|)\partial_{xx} & -\frac{h}{2}(|u|\partial_{xx}+\frac{2v^2+\beta}{\sqrt{v^2+\beta}}\partial_{yy}) & -\frac{h}{2}v(\frac{1}{\sqrt{v^2+\beta}}\partial_{yy}+\frac{\sqrt{u^2+\beta}-|u|}{u^2+\beta}\partial_{xx}) \\ \beta\partial_x & \beta\partial_y & \\ -\frac{h}{2}\frac{\beta u}{\sqrt{u^2+\beta}}\partial_{xx} & -\frac{h}{2}\frac{\beta v}{\sqrt{v^2+\beta}}\partial_{yy} & -\frac{h}{2}\beta(\frac{1}{\sqrt{u^2+\beta}}\partial_{xx}+\frac{1}{\sqrt{v^2+\beta}}\partial_{yy}) \end{bmatrix}. \tag{2.4.11}$$

The operator is written like a matrix that would be multiplied from the right by the vector of flow variables $\mathbf{q} := [u,v,p]^T$ and the entries of the matrix are each written across two lines, the first line forming the original operators from the Navier-Stokes equations and the second denoting the stabilisation terms. As all the stabilisation terms are of order $\mathscr{O}(h)$, the overall spatial discretisation is only first order accurate.

The boundary conditions are also implemented in different ways for the inviscid and viscous fluxes. The inviscid fluxes require the definition of the velocities and pressure on

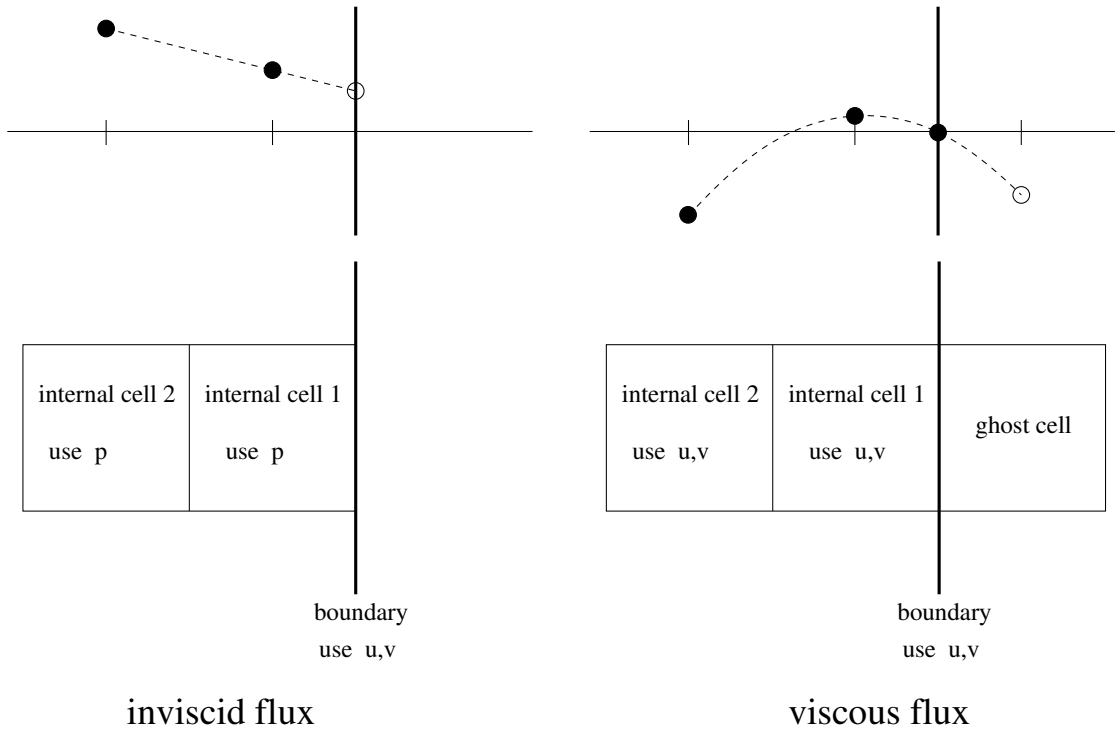


Figure 2.4: Definition of the ghost cell data

the boundary. The velocities are prescribed by the boundary conditions while the pressure is extrapolated linearly from the two adjacent inner cells, see the left-hand side of Figure 2.4 for an illustration. This procedure guarantees that the total mass flux through the boundary of the domain is exactly that described by the boundary conditions, i.e. if the boundary conditions dictate balancing in and out flux then the net mass flux through the boundary will be zero. The viscous flux requires the normal derivatives of the velocities on the boundary. This has been implemented using "ghost" cells, as illustrated in the right-hand side of Figure 2.4. Utilising the velocity values at the two adjacent inner cells and the values on the boundary, quadratic extrapolation is used to define values at the "ghost" cell. A central difference for the internal and extrapolated "ghost" cell values is then used as the normal derivative, reusing the code for the internal cells. This approach reproduces the solution for channel flow exactly.

This completes the definition of the spatial discretisation. Defining the vector of discrete states ω and the vector of discrete residuals $R(\omega)$, this yields

$$M\frac{\partial \omega}{\partial t} + R(\omega) = 0,$$

where M is a diagonal matrix with the cell volumes as entries. Temporal discretisation

can for example be done using the explicit Euler scheme,

$$M \frac{\omega^{(k+1)} - \omega^{(k)}}{t^{(k+1)} - t^{(k)}} + R(\omega^{(k)}) = 0,$$

which implies CFL-type restrictions on the time-step size $\tau := t^{(k+1)} - t^{(k)}$ (e.g. [31, Section 6.3.1]). As the focus here is on steady solutions, the implicit Euler scheme

$$M \frac{\omega^{(k+1)} - \omega^{(k)}}{t^{(k+1)} - t^{(k)}} + R(\omega^{(k+1)}) = 0, \tag{2.4.12}$$

is advantageous, since it allows larger time-steps and thus allows one to arrive at a steady state in fewer (but more expensive) time-steps.

2.4.3 Properties of the scheme

A property of the artificial compressibility approach as a whole is that it may allow for a pseudo stationary state, where a constant mass production in the domain might be present. If $\nabla.\mathbf{u} = \text{const}$ holds, it follows $\partial p/\partial t = \text{const}$. But if the pressure changes at a constant rate throughout the whole domain, ∇p does not change. Thus the momentum equations may be stationary while only a constant mass production rate is present. However, if the boundary conditions are conforming, in the sense that the integral over the normal velocity on the boundary is zero (balancing of in- and out-flux), this behaviour can not occur, since this implies that the only such constant mass production is "constant zero". It is desirable that this property is reflected by the discrete system. For this the correct implementation of the boundary conditions is crucial, e.g. if the ghost-cell approach were to be used for the velocities in both the viscous and inviscid fluxes, the up-winding of the Roe scheme would destroy this mass conservation property. But if the inviscid flux on the boundary is evaluated using the prescribed velocities directly, as described in Section 2.4.2, and the discrete integral over the boundary mass flux is zero, the discrete scheme can also only converge to constant mass production if this mass production is zero.

The incompressible Navier-Stokes equations define the pressure only up to an additive constant. The FV scheme from Section 2.4.2 shows this property as well in the form

$$R(\omega) = R(\omega + \alpha e_p)$$

for any vector ω of discrete flow variables and any constant α, while e_p denotes a vector containing zeros for the velocity variables and ones for the pressure variables. Thus the solution of $R(\omega) = 0$ can only be well defined up to an additive constant pressure. If no

attention is paid to this, inaccuracies can build up in the iterative solution procedure which can potentially result in very large constant pressure components with the adverse effects these imply in finite precision arithmetic. To avoid this, the pressure is normalised after each time-step by subtracting the mean pressure, $p := p - e(w^T p)$, where e is a vector of all ones, and w is a weight vector consisting of the cell volumes, scaled such that the sum of the weights is one, i.e. $w_i := \mathrm{d}V_i/V$, which implies $w^T e = 1$. This is equivalent to projecting the pressure such that the integral of the pressure over the whole domain is zero. Applying the projection step at each solution update can be seen as to seek a solution ω to

$$0 = \begin{bmatrix} R(\omega) \\ w_p^T \omega \end{bmatrix},$$

where w_p denotes a vector containing zeros for the velocity variables and the weight from w for the pressure variables. As this is a system of $N+1$ equations in N unknowns, it is not yet quite complete. However, by defining an extended state vector $\widetilde{\omega} := (\omega, \lambda)$, a regularised discrete system may be defined as

$$0 = \widetilde{R}(\widetilde{\omega}) := \begin{bmatrix} R(\omega) + \lambda w_p \\ w_p^T \omega \end{bmatrix}. \tag{2.4.13}$$

The additional term λw_p ensures that the system $\widetilde{R}(\widetilde{\omega}) = f$ is solvable for arbitrary right-hand side, removing the irregularity resulting from the "constant mass production" issues discussed at the beginning of this section. Using Matlab, it has been confirmed that the Jacobian $\widetilde{J}$ of $\widetilde{R}$,

$$\widetilde{J} := \left[\frac{\partial \widetilde{R}}{\partial \widetilde{\omega}}\right] = \begin{bmatrix} J & w_p \\ w_p^T & 0 \end{bmatrix}, \qquad J := \left[\frac{\partial R}{\partial \omega}\right],$$

is non-singular at the evaluated solution, thus this solution is (at least) locally unique.

2.4.4 Some properties of the Jacobian $\partial R/\partial \omega$

If the discrete variables are ordered as

$$\omega = [u_1, u_2, \ldots, u_N, v_1, \ldots, v_N, p_1, \ldots, p_N],$$

and the equations are ordered in an analogous way, the Jacobian of the scheme can be written in the block structure

$$J := \frac{\partial R}{\partial \omega} = \begin{bmatrix} F & B_T \\ B & C \end{bmatrix}. \tag{2.4.14}$$

Looking at the scheme's representation in (2.4.11) it is obvious that the B_T and C blocks map constant pressure vectors to zero, and therefore, that the resulting system has a (right) eigenvector of the form $e_p = [0, e^T]^T$ to the eigenvalue $\lambda = 0$,

$$\begin{bmatrix} F & B_T \\ B & C \end{bmatrix} \begin{bmatrix} 0 \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.4.15}$$

This singularity is the discrete analogue to the non-uniqueness of the static pressure p for the steady state of the continuous system. The stabilisation terms in (2.4.11) imply that $B_T \neq B^T$ for the Jacobian of this scheme, which allows for the corresponding left eigenvector to be different. However, this left eigenvector has been evaluated for different discrete problems using Matlab, and it was found to be identical to the right eigenvector. This is somewhat surprising as the constant mass production issues discussed at the beginning of Section 2.4.3 would point toward an eigenvector of the structure w_p rather than e_p, which is observed.

To explain why one would expect right eigenvectors of the form w_p we observe that the left eigenvector to eigenvalue $\lambda = 0$ has to be orthogonal to the image of J. Thus it has to be a vector that is not obtainable as product Jy for any $y \in \mathbb{R}^N$. Looking at the differential operator defining the B block, we know that $div(\mathbf{u}) = c = const$ is only possible for $c = 0$ due to definition of the boundary conditions. Therefore the integral over a constant should not be in the image of the pressure space residual, i.e. w_p should be left eigenvector to eigenvalue $\lambda = 0$.

It is common practice in FV solvers to avoid the problems resulting from this zero eigenvalue simply by using finite time-steps of length $\tau_k = t^{(k+1)} - t^{(k)}$, even when a steady-state solution is sought. For backward Euler time discretisation and uniform spatial meshes this is equivalent to adding $(1/\tau_k)|V_h|$ times the identity matrix to the Jacobian, which shifts the spectrum and thereby makes the resulting system non-singular (for an appropriate time-step τ_k). This in itself presents a dilemma. If the τ_k is chosen to be very small, many time-steps will be needed before the solution approaches a steady state. On the other hand, if the τ_k is chosen very large then $1/\tau_k$ will be very small. Therefore, for $\tau_k \to \infty$ the properties of the system deteriorate towards the singular behaviour that is seen when no time terms are added. Note that the absence of the time terms can also be interpreted as $\tau_k = \infty$.

Algorithm 1 Solution Procedure

$\omega := 0$ {initialisation}
$\lambda := -\frac{w_p^T R(\omega)}{w_p^T w_p}$ {initial evaluation of λ and residual}
$r := R(\omega) + \lambda w_p$ {residual of (2.4.13)}
while $||r|| > \varepsilon$ **do**
  solve $(J+D)x = r$ {compute one Newton step as update to solution}
  $\omega := \omega - x$ {apply time-step}
  $\omega := \omega - e_p(w_p^T \omega)$ {make sure $w_p^T \omega = 0$}
  $\lambda := -\frac{w_p^T R(\omega)}{w_p^T w_p}$ {update λ and residual of (2.4.13)}
  $r := R(\omega) + \lambda w_p$
end while

2.4.5 A solution procedure

The actual computation of the steady-state solution is performed using a variant of a backward Euler time advancement formula (2.4.12), where at each time-step the first iterate of Newton's method for solving the resulting nonlinear system is used as solution at the end of the time-step. This procedure is of course not time accurate, but it allows one to arrive at a steady solution quickly. Note that if the system converges to a steady solution, this does fulfil (2.4.13).

The time advancement is performed according to Algorithm 1, where J is the Jacobian of $R(\omega)$ and the diagonal matrix D contains the cell volumes over the fixed time-step size $D := (1/\tau_k)M$. Again, note that J is the Jacobian of the original FV scheme R rather than the regularised $\widetilde{R}$. Use of the finite time-step ensures the system matrix is non-singular. To solve the equations system $(J+D)x = r$ we recommend to use a multigrid preconditioned GMRES algorithm, see Section 2.5.3.2. The regularisation and the adjusting of the mean pressure are explicit steps of the algorithm.

2.5 Efficient solution strategies for the linearised systems

Now that we have introduced two discretisations for the incompressible Navier-Stokes equations, one of finite element and one of finite volume type, the question remains how to solve the resulting linearised systems of equations efficiently. Typically the number of unknowns for these systems is very large, in the range from tens of thousands to tens of millions. The upper limit is only implied by the available computer systems and efficiency of the methods used to solve the systems. The bigger the system that can be solved, the more accurate an approximation to the continuous solution can be computed. For fairly simple problems a few hundred degrees of freedom may well be enough to get reasonable results. But for complex three dimensional geometries it is certainly the case that computing resources are limiting the accuracy which can be achieved.

We remark that the question of efficient solution strategies is of special importance if optimisation is to be performed, since this will invariably require repeated evaluation of the quantities of interest, irrespective of the quality of the optimisation approach. Efficient optimisation methods may require significantly fewer evaluations than other techniques, but generally a number of evaluations significantly larger than one will still be required!

The outline of this section is as follows. In Subsection 2.5.1 efficient solution strategies are briefly discussed in general, giving examples of techniques for the case of FEM discretisation of the Poisson equation, and highlighting differences and challenges posed by the Navier-Stokes systems. Two solution approaches for the Navier-Stokes systems which have been proposed in the literature, the F_p preconditioner (e.g. [29]) and box-smoother multigrid (e.g. [95]), are then applied to the FEM discretisation first, Subsection 2.5.2, and to the FV discretisation second, Subsection 2.5.3.

2.5.1 Introduction

As already stated above the size of the systems of equations resulting from PDE discretisation can be arbitrarily large. For meshes defined by uniform refinement of a given coarse mesh, the number of degrees of freedom (DOFs) N grows like $N \sim h^{-d}$, where d is again the spatial dimension and h the discretisation length scale. The more accurate a solution to the PDE is required, the smaller h has to be chosen, the larger the size of the linear systems N. Fortunately, discretisation by finite elements or finite volumes results in sparse linear systems. That means that most of the entries of the system matrix are zero, apart from a few entries in each row or column. As the number of non-zero entries per row is bounded, with a bound depending on the discretisation method and the base mesh, the memory requirements for storing the system matrices are $\mathcal{O}(N)$, rather than N^2 in the

general case of dense matrices.

Due to the size of the matrices standard factorisation approaches like LU-decomposition by Gaußian elimination are computationally very expensive, as they require $\mathcal{O}(N^3)$ operations. Further, they destroy the sparsity of the matrices (fill-in) resulting in large memory requirements[73]. An improvement can be achieved if the sparsity of the matrix is exploited and (partially) preserved, giving rise to sparse direct solvers like SuperLU [26]. However, while these techniques improve on the complexity of non-sparse direct solvers, their asymptotic memory requirements and computational costs are still sub-optimal. Krylov subspace solution techniques, for example conjugate gradients (CG) [73, Section 6.7] for symmetric positive matrices or the generalised minimal residual (GMRES) method [73, Section 6.5], for the general case, provide an alternative which does not require the computationally expensive factorisation of the matrices. In the k-th step these iterative methods approximate the solution x of a linear system $Ax = b$ as its projection on the Krylov subspace

$$K_k(A, r_0) = span(r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0)$$

where $r_0 := Ax_0 - b$ is the residual for the initial guess x_0 of the solution. The individual methods arise by different ways in which the projection on the subspace is defined. CG uses a projection orthogonal to $K_k(A, r_0)$ while GMRES uses projection orthogonal to $AK_k(A, r_0)$. For reference we provide basic versions of the preconditioned variants of these algorithms as Algorithm 2 (CG in the notation of [57]) and Algorithm 3 (GMRES[4]).

Both algorithms require one matrix-vector product Ax per iteration, which, due to the sparsity of A, can be implemented in $\mathcal{O}(N)$ operations. To define the projection, CG requires a constant number of vector-vector operations as well, whose overall computational cost is $\mathcal{O}(N)$. The symmetry of the matrix A is used in CG to define a sequence of orthogonal search directions without storing more than one of these directions at any time. A similar recurrence definition of the search directions is not known in the general case, so GMRES builds and stores a basis of orthogonal search directions explicitly. Each new search direction has to be orthogonalised to all previous search directions. Thus, its memory requirements as well as the required operations per iteration grow with each it-

[4]Note that for simplicity of presentation this version does not check the norm of the residual at each step. A real implementation specifies a maximal number of steps m, computes the Hessenberg matrix H and the orthogonal search directions v_i according to Algorithm 3 and, at each step, applies rotations to transform the Hessenberg matrix H into an upper triangular matrix R and βe_1 into g_j. The resulting triangular least squares problem $\min(\|g_j - Ry_j\|)$ provides the residual after the j-th step directly and, when a stopping criterion is fulfilled and the solution approximation $x^{(m)}$ has to be computed, allows easy computation of y_m. See [73, Subsection 6.5.3] for a detailed discussion of these implementation issues.

Algorithm 2 Conjugate Gradients (CG)

Input: $x = x^{(0)}$ {initial guess}
$r := Ax - b$ {compute initial residual}
$w := C^{-1}r$ {initial application of the preconditioner}
$q := w$ {initial search direction}
$\gamma := w^T w$
for $i = 1, 2, \ldots$ **do**
  $z := Aq$ {multiplication with matrix}
  $p := C^{-1}z$ {application of the preconditioner}
  $\delta := p^T q$
  $\alpha := -\gamma/\delta$
  $x := x + \alpha q$ {update solution}
  $w := r + \alpha p$ {update preconditioned residual}
  $\gamma_{new} := w^T w$
  $\beta := \gamma_{new}/\gamma$
  $q := w + \beta q$ {update search direction}
  check convergence ($\|w\| < \ldots$), stop if satisfied
  replace $\gamma := \gamma_{new}$
end for

eration. The overall cost for m iterations of GMRES is $\mathcal{O}(m^2 N)$ in contrast to $\mathcal{O}(mN)$ for m iterations of CG. So, if m, the number of iterations to achieve the required accuracy[5], can be kept small in relation to N, then these solvers are far more efficient than standard factorisation approaches.

At this point the condition number of the matrices becomes important. The well known result for CG (e.g. [73, Section 6.11.3])

$$\|x_* - x_m\|_A \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^m \|x_* - x_0\|_A \tag{2.5.1}$$

relates the A-norm of the error of the m-th iterate $x_* - x_m$ to that of the initial error $x_* - x_0$. The error bound (2.5.1) shows exponential convergence with respect to m, with a constant that depends on the condition number κ, which is defined as the ratio of the largest to the smallest eigenvalue of A. Thus, the number of iterations required to reduce the error to a given fraction of the initial error is dependent on κ. Unfortunately, for the PDE discretisations which are of interest here, the condition number κ grows as the discretisation is refined, e.g. for the Poisson equation it is well known (e.g. [88, Section 5.2] or [19, Kapitel IV, §2]) that $\kappa \sim \mathcal{O}(h^{-2})$. Thus, as h is reduced, not only is the size N of the equation

[5] As the equation systems arise in the discretisation of PDEs, which implies a certain discretisation error, solution of the linear equation systems to the same order of accuracy as the discretisation error is sufficient to maintain the convergence order of the PDE discretisation.

Algorithm 3 *m* steps of the General Minimal Residual method (GMRES)

Input: $x^{(0)}$ {initial guess}
$r := Ax^{(0)} - b$ {compute initial residual}
$w := C^{-1}r$ {initial application of the preconditioner}
$\beta := \|w\|$ {initial residual norm}
$v_1 := w/\beta$ {first search direction}
define $H \in \mathbb{R}^{m+1,m}$, set $H := 0$
for $j = 1, 2, \ldots, m$ **do**
  $z := Aq$ {multiplication with matrix}
  $w := C^{-1}z$ {application of the preconditioner}
  for $i = 1, 2, \ldots, j$ **do**
    $h_{i,j} := w^T v_i$
    $w := w - h_{i,j} v_i$ {Gram-Schmidt orthogonalisation}
  end for
  $h_{j+1,j} := \sqrt{w^T w}$
  if $h_{j+1,j} = 0$ **then**
    set $m := j$ (reduces the size of H),
    compute y_m the minimiser of $\|\beta e_1 - Hy\|_2$,
    $x^{(m)} := x^{(0)} + [v_1, \ldots, v_m] y_m$ {least squares solution in Krylov subspace}
    stop
  end if
  $v_{j+1} := w/h_{j+1,j}$ {new search direction}
end for
compute y_m the minimiser of $\|\beta e_1 - Hy\|_2$,
$x^{(m)} := x^{(0)} + [v_1, \ldots, v_m] y_m$ {least squares solution in Krylov subspace}

system increased, but the number of iterations required to achieve a given reduction of the error increases too.

This effect of ill-conditioning can be averted by appropriate preconditioning. In the case of the FEM discretised Poisson equation, for example, very successful solution strategies have been developed. One such approach is to combine CG with the BPX preconditioner [20], which relies on hierarchical meshes. The condition numbers of the preconditioned system $C^{-1}Ax = C^{-1}b$ are of order $\mathcal{O}(1)$, i.e. independent of h. Such an approach is of course only useful if the preconditioner C^{-1} itself can be implemented in an efficient way. For the BPX preconditioner this is the case and the overall cost for solving the systems is $\mathcal{O}(N)$, which is optimal.

Geometric multigrid (GMG) or algebraic multigrid (AMG) techniques (e.g. [95] for an introduction) have been proven to provide optimal ($\mathcal{O}(N)$) solution strategies for the Poisson equation as well. These techniques do not only use the equation system as it is given on the current mesh, but a sequence of equation systems, corresponding to different refinement levels. In the case of GMG this sequence of equation systems is generated by building the system matrices for a hierarchical sequence of increasingly fine meshes, while in the case of AMG the coarse versions of the equation system are constructed from the system matrix A itself. In both cases simple iterative methods, like Gauß-Seidel for example, are used to reduce the highest frequency components of the solution error on the finest mesh (smoothing). The remaining residual is then restricted to a coarse-grid representation, which is used to obtain an improved approximation of the low frequency components of the solution. This coarse grid solution is then interpolated onto the fine mesh and used to update the fine solution approximation (coarse grid correction). A further smoothing step is then usually used to reduce high frequency error components introduced by the coarse grid correction. This basic process can either be repeated until the residual of the equation system becomes sufficiently small, i.e. using multigrid as a solver, or it can be used as preconditioner inside a Krylov subspace solver, like CG for example. The latter variant, using multigrid as a preconditioner in a Krylov subspace solver, is found to be more robust [95, Section 7.8].

The improved approximation of the low frequency components of the solution on the coarse mesh can be obtained the same way, i.e. pre-smoothing, coarse grid correction (if there is a coarser grid), post-smoothing, because the highest frequency components on the coarse grid have lower frequency than those on the fine mesh. Often the equation systems on the coarsest grid are solved by direct solvers. Different variants of the basic multigrid idea emerge as the operators for the smoothing, interpolation, and restriction are defined, and by choosing the sequence and the number of times the pre-smoothing, coarse grid

correction and post-smoothing steps are applied.

In addition to the ill-conditioning known from discretisation of the Poisson equation, the system matrices of the problems under consideration, the discretisations of the incompressible Navier-Stokes equations, are non-symmetric and indefinite. Thus, CG can not be applied in this case, but the more expensive GMRES can instead be used. Standard multigrid approaches fail due to their inability to resolve the velocity-pressure coupling.

These systems appear to be much more challenging than those in the case of the Poisson equation. The search for optimal solution techniques for these problems has been an active research area for several years and a large number of approaches have been reported with varying success, e.g. [5, 27, 28, 29, 52, 64, 96, 102, 103]. It would be far beyond the scope of this work to consider them all, and many of them would not be applicable to the discretisation schemes under consideration, as often the discretisation and solution strategy are tightly related.

Instead the investigation in the remainder of this section concentrates on two approaches which may be considered good candidates for a universal approach, the F_p preconditioner in conjunction with GMRES (e.g. [29, 52, 102, 103]) and box-smoother multigrid (e.g. [95]). These techniques are applied to the two discretisations at hand. It turns out that each appears to work well for one discretisation, but less so, or even not at all for the other. In one case optimal behaviour is seen for a fixed Reynolds number, but the behaviour deteriorates badly with increasing Reynolds number. To this date a universally optimal solution approach for the linearised Navier-Stokes systems remains unknown to the author.

At this point it should be noted that the very common preconditioning approach of incomplete triangularisation (ILU) has not been considered here, as it is known that this approach does not perform optimally for the Poisson case [42][6]. Even though this approach may work well in certain situations, the lack of efficiency in the Poisson case implies that this approach is not a candidate for an universal approach for the systems considered here.

[6]Gustafsson showed that his modified incomplete factorisations obtain condition numbers of order $o(h^{-1})$ while he states that other approaches yield $o(h^{-2})$ only. To obtain optimal convergence $o(1)$ condition numbers are necessary.

2.5.2 Finite Element Method

2.5.2.1 The F_p preconditioner

If the discrete variables from the linearised FE discretisation are ordered such that all the velocity variables come before the pressure variables, and the equations are ordered in an analogous way, then the system matrices K for (2.3.9) and (2.3.10) have the block structure

$$K = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}, \tag{2.5.2}$$

where the F block results from the $a(.,.)$ and $c(.,.,.)$ terms while the B and B^T blocks arise from the $b(.,.)$ terms of the linearisations of weak form (2.3.7). For discretisations which are not inherently div-stable but stabilised in some way, the pressure-pressure block of K might be non-zero as well. But as the div-stable Taylor-Hood element pair is used in this work, such stabilisation is not necessary here.

In [29, 52, 101] a right preconditioner is presented based upon approximating the inverse of the block triangular matrix

$$\widetilde{C}_R := \begin{bmatrix} F & B^T \\ 0 & -S \end{bmatrix},$$

where S denotes the Schur complement

$$S := BF^{-1}B^T. \tag{2.5.3}$$

In order to derive an analogous *left* preconditioner, the same approach is applied to the matrix

$$\widetilde{C}_L := \begin{bmatrix} F & 0 \\ B & -S \end{bmatrix}. \tag{2.5.4}$$

The inverse of $\widetilde{C}_L$ is

$$\begin{aligned} \widetilde{C}_L^{-1} &= \begin{bmatrix} F^{-1} & 0 \\ S^{-1}BF^{-1} & -S^{-1} \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ 0 & -S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}, \end{aligned} \tag{2.5.5}$$

where for computations the latter factorisation is advantageous as it demonstrates the three

major steps necessary to apply this inverse to a vector: solve with F, multiply by B and finally solve with $-S$.

The most important ingredient of the F_p preconditioning technique is to use an approximation $X^{-1} \approx -S^{-1}$ instead of the inverse of S. Solving equation systems with S would be very expensive, as the definition of S already contains the inverse of F. Thus, if iterative solvers like GMRES where used to solve with S, for each multiplication by S a system with F would have to be solved. The particular approximation X^{-1} used in the F_p preconditioner is motivated by the following argument: S is a discretisation of the differential operators

$$-S \quad \sim \quad -\nabla.(-\nu\Delta + \mathbf{u}.\nabla + \sigma(\nabla\mathbf{u}))^{-1}\nabla,$$

where the constant σ is used to distinguish between the F operator arising from Newton linearisation $\sigma = 1$ and that from Picard iteration $\sigma = 0$. Now, if for the purpose of motivation it is assumed that the operators are commutative, and if the dimensions of the various arguments are ignored, the expression

$$-\Delta(-\nu\Delta + \mathbf{u}.\nabla + \sigma(\nabla\mathbf{u}))^{-1} \tag{2.5.6}$$

can be derived. This would suggest to use the inverse of a Laplacian A_p and multiplication by an F-like discrete operator F_p as an approximation to $-S^{-1}$,

$$X^{-1} := M_p^{-1} F_p A_p^{-1}. \tag{2.5.7}$$

Here M_p is a mass matrix (discretisation of the identity operator) which is added to improve scaling. The subscript $*_p$ indicates that these operators are to be applied to pressure space vectors, respectively functions from S_0^h. A rigorous analysis of the resulting preconditioner can be found in [29, 52, 101]. Here we focus on practical issues related to the application of this approach.

Let us draw attention to the fact that the preconditioner, as it is written in (2.5.5) and (2.5.7), contains the inverse matrices of F, A_p and M_p. Of course computing these inverse matrices would be a computationally very expensive task, and would render the approach similarly expensive as direct solution methods for the whole matrix K. However, as this preconditioner is to be used within a Krylov subspace solver, it is sufficient to supply routines which apply the preconditioner to a given vector. Therefore it suffices to be able to solve equation systems with each of the matrices F, A_p and M_p, representing the action of the inverse matrices. Such solves do not necessarily have to be very accurate,

and a preconditioned Krylov subspace solver may be used to approximate the solutions to a specified tolerance. Of course the efficiency of the outer solution technique, for the linearised Navier-Stokes problems, strongly depends on the efficiency of the methods used to solve the subproblems which arise in the preconditioner. That is, an optimal solver can only be achieved if the subproblem solvers are optimal.

Very efficient methods for solving equation systems arising from discretisations of the Laplacian operator are available, as already discussed in Subsection 2.5.1. The implementation in this work uses CG with the BPX preconditioner to solve the systems with A_p. This results in computational costs linear in the number of unknowns, and thus is considered an optimal solver for this subproblem. The subproblems with the mass matrix require even less effort, as it is known [104] that the condition number of $diag(M_p)^{-1}M_p$ is mesh independent. Therefore applying CG preconditioned by the inverse of the diagonal of M_p to solve these problems results in an optimal solver. However, the remaining subproblem, application of F^{-1} turns out to be more challenging. Detailed discussion of this part is left to Subsection 2.5.2.2. For now let us assume that a good preconditioner for these systems is available such that GMRES with this preconditioner allows robust, efficient solution of equation systems with F.

When iterative solution techniques are used for the inner solves, to approximate the application of the inverse operators, F^{-1}, A_p^{-1}, M_p^{-1}, care has to be taken that the resulting outer preconditioner (2.5.5) remains a linear operator. If, for example, a fixed relative reduction of the initial residual of the inner equation systems is used as a stopping criterion in the inner GMRES or CG solvers, this will generally result in a different number of iterations required to fulfil this criterion, depending on the right-hand side to which the algorithms are applied. These different iteration numbers imply that slightly different approximations of the inverse matrices are used in each call of the outer preconditioner, i.e. 20 steps of GMRES will result in a better approximation of F^{-1} than 10 steps, provided the same preconditioner is used. Thus, one would apply the outer GMRES solver to a preconditioned matrix $C_{L,(k)}^{-1}K$ which is not constant, as $C_{L,(k)}^{-1}$ is slightly different in each call k. This is a situation for which the GMRES algorithm has not been designed and consequently it will fail, or at least return less accurate results than predicted. On the other hand, if a fixed number of iterations and initial guess $x_0 = 0$ is used for each of the inner solves, these become constant linear operators, since the solution approximations resulting from m steps of GMRES [73, Lemma 6.6] as well those from m steps of CG [73, Lemma 6.5] can be written as

$$x_m = x_0 + q_m(A)(b - Ax_0),$$

where $q_m(.)$ is a polynomial of degree m which is independent of the right-hand side b and the initial guess x_0.

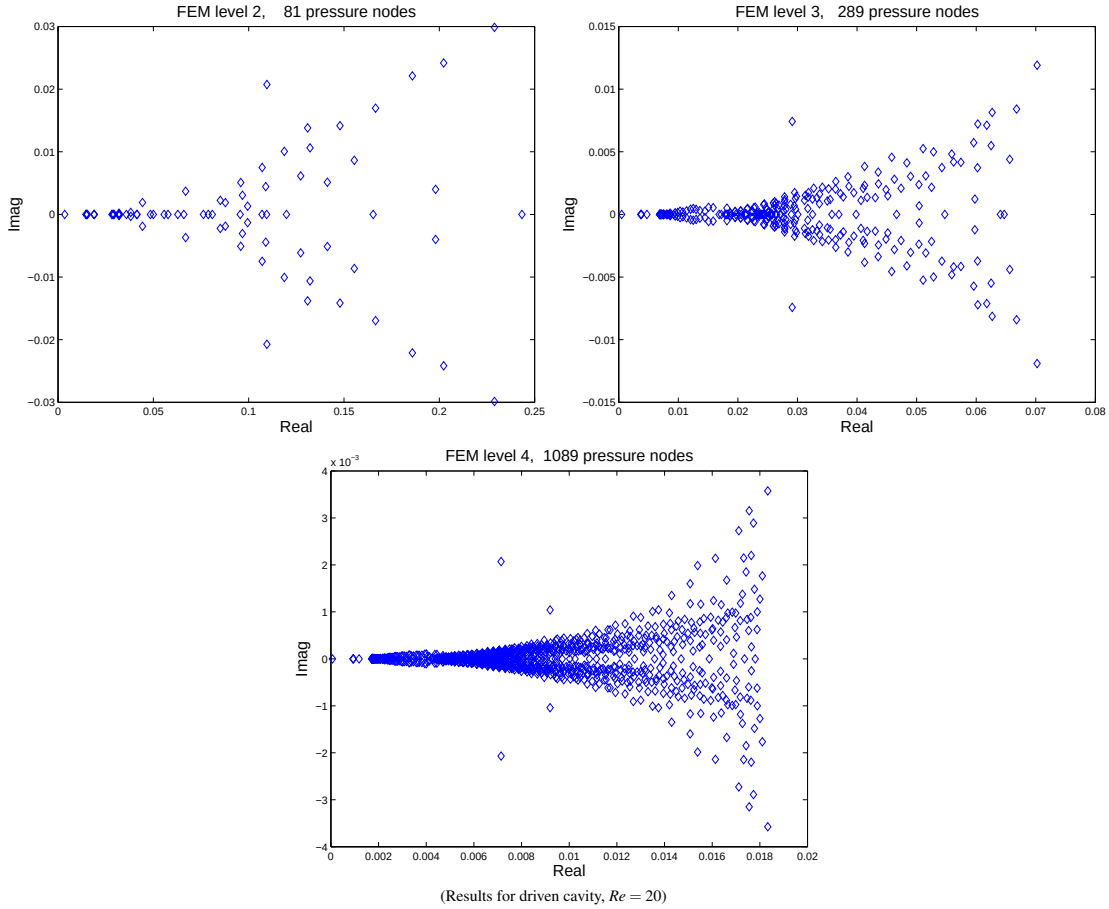
However, just using a problem independent fixed iteration number m for the inner solves would be either unreliable (if m is too small) or inefficient (if m is too large). Computational experiments indicate that a more reasonable choice for defining a fixed iteration number for the inner solves is to use a fixed relative reduction, e.g. by 10^{-3}, as stopping criterion in the first call of each of the inner solves, and to set the resulting iteration number as stopping criterion for all subsequent calls. In a sense the first call in this approach is used to test the efficiency of each of the inner solution approaches and to set the fixed iteration number accordingly, so as to guarantee a problem independent accuracy of the inner solves.

This preconditioning technique has first been developed for the systems arising in the Picard iteration [52], where it performs best, and has been applied to the systems from Newton linearisation later [29]. In the case of the Newton linearisation the performance is worse in two ways. Firstly, the number of iterations for the outer solver deteriorates more strongly as the Reynolds number Re increases, as analysed in [29]. Secondly, and equally importantly, the performance of the multigrid solvers for the subsystems with the F-block deteriorates badly with increasing Re, as is explained in more detail in Subsection 2.5.2.2. The combined increase makes this solver approach less attractive for higher Re, but it is still competitive due to the fast convergence of the Newton linearisation in terms of the nonlinear system. These issues are illustrated in Table 2.1, where for comparison a driven cavity problem has been solved with both linearisation techniques. In both cases, Picard linearisation and Newton linearisation, the F_p preconditioner has been utilised, with the same accuracy settings for the inner and outer solves. The desired Reynolds number $Re = 100$ could not be used on the coarse meshes due to stability issues explained at the end of Section 2.3.1. The table lists the mesh refinement level ℓ, the Reynolds number Re used on that level, the number of degrees of freedom (#dof), the number of linear steps (#steps) to approximate the nonlinear solution to the specified accuracy, the maximal number of F_p preconditioned GMRES iterations required to solve the linear systems ($\max(\#it)$, the maximum is over the linearisation steps), the maximal number of inner GMRES iterations for the F block per outer GMRES iteration ($\max(\frac{\#\text{inner}_F}{\#\text{outer}})$), the maximal time it took to solve one of the (outer) linear equation systems on the level ($\max(t_{\text{solve}})$) and the total time for the computations from the coarsest to the finest level. In both cases the most efficient multigrid preconditioning variant for the F-block solves has been used, i.e. *stab0* for the Picard linearisation, and *stab3* for Newton linearisation, see Section 2.5.2.2 for a detailed discussion.

Picard linearisation						
ℓ	Re	#dof	#steps	$\max(\#it)$	$\max(\frac{\#\text{inner}_F}{\#\text{outer}})$	$\max(t_{\text{solve}})$
1	5.9	59	2	11	2	$5.0e-04$
2	17.9	187	2	17	9	$1.2e-02$
3	20.8	659	2	18	9	$8.7e-02$
4	32.3	2467	2	20	11	$6.6e-01$
5	40.0	9539	2	21	13	$3.4e+00$
6	76.9	37507	4	24	18	$2.5e+01$
7	100.0	148739	4	25	22	$1.4e+02$
8	100.0	592387	4	25	22	$6.1e+02$
					total time	$2.6e+03$

Newton linearisation						
ℓ	Re	#dof	#steps	$\max(\#it)$	$\max(\frac{\#\text{inner}_F}{\#\text{outer}})$	$\max(t_{\text{solve}})$
1	5.9	59	2	11	2	$4.7e-04$
2	17.9	187	2	17	10	$1.8e-02$
3	20.8	659	2	20	10	$1.4e-01$
4	32.3	2467	2	23	14	$1.3e+00$
5	40.0	9539	2	25	16	$6.9e+00$
6	76.9	37507	3	30	25	$6.0e+01$
7	100.0	148739	3	34	32	$4.1e+02$
8	100.0	592387	3	34	32	$1.8e+03$
					total time	$4.7e+03$

Table 2.1: Comparison Picard and Newton linearisations with F_p preconditioner, driven cavity at $Re = 100$



(Results for driven cavity, $Re = 20$)

Figure 2.5: Spectra of the Schur complement S, FE-case (no preconditioning)

Note that the increase in the iteration numbers as the mesh is refined is mainly related to the increasing Re. Once the desired Re can be used on the mesh (stability constraints are fulfilled), further refinement does not result in increased iteration numbers. The effects explained above are clearly visible in this example, i.e. the Newton linearisation requires fewer steps to converge to the solution of the nonlinear system, but the solution of the linear systems arising in the Picard linearisation is computationally more efficient. Note that for the same problem at $Re = 10$ there would be no noticeable difference between the two linearisation methods, due to the dominance of the linear terms in this case.

Yet, besides the fast convergence of the nonlinear systems, there is another important reason that makes the Newton linearisation a more favourable choice: it is required for the discrete adjoint method.

We conclude this subsection by illustrating the effects of the Schur complement preconditioner in figures 2.5, 2.6 and 2.7 for a driven cavity model problem at $Re = 20$ ($\nu = 0.05$). Figure 2.5 shows the spectra of the Schur complement $S := BF^{-1}B^T$ for three levels of mesh refinement in the case of Newton linearisation. The eigenvalues are dis-

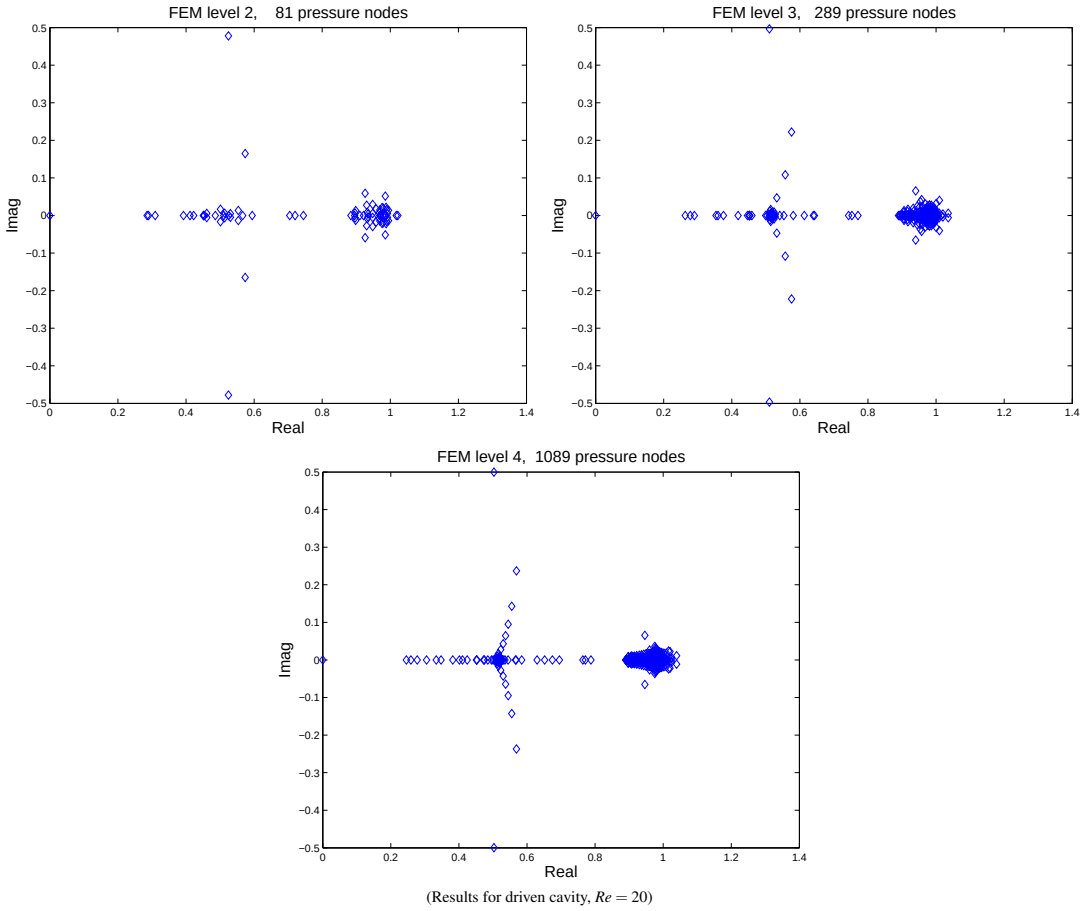
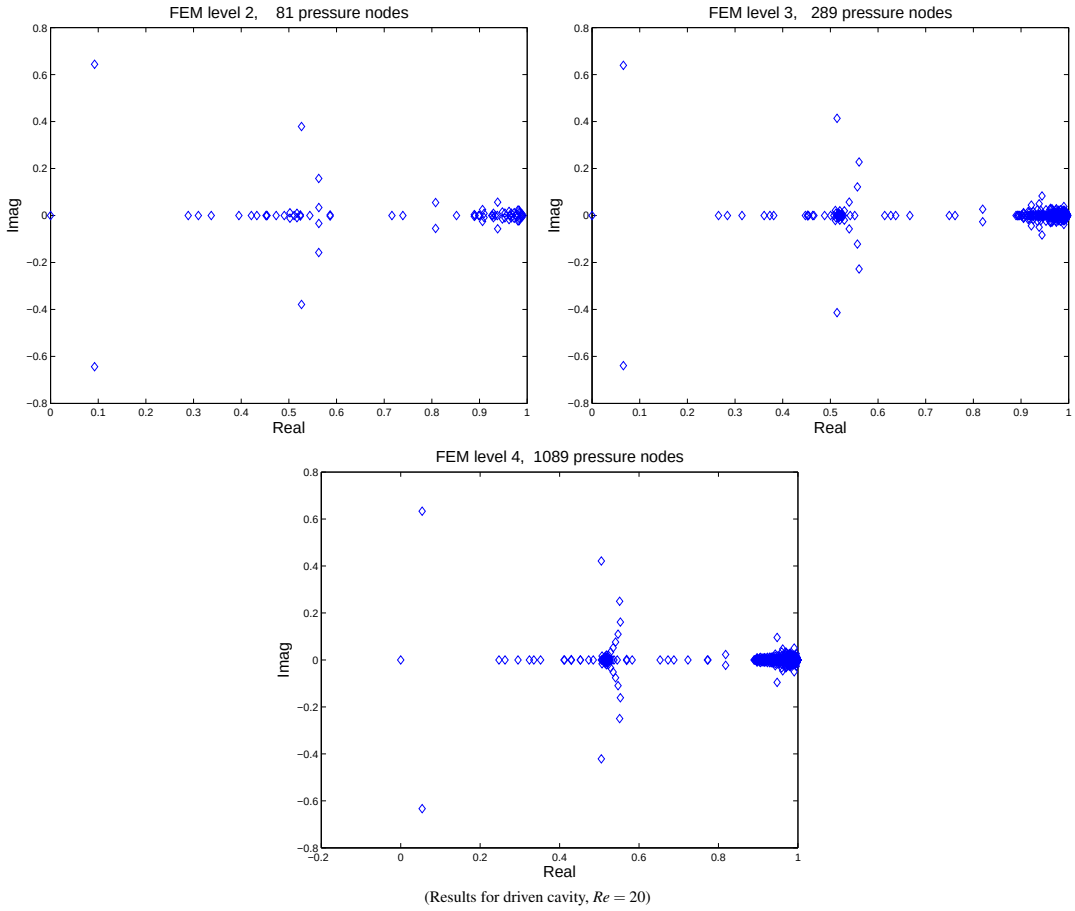


Figure 2.6: Spectra of the F_p-preconditioned Schur complement, $X_{F_p}^{-1}S$, FE-case, Picard linearisation



(Results for driven cavity, $Re = 20$)

Figure 2.7: Spectra of the F_p-preconditioned Schur complement, $X_{F_p}^{-1}S$, FE-case, Newton-linearisation

tributed over a region which depends on the refinement level. This is contrasted by the spectra of the preconditioned Schur complement $X_{F_p}^{-1}S$ in figures 2.6 and 2.7 in the case of Picard linearisation and Newton linearisation respectively. In both cases clustering of the eigenvalues in two small regions may be observed. Note that in all three of these figures a zero eigenvalue is present. This stems from the rank deficiency of the B-block, which is not treated explicitly at this point, but will be discussed in detail in Subsection 2.5.2.3. Note also that in contrast to the case of CG, the spectrum of the matrix alone is not sufficient to describe the convergence behaviour of GMRES [82], even though it is of importance.

2.5.2.2 The F-block of the linearised discrete Navier-Stokes equations

Recall that the F-block is the discretisation of the convection-reaction-diffusion operator

$$-\nu\Delta + \mathbf{u}.\nabla + \sigma(\nabla\mathbf{u}), \tag{2.5.8}$$

where the reaction part with reaction coefficient $(\nabla\mathbf{u})$ is only present in the case of Newton linearisation ($\sigma = 1$ in this case, $\sigma = 0$ otherwise). There are two terms in this operator which make efficient solution difficult, the reaction term and the convection term.

While reaction-diffusion equations with non-negative reaction coefficients can, for example, be efficiently solved by CG with BPX preconditioning [20], the non-negativity condition can clearly not be guaranteed for $\nabla\mathbf{u}$. With increasing Reynolds number the velocity components $\mathbf{u}$ of the solutions of the Navier-Stokes equations usually develop boundary layers, where $\mathbf{u}$ changes rapidly from its free-stream behaviour to conform with the boundary conditions. This change occurs in a layer of thickness $\mathscr{O}(\nu)$ [65] around Dirichlet boundaries. Thus the reaction coefficients can be expected to be of order $\mathscr{O}(\nu^{-1})$ in these areas, while more moderate values can be expected in the rest of the domain. Negative reaction coefficients can produce oscillatory behaviour of the continuous solution [45]. Such oscillations may be impossible to approximate on coarse meshes, which may form an obstacle for successful multigrid solution of these equations. Further, standard theory regarding existence and uniqueness of solutions of elliptic problems uses the positivity of the bilinear part of the weak form of the PDE. This positivity can easily be violated if negative reaction coefficients are permitted. Thus, most publications restrict themselves to non-negative reaction coefficients, for example [33].

If the reaction terms in (2.5.8) are ignored, another model operator, convection-diffusion,

$$-\nu\Delta + \mathbf{w}.\nabla, \tag{2.5.9}$$

is obtained, where $\mathbf{w}$ denotes the convection "wind" which is $\mathbf{u}$ in (2.5.8). As already briefly discussed in Section 2.3.1, the Galerkin FE method applied to convection-diffusion may yield unstable discretisations if convection is dominant and the discretisation is too coarse to resolve the diffusion length scales (e.g. [51, Section 9.5]). These phenomena are characterised by the mesh Peclet number (e.g. [68]),

$$Pe := \frac{h\|\mathbf{w}\|}{2\nu}, \tag{2.5.10}$$

i.e. if $Pe > 1$ the Galerkin FE discretisation may be unstable and oscillations may occur which are not typical of the continuous solutions. In the context of multigrid, good approximations on coarse meshes are required, such that obtaining $Pe \leq 1$ is not generally feasible on all meshes. To avoid the spurious oscillations stabilised discretisations have been introduced, e.g. Streamline-Upwind Petrov Galerkin (SUPG) and Sub-Grid Stabilisation (SGS) techniques have been proposed as a remedy, see e.g. [47] for an overview of different approaches. These schemes modify the discrete weak form by adding terms which improve stability for $Pe > 1$ and vanish as $h \to +0$. For example in the SUPG method, the Galerkin discrete form

$$a(u_h, v_h) := \nu(\nabla u_h, \nabla v_h) + (\mathbf{w} \cdot \nabla u_h, v_h) = (f, v_h) \tag{2.5.11}$$

is modified by changing the test space functions $v_h \in \mathbb{V}_h$ to $(v_h + \delta\ \mathbf{w} \cdot \nabla v_h)$, resulting in the stabilised form

$$a_\delta(u_h, v_h) := \nu(\nabla u_h, \nabla v_h) + (\mathbf{w} \cdot \nabla u_h, v_h + \delta\ \mathbf{w} \cdot \nabla v_h) = (f, v_h + \delta\ \mathbf{w} \cdot \nabla v_h) \tag{2.5.12}$$

in the case of linear elements. The additional term

$$a_\delta(u_h, v_h) - a(u_h, v_h) = (\mathbf{w} \cdot \nabla u_h, \delta\ \mathbf{w} \cdot \nabla v_h) \tag{2.5.13}$$

is equivalent to diffusion along direction $\mathbf{w}$, i.e. the streamline. This together with the classification as a Petrov-Galerkin discretisation, due to the use of discrete test functions $v_h + \mathbf{w} \cdot \nabla v_h$ which are from a different discrete function space than the ansatz functions u_h, gives this method its name, streamline-upwind[7] Petrov Galerkin method. Note that testing with $(v_h + \delta\ \mathbf{w} \cdot \nabla v_h)$ implies terms of the form $(-\nu \Delta u_h, \mathbf{w} \cdot \nabla v_h)$ which vanish for linear elements, but not for higher order elements. The choice of the stabilisation

[7] The term up-winding is generally used for stabilisation approaches for convection discretisations. It stems from finite difference approaches where forward differences in direction of the wind are unstable while backward differences according to the wind direction are stable.

parameter $\delta \geq 0$ is thereby crucial for the success of the approach. Large values of δ smooth out important solution features like boundary layers, while too small values result in unstable discretisations. A typical choice in the case of SUPG is, for example [68],

$$\delta := \frac{\bar{h}}{\|\bar{\mathbf{w}}\|} \max\left(0, \frac{1}{2} - \frac{\nu}{\bar{h}\|\bar{\mathbf{w}}\|}\right), \tag{2.5.14}$$

where $\bar{h}$ is a measure of the element size along the wind direction $\mathbf{w}$, and $\|\bar{\mathbf{w}}\|$ is the Euclidean norm of the wind vector at the centre of the element. The stabilisation allows meaningful (i.e. non-oscillatory) approximations on relatively coarse meshes and, because the stabilisation terms vanish as $h \to +0$, asymptotic convergence is maintained. Thereby the h order of the stabilisation terms influences the overall convergence order of the discretisation and has to be chosen to allow fast convergence while maintaining stability.

In [68] the SUPG approach has been used to develop a multigrid preconditioner for GMRES, to solve convection-diffusion problems in h independent number of GMRES steps. Three different prolongation operators have been considered, standard FE interpolation, and two types of operator dependent prolongation. Restriction was chosen to be the transpose of the prolongation operator in each of the cases. For geometric multigrid, i.e. the coarse grid matrices are assembled on coarse meshes rather than constructed from the fine grid matrices directly, the tests in [68] indicate that the standard interpolation performed at least as well as the operator dependent prolongations. Further, the results in [68] demonstrate that in order to achieve mesh independent iteration counts for GMRES, appropriate choice of the stabilisation parameter is crucial.

The choice of the smoother for convection-diffusion problems deserves more attention than in the Poisson case. In the case of Gauß-Seidel smoothing the node numbering, or alternatively the order in which the rows of the equation system are traversed by the Gauß-Seidel sweep, is far more important than in the Poisson case. The convection wind $\mathbf{w}$ dictates a preferred direction for the transport of information. For pure convection, a forcing term located at a specific node has strong influence on the solution at all nodes in downstream direction of that location, while it has no influence on the solution upstream of that node. For convection-diffusion with dominating convection this effect is not as strong, but still important. Thus, it is advantageous in a Gauß-Seidel sweep if the solution approximation at the upstream nodes is improved first, because any changes to it affect all downstream nodes, whereas changes at the downstream nodes have little influence on the upstream solution. For some problems, e.g. flow in a pipe, it is very easy to number the nodes of the mesh accordingly, by simply sorting them with increasing distance from

the inflow boundary. However, in most interesting flows this is not as easy, because recirculations occur. Due to the recirculation some nodes of the mesh are upstream of themselves. Simple geometrical ordering according to the distance from the inflow boundary is not optimal in these cases. Also for some problems, like the driven cavity problems for example, there is no inflow boundary, as the boundary conditions specify tangential velocities at all parts of the boundary. As a remedy, two Gauß-Seidel sweeps are used as smoother in [68], the first ordering the nodes left-to-right, the second top-to-bottom. Typically the Gauß-Seidel sweeps for the pre- and post-smooths are done in opposing directions, thus the pre-smoothing might be done left-to-right then top-to-bottom, and the post-smoothing bottom-to-top then right-to-left. This way the information can travel in any direction within one V-cycle.

Inspired by the success of the multigrid preconditioned GMRES solver in [68] this strategy has also been adopted in this present work. However, in contrast to the bilinear quadrangular elements used in [68], quadratic triangular elements are used here in the definition of the discrete systems corresponding to the F-block. Even though this type of element is frequently used in practice, e.g. as part of the Taylor-Hood element pair in this present work, the author has been unable to find any reference in the literature that investigates stabilisation of convection-diffusion or multigrid solution of convection-diffusion for this type of element. Yet for linear and bilinear elements there are numerous publications on this type of stabilisation, e.g. [17, 23, 33, 45, 47, 62, 63, 68] (and many more).

Often the analysis of these stabilisation approaches contain terms which vanish for the low order elements for which they are used, e.g. [47, Equation (77)] proposes a scheme with

$$a_\delta(u_h, v_h) - a(u_h, v_h) = (\mathscr{L}u_h - f, \delta(-\mathscr{L}^*\nabla v_h)), \tag{2.5.15}$$

where $\mathscr{L}$ denotes the differential operator of the PDE, $\mathscr{L}^*$ its adjoint and the inner product is evaluated element wise (ignoring jump terms at element interfaces). In the case of convection-diffusion $\mathscr{L} = -\nu\Delta + \mathbf{w}\cdot\nabla$, the resulting integral contains terms of the form Δu_h and Δv_h which are zero for linear and bilinear elements. This approach has been tested by the author during the preparation of this thesis, together with the proposed stabilisation parameter

$$\delta_{\text{Hughes}} := \frac{h}{2\|\mathbf{w}\|}\left(\coth(Pe) - \frac{1}{Pe}\right). \tag{2.5.16}$$

If the coarse grid discretisations were defined that way for the convection-diffusion subproblems that arise when the F_p preconditioner is applied to the Picard linearisation dis-

cretised by quadratic elements, this resulted in a breakdown of the subproblem solver (convergence far too slow)[8], under the same conditions under which the choice (2.5.13) with (2.5.14) works flawlessly.

As this example demonstrates, extending the approaches known to work well for the linear/bilinear elements to quadratic elements is non-trivial. Two important questions have to be raised: under which conditions on the ratio of convection and diffusion is the standard Galerkin scheme stable, and how to stabilise it when it is unstable. However, to answer these questions rigorously is not the aim of this work. Instead, the criterion $Pe \leq 1$ is used and a number of variations of stabilisation approaches centring around the SUPG discretisation (2.5.12) with stabilisation parameter (2.5.14) are tested for the quadratic elements and heuristic arguments are given as to why these approaches are feasible.

The criterion $Pe \leq 1$ may be viewed in the context of [62], where Oñate derives an explanation for the stabilisation terms from analysing and extending the derivation of the differential operator itself. Usually these operators are derived by considering control volumes of finite length scale $\widetilde{h}$, for which conservation laws (e.g. force balancing or energy conservation) are applied and the limit $\widetilde{h} \to +0$ implies the PDE. Oñate finds that if, in addition to the linear terms which persist in the limit $\widetilde{h} \to +0$, quadratically vanishing terms are considered as well, the standard stabilisation techniques for finite elements can be derived. These equations contain a discretisation length scale parameter equivalent to δ in the notation of this work. One might argue that this derivation of the stabilised equations indicates that if a higher order data representation is used within the mesh cells, this alone may result in the discretisation being stable in conditions where a lower order data representation results in un-stable discretisations. On the other hand, for non-stationary convection-diffusion problem it is well known (e.g. [89, Section 6.4]) that first order finite difference methods, if chosen appropriately, produce non-oscillatory solutions even on coarse meshes where second order methods produce non-physical oscillations. So this argument does not hold in general. However, the authors computational experiments with the Taylor-Hood element pair for Navier-Stokes show no indication of oscillations when $Pe \leq 1$ is obeyed. Thus it is assumed to be a sufficient condition for the quadratic velocity elements as well.

An overview of the basic stabilisation approaches for the coarse grid discretisations of (2.5.8) is given in Table 2.2. For reference the pure Galerkin discretisation, that is

[8] [47, Subsection 5.3, REMARK 4] indicates that an element wise constant δ may not be optimal for quadratic elements. However, parts of the derivation of the stabilisation in [47] are based on assumptions that are impossible to justify rigorously, i.e. vanishing of the sub-grid scales on element boundaries. Thus, even though the work provides one possible explanation for the stabilisation approaches in the case of low order elements, it is not necessarily a good basis for extension to higher order elements.

without stabilisation, is included in this list under the name *Stab0*. The simplest approach to stabilisation is to modify the differential operator by increasing the viscosity constant ν such that the stable regime is maintained. This approach is generally referred to as artificial diffusion (e.g. [51]) and is known to give poor approximations because it changes the properties of the differential operator. However, as the stabilised approaches are only used in the coarse grid discretisations which are used in a multigrid preconditioner here, it is not necessary that these discretisations give the most accurate discrete solution, but only something that approximates the low frequencies of the solution well. The diffusion parameter ν is set to a mesh wide constant value, which is increased by a factor of two with each mesh coarsening. This stabilisation variant is denoted by *Stab1*.

A refined approach is to add diffusion only in the streamline direction, which is achieved by using formula (2.5.12). In the case of linear elements this can be seen as a Petrov-Galerkin discretisation, but for quadratic elements this does not hold. Instead, (2.5.12) can be seen as a perturbation of the Galerkin formulation. This approach is denoted by *Stab2*. A simplification is possible by using linear elements for the coarse grid discretisations. This can be implemented using linear element functions on the finest mesh as the second finest discretisation in the hierarchy, and applying coarsening to define the remaining levels of the hierarchy. An important advantage of this approach is that for linear elements the stiffness matrices have fewer non-zero entries per row and thus storage requirements are lower and fewer floating point operations are required for the multigrid ingredients of smoothing, and evaluation of the residuals. Further, the stabilised methods are readily available for linear elements. The resulting approach is denoted by *Stab3*. If the reaction terms are present, i.e. for Newton linearisation, a further simplification can be made by dropping the reaction terms on the coarse grids, resulting in a further reduction of the memory requirements as the coupling between the velocity components is removed. This approach is referred to as *Stab4*.

Finally, application of the Sub-Grid Stabilisation (SGS) approach [47] is attempted, denoted by *Stab5* in the overview Table 2.2. However, as already described earlier in this subsection, this approach failed outright even in the tests for Picard linearisation at low *Re*. Thus, no results for *Stab5* are included in the test result tables below.

Now that all these aspects of the multigrid preconditioning of the F-block subproblems have been discussed we present some exemplary results for the driven cavity problem. Both Picard and Newton linearisation are considered, for two Reynolds number regimes, $Re = 10$ and $Re = 100$. The nonlinear solver steps are performed until both the Euclidean norm of the residual and the update are less than 10^{-1} for the coarse levels and less than $10^{-}6$ on the finest mesh. The stopping criterion for the outer linear solves is set

Name	Description
Stab0	No stabilisation. (Standard Galerkin discretisation.)
Stab1	Artificial diffusion. The viscosity constant ν is doubled with every level of mesh coarsening.
Stab2	Streamline diffusion. Only the streamline diffusion term (2.5.12) with δ according to (2.5.14) is added to the standard Galerkin discretisation. (Not a Petrov-Galerkin discretisation.)
Stab3	Linear elements and SUPG. Only linear elements are used in the coarse meshes and SUPG according to (2.5.12) and (2.5.14). (True Petrov-Galerkin discretisation.)
Stab4	Same as *Stab3*, but if reaction terms are present (Newton linearisation) these are switched off on coarse meshes.
Stab5	Sub-grid Stabilisation. The term stabilisation term (2.5.15) is used in conjunction with (2.5.14).

Table 2.2: Stabilisation approaches for the coarse grid representations of the F-block

			stab1	stab2	stab3	stab4
	ν	#dof	it_F/it_{outer}			
	1.0e-01	162	11	10	9	9
	1.0e-01	162	10	9	9	9
	1.0e-01	578	14	9	9	10
	1.0e-01	578	14	9	9	10
	1.0e-01	2178	18	10	9	10
	1.0e-01	2178	19	10	10	10
	1.0e-01	8450	22	10	9	10
	1.0e-01	8450	26	10	10	10
	1.0e-01	33282	26	9	9	10
	1.0e-01	33282	35	10	10	11
	1.0e-01	132098	40	9	9	10
	1.0e-01	132098	53	10	10	10
	1.0e-01	526338	26	9	10	10
	1.0e-01	526338	72	10	10	11
	1.0e-01	526338	81	10	10	11
		total time	4.0e+03	9.5e+02	9.3e+02	9.3e+02

Table 2.3: Inner iteration counts F-block for driven cavity at $Re = 10$, Newton linearisation, V-cycle, various stabilisations

			stab0	stab2	stab3	stab4
	ν	#dof	it_F / it_{outer}			
*	5.6e-02	162	9	10	8	8
*	5.6e-02	162	8	10	8	8
*	4.8e-02	578	9	11	9	9
*	4.8e-02	578	9	10	9	9
*	3.1e-02	2178	11	12	11	11
*	3.1e-02	2178	11	13	11	11
*	2.5e-02	8450	12	14	13	13
*	2.5e-02	8450	13	15	13	13
*	1.3e-02	33282	18	21	19	19
*	1.3e-02	33282	18	21	19	19
*	1.3e-02	33282	18	21	19	19
*	1.3e-02	33282	18	21	18	18
	1.0e-02	132098	22	26	22	22
	1.0e-02	132098	22	26	22	22
	1.0e-02	132098	22	25	22	22
	1.0e-02	132098	22	26	22	22
	1.0e-02	526338	22	26	22	22
	1.0e-02	526338	22	26	23	23
	1.0e-02	526338	22	26	23	23
	1.0e-02	526338	22	26	23	23
		total time	2.6e+03	3.4e+03	2.8e+03	2.8e+03

Table 2.4: Inner iteration counts F-block for driven cavity at $Re = 100$, Picard linearisation, V-cycle, various stabilisations

			stab0	stab2	stab3	stab4
	ν	#dof	it_F/it_{outer}			
*	5.6e-02	162	10	11	10	10
*	5.6e-02	162	10	11	10	10
*	4.8e-02	578	10	12	10	11
*	4.8e-02	578	11	12	10	12
*	3.1e-02	2178	13	15	14	15
*	3.1e-02	2178	14	15	14	15
*	2.5e-02	8450	16	17	16	18
*	2.5e-02	8450	16	17	16	18
*	1.3e-02	33282	26	27	25	27
*	1.3e-02	33282	26	27	24	28
*	1.3e-02	33282	26	27	25	27
	1.0e-02	132098	63	37	32	37
	1.0e-02	132098	54	37	31	35
	1.0e-02	132098	54	35	32	39
	1.0e-02	526338	39	29	27	29
	1.0e-02	526338	54	37	32	38
	1.0e-02	526338	52	38	31	34
		total time	8.3e+03	5.5e+03	4.7e+03	5.0e+03

Table 2.5: Inner iteration counts F-block for driven cavity at $Re = 100$, Newton linearisation, V-cycle, various stabilisations

to 10^{-5} improvement in the norm of the preconditioned residual. For the inner solves a relative accuracy of an improvement by 10^{-6} in the preconditioned residual is required for the first call of the inner solver within the solve of each outer system, and the resulting number of iterations is fixed for all consecutive calls. Iteration counts for the inner subproblems with the F-block are presented in tables 2.3, 2.4 and 2.5. In each case the table lists the actual ν used on each refinement level, the number of degrees of freedom (#dof) in the outer nonlinear system, and for each multigrid preconditioner variant the number of inner iterations for the F-block per outer iteration (it_F/it_{outer}). The refinement levels are separated by lines while the individual rows within each block constitute the steps of the nonlinear solver.

In Table 2.3 the results are presented for Newton linearisation in the case of $Re = 10$. Apart from the stabilisation *Stab1* (artificial viscosity) all the columns show textbook multigrid performance. The degradation in the case of *Stab1* shows that this approach is unsuitable for constructing a competitive multigrid preconditioner. A column for *Stab0*, i.e. no stabilisation, as well as a similar table for Picard iteration have been omitted because the results are virtually identical. However, the situation changes if Re is increased to $Re = 100$, see tables 2.4 and 2.5. Here it is actually the case that without increasing ν the stability condition $Pe < 1$ is only fulfilled on the two finest meshes used in the tests. The levels where ν was increased to guarantee this stability condition are marked with a * in the first column. For Picard linearisation, Table 2.4, a rather surprising result is obtained. The variant without stabilisation of the coarse grid discretisations, *Stab0*, actually performs better than any of the stabilised variants. However, the variants *Stab3* and *Stab4* show almost the same performance as *Stab0*, while *Stab2* results in slightly higher iteration numbers. Mesh independent iteration numbers are obtained in all four stabilisation variants as there is only an increase between the blocks where the diffusion parameter ν is decreased. Thus optimal performance is achieved for all four variants, but a deterioration is observed as the Reynolds number is increased.

For the Newton linearisation the situation is similar in the sense that the strongest differences in iteration numbers occur where the diffusion parameter ν is decreased, see Table 2.5. But here the differences between the stabilisation approaches are more pronounced, and the deterioration as Re is increased is also more significant. The non-stabilised variant *Stab0* actually performs worst, while among the stabilised variants *Stab3*, streamline diffusion on linear elements, performs best in multiple ways: it results in the lowest overall time to solution, the lowest iteration numbers and the smallest variations between consecutive Newton steps. Thus we conclude that *Stab3* is the most preferable of the listed stabilisation approaches.

Note that the variant *Stab1* has been omitted for the $Re = 100$ tests due to the poor results in the $Re = 10$ test case. Note also that this is only one example problem, but it shows the general trends observed in more extensive tests carried out by the author. As one such test ordering of the nodes as described in [68] has been tested, i.e. performing two ordered Gauß-Seidel sweeps, one sorted by the x component of the node positions and the second sweep sorted by the y component. This resulted in slightly decreased iteration numbers, but this did not compensate for the higher cost due to the multiple smoother sweeps. The deterioration of the multigrid performance was unaffected by these ordered smoother sweeps.

Considering the partial character of the success of the geometric multigrid (GMG) approach for the problems considered in this subsection, the question remains if other approaches might provide a better base for a preconditioner. In [106] both geometric and algebraic multigrid (AMG) approaches have been considered for convection-diffusion problems, with the result that both approaches show deficiencies as the diffusion parameter is decreased. However, there it was found that for convection dominated problems GMRES with an AMG preconditioner was the best of the solvers considered. Another recent approach is to utilise hierarchical matrix ($\mathscr{H}$-matrix) approximative LU factorisations [15] as preconditioner for GMRES. For the streamline-diffusion FE discretisation of the convection-diffusion equation the approximate LU decompositions can be computed with a given fixed accuracy in logarithmic-linear complexity in both memory and computational cost [15], rendering this an attractive approach. However, both of these approaches have so far only been applied to stabilised discretisations of convection-diffusion problems. Application to the F-block subproblems considered in this work is a possible future research direction. Especially the $\mathscr{H}$-matrix approach may be promising for the subproblems arising in the Newton-linearisation, as it allows to specify an accuracy of the approximate LU factorisation even in the general case, i.e. for arbitrary system matrices. The special properties of the system matrices, which arise due to the properties of the discretised PDEs, are required to guarantee the boundedness of memory and computational cost, but the approach should work reliably (albeit more expensively) for more general systems as well.

2.5.2.3 Treatment of boundary conditions and zero-mean-pressure condition by projections

An important aspect that has been largely ignored in the previous subsections is the efficient implementation of two side-constraints to the linearised Navier-Stokes systems: the

zero-mean-pressure condition (ZMPC) and the Dirichlet boundary conditions. In both cases a finite element function space $\mathbb{V}^h$ is replaced by a lower dimensional subspace $\mathbb{V}_0^h$. The linearised discrete problem becomes

$$K\underline{u} = b,$$

with

$$\begin{aligned} K &= \left[a(\varphi_j, \varphi_i)\right]_{i,j=1}^{n-k}, \\ b &= \left[b(\varphi_i) - a(g_0, \varphi_i)\right]_{i=1}^{n-k}, \end{aligned}$$

$\{\varphi_1, \ldots, \varphi_{n-k}\}$ denoting a basis of the subspace $\mathbb{V}_0^h$, g_0 denoting a function fulfilling the Dirichlet boundary conditions which is extended identical to zero for all nodes in the interior of the domain in the case of Dirichlet boundary conditions, and $g_0 = 0$ in the case of the ZMPC. In most cases it is disadvantageous to assemble the stiffness matrix K and right-hand side b directly. For example in the case of Dirichlet boundary conditions this would involve consideration of special cases for each element, i.e. checking if one or more nodes of the element are boundary nodes. The inclusion of such conditional statements in what is a nested loop of computationally intensive operations has a significant negative impact on the speed of execution! Instead often the matrix $\widetilde{K}$ is assembled ignoring the constraints, i.e.

$$\widetilde{K} = \left[a(\varphi_j, \varphi_i)\right]_{i,j=1}^{n}$$

with $\{\varphi_1, \ldots, \varphi_n\}$ denoting a basis of the whole FE function space $\mathbb{V}^h$ (and $\widetilde{b}$ accordingly). The constraints are then easily incorporated by modifying the discrete system,

$$K = P^T \widetilde{K} P \tag{2.5.17a}$$

$$b = P^T \widetilde{b} - P^T \widetilde{K} \underline{g}_0, \tag{2.5.17b}$$

where P denotes an appropriate projector onto the discrete subspace $\mathbb{V}_0^h \subset \mathbb{V}^h$ and $\underline{g}_0$ the coefficient vector for function g_0. In the case of Dirichlet boundary conditions for example this is simply to delete rows and columns from the stiffness matrix, to delete rows from the right-hand side vector and to adjust the right-hand side to incorporate $a(., g_0)$. Thus,

$$P := \begin{bmatrix} I_{n-k} \\ 0 \end{bmatrix} \in \mathbb{R}^{n,n-k}.$$

However, this modification of the matrix requires the size of the matrix to change and a reordering of entries if the boundary nodes are not as conveniently numbered as in the notation here. Further this approach requires the handling of coefficient vectors of size $n-k$ as well as of size n, including the boundary nodes.

An alternative approach, circumventing these difficulties, is taken in [57] in the context of hanging nodes in adaptive mesh refinement, remarking on its application for the handling of Dirichlet boundary conditions as well. The first modification this approach makes is to define the projector P as $P \in \mathbb{R}^{n,n}$, i.e. it projects into a lower dimensional subspace of $\mathbb{R}^n$ rather than the lower dimensional space $\mathbb{R}^{n-k}$. It may then be observed that if appropriate projection steps are introduced in an iterative solver (only CG is considered in [57]) it can be achieved that only residuals of the projected type $r = P^T(\widetilde{K}x - \widetilde{b})$ are ever used in the solver and that if the initial guess for the solution is within the subspace $\mathbb{V}_0^h$ and all search directions are projected, $w = P\hat{w}$, then all the iterates for the solution remain within the subspace $\mathbb{V}_0^h$. Since all the arithmetic of the solver is performed in $\mathbb{V}_0^h$ this is actually equivalent to applying the iterative solver to a lower dimensional problem. This approach implies a small computational overhead because all operations are performed in $\mathbb{R}^n$ rather than $\mathbb{R}^{n-k}$, but it saves on the cost for setting up the problem.

In [57] it is found that this approach can be implemented using a standard CG implementation, simply by using a projected version of the preconditioner for the larger space,

$$C^{-1} := P\widetilde{C}^{-1}P^T. \tag{2.5.18}$$

(This includes the possibility $\widetilde{C}^{-1} = I$.) Analysing this idea it was found that the optimality of the BPX preconditioner for the Poisson problem is preserved by this approach (for the hanging nodes treatment). Unfortunately the technique of proof, the fictitious space lemma, is not directly applicable to the systems under consideration in this present work here, due to their non-symmetric character.

The question whether implementation in GMRES can be done in the same simple way, by using the projected preconditioner (2.5.18) can be answered in the affirmative. Substituting the preconditioner in Algorithm 3 by (2.5.18), one can immediately see that only projected residuals are ever considered and that all search directions conform with the subspace $\mathbb{V}_0^h$.

In the remainder of this subsection, first the application of this approach to the Dirichlet boundary conditions and later its use to implement the ZMPC will be considered in more detail.

Dirichlet boundary conditions. As the application to Dirichlet boundary conditions served as the main example in the introduction above, the overall idea of how this approach applies in this case is clear. The appropriate projectors take the form

$$P = diag([\alpha_1, \ldots, \alpha_n]),$$

with

$$\alpha_i := \begin{cases} 0 & \forall i : \text{index } i \text{ is associated with a Dirichlet node,} \\ 1 & \text{otherwise.} \end{cases}$$

A special feature of this case is that if the initial guess of the Krylov subspace solver is chosen to fulfil non-homogeneous boundary conditions, i.e. $x^{(0)} = \underline{g}_0$ with $\underline{g}_0$ as defined above, then all iterates are in the affine subspace $(\mathbb{V}_0^h + g_0) \subset \mathbb{V}^h$. However, as all updates to this $x^{(0)}$ are in $\mathbb{V}_0^h$, this is equivalent to solving in the subspace V_0^h with a right-hand side equivalent to (2.5.17b).

Numerical experiments indicate that preservation of optimality of multigrid as a preconditioner, as proven for BPX and the hanging node treatment in [57], is not trivially achieved. If the boundary conditions are ignored for the coarse grid representations the iteration numbers required to achieve a prescribed relative residual reduction in GMRES increase. This is improved if due consideration is given to the boundary conditions on the coarse meshes. In a manner similar to the basic projection idea this can easily be achieved by a slight modification of the Gauß-Seidel smoother. If the vector $d = [d_0, \ldots, d_n]$ of reciprocals of the diagonal entries of the system matrix A is stored,

$$d_i := \frac{1}{A_{i,i}}, \tag{2.5.19}$$

then the update step of the i-th degree of freedom in the Gauß-Seidel sweep can be implemented as

$$x_i := x_i + d_i(b_i - \sum_{k=1}^{n} A_{i,k} x_k).$$

Besides the advantage that at each step this saves a division, which on current hardware is computationally more expensive than a multiplication, this approach also allows to set the values of d_i to zero for all boundary degrees of freedom, i.e. $d := Pd$. As this can easily be done on all levels of the multigrid hierarchy, the V-cycle with these projected smoothers becomes equivalent to standard Gauß-Seidel smoothing for the systems where the boundary degrees of freedom have been removed. This minor modification improves the performance of multigrid as projected preconditioner (2.5.18). This is demonstrated in Table 2.6, where iteration counts are given for the inner iterations of the F blocks of

the F_p preconditioner. The table is organised in the same manner as the result tables at the end of Subsection 2.5.2.2, see page 61 for a detailed description. The variant *normal diag* denotes the implementation of the projected preconditioner without projection of d on all levels, while *projected diag* denotes the results for the same problem with the projected smoother on all levels $d = Pd$. Note that the difference lies only in different variants of $\widetilde{C}^{-1}$.

If the coarsest mesh of the hierarchy is finer than in the example given in Table 2.6, it becomes important to apply a direct solver on the coarsest grid, because very low frequency components of the error are not sufficiently removed by smoothing alone if the coarsest grid is too fine. To apply a direct solver it is necessary to implement the Dirichlet boundary conditions directly on the coarsest mesh. This variant is denoted *projected diag exactCGS*. As the values in Table 2.6 demonstrate, this gives only a very slight improvement in this case where the coarsest grid consists of only 5×5 nodes. However, for examples with finer and less regular coarse grids noticeable improvements are obtained by the direct coarse grid solves.

Even though this slight modification is necessary in order to achieve the best multigrid performance, the projection approach is beneficial for the implementation of the multigrid components. It simplifies the implementation of restriction and prolongation operators, since these operators can be exactly the same for interior and boundary elements, due to the projected smoother.

The zero mean pressure condition. While the zero-mean-pressure condition (ZMPC) $p \in L^2_0(\Omega)$, i.e. $\int_\Omega p \, \mathrm{d}\Omega = 0$, is frequently used in the theory on the existence and uniqueness of the solutions to Dirichlet problems of the Stokes and Navier-Stokes equations, often very little is said on how this or similar conditions are enforced for the discrete problems. The purpose of this condition is to enforce uniqueness of the pressure part of the solution to these problems, as without any further constraints p is only defined up to an additive constant. One common approach is to fix the pressure to a specified value, usually zero, at one node of the mesh, e.g. [38, Vol. 2, Section 3.8.2, Remark 1]. While this approach is reasonable if this node is chosen well, it may be difficult in general to make such a choice. If, for example, the pressure is fixed at a location where a pressure spike occurs, this may result in relatively large pressure values in parts of the domain where p shows very little variation. This combined with the limitations of finite precision arithmetic may result in unnecessary cancellation effects.

Formulation of the ZMPC for the discrete systems directly leads to an equation of the

			normal diag	projected diag	projected diag exactCGS
	ν	#dof	it_F / it_{outer}		
*	1.7e-01	50	8	7	2
*	1.7e-01	50	8	8	2
	1.0e-01	162	11	9	9
	1.0e-01	162	11	9	9
	1.0e-01	578	12	9	9
	1.0e-01	578	12	9	9
	1.0e-01	2178	13	9	10
	1.0e-01	2178	13	9	9
	1.0e-01	8450	13	10	9
	1.0e-01	8450	13	9	9
	1.0e-01	33282	14	10	9
	1.0e-01	33282	14	9	9
	1.0e-01	132098	15	10	9
	1.0e-01	132098	15	9	9
	1.0e-01	132098	16	10	9
	1.0e-01	526338	16	9	9
	1.0e-01	526338	16	10	9
	1.0e-01	526338	16	9	9
	1.0e-01	526338	16	9	9
		total time	1.3e+03	8.5e+02	9.5e+02

Table 2.6: Inner iteration counts F-block for driven cavity at $Re = 10$, Picard linearisation, V-cycle, *Stab2*, implementations of boundary conditions on coarse grids

form

$$w^T \underline{p} = 0, \tag{2.5.20}$$

where w is a weight vector, containing the integrals over the pressure basis functions, and $\underline{p}$ is the coefficient vector of the FE pressure solution. One possible approach to incorporate this equation into the discrete system is to use a Lagrange multiplier approach, leading to systems of the form

$$\begin{bmatrix} F & B^T & 0 \\ B & 0 & w \\ 0 & w^T & 0 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{p} \\ \lambda \end{bmatrix} = \begin{bmatrix} f_u \\ f_p \\ 0 \end{bmatrix}, \tag{2.5.21}$$

where the scalar λ is an auxiliary variable, the Lagrange multiplier. While this approach is feasible, its effect on the conditioning of the system may be disadvantageous. Further, it has the disadvantage of changing the system size, which results in significant overhead in the setup of the problem and during the solution process. Incorporating the weight vector w^T into the sparse system matrix may be disadvantageous due to the relatively high number of non-zeros of the resulting row[9]. An equivalent system is obtained by replacing one row, e.g. the last, of the last block row of (2.5.2) with (2.5.20), as it has for example been studied in our previous work [75] for the formulation of the discrete adjoint system. Proof of the equivalence of these two formulations is given in Appendix B. While this approach avoids the increased size of the system, the issue of its effects on the conditioning of the resulting problem remains.

In this subsection two more approaches are studied, which turn out to be related. The first is a "do nothing" approach which under certain conditions can be interpreted as a projection approach, but with respect to a different subspace than that induced by the ZMPC. The second approach is based on a projection onto the ZMPC subspace.

In order to analyse these approaches a property of the B-block is required. For this we recall the definition of B as derived from (2.3.7b) and (2.3.8),

$$\begin{aligned} B &= \left[b_{i,j}\right] \in \mathbb{R}^{M,dN}, \\ b_{i,j} &= b(\varphi_{(j)}, \theta_i) \\ &= \int_\Omega \theta_i \ \mathrm{div}(\varphi_{(j)}) \ \mathrm{d}\Omega, \end{aligned} \tag{2.5.22}$$

[9] Sparse matrix data structures with constant maximum number of non-zeros per row or column can not be used in this case or have to be used with disadvantageous parameters. Banded matrix storage is also not applicable.

where the θ_i are pressure space basis functions, M the dimension of the discrete pressure space S^h, the $\varphi_{(j)}$ are velocity space basis functions and dN the dimension of the velocity space $\mathbf{V}_0^h$. Note that the notation $\varphi_{(j)}$ is used here to symbolise the j-th basis function of $\mathbf{V}_0^h$, avoiding the additional index φ_k^ℓ which was used in (2.3.8) to distinguish the components of the velocity vector. Let us assume Dirichlet boundary conditions for the velocity on the whole boundary $\partial\Omega$ and that these boundary conditions have been treated accordingly, i.e. $\varphi_{(j)}$ are a basis of $\mathbf{V}_0^h$ rather than $\mathbf{V}^h$, or equivalently $\varphi_{(j)}(x) = 0 \; \forall x \in \partial\Omega$. Let us also assume that the ZMPC is ignored, that is the θ_i form a basis of S^h rather than S_0^h. For the usual nodal basis functions and $\underline{e} = [1, 1, \ldots, 1]^T$, i.e. a vector of ones, this implies

$$\underline{e}^T B = \left[b(\varphi_{(j)}, 1) \right]_{j=1}^{dN}.$$

This can be simplified using Greens formula and (2.5.22), giving

$$\begin{aligned} b(\varphi_{(j)}, 1) &= \int_\Omega 1 \; \mathrm{div}(\varphi_{(j)}) \; \mathrm{d}\Omega \\ &= \int_{\partial\Omega} n \cdot \varphi_{(j)} \; \mathrm{d}\Gamma \\ &= 0. \end{aligned}$$

Thus,

$$\underline{e}^T B = 0. \tag{2.5.23}$$

Let $\underline{e}_p$ be defined as the vector containing zeros for all velocity degrees of freedom (DOFs) and ones for all pressure DOFs. Then (2.5.23) implies for the matrix K as defined in (2.5.2)

$$K\underline{e}_p = 0 \quad \text{and} \quad \underline{e}_p^T K = 0. \tag{2.5.24}$$

Knowledge of these singular vectors of K allows application of the following Lemma.

Lemma 1. *Let $A \in \mathbb{R}^{n,n}$, $\mathrm{rank}(A) = n - 1$, with known singular vector $v_0 \in R^n$, $\|v_0\| = 1$,*

$$Av_0 = 0 \quad \text{and} \quad v_0^T A = 0, \tag{2.5.25}$$

and let $w \in \mathbb{R}^n$ be such that $v_0^T w = 1$. Then $P := I - v_0 w^T$ defines a projector for which it holds

$$P^T A P = A \tag{2.5.26}$$

and there exists an orthogonal matrix $Q \in \mathbb{R}^{n,n}$ such that

$$Q^T A Q = \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{A} \end{bmatrix}, \tag{2.5.27}$$

with $\widetilde{A} \in \mathbb{R}^{n-1,n-1}$ non-singular. Further, P^T is also a projector.

Proof. The definition of P and (2.5.25) give

$$\begin{aligned} P^T A P &= \left(I - w v_0^T\right) A P \\ &= \left(A - w(v_0^T A)\right) P \\ &= A\left(I - v_0 w^T\right) \\ &= A - (Av_0)w^T \\ &= A. \end{aligned}$$

P and P^T are projectors because

$$\begin{aligned} PP &= (I - v_0 w^T)(I - v_0 w^T) = I - 2v_0 w^T + v_0(w^T v_0)w^T \\ &= I - v_0 w^T = P \end{aligned}$$

and

$$\begin{aligned} P^T P^T &= (I - v_0 w^T)^T (I - v_0 w^T)^T = (I - w v_0^T)(I - w v_0^T) = I - 2w v_0^T + w(v_0^T w)v_0^T \\ &= I - w v_0^T = P^T. \end{aligned}$$

For the second part, we observe that the set of vectors $\{v_0\}$ can be extended to an orthonormal basis of $\mathbb{R}^n$ by orthogonalising the canonical basis. The resulting orthonormal basis $\{v_0, v_1, \ldots, v_{n-1}\}$ defines an orthogonal matrix Q as

$$Q^T := \begin{bmatrix} v_0 & v_1 & \ldots & v_{n-1} \end{bmatrix}. \tag{2.5.28}$$

Equation (2.5.27) is then easily verified by applying (2.5.25). The rank of a matrix is invariant under multiplication with non-singular matrices, thus

$$n - 1 = \operatorname{rank}(A) = \operatorname{rank}(Q^T A Q) = \operatorname{rank}(\widetilde{A}).$$

□

Remark 1. *The non-singularity of $\widetilde{A}$ implies that $Ax = P^T b$ has a unique solution in the subspace orthogonal to v_0. For the solution to exists it is necessary that the right-hand side of the equation system is orthogonal to v_0, which is ensured by the projection $P^T b$.*

Remark 2. *Note that the condition $\|v_0\| = 1$ was only used to simplify the definition of Q. For the definition of P it is sufficient to have $v_0 \neq 0$ and $v_0^T w = 1$.*

An interesting observation is that since $v_0^T A = 0$, a Krylov subspace solver without preconditioning ($C^{-1} = I$) automatically operates in the subspace orthogonal to v_0, because all search directions are formed as $A^k r_0$. Thus if the right-hand side b of the system $Ax = b$ and the initial iterate are orthogonal to v_0 as well, it will converge to a unique solution orthogonal to v_0. This "do nothing" approach is equivalent to a projection approach with $\widehat{P} := I - v_0 v_0^T$. In the more general case, if a preconditioner with

$$w^T (C^{-1} v) = 0 \quad \forall v \in \mathbb{R}^n : v^T v_0 = 0, \tag{2.5.29}$$

is used where $w^T v_0 = 1$, then all iterates will be in the subspace orthogonal to w, because the search space is spanned by $\left\{(C^{-1}A)^k r_0\right\}_{k=0}^{m}$. The corresponding projector is then $P := I - v_0 w^T$. Note that $\widehat{P} = I - v_0 v_0^T$ is a special case of the projector $P = I - v_0 w^T$ in Lemma 1.

At this point two questions arise: Can convergence on appropriate subspaces be guaranteed for preconditioned Krylov solvers with singular preconditioners C^{-1}? And, what effect does this have on the conditioning of the system? The first of these questions is of special importance, because the core of the projection idea is to use $C^{-1} := P\widetilde{C}^{-1}P^T$ as preconditioner which is inevitably a singular matrix.

Remark 3. *Note that C^{-1} is not necessarily invertible and thus not necessarily the inverse of any $C \in \mathbb{R}^{n,n}$. The notation C^{-1} is used to highlight its use as preconditioner.*

In pursuit of the above questions, let C^{-1} be defined as

$$C^{-1} = \widehat{P} Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T \widehat{P}^T, \tag{2.5.30}$$

where $Q^T := [v_0, v_1, \ldots, v_{n-1}]$ is defined as in the proof of Lemma 1, $\widehat{P} = I - v_0 v_0^T$, and $\widetilde{C}^{-1} \in \mathbb{R}^{n-1,n-1}$ is a preconditioner for $\widetilde{A}$. By using repeatedly

$$y = \widehat{P}^T y = \widehat{P} y \quad \forall y \in \mathbb{R}^n : \; v_0^T y = 0$$

and observing

$$v_0^T A y = 0, \quad \text{and} \quad v_0^T Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} y = 0 \quad \forall y \in \mathbb{R}^n,$$

the following becomes apparent.

$$\begin{aligned} C^{-1}A &= \widehat{P}\left(Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} \right) Q^T (\widehat{P}^T A) \\ &= Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T A \\ &= Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{A} \end{bmatrix} Q^T \\ &= Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{A} \end{bmatrix} Q^T \\ &= Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1}\widetilde{A} \end{bmatrix} Q^T. \end{aligned} \tag{2.5.31}$$

Thus, if $\widetilde{C}^{-1}$ is non-singular, then the product $C^{-1}A$ is non-singular on the subspace orthogonal to v_0.

These ideas from the general case can now be applied to analyse the performance of the projected F_p-preconditioner. For $A = K$ the singular vector is $v_0 = \underline{e}_p / \|\underline{e}_p\|$ due to (2.5.24). Under the assumption that the velocity boundary conditions are implemented by reducing the dimension of the FE ansatz space and due to the LBB-condition, the assumption $\text{rank}(K) = n - 1$ holds. Thus, Lemma 1 is applicable to this case. In order to apply (2.5.31), it has to be verified that the F_p-preconditioner (2.5.5) with (2.5.7) applied with the modified pressure space has the structure

$$C_{F_p}^{-1} = Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T,$$

or equivalently

$$\underline{e}_p^T C_{F_p}^{-1} = 0 \quad \text{and} \quad 0 = C_{F_p}^{-1} \underline{e}_p. \tag{2.5.32}$$

Recalling,

$$C_{F_p}^{-1} = \begin{bmatrix} F^{-1} & 0 \\ -X^{-1}B\,F^{-1} & -X^{-1} \end{bmatrix} \tag{2.5.33}$$

$$= \begin{bmatrix} I & 0 \\ 0 & -X^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}, \tag{2.5.34}$$

and that $\underline{e}_p = \left[0^T, \underline{e}^T\right]^T$ in this block notation, (2.5.32) is equivalent to

$$\underline{e}^T X^{-1} = 0 \quad \text{and} \quad 0 = X^{-1}\underline{e}. \tag{2.5.35}$$

Considering $X^{-1}\underline{e} = M_p^{-1}F_p\,A_p^{-1}\underline{e}$, where A_p is the pressure space discretisation of the Laplace operator with homogeneous Neumann boundary conditions everywhere, we observe that $\underline{e}$ is not in the image of the singular matrix A_p and thus that $A_p^{-1}\underline{e}$ is not well defined. However, it is natural to apply the same projection approach to this system. Again there is no difference compared to the case when the constant function is actually removed from the pressure FE space. In this sense $A_p^{-1}\underline{e} = 0$ holds and $X^{-1}e = \underline{0}$ is obtained. Slightly different is the situation for $\underline{e}^T X^{-1}$. The inverse of M_p does exist, thus from this point of view the projection is not necessary. But in order to be consistent with the definition of A_p^{-1}, M_p^{-1} may be defined applying the projection approach[10]. This ensures $\underline{e}^T M_p^{-1} = 0$ and thus $\underline{e}^T X^{-1} = 0$. So under these conditions, i.e. that the inverses of A_p and M_p be defined in the sense of the projected systems $(I - \frac{\underline{e}\underline{e}^T}{\underline{e}^T\underline{e}})^T A_p (I - \frac{\underline{e}\underline{e}^T}{\underline{e}^T\underline{e}})x = (I - \frac{\underline{e}\underline{e}^T}{\underline{e}^T\underline{e}})b$ and M_p analogously, (2.5.35) and consequently (2.5.32) hold.

This gives us the following spectral equivalence.

Lemma 2. *Let $K \in \mathbb{R}^{n,n}$ be the discrete linearised Navier Stokes system (2.5.2) with Dirichlet boundary conditions implemented directly and standard FE pressure space, i.e. ZMPC or equivalent conditions not incorporated into this space, and let $v_0 = \frac{\underline{e}_p}{\|\underline{e}_p\|}$, $\widehat{P} = I - v_0 v_0^T$. Let $Q^T := [v_0, v_1, \ldots, v_{n-1}]$ be defined as in the proof of Lemma 1 and let $\widetilde{C}^{-1} \in \mathbb{R}^{n-1,n-1}$ be a preconditioner with*

$$\underline{\gamma} \leq \frac{\|\widetilde{C}^{-1}\widetilde{A}x\|}{\|x\|} \leq \overline{\gamma} \qquad \forall x \in \mathbb{R}^{n-1} : x \neq 0. \tag{2.5.36}$$

[10] Note that $\underline{e}^T X^{-1} = 0$ does not hold otherwise.

Then

$$\underline{\gamma} \leq \frac{\|C^{-1}Ax\|}{\|x\|} \leq \overline{\gamma} \qquad \forall x \in \mathbb{R}^n : \; x \neq 0, \; v_0^T x = 0, \tag{2.5.37}$$

where C^{-1} *is defined as*

$$C^{-1} = \widehat{P} Q \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T \widehat{P}^T. \tag{2.5.38}$$

Proof. The assertion is implied by (2.5.31) and the construction of Q as orthogonal matrix. □

Lemma 2 shows the equivalence of a preconditioner $\widetilde{C}^{-1}$ constructed with pressure space S_0^h to the projected analogue preconditioner (2.5.38) constructed with S^h. Since optimality of the F_p preconditioner was proven in [29] in the sense of $\widetilde{C}^{-1}$, this implies that the $\widehat{P}$ projected F_p-preconditioner is optimal as well.

One last step remains in order to arrive at a preconditioner that guarantees the ZMPC. The projection approach with $\widehat{P} = I - v_0 v_0^T$ does construct solutions which are orthogonal to solutions with constant pressure, but in the sense of the standard scalar product of $\mathbb{R}^n$ rather than the $L^2(\Omega)$ scalar product. So a modification is required to guarantee the ZMPC. If instead of $\widehat{P}$ a projector $P = I - v_0 w^T$ with weight vector w defined as

$$w = \frac{\widetilde{w}}{\widetilde{w}^T \underline{e}_p}, \tag{2.5.39}$$

where

$$\widetilde{w} := [\widetilde{w}_i]_{i=1}^n,$$
$$\widetilde{w}_i := \begin{cases} 0 & \text{for velocity DOFs} \\ \int_\Omega \theta_i \, \mathrm{d}\Gamma & \text{for pressure DOFs.} \end{cases}$$

Note that P projects orthogonal to $\underline{e}_p$ onto the subspace orthogonal to w, as illustrated in Figure 2.8. The difference between two projected vectors is a multiple of $\underline{e}_p$, thus it does not influence the result of a subsequent multiplication by the singular stiffness matrix K. Therefore, the projected preconditioner

$$C^{-1} = PQ \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{C}^{-1} \end{bmatrix} Q^T P^T \tag{2.5.40}$$

does preserve optimality of the F_p preconditioner and guarantees that GMRES constructs

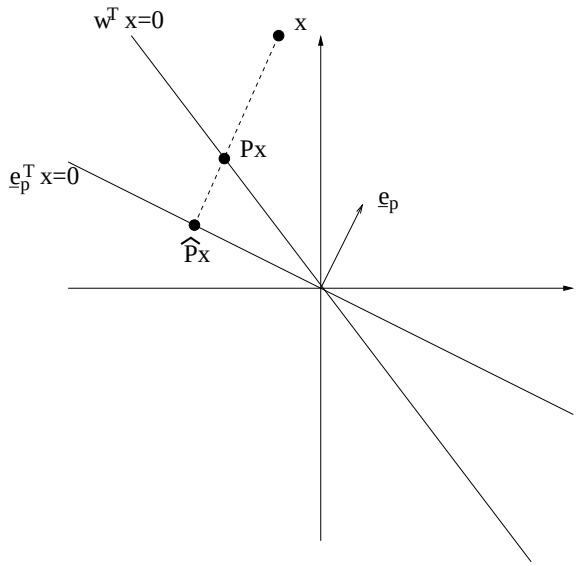


Figure 2.8: Difference between the projectors $\widehat{P}$ and P

a solution fulfilling the ZMPC.

The F_p preconditioner requires a multigrid solver for the subproblems defined by the F-block, see Subsection 2.5.2.2. An alternative approach is to construct a multigrid scheme for the whole linearised discrete Navier-Stokes system. One approach to do so is the subject of the following subsection.

2.5.2.4 Geometric multigrid using the box-smoother

Implementing a geometric multigrid (GMG) solution procedure for the linearised Navier-Stokes equations is a relatively difficult task compared to simpler operators such as the Laplacian [95]. In particular selecting a good smoother is not trivial. For example Gauß-Seidel smoothing, a very common choice for elliptic self-adjoint PDEs, is not possible because of the zero diagonal entries in the pressure-pressure block of the Jacobian K [11]. Even if this difficulty is avoided, for example by the use of artificial compressibility or stabilised methods, the velocity-pressure coupling is not properly accounted for.

The basic idea of the smoother. In [95, pp. 320–322] the so called box-smoother for the Navier-Stokes equations is discussed. In contrast to the Gauß-Seidel smoothing procedure, where only one component of the solution vector is updated at a time, the box smoother updates all unknowns related to a small part of the mesh simultaneously by solving a local (small) system of equations. The basic procedure is summarised in

[11] For inherently stable discretisations the entire pressure-pressure block is zero.

Algorithm 4 Box smoother

Seeking an approximate solution to $Kx = b$, where $K = \begin{bmatrix} F & B_T \\ B & C \end{bmatrix}$.

Using damping parameter $\gamma \in (0,1]$.

Require: Let $\mathscr{I}$ be an index set and $\forall i \in \mathscr{I}$ let s_i denote a set of local degrees of freedom, such that $\bigcup_{i \in \mathscr{I}} s_i$ contains all degrees of freedom of the system.

Set x to an initial guess, $x = x_0$.

for all $i \in \mathscr{I}$ **do**

Collect the K-entries belonging to s_i into K_{loc},

$$K_{\text{loc}} = \begin{bmatrix} F_{\text{loc}} & B_{T,\text{loc}} \\ B_{\text{loc}} & C_{\text{loc}} \end{bmatrix}.$$

Compute the part of $r = b - Kx$ that corresponds to s_i, denote it by r_{loc}.

Solve

$$K_{\text{loc}} x_{\text{loc}} = r_{\text{loc}}. \tag{2.5.41}$$

Add $\gamma\, x_{\text{loc}}$ to the part of x that corresponds to s_i.

end for

Algorithm 4. Note that this algorithm is based on a more general Jacobian matrix K than that arising from a stable FE discretisation, to allow its use in the FV setting later on as well. In a sense this procedure is a generalisation of the Gauß-Seidel smoother, but unfortunately it is computationally more expensive. Note that in other publications this smoother idea is often referred to as Vanka smoother, honouring its first appearance in [97].

Definition of the local systems. The definition of the local degrees of freedom (LDOFs) is an important issue. Since general dense matrix solvers are used, the cost for setting up and solving the local systems grows like n^3, where n is the number of local degrees of freedom. Thus n is desired to be kept very small. On the other hand, it is necessary to define these systems such that sufficient smoothing takes place. In [95] this smoother is applied to a staggered finite difference discretisation. In this setting use of the degrees of freedom associated with the individual mesh cells (see Figure 2.9) yields very small systems (9 by 9) and the number of systems (i.e. $|\mathscr{I}|$) is equal to the number of cells, but the resulting procedure still possesses very good smoothing properties. For the FE discretisation employed in this thesis, i.e. using Taylor-Hood elements, the definition of the local degrees of freedom is not as straightforward. References applying this type of smoother to triangular Taylor-Hood elements are rare. The method is much more frequently applied to lowest order stabilisations, e.g. the $Q_1^{rot} - Q_0$ (Rannacher-Turek) elements, see for ex-

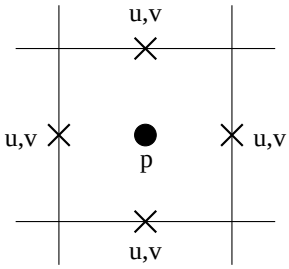


Figure 2.9: Local DOF for a staggered finite difference scheme

ample [96]. In [50] this type of smoother has been applied in multigrid schemes to various types of finite elements for incompressible flow. The main topic of [50] were higher order discretisation schemes, of which the triangular Taylor-Hood elements are one example. The local DOFs in [50] have been defined as

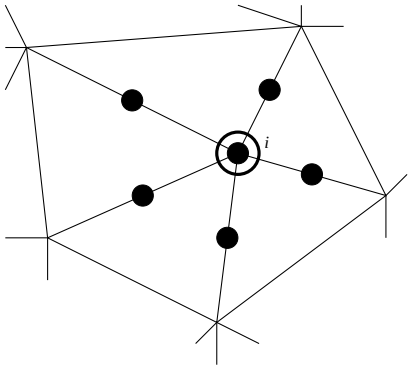
a. all DOFs belonging to an element in case of discontinuous pressure elements

b. a pressure DOF and all velocity degrees of freedom related to it in the case of continuous pressure FE spaces.

Both these approaches are applied to the (continuous pressure) elements considered in this thesis, as summarised in Figure 2.10. In order to refine the definition of "all velocity degrees of freedom related" to a pressure node, two distinct possible cases are considered. If actually all velocity DOFs of the elements containing a pressure node are included in the definition of a local problem, see the *p-support Neumann* part of Figure 2.10, no actually physical meaning can be associated with the local problem. It is neither a local Dirichlet nor a local Neumann problem[12]. Nevertheless the notation *p-support Neumann* is used here because the problems are more similar to the Neumann case. A physically meaningful local problem, i.e. a local Dirichlet problem, is obtained if the velocity DOFs corresponding to the boundary of the local element patch are dropped, see the *p-support Dirichlet* part of Figure 2.10. This definition also has the advantage of a reduced size for the local problems, resulting in lower costs for solving them.

Interpolation and restriction. As the finite element discretisation naturally defines an interpolation, this interpolation operator is used here, i.e. P_2 interpolation for the velocities and P_1 interpolation for the pressure. As usual, the restriction operator is chosen to be the transpose of the interpolation operator.

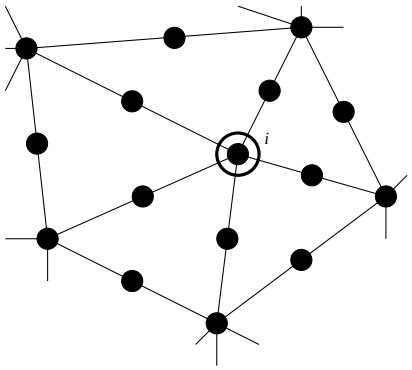
[12] It is not a Neumann problem because the entries of the local matrix corresponding to boundary nodes of the element patch contain contributions from neighbouring elements.



p−support Dirichlet

index set: *pressure nodes*

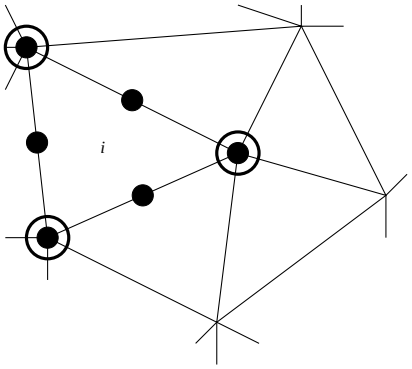
local DOFs: *(p,u,v) at the node, (u,v) at adjacent edge midnodes*



p−support Neumann

index set: *pressure nodes*

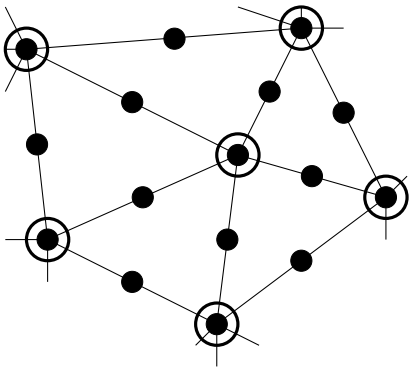
local DOFs: *(p,u,v) at the node, (u,v) of adjacent element*



Element based

index set: *elements*

local DOFs: *all DOFs of the element*



Base mesh and legend:

velocity DOFs (u,v)

pressure DOF (p)

Figure 2.10: Local DOF variants for Taylor-Hood elements

Stabilisation. The issue of stabilisation arises again in this context, because even if the finest mesh of the hierarchy is chosen such that the Galerkin FE discretisation of the convection operator is stable, the discretisations on the coarser meshes may be unstable. In this case not only the convection-reaction-diffusion operator has to be stabilised, but the whole Navier-Stokes system. For this purpose the Streamline Diffusion Finite Element Method (SDFEM) as described and analysed in [94] together with the parameter choice of [34] is used here. In this work only the steady state equations are considered and the pressure FE space consists of continuous functions. Thus [94, eq.(4.6)], modified according to [94, Remark 3.2], simplifies to: find $(\mathbf{u}^h, p^h) \in \mathbf{V}_g^h \times S_0^h$ such that

$$\begin{aligned} 0 =& \nu(\nabla\mathbf{u}^h,\ \nabla\mathbf{v}^h) + (\mathbf{u}^h\cdot\nabla\mathbf{u}^h,\ \mathbf{v}^h) - (p^h,\ \nabla\cdot\mathbf{v}^h) + (q^h,\ \nabla\cdot\mathbf{u}^h) + \nu\alpha\delta(\nabla\cdot\mathbf{u}^h,\ \nabla\cdot\mathbf{v}^h) \\ &+ \frac{1}{\nu}\delta\sum_{T\in\mathscr{T}_h} h_T^2\left(-\nu\Delta\mathbf{u}^h + \mathbf{u}^h\cdot\nabla\mathbf{u}^h + \nabla p^h,\ \ \mathbf{u}^h\cdot\nabla\mathbf{v}^h + \nabla q^h\right)_T \end{aligned} \tag{2.5.42}$$

for all $(\mathbf{v}^h, q^h) \in \mathbf{V}_0^h \times S_0^h$. Equation (2.5.42) replaces the Galerkin FE discretisation (2.3.7). Similar to (2.3.7), (2.5.42) can be expressed in the notation of [40], i.e. using the bilinear forms $a(.,.)$, $b(.,.)$ for the diffusive and divergence terms respectively and the trilinear form $c(.,.,.)$ for the convective terms. The stabilisation terms are denoted by $d_1(.,.,.,.,.,.)$ and $d_2(.,.)$, which are also linear in each of their arguments respectively and are only present in the SDFEM formulation. So (2.5.42) becomes

$$0 = a(\mathbf{u}^h,\mathbf{v}^h) + c(\mathbf{u}^h,\mathbf{u}^h,\mathbf{v}^h) - b(p^h,\mathbf{v}^h) + b(q^h,\mathbf{u}^h) + d_1(\mathbf{u}^h,\mathbf{u}^h,\mathbf{u}^h,p^h,\mathbf{v}^h,q^h) + d_2(\mathbf{u}^h,\mathbf{v}^h), \tag{2.5.43}$$

where

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &:= \nu(\nabla\mathbf{u},\ \nabla\mathbf{v}), \\ c(\mathbf{w},\mathbf{u},\mathbf{v}) &:= (\mathbf{w}\cdot\nabla\mathbf{u},\ \mathbf{v}), \\ b(p,\mathbf{v}) &:= (p,\ \nabla\cdot\mathbf{v}), \\ d_1(\mathbf{w},\mathbf{z},\mathbf{u},p,\mathbf{v},q) &:= \frac{1}{\nu}\delta\sum_{T\in\mathscr{T}_h} h_T^2\left(-\nu\Delta\mathbf{u} + \mathbf{w}\cdot\nabla\mathbf{u} + \nabla p,\ \ \mathbf{z}\cdot\nabla\mathbf{v} + \nabla q\right)_T \\ d_2(\mathbf{u},\mathbf{v}) &:= \nu\alpha\delta(\nabla\cdot\mathbf{u},\ \nabla\cdot\mathbf{v}). \end{aligned}$$

In order to simplify the notation the superscript h will be omitted for the remainder of this subsection, as only discrete functions are considered anyway.

Application of Newton's method to solve the nonlinear system (2.5.43) requires one

to solve

$$\begin{aligned} & a(\mathbf{u}_{k+1},\mathbf{v}) + c(\mathbf{u}_{k+1},\mathbf{u}_k,\mathbf{v}) + c(\mathbf{u}_k,\mathbf{u}_{k+1},\mathbf{v}) + b(q,\mathbf{u}_{k+1}) - b(p_{k+1},\mathbf{v}) \\ & + d_1(\mathbf{u}_{k+1},\mathbf{u}_k,\mathbf{u}_k,p_k,\mathbf{v},q) + d_1(\mathbf{u}_k,\mathbf{u}_{k+1},\mathbf{u}_k,p_k,\mathbf{v},q) + d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_{k+1},p_k,\mathbf{v},q) \\ & + d_2(\mathbf{u}_{k+1},\mathbf{v}) + d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_k,p_{k+1},\mathbf{v},q) \\ = \; & c(\mathbf{u}_k,\mathbf{u}_k,\mathbf{v}) + 3d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_k,p_k,\mathbf{v},q). \end{aligned} \tag{2.5.44}$$

However, in the computational experiments performed in this work the Newton iteration (2.5.44) was found to be divergent even for moderate Re. Thus for the results presented later in this subsection the more robust Picard iteration has been applied:

$$\begin{aligned} & a(\mathbf{u}_{k+1},\mathbf{v}) + c(\mathbf{u}_k,\mathbf{u}_{k+1},\mathbf{v}) + b(q,\mathbf{u}_{k+1}) - b(p_{k+1},\mathbf{v}) \\ & + d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_{k+1},p_k,\mathbf{v},q) + d_2(\mathbf{u}_{k+1},\mathbf{v}) + d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_k,p_{k+1},\mathbf{v},q) \\ = \; & d_1(\mathbf{u}_k,\mathbf{u}_k,\mathbf{u}_k,p_k,\mathbf{v},q). \end{aligned} \tag{2.5.45}$$

The parameter choice of [34, equation (17)], with constant according to [34, Experiment 5.5], gives in the notation of (2.5.42)

$$\alpha = \frac{1}{\nu^2}, \quad \delta = \nu. \tag{2.5.46}$$

In order to verify the correct implementation of this discretisation scheme it has been applied to the standard test problem of driven cavity flow, see Section 2.7.2.1.

Variants arising by the choice of smoother parameters. In our tests all three smoother variants, *p-support Dirichlet*, *p-support Neumann* and *elem based*, required significant damping in order to get robust and fast convergence for the multigrid preconditioned GMRES solver. As the most successful schemes are obtained for damping factors of 0.6 up to 0.8 (see result tables later in this subsection), the question arises if this might be due to a systematic requirement of the approach. Indeed, such a systematic scaling of the updates can be found in [79], where the smoothing properties of this type of smoother applied in a Jacobi fashion (rather than in a Gauß-Seidel fashion) have been analysed for mixed FEM discretisations of the Stokes equations. There the velocity updates are scaled according to the number of appearances of each velocity DOF in a smoother sweep. To achieve this in a symmetry-preserving manner (2.5.41) is replaced by

$$x_{\text{loc}} = D^{-1}K_{\text{loc}}^{-1}D^{-1}r_{\text{loc}}, \tag{2.5.47}$$

category	label	function
sorting	*nosort*	one sweep unsorted
	(no label)	one sweep sorted by (x,y)
	2sort	one sweep sorted by (x,y) and one by $(-y,-x)$
local DOFs	*elem*	element based
	p-dir	p-support Dirichlet
	p-neum	p-support Neumann
weighting	*Vweights*	weighting of velocity DOFs according to (2.5.47)
	noVweights	no weighting, basic (2.5.41)
damping	*dam* γ_i	damping parameter $\gamma = \gamma_i$

Table 2.7: Overview box-smoother parameters

where D is a diagonal matrix with $\sqrt{(\text{number of appearances})}$ as entries for the velocity DOFs and ones for the pressure DOFs. As this introduces weighting of the velocity DOFs into the smoother, this variant is denoted by *Vweights* in the tables at the end of this subsection.

Further, due to the potentially dominating convection, the influence of the order in which the local problems appear in the Gauß-Seidel sweeps may again be of importance, as in Subsection 2.5.2.2. In order to analyse this issue, tests have been performed with the cell patches in their natural order created by the refinement algorithm[13] (label *nosort*) and for comparison sorted by increasing x coordinate first and for equal x by increasing y coordinate (no label). Motivated by the approach in [68] a third sorting approach is also considered in the tests, for which after a sweep sorted by increasing x and increasing y, a second smoother sweep is performed sorted by decreasing y and for equal y sorted by decreasing x. This third sorting mode is labelled *2sort* in the result tables. Since the post-smooth sweeps the mesh in the opposite order than the pre-smooth, information can travel in all four basic directions (right, down, up, left) on each level, within one multigrid cycle.

The final parameter in this multigrid preconditioner is the damping parameter $\gamma \in (0,1]$ of Algorithm 4. As the undamped variant ($\gamma = 1$) was unreliable in all tests performed in the preparation of this thesis, only the values $\gamma_i = i/10$, $i = 1,\dots 9$ have been considered in the production of the result tables at the end of this subsection.

Table 2.7 summarises all the parameters considered in the result tables.

Test results. The smoother described in this subsection has been tested in a multigrid preconditioner for a GMRES solver for the SDFEM discretised, Picard linearised incom-

[13] New nodes and elements are always inserted at the end of the current lists, while one of the four child elements replaces the parent element.

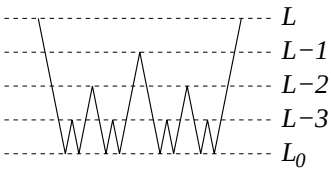


Figure 2.11: Modified W-cycle

pressible Navier-Stokes equations (2.5.45). To avoid problems with unreliable approximations on coarse meshes while minimising sacrifices of performance, a variant of a W-cycle has been implemented. It comprises (see Figure 2.11) of one pre-smooth on the finest mesh L, one coarse grid correction on $L-1$, and one post-smooth. The coarse grid correction on $L-1$ itself is a proper W-cycle, with pre-smooth, coarse grid correction $(L-2)$, post-smooth, pre-smooth, second coarse grid correction $(L-2)$ and a final post-smooth. The coarse grid correction on the lower level is performed according to the same scheme. On the coarsest level (L_0) the systems are solved directly by a general purpose banded matrix LU-solver from `LAPACK` [60]. Compared to the standard W-cycle this variant performs less smoothing on the finest level, thus resulting in lower computational cost and leaving more of the work on the finest level to the outer GMRES solver.

All combinations of the smoother parameters summarised in Table 2.7 have been tested for two test problems, a lid driven cavity at $Re = 100$ and at $Re = 1000$, see tables 2.8 and 2.9. The linear systems have been solved to a relative accuracy of 10^{-5}. The tables list the maximum number of iterations on each level of the mesh refinement hierarchy, i.e. that of the Picard step with the most GMRES iterations. The row *total time* lists the overall solution time for the nonlinear Navier-Stokes system, from coarsest to finest level. The listed results are for the best damping parameter in each test series, the values of which are indicated in the column captions.

From the comparison of tables 2.8 and 2.9 it becomes visible that the *p-support Dirichlet* variant of the smoother is the best in terms of overall compute time to solution. The non-sorted and the simple sorted Gauß-Seidel like sweeps do not differ significantly in the number of iterations or the compute time. The larger size of the local problems in the case of the *p-support Neumann* and *element based* variants implies more costs for solving the local problems, which is not compensated for by faster convergence of the solver. Comparing the *Vweights* and *noVweights* variants of each smoother it is evident that the *noVweights* variant is better in each case. The comparison does not show a major difference in performance of the sorted nor non-sorted variants of each respective smoother. Even more, the additional smoothing in the *2sort* variant does not significantly

#dof	nosort elem Vweights dam 0.8	nosort elem noVweights dam 0.8	nosort p-dir Vweights dam 0.7	nosort p-dir noVweights dam 0.6	nosort p-neum Vweights dam 0.7	nosort p-neum noVweights dam 0.7	elem Vweights dam 0.8	elem noVweights dam 0.8	p-dir Vweights dam 0.6	p-dir noVweights dam 0.8	p-neum Vweights dam 0.7	p-neum noVweights dam 0.6	2sort elem noVweights dam 0.8	2sort p-dir noVweights dam 0.6	2sort p-neum noVweights dam 0.7
	max(it)														
659	21	16	20	16	20	15	21	16	21	15	20	16	13	13	13
2467	25	19	23	20	23	18	25	20	25	18	23	19	16	16	15
9539	26	19	24	20	24	19	25	20	26	19	24	20	17	17	16
37507	26	20	25	21	24	20	26	20	26	20	25	21	18	18	17
total time	3.1e+02	2.8e+02	1.7e+02	1.4e+02	1.8e+02	1.5e+02	3.1e+02	2.6e+02	1.7e+02	1.4e+02	1.8e+02	1.6e+02	4.5e+02	2.4e+02	2.6e+02

Table 2.8: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 100$, FEM discretisation, Picard linearisation, W-cycle, best damping parameter in each case

	$\max(it)$, #dof 659	2467	9539	37507	total time
2sort p-neum noVweights dam 0.7	44	75	102	117	2.7e+03
2sort p-dir noVweights dam 0.7	46	76	103	119	2.4e+03
2sort elem noVweights dam 0.*	0	0	0	0	0.0e+00
p-neum noVweights dam 0.7	52	87	122	141	1.6e+03
p-neum Vweights dam 0.7	65	108	147	167	2.0e+03
p-dir noVweights dam 0.7	55	95	126	143	1.5e+03
p-dir Vweights dam 0.6	71	119	162	171	1.9e+03
elem noVweights dam 0.8	55	91	120	136	2.8e+03
elem Vweights dam 0.7	72	122	153	174	3.7e+03
nosort p-neum noVweights dam 0.7	53	89	120	140	1.6e+03
nosort p-neum Vweights dam 0.6	69	114	155	172	2.0e+03
nosort p-dir noVweights dam 0.7	57	97	129	146	1.5e+03
nosort p-dir Vweights dam 0.7	68	114	158	167	1.9e+03
nosort elem noVweights dam 0.8	55	89	117	133	2.9e+03
nosort elem Vweights dam 0.8	68	116	152	172	3.6e+03

Table 2.9: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 1000$, FEM discretisation, Picard linearisation, W-cycle, best damping parameter in each case

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
#dof	max(it)								
659	31	24	21	19	17	16	16	0	15
2467	36	28	25	22	21	20	19	0	18
9539	39	30	25	23	21	20	20	0	19
37507	39	30	26	24	22	21	20	0	19
total time	2.5e+02	1.9e+02	1.6e+02	1.5e+02	1.5e+02	1.4e+02	1.4e+02	0.0e+00	1.4e+02

Table 2.10: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 100$, FEM discretisation, Picard linearisation, W-cycle, smoother variant *nosort p-dir noVweights*, various damping parameters γ

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
#dof	max(it)								
659	98	82	72	66	62	59	57	56	54
2467	174	138	121	110	103	100	97	93	90
9539	262	182	163	150	141	135	129	126	123
37507	273	187	171	193	166	149	146	144	142
total time	2.8e+03	2.2e+03	1.9e+03	2.1e+03	1.7e+03	1.6e+03	1.5e+03	1.5e+03	1.5e+03

Table 2.11: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 1000$, FEM discretisation, Picard linearisation, W-cycle, smoother variant *nosort p-dir noVweights*, various damping parameters γ

reduce the number of iterations required, and thus the additional cost for this approach can not be justified on the basis of these tests. Note that the *2sort elem noVweights* column in Table 2.9 shows zero iteration counts, because for all tested damping parameters the solver failed or took excessively long.

The only major difference between the results of the two best smoothing variants *nosort p-dir noVweights* and *p-dir noVweights* lies in the most successful damping parameter. To allow assessment of the robustness of these methods tables 2.10, 2.11, 2.12 and 2.13 list the results for all tested damping parameters. Both variants fail outright for some damping parameters in the $Re = 100$ case. The fact that these failing damping parameters are in a similar range as the most successful parameters indicates that even these best parameter choices may not be particularly reliable.

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
#dof	max(it)								
659	31	0	21	0	17	16	0	15	15
2467	36	0	24	0	21	20	0	18	18
9539	39	0	25	0	21	20	0	19	19
37507	39	0	26	0	22	21	0	20	19
total time	2.6e+02	0.0e+00	1.7e+02	0.0e+00	1.5e+02	1.5e+02	0.0e+00	1.5e+02	1.4e+02

Table 2.12: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 100$, FEM discretisation, Picard linearisation, W-cycle, smoother variant *sorted p-dir noVweights*, various damping parameters γ

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
#dof	max(it)								
659	98	82	72	65	60	57	55	54	52
2467	174	138	120	109	101	98	95	90	87
9539	261	180	160	147	138	131	126	122	119
37507	273	186	170	159	152	147	143	141	139
total time	2.8e+03	2.2e+03	1.9e+03	1.8e+03	1.6e+03	1.6e+03	1.5e+03	1.5e+03	1.4e+03

Table 2.13: Iteration counts for box-smoother multigrid preconditioned GMRES, driven cavity at $Re = 1000$, FEM discretisation, Picard linearisation, W-cycle, smoother variant *sorted p-dir noVweights*, various damping parameters γ

It may be observed that even the best of these tested smoother variants does not produce uniform behaviour with respect to the Reynolds number Re. While for this test problem the solver performs reasonably well for $Re = 100$, iteration numbers increase drastically for $Re = 1000$. In addition more Picard iterations are required as well for the higher Re due to the stronger dominance of the nonlinear convection term in this case.

We emphasise that all the results in this subsection are for the Picard iteration case only, but for the discrete adjoint method the true Jacobian of the nonlinear discretisation scheme has to be used in order to obtain sensitivity values which are consistent with the discrete functional evaluations themselves. Thus it has to be concluded that in conjunction with the finite element discretisation at hand this approach does not appear to be suitable to construct an efficient solver base for an adjoint code. However, in Section 2.5.3.2 it will be demonstrated that this solver approach does indeed have applications where it produces a reliable and efficient base for an adjoint code, i.e. for the finite volume discretisation described in Section 2.4.

2.5.3 Finite volume method

2.5.3.1 The F_p preconditioner and the FV scheme

Unfortunately the F_p preconditioning technique which has proved quite successful for the FE discretisation turns out not to deliver efficient results for the FV scheme presented in Section 2.4.2. To investigate the reason for this, consider the $(3,3)$ block of the stabilised operator (2.4.11), which is essentially a (directionally-scaled) diffusion operator scaled by $h\beta/2$. If the directional scaling is ignored (or assumed $u = v$), then this becomes a Laplacian operator, scaled proportional to h, acting on the pressure. This observation forms the basis for our analysis of why the F_p preconditioner does not produce h-independent behaviour for the problems arising from the Newton linearisation of the FV scheme under consideration.

Let us recall a few features of this preconditioning technique. As usual, let the ordering of the discrete variables be such that the linearised discrete system has the block form

$$\begin{bmatrix} F & B_T \\ B & C \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{p} \end{bmatrix} = \begin{bmatrix} r_u \\ r_p \end{bmatrix}. \tag{2.5.48}$$

In [52] the F_p preconditioner is compared to optimal Schur complement preconditioning techniques for the Stokes systems. The F_p preconditioner's most significant ingredient is to use

$$X_{F_p}^{-1} = M_p^{-1} F_p A_p^{-1} \tag{2.5.49}$$

as a preconditioner for the Schur complement

$$S = BF^{-1}B_T - C. \tag{2.5.50}$$

This can be seen as a natural extension from the Stokes case, where

$$X_M^{-1} := \nu M_p^{-1} \tag{2.5.51}$$

is used. Note that the $F_p A_p^{-1}$ term defaults to ν in the Stokes limit, where convection terms can be neglected, giving $F_p \approx \nu A_p$.

To analyse the effect of the stabilisation terms, we consider an analogous FEM-discretised Stokes problem and follow the arguments in [102] and [103]. Good performance of the Stokes preconditioner X_M^{-1} depends on the spectral equivalence of M_p to the Schur complement S, i.e. there exist constants $\underline{\gamma} > 0$ and $\overline{\gamma}$ independent of h such that

$$\underline{\gamma} \leq \frac{p^T(BA^{-1}B_T - C)p}{p^T M_p p} \leq \overline{\gamma} \tag{2.5.52}$$

for all pressure vectors $p \in P_h, p \neq 0$, $P_h := \{p : p^T e = 0\}$, $e := [1,\ldots,1]^T$. For LBB-unstable finite element discretisations the non-stabilised version (C=0) of (2.5.52) holds only for $\underline{\gamma} = 0$. In [102] a way to overcome this is suggested, based upon splitting the pressure space P_h into a stable part Q_h and mode part Q_h^T such that P_h is the orthogonal sum of the two spaces and (2.5.52) holds on Q_h even with $C = 0$. Once this is achieved the following restrictions

$$\phi^2 \leq \frac{p^T C p}{p^T M_p p} \leq \Phi^2 \tag{2.5.53}$$

$$\frac{p^T(BA^{-1}B_T)p}{p^T M_p p} \leq \alpha^2 \tag{2.5.54}$$

for all $p \in Q_h^T$, with h-independent constants $\phi > 0, \Phi, \alpha$ ensure that (2.5.52) holds on $P_h \setminus \{0\}$.

It is well established that one of the unstable pressure modes of standard methods on

a uniform mesh is the so-called checker board mode:

$$\begin{bmatrix} + & - & + & - & + & - & + & - & + \\ - & + & - & + & - & + & - & + & - \\ + & - & + & - & + & - & + & - & + \\ - & + & - & + & - & + & - & + & - \\ + & - & + & - & + & - & + & - & + \\ - & + & - & + & - & + & - & + & - \end{bmatrix},$$

i.e. neighbouring nodes have pressure of opposite sign but the same absolute value ($= 1$). Let the corresponding pressure vector be denoted as p_m and consider this mode for a FEM discretisation of the operator

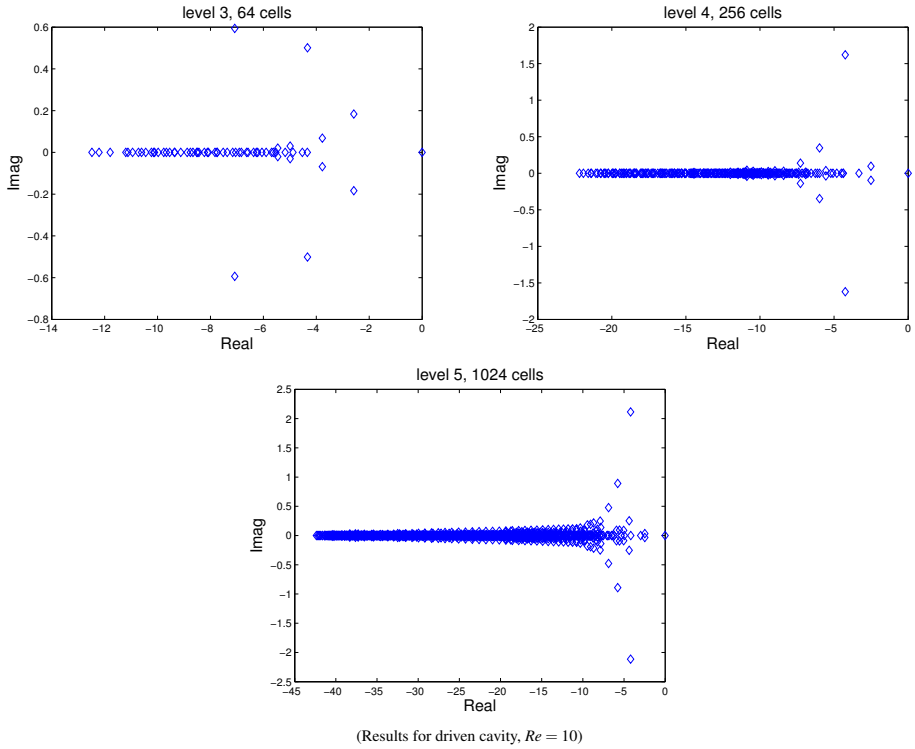
$$\begin{bmatrix} -\Delta & & \partial_x \\ & -\Delta & \partial_y \\ \partial_x & \partial_y & -c_1 h\Delta \end{bmatrix}, \tag{2.5.55}$$

which is equivalent to $C = c_1 h A_p$ and forms an analogue to the FV scheme (2.4.11). Thus, on a unit square, by considering the periodic properties of the function p_m^h (corresponding to p_m) and integrating on a particular linear element K, it follows

$$\begin{aligned} \frac{p_m^T C p_m}{p_m^T M_p p_m} &= \frac{c_1 h \int_\Omega \|\nabla p_m^h\|^2 \, \mathrm{d}\Omega}{\int_\Omega |p_m^h|^2 \, \mathrm{d}\Omega} \\ &\approx \frac{c_1 h \frac{1}{h^2} \int_K \|\nabla p_m^h\|^2 \, \mathrm{d}\Omega}{\frac{1}{h^2} \int_K |p_m^h|^2 \, \mathrm{d}\Omega} \\ &\approx \frac{c_1 h \frac{1}{h^2} 4}{\frac{1}{h^2} \frac{1}{6} h^2} \\ &= \mathscr{O}(h^{-1}), \end{aligned} \tag{2.5.56}$$

which implies that the upper bound in (2.5.53) can not hold uniformly in h. This indicates that F_p-type preconditioning will not achieve h-uniform performance if the stabilisation term C is a multiple of a Laplacian scaled by h.

Even though we have highly simplified the problem, the resulting $\mathscr{O}(h^{-1})$ upper bound appears to be representative and numerical experiments confirm a spreading of the spectrum of that order. This is demonstrated in Figure 2.12 where the eigenvalues of the preconditioned Schur complement $X_{F_p}^{-1} S$ for different levels of refinement of the mesh are plotted in the complex plane. It may be observed that the real part of the eigenvalue of largest absolute value (the left-most in each plot) roughly doubles with each refinement:



(Results for driven cavity, $Re = 10$)

Figure 2.12: Spectra of the F_p-preconditioned Schur complement, $X_{F_p}^{-1}S$

ca. -12, -22, -43. In particular, the plot for the finest mesh, level 5, demonstrates that there is no significant clustering of the eigenvalues as has been observed in the finite element case (figures 2.6 and 2.7).

We remark that in [102, Section 4.2] a stabilisation for FE methods is discussed which is basically $C = c_2 h^2 A_p$ and that for this second order stabilisation the above arguments involving p_m revert to $\mathscr{O}(1)$ and good performance of this type of preconditioning has been reported, e.g. in [103]. This leaves the possibility that F_p preconditioning might still perform well for a *second order* FV scheme. However, as the multigrid method which is considered in Section 2.5.3.2 performs optimally, this hypothesis is not followed up here.

A further problem with the F_p preconditioning technique arises from the pressure time terms in the FV scheme. In [52] it is recommended to use only homogeneous Neumann boundary conditions in the definition of the operators F_p and A_p, in order to reflect the property of the incompressible Navier-Stokes equations that the pressure is only defined up to an additive constant. For infinite time-steps this makes both of these operators singular, but the vectors that A_p^{-1} is applied to are in the range of A_p, so the solution is well defined up to a constant and this additive constant does not influence the result of the subsequent multiplication by F_p.

The FV approach under consideration, with artificial compressibility and finite time-

steps in the pressure terms, results in a somewhat different situation. The time terms uniquely define the pressure at the next time-step, even though additive constants have no influence on the residual with respect to steady state. Time terms enter F_p quite naturally but not A_p. Applying A_p^{-1} with Neumann boundary conditions requires the right-hand side to be in the range of A_p, thus care needs to be taken of this issue.

The motivation arguments (2.5.6) (as taken from [52]), used in the derivation of the F_p preconditioner, would suggest something like

$$X^{-1}_{\text{with p time term}} = \left(F_p^{-1}A_p + \alpha M_p\right)^{-1}, \tag{2.5.57}$$

because the time term adds a zero order operator $S = BF^{-1}B^T + \alpha M_p$. The expression (2.5.57) can be approximated for very large or very small time-steps by neglecting the appropriate term. Yet, for time-steps of intermediate size, no satisfactory approximation is known to the author.

2.5.3.2 Geometric multigrid using the box-smoother

A geometric multigrid (GMG) solution procedure based on the box-smoother (Algorithm 4, Subsection 2.5.2.4) can be designed for the linearised finite volume discretisation of the Navier-Stokes equations as well. Before we continue to define the ingredients of this approach properly we emphasise that, in contrast to the finite element case from Subsection 2.5.2.4, this approach does work reliably in a straightforward manner for the finite volume discretisation described in Section 2.4.

A canonical way of defining the local degrees of freedom (local DOFs) for Algorithm 4 is to sweep over all mesh cells and for each cell to use all DOFs of the cell itself and all neighbouring cells, as depicted in Figure 2.13. This results in local equation systems with 15 unknowns for each interior cell patch, as these consist of 5 cells with 3 unknowns each, see Figure 2.13. More complicated definitions are of course feasible as well, but unnecessary, as this simple approach works reasonably well.

The interpolation operator for the multigrid which is used in this work is defined analogously to the case of standard bilinear elements in FEM. That is, the interpolated value $q(x)$ in child cell x is defined by the values in the parent cells a, b, c, d as

$$q(x) = \frac{9}{16}q(a) + \frac{3}{16}q(b) + \frac{1}{16}q(c) + \frac{3}{16}q(d),$$

see Figure 2.14. The interpolation for the remaining child cells is defined by rotating the setup from Figure 2.14 accordingly. The restriction operator is defined as the transpose of

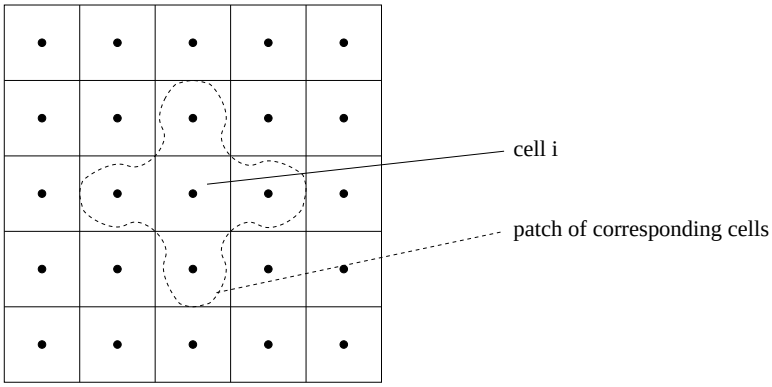


Figure 2.13: Cell patches used in the Box-smoother

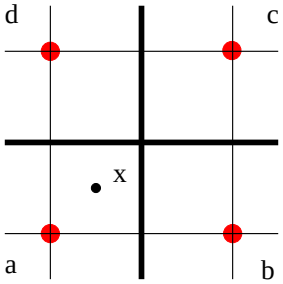


Figure 2.14: Cell centres a, b, c, d of parent cells to child cell x

the interpolation operator.

This smoother has been tested as a multigrid preconditioner for a GMRES solver for the linearised systems $(J+D)x = r$ from Algorithm 1. Table 2.14 lists some results for this type of solution procedure for a driven cavity problem at different Reynolds numbers Re and using different multigrid cycles as preconditioners. All results use a fixed time-step size of $\tau_k = 10$. A fixed τ_k has been chosen to allow simple comparison of the multigrid strategy for different Reynolds numbers and mesh refinement levels. The value chosen was found to form a reasonable compromise between computational expense for the solution of the linear systems and expense due to slow convergence to steady state at the higher end of the range of Reynolds numbers considered here. The linear systems are solved to a relative reduction to 10^{-3} of the preconditioned residual. This rather inaccurate solve is justified as finite time-steps are used, which limits the convergence of the nonlinear residual anyway. On the other hand even more inaccurate solves tend to hinder nonlinear convergence due to the non-uniform behaviour of the multigrid preconditioner. Tests for a range of problems and Reynolds numbers led to the choice of 10^{-3} as stopping criterion as it proved a reasonable trade-off between robustness and computational cost.

Time-steps according to Algorithm 1 are performed until the overall residual on the

current mesh is reduced to 10^{-8} of its initial norm. It is evident that for higher *Re* more time-steps are necessary. This reflects the increasingly nonlinear nature of the problem and longer time-scales for the diffusion of initial perturbations. Further, it becomes more pronounced on the finer meshes where artificial diffusion becomes small.

All tests listed in the Table 2.14 have been performed with damping parameter $\gamma = 1.0$, that is without damping. The first three columns of the table show results for the use of a simple V-cycle as preconditioner. While this performs well at low *Re*, its performance deteriorates significantly for higher *Re*. This deterioration may be attributed to the bad approximation of even the low frequency solution components on the coarse meshes due to the low order stabilisation terms introduced by the Roe scheme (2.4.11). Examples of the convergence of the solution of a lid driven cavity problem are given in Section 2.7.2.1 where the $Re = 1000$ results demonstrate this effect. To verify that this is actually the source of the deterioration the results of the simple V-cycle (a) (with one pre- and one post-smoothing step) are compared to those of a V-cycle with more smoothing (b) (one pre- and two post-smoothing steps) and a W-cycle (c) (pre-smoothing, coarse grid correction (CGC), post- and pre-smoothing, one more CGC, post-smoothing), see the last three columns in Table 2.14.

The improved smoothing in (b) produces lower iteration counts for GMRES but is not sufficient to avoid deterioration as the mesh is refined. On the other hand the more expensive W-cycle (c) achieves this and results in what appear to be mesh independent iteration counts, compensating for the higher cost per cycle. This is illustrated by the total CPU time in each of the cases, which is the sum of the times taken on each level listed.

Note that the performance of the smoother for convection-dominated problems (high *Re*) may be improved by ordering the cells stream-wise, from inflow to outflow. However for the driven cavity problem it is difficult to define such an ordering due to the recirculation and the absence of in- and outflow boundaries. Therefore the results presented in Table 2.14 were produced with a simple coordinate based ordering which conforms with the flow direction at the boundary that drives the flow (the lid).

Overall we conclude that the box smoother used in a W-cycle as preconditioner for a GMRES solver provides a satisfying basis for a solver in the sense that the costs for the linear solves are linear in the number of unknowns and, in the Reynolds number range considered here, almost independent of *Re*.

		Re				
level	cells	10	100	1000	1000	1000
2	64	5×3	6×3	6×3	6×3	6×2.8
3	256	4×3	6×4	7×5	7×3.7	7×4.1
4	1024	4×3	6×4	8×6	8×4.8	8×4.8
5	4096	4×3	6×5.5	9×7.3	9×9	9×5.2
6	16384	4×4	6×6	11×10.5	11×8.5	11×6.1
7	65536	4×4.5	6×7	12×15.7	12×12.1	12×6.2
8	262144	6×5.5	6×7	21×21.1	14×15.1	14×5.1
		(time-steps) × (average GMRES iterations)				
total CPU time		32m52s	42m13s	197m38s	228m03s	118m41s
final residual		6.1e-09	1.4e-10	9.6e-11	9.5e-11	9.5e-11
cycle		a	a	a	b	c

cycle types:		
a	V-cycle	down, CGC, up
b	V-cycle	down, CGC, up, down
c	W-cycle	2×(down, CGC, up)

smoother:	
up	box smoother, cell ordering forward
down	box smoother, cell ordering backward

Table 2.14: Iteration counts for the FV solver

2.6 Application of the discrete adjoint method

2.6.1 The discrete adjoint method and regularisation of the incompressible Navier-Stokes system

Both discretisations of the incompressible Navier-Stokes equations (2.1.4) considered in this thesis, the finite element (FE) method from Section 2.3 and the finite volume (FV) discretisation from Section 2.4, require a regularisation in order to arrive at non-singular linearised systems. In both cases the source of the singularity is that the incompressible Navier-Stokes equations define the pressure only up to an additive constant. This results in a single zero eigenvalue for the linearised systems, with a corresponding right eigenvector v_R of constant pressure and zero velocity, $v_R = e_p$. The corresponding left eigenvector v_L forms an analogue to the condition that the net mass influx into the domain is zero, or equivalently that there is no net mass production in the domain, because a solution to the linear systems exists only if the right-hand side b satisfies $v_L^T b = 0$.

For the forward solution procedures this singularity is treated in slightly different ways for the two discretisations, projection at each linear solver iteration in the FE case and projection at each nonlinear iteration in the FV case. Before individual implementation issues of the discrete adjoint method for these discretisations are discussed we will analyse the issues resulting from the regularisation in a unified way.

For this purpose let the non-regularised discrete equations define the nonlinear residual $R(\omega, \mathscr{F})$, where ω is the vector of discrete state variables of the flow and $\mathscr{F}$ are the shape parameters as throughout this thesis. Let us consider the discrete equations as regularised by a Lagrange multiplier approach, defining an augmented residual $\widetilde{R}(\widetilde{\omega}, \mathscr{F})$ and augmented state $\widetilde{\omega}$ as

$$\widetilde{R}(\widetilde{\omega}, \mathscr{F}) := \begin{bmatrix} R + \lambda w_p \\ w_p^T \omega \end{bmatrix} \qquad (2.6.1)$$

$$\widetilde{\omega} := \begin{bmatrix} \omega \\ \lambda \end{bmatrix}, \qquad \lambda \in \mathbb{R}.$$

The regularising weight vector w_p must be non-orthogonal to both the singular left and right eigenvectors of the system. Then $\widetilde{R} = 0$ defines a unique solution $\widetilde{\omega}$ because $R = 0$ defines a solution ω unique up to an additive constant pressure, which is fixed by the condition $w_p^T \omega = 0$ and the resulting unique solution is extended to $\widetilde{\omega}$ by $\lambda = 0$. Thus the case $\lambda \neq 0$ can only arise if no solution exists to the non-regularised system $R = 0$. Since

$\widetilde{R}$ defines a unique solution, its Jacobian $\widetilde{J}$ is non-singular. It has the block structure

$$\widetilde{J} := \frac{\partial \widetilde{R}}{\partial \widetilde{\omega}} = \begin{bmatrix} J & w_p \\ w_p^T & 0 \end{bmatrix}, \tag{2.6.2}$$

where $J := \frac{\partial R}{\partial \omega}$.

Now let us apply the discrete adjoint technique to the augmented system $\widetilde{R} = 0$. As described in Section 1.1, the overall derivatives $\mathrm{D}I/\mathrm{D}\mathscr{F}$ of the performance criterion $I := \widetilde{I}(\widetilde{\omega}(\mathscr{F}), \mathscr{F})$ can be evaluated by solving the discrete adjoint equation

$$\left[\frac{\partial \widetilde{R}}{\partial \widetilde{\omega}}\right]^T \widetilde{\Psi} = \frac{\partial \widetilde{I}}{\partial \widetilde{\omega}} \tag{2.6.3a}$$

and computing

$$\frac{\mathrm{D}I}{\mathrm{D}\mathscr{F}} = \frac{\partial \widetilde{I}^T}{\partial \mathscr{F}} - \widetilde{\Psi}^T \frac{\partial \widetilde{R}}{\partial \mathscr{F}}. \tag{2.6.3b}$$

First of all, two properties of $\partial \widetilde{I}/\partial \widetilde{\omega}$ may be pointed out. Since practically relevant performance criteria do not depend on the regularisation variable λ, $\partial \widetilde{I}/\partial \lambda = 0$ is obvious. Thus the adjoint equation (2.6.3a) has the structure

$$\begin{bmatrix} J^T & w_p \\ w_p^T & 0 \end{bmatrix} \begin{bmatrix} \Psi_\omega \\ \Psi_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \widetilde{I}}{\partial \omega} \\ 0 \end{bmatrix}. \tag{2.6.4}$$

Further, as the incompressible Navier-Stokes equations define the pressure only up to an additive constant, the static pressure, meaningful performance criteria should not depend on this static pressure. In our notation

$$e_p^T \frac{\partial \widetilde{I}}{\partial \omega} = 0.$$

Thus, the right-hand side of the singular system

$$J^T \Psi_\omega = \frac{\partial \widetilde{I}}{\partial \omega} \tag{2.6.5}$$

is orthogonal to the singular vector $v_R = e_p$ and hence the system has a solution

$$\Psi_\omega = \Psi_\omega^* + s\, v_L, \tag{2.6.6}$$

which is unique up to multiples of the singular left eigenvector v_L of J, as represented by the scalar parameter $s \in \mathbb{R}$. The last block row of the adjoint equation (2.6.4), $\Psi_\omega^T w_p = 0$, fixes the remaining degree of freedom, the parameter s, so that a unique solution $\widetilde{\Psi} = [\Psi_\omega, \Psi_\lambda]$ for the adjoint equation (2.6.3a) exists which fulfils $\Psi_\lambda = 0$. This represents an interesting fact as (2.6.3b) implies that the derivative of the residual of the last block row of (2.6.1), the regularising equation $w_p^T \omega = 0$, has no influence on the derivative $\mathrm{D}I/\mathrm{D}\mathscr{F}$. Yet, the regularisation is important in order to get correct results for the sensitivities, which can be verified by comparison with finite difference results. To illustrate this, consider the solutions (2.6.6) to the adjoint (2.6.5) of the singular system again, which leads to

$$\Psi_\omega^T \frac{\partial R}{\partial \mathscr{F}} = (\Psi_\omega^*)^T \frac{\partial R}{\partial \mathscr{F}} + s \left(v_L^T \frac{\partial R}{\partial \mathscr{F}} \right).$$

As $v_L^T (\partial R / \partial \mathscr{F}) = 0$ may not generally be the case, the non-uniqueness of Ψ_ω would have an influence on $\mathrm{D}I/\mathrm{D}\mathscr{F}$ in the adjoint representation (2.6.3b). This deficiency is prevented by the regularisation.

Now that this common issue has been discussed the remaining two subsections will detail some of the aspects which are specific to the application of the discrete adjoint method for the individual discretisations considered in this work.

2.6.2 The discrete adjoint method applied in the finite element context

Since the finite element (FE) discretisation in this work uses unstructured triangular meshes combined with an automatic mesh generation and the mesh deformation approach, as briefly outlined in Section 2.2.6, it can be applied to a wide range of geometries. This approach implies that the dependency of the discrete approximation of the performance functional I is actually realised by a chain of dependencies. The shape parameter vector $\mathscr{F}$ defines a mesh with node positions $\underline{\mathbf{s}}$, which defines the discrete Navier-Stokes equations $R(\omega, \underline{\mathbf{s}}) = 0$ with discrete solution ω, which in turn defines the discrete performance functional $I(\underline{\mathbf{s}}) := \widetilde{I}(\omega(\underline{\mathbf{s}}), \underline{\mathbf{s}})$. Just like this chain of definitions in the forward problem the derivatives $\mathrm{D}I/\mathrm{D}\mathscr{F}$ are defined as a chain but in the opposite direction. The discrete adjoint method is actually applied to the dependency of the discrete functional on the node positions of the mesh. In Subsection 2.6.2.1 an important ingredient of this approach is discussed, the derivatives of the FE discretisation with respect to the node positions of the mesh. Subsection 2.6.2.2 is devoted to the next step in the adjoint chain, the derivatives of the node positions with respect to the shape parameters, using this opportunity

to describe the mesh deformation approach in more detail. The efficient solution of the discrete adjoint equation is finally discussed in Subsection 2.6.2.3.

2.6.2.1 Derivatives with respect to node positions

Evaluation of the total derivative $\mathrm{D}I/\mathrm{D}s$ by way of the discrete adjoint method (1.1.7) requires evaluation of terms of the form

$$\frac{\mathrm{D}I}{\mathrm{D}s} = \frac{\partial \widetilde{I}}{\partial s} - \Psi^T \frac{\partial R}{\partial s}, \tag{2.6.7}$$

i.e. partial derivatives of the discrete residual vector R with respect to the node positions in the mesh as well as partial derivatives of the performance criterion $\widetilde{I}$ with respect to the node positions. In essence this requires differentiation of the algorithms which perform the computation of R and $\widetilde{I}$. There are three common approaches to this task: applying finite differences, using automatic differentiation software [39] and hand calculations. While the first two of these approaches offer the possibility to avoid the work-intensive hand calculations, they may result in significant performance loss. Generally, all these approaches should be applied on the element level only, i.e. for the local contributions to R rather than the whole vector R, in order to avoid a possible increase in computational complexity. Automatic differentiation software (see e.g. [39]) is designed to differentiate computer programs by generating a modified source code. Unfortunately mature tools of this kind are not available for all programming languages. Thus, "hand coding" the derivatives remains an important alternative. Further, if approached in a systematic manner, exploiting the knowledge about the formulas that are implemented by the software, then it is a simpler task than one might think. In this subsection such a systematic approach is demonstrated for a simplified model problem. The model problem is chosen in order to keep notation simple and the focus on the general methodology. Extension to the incompressible Navier-Stokes equations is straightforward.

As model problem the Poisson equation with constant forcing term $f \equiv 1$ is chosen,

which gives rise to the discrete formulation

$$R(\underline{u}, s) = [a(u_h, \varphi_i) - b(\varphi_i)]_{i=1}^{n}, \tag{2.6.8a}$$

$$a(u, v) := \int_{\Omega} \nabla u \cdot \nabla v \, \mathrm{d}\Omega, \tag{2.6.8b}$$

$$b(v) := \int_{\Omega} 1 \, v \, \mathrm{d}\Omega. \tag{2.6.8c}$$

$$\tag{2.6.8d}$$

The integrals contained in the definition of $a(.,.)$ and $b(.)$ are usually computed using numerical integration on each element and then summing over the elements. Hence the i-th row of the discrete residual vector R may be expressed as

$$[R]_i = \sum_{\substack{T \in \mathscr{T}_h \\ s_i \in \overline{T}}} \sum_{j=1}^{n} \left(\sum_{\ell=1}^{m} w_\ell \left(\nabla_T \varphi^{(g_T(j))} \cdot \nabla_T \varphi^{(i)} - 1 \, \widehat{\varphi}^{(i)}(\hat{x}_\ell) \right) \right) |\det(J_T)| \, u_{g_T(j)}, \tag{2.6.9}$$

where the outermost sum is over those elements $T \in \mathscr{T}_h$ which have node i as a vertex, n is the number of Lagrange basis functions per element, $\hat{x}_\ell$ ($\ell = 1, \ldots, m$) are the quadrature points on the master element and w_ℓ are the quadrature weights. Further, $g_T(j)$ is the global node number of vertex j of element T, ∇_T denotes the gradient restricted to element T evaluated at the point x_ℓ (which corresponds to $\hat{x}_\ell$ via the isoparametric element mapping), $\widehat{\varphi}^{(j)}(\hat{x})$ is the basis function corresponding to node j evaluated at point $\hat{x}$ within the master element and J_T is the Jacobian of the element mapping. In this expression the parts that are non-constant with respect to the node positions are

$$\nabla_T \varphi^{(g_T(j))}, \quad \nabla_T \varphi^{(i)} \quad \text{and } |\det(J_T)|,$$

in each element T. Applying the product rule of differentiation gives for the derivative of

$[R]_i$ with respect to $[s_k]_t$, the t-th coordinate of node k,

$$\begin{aligned}\frac{\partial [R]_i}{\partial [s_k]_t} = \sum_{\substack{T\in\mathscr{T}_h:\\ (s_i\in\overline{T}\\ \wedge s_k\in\overline{T})}} \sum_{j=1}^{n} \Bigg[\Bigg(\sum_{\ell=1}^{m} w_\ell \Big(\frac{\partial \nabla_T \varphi^{(g_T(j))}}{\partial [s_k]_t} \cdot \nabla_T \varphi^{(i)} \\ + \nabla_T \varphi^{(g_T(j))} \cdot \frac{\partial \nabla_T \varphi^{(i)}}{\partial [s_k]_t} \Big) \Bigg) |\det(J_T)| \\ + \Bigg(\sum_{\ell=1}^{m} w_\ell \Big(\nabla_T \varphi^{(g_T(j))} \cdot \nabla_T \varphi^{(i)} \\ + 1\, \widehat{\varphi}^{(i)}(\hat{x}_\ell) \Big) \Bigg) \frac{\partial |\det(J_T)|}{\partial [s_k]_t} \Bigg] u_{g_T(j)}.\end{aligned} \tag{2.6.10}$$

Assuming that each of the individual terms in (2.6.10) may be evaluated it is now a straightforward task to compute $\underline{\Psi}^T \partial R/\partial s$.

Remark 4. *Note that it is more efficient to compute and sum the element contributions to $\underline{\Psi}^T \partial R/\partial s$ because this avoids the cost, and the higher memory requirements, of assembling the sparse Jacobian matrix $\partial R/\partial s$. This matrix would be used for only one matrix vector product anyway.*

It remains to derive expressions for the derivatives of the gradient $\nabla_T \varphi^{(i)}$ and the determinant $|\det(J_T)|$. For the impatient we present the result of the rather technical derivation in advance.

Proposition 1.

$$\frac{\partial [\nabla_T \varphi^{(i)}]_u}{\partial [s_k]_t} = -\left[\nabla_T \varphi^{(k)}\right]_u \left[\nabla_T \varphi^{(i)}\right]_t \tag{2.6.11}$$

$$\frac{\partial |\det(J_T)|}{\partial s_{g_T(k)}} = |\det(J_T)| J_T^{-T} \hat{\nabla} \widehat{\varphi}^{(k)}(\hat{x}_\ell). \tag{2.6.12}$$

Note that similar representations of these derivatives have for example been presented in [43], but only for linear triangular elements, while the derivation that follows is valid for isoparametric elements of arbitrary degree and shape. Note also that the conditions for these expressions are different to similar formulas in [49], because we consider the evaluation point as fixed in the reference element, whereas [49] considered it fixed in the world element and restricted themselves to elements of simplex type, i.e. excluding curved element boundaries.

The remainder of this subsection is devoted to the proof of this proposition.

Proof of Proposition 1. Let us recall some properties of isoparametric finite elements (e.g. [21]): the definition of the element mapping function $M_T(.)$ (2.6.13), the resulting expression for its Jacobian J_T (2.6.14), the definition of the ansatz functions $\varphi^{(i)}$ on the world element by means of the ansatz functions $\widehat{\varphi}^{(i)}$ on the reference element (2.6.15) and the resulting expression for the world element gradient $\nabla_T \varphi^{(i)}(x)$ (2.6.16),

$$x = M_T(\hat{x}) := \sum_{k=1}^{n} s_k \widehat{\varphi}^{(k)}(\hat{x}), \tag{2.6.13}$$

$$J_T = \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_{k=1}^{n} s_k \left[\hat{\nabla}\widehat{\varphi}^{(k)}(\hat{x})\right]^T, \tag{2.6.14}$$

$$\varphi^{(i)}(x) := \widehat{\varphi}^{(i)}(M_T^{-1}(x)), \tag{2.6.15}$$

$$\nabla_T \varphi^{(i)}(x) = J_T^{-T} \hat{\nabla}\widehat{\varphi}^{(i)}(M_T^{-1}(x)). \tag{2.6.16}$$

Here s_k denotes the coordinate vector of the k-th node of the element T and $\hat{\nabla}$ denotes the gradient on the reference element, i.e. the partial derivatives with respect to the reference element coordinates $\hat{x}$. Two immediate consequences are

$$\begin{aligned}\frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t} &= \sum_{r=1}^{n} \delta_{k,r}\delta_{t,j}\left[\hat{\nabla}\widehat{\varphi}^{(r)}(\hat{x})\right]_v && \text{(by (2.6.14))} \\ &= \delta_{t,j}\left[\hat{\nabla}\widehat{\varphi}^{(k)}(\hat{x})\right]_v, && (2.6.17)\end{aligned}$$

where $\delta_{i,j}$ denotes the Kronecker delta.

Due to the application of the numerical integration based on the reference element, the world gradient is only evaluated for points corresponding to the quadrature points on the reference element $\hat{x}_\ell$,

$$\nabla_T \varphi^{(i)} := \nabla_T \varphi^{(i)}(M_T(\hat{x}_\ell)) = J_T^{-T} \hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell). \tag{2.6.18}$$

Since $\hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell)$ is independent of the node positions, $\nabla_T \varphi^{(i)}$ depends on the node positions only via the transpose of the element Jacobian J_T^T. The derivatives with respect to the node positions can be calculated using the implicit function theorem on the reformulated (2.6.18),

$$0 = J_T^T([s_k]_t)\nabla_T \varphi^{(i)} - \hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell). \tag{2.6.19}$$

This gives

$$\frac{\partial \nabla_T \varphi^{(i)}}{\partial [s_k]_t} = -J_T^{-T} \frac{\partial J_T^T}{\partial [s_k]_t} \nabla_T \varphi^{(i)}, \tag{2.6.20}$$

where the derivative of the transposed Jacobian is in the component wise sense. Taking a

closer look at the derivative of the u-th component of $\nabla_T \varphi^{(i)}$ we get

$$\begin{aligned}
\frac{\partial [\nabla_T \varphi^{(i)}]_u}{\partial [s_k]_t} &= -\sum_{v=1}^{d} \left[J_T^{-T}\right]_{(u,v)} \sum_{j=1}^{d} \frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t} \left[\nabla_T \varphi^{(i)}\right]_j \\
&= -\sum_{v,j=1}^{d} \left[J_T^{-T}\right]_{(u,v)} \frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t} \left[\nabla_T \varphi^{(i)}\right]_j \\
&= -\sum_{v,j=1}^{d} \left[J_T^{-T}\right]_{(u,v)} \delta_{t,j} \left[\hat{\nabla} \widehat{\varphi}^{(k)}(\hat{x}_\ell)\right]_v \left[\nabla_T \varphi^{(i)}\right]_j \qquad \text{(by (2.6.17))} \\
&= -\sum_{v=1}^{d} \left[J_T^{-T}\right]_{(u,v)} \left[\hat{\nabla} \widehat{\varphi}^{(k)}(\hat{x}_\ell)\right]_v \left[\nabla_T \varphi^{(i)}\right]_t \\
&= -\left[\nabla_T \varphi^{(k)}\right]_u \left[\nabla_T \varphi^{(i)}\right]_t \qquad \text{(by (2.6.18))},
\end{aligned}$$

i.e. the first part of the proposition.

The derivative of $|\det(J_T)|$ can also be reduced to a derivative of J_T due to the identity

$$\left[\frac{\partial}{\partial a_{i,j}} \det(A)\right] = A^{-T} \det(A)$$

which follows from expansion to sub-determinants and the adjoint representation of the inverse of the matrix A. Applying this formula and an analogue of (2.6.17) we get

$$\begin{aligned}
\frac{\partial}{\partial [s_k]_t} |\det(J_T)| &= |\det(J_T)| \sum_{u,v=1}^{d} \left[J_T^{-T}\right]_{u,v} \frac{\partial [J_T]_{u,v}}{\partial [s_k]_t} \\
&= |\det(J_T)| \sum_{u,v=1}^{d} \left[J_T^{-T}\right]_{u,v} \delta_{t,u} \left[\hat{\nabla} \widehat{\varphi}^{(k)}(\hat{x}_\ell)\right]_v \\
&= |\det(J_T)| \sum_{v=1}^{d} \left[J_T^{-T}\right]_{t,v} \left[\hat{\nabla} \widehat{\varphi}^{(k)}(\hat{x}_\ell)\right]_v .
\end{aligned}$$

□

2.6.2.2 Influence of the mesh deformation mapping

As already discussed in Section 2.2.6, automatic mesh generators, such as [77, 80] for example, may result in meshes which depend in a discontinuous manner on the shape parameters $\mathscr{F}$, even the number of elements may not be constant. As a compromise between the smoothness properties of parametric meshes and the geometric flexibility of automatically generated meshes, we proposed to use a hybrid approach of defining a base

mesh by means of an automatic mesh generator and then use deformed versions of it as a parametric mesh in a neighbourhood of the base shape parameters. The definition of the deformations as well as a discussion of how these have to be taken into account in the evaluation of the derivatives with respect to the shape parameters are the subject of this subsection.

The stationary Lamé equations (linear elasticity) are a mathematical model for elastic deformations of solid bodies under internal or external forces, see e.g. [19]. The pure Dirichlet problem, find $\mathbf{u} \in \mathbf{H}^1_{\mathbf{g}}(\Omega)^2$ such that

$$\mathrm{a}_L(\mathbf{u},\mathbf{v}) = 0 \qquad \forall \mathbf{u} \in \mathbf{H}^1_0(\Omega)^2 \tag{2.6.21}$$

$$\mathrm{a}_L(\mathbf{u},\mathbf{v}) := \int_\Omega \left(\lambda \left(\nabla\cdot\mathbf{u}\right)\left(\nabla\cdot\mathbf{v}\right) + 2\mu \sum_{i,j=1}^{2} \varepsilon_{ij}(\mathbf{u})\,\varepsilon_{ij}(\mathbf{v}) \right) \mathrm{d}\Omega, \tag{2.6.22}$$

$$\varepsilon_{ij}(\mathbf{u}) := \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right),$$

can be used to propagate a deformation $\mathbf{g}$ of the boundary of a domain into the interior, resulting in a deformation vector field $\mathbf{u}$ of minimal potential energy. This model contains two material constants $\lambda > 0$ and $\mu > 0$ (Lamé constants) which form a continuous analogue to spring constants in a discrete network of springs. In this present work they are chosen to take the values for steel,

$$\mu = 7.7 \cdot 10^4, \quad \lambda = 1.15 \cdot 10^5. \tag{2.6.23}$$

Finite element discretisation of (2.6.21) on the reference mesh allows a node-wise specification of a deformation $\mathbf{g}_h$ of the boundary, i.e. the displacement vectors for each boundary node. In order to define the displacement of the interior nodes an equation system of the form

$$\begin{bmatrix} K_{\mathrm{i,i}} & K_{\mathrm{i,b}} \\ 0 & I \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}}_{\mathrm{i}} \\ \underline{\mathbf{u}}_{\mathrm{b}} \end{bmatrix} = \begin{bmatrix} 0 \\ \underline{\mathbf{g}} \end{bmatrix} \tag{2.6.24}$$

has to be solved, where the subscript $*_{\mathrm{i}}$ denotes a block corresponding to interior nodes while $*_{\mathrm{b}}$ denotes a block corresponding to boundary nodes. The vector $\underline{\mathbf{s}}$ of node positions in the deformed mesh is defined as the positions in the base mesh $\underline{\mathbf{x}}_0$ plus the displacements due to the deformation,

$$\underline{\mathbf{s}} = \underline{\mathbf{x}}_0 + \underline{\mathbf{u}}. \tag{2.6.25}$$

This basic linear elasticity approach has one major disadvantage: regions of the mesh with relatively small elements (e.g. to resolve boundary layers) are treated the same way

as those with large elements, even though deformations in these regions of small elements have a much stronger influence on the mesh quality than in the coarser regions of the mesh. In [87] a modification to the discretised Lamé equations was proposed in order to address this issue. From a modelling point of view this modification adapts the parameters λ and μ such that the material is more rigid in regions of small elements, thus the deformations in those areas will tend to be smaller than in regions of large elements. This can easily be implemented by choosing λ and μ to be element-wise constant and multiplying the base values (2.6.23) by $|A_k|^\alpha$, where $\alpha \in \mathbb{R}$ is a parameter and $|A_k|$ denotes the surface area of element k. The tests in [87] suggest that $\alpha = -1$ forms a reasonable compromise, balancing the deformations between areas of small elements and areas of larger elements. Conveniently this parameter choice is equivalent to dropping the term $|A_k|$ completely from the assembly routines[14], thus it even simplifies the computations. For these reasons the choice $\alpha = -1$ is used in this present work as well. Note that even with these modifications a multigrid preconditioned CG solver performs optimally for the symmetric reformulation of (2.6.24). Thus, the costs for solving these problems are comperatively small in the context of the Navier-Stokes solver.

To illustrate the resulting mesh deformations, Figure 2.15 shows a close-up of two superimposed versions of the mesh around an interior obstacle in a channel (Example 2.2 from Section 2.7.1.2): one before and one after the mesh deformation. For moderate deformations of the boundary the mesh quality in terms of the size of the interior angles of the elments is usually maintained. However, strong deformations may result in degenerating meshes and thus re-definition of the base mesh may be required as discussed in Section 2.2.6.

Remark 5. *Note that simply deforming the boundary of the mesh only is not appropriate, because this would drastically restrict the amount by which the shape parameters can change before remeshing becomes necessary. For example a mesh cell can degenerate if the boundary is moved inward. Further, the quality of the boundary cells would strongly depend on the shape parameters, and the changes in this quality might even affect the quality of the discrete approximation of the performance functional I more strongly than the actual change of the boundary geometry. On the other hand, if the interior mesh nodes are moved in an appropriate manner as well, then these effects are far less pronounced. To illustrate these issues we ask the reader to look again at Figure 2.15. Note that some of the smaller elements at the top of the obstacle would have almost collapsed if the interior nodes of the mesh where not moved as well. Yet, the fully deformed mesh is of similar quality as the initial mesh.*

[14] The surface area $|A_k|$ is equivalent to the term $|\det(J_T)|$ in (2.6.9), for example.

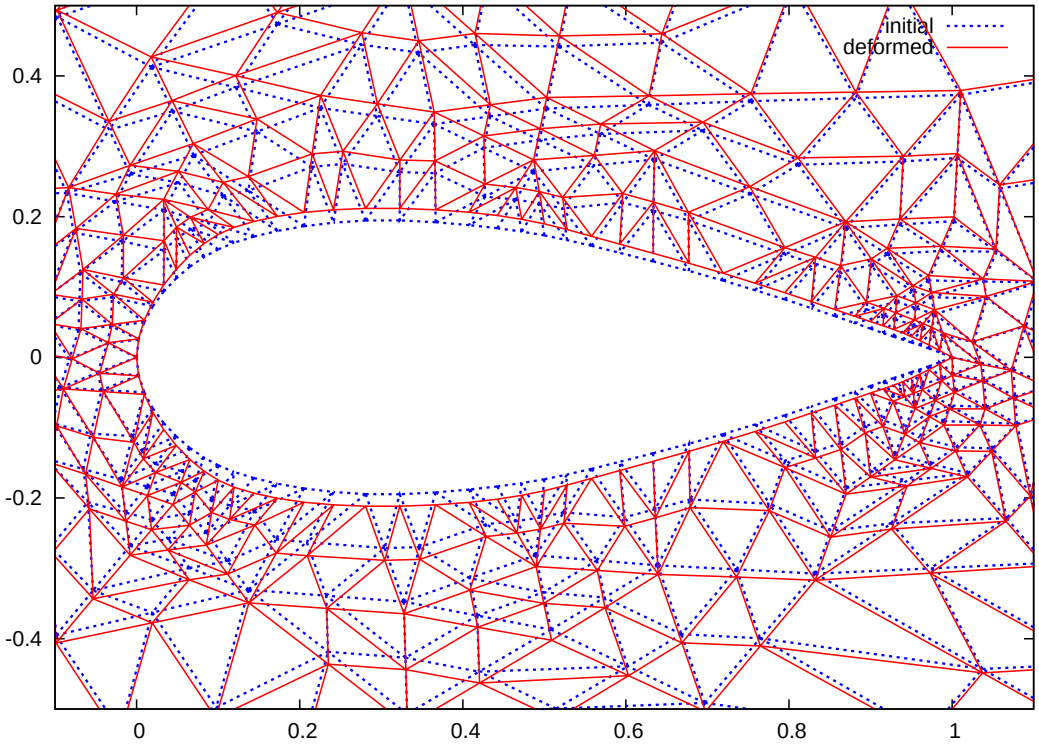


Figure 2.15: Mesh deformation by means of linear elasticity

An implication of equations (2.6.25) and (2.6.24) is that for the deformed mesh the node positions of the interior nodes are linearly dependent on the boundary displacements $\underline{\mathbf{g}}$,

$$\underline{\mathbf{s}}_{\mathrm{i}} = \underline{\mathbf{x}}_{0,\mathrm{i}} - K_{\mathrm{i,i}}^{-1} K_{\mathrm{i,b}} \underline{\mathbf{g}} \quad \text{and} \quad \underline{\mathbf{s}}_{\mathrm{b}} = \underline{\mathbf{x}}_{0,\mathrm{b}} + \underline{\mathbf{g}}.$$

This dependency has to be taken into account in the computation of the derivatives $\mathrm{D}I/\mathrm{D}\mathscr{F}$. This is easily done applying the adjoint equation approach, yielding

$$\frac{\mathrm{D}I}{\mathrm{D}\underline{\mathbf{g}}} = \frac{\mathrm{D}I}{\mathrm{D}\underline{\mathbf{s}}_{\mathrm{b}}} - K_{\mathrm{i,b}}^{T} K_{\mathrm{i,i}}^{-T} \frac{\mathrm{D}I}{\mathrm{D}\underline{\mathbf{s}}_{\mathrm{i}}}, \tag{2.6.26}$$

where in analogy to (2.6.24) the vector of node positions $\underline{\mathbf{s}}$ is split into interior and boundary parts. The derivatives with respect to the node positions in the mesh $\mathrm{D}I/\mathrm{D}\underline{\mathbf{s}}$ are computed using the discrete adjoint method with the approaches discussed in Subsection 2.6.2.1. The vector of boundary displacements $\underline{\mathbf{g}}$ itself is a function of the shape parameters $\mathscr{F}$. Thus the computation of $\mathrm{D}I/\mathrm{D}\mathscr{F}$ is completed by

$$\frac{\mathrm{D}I}{\mathrm{D}\mathscr{F}} = \frac{\mathrm{D}I}{\mathrm{D}\underline{\mathbf{g}}} \left[\frac{\partial \underline{\mathbf{g}}}{\partial \mathscr{F}} \right].$$

Remark 6. *Note that this deformation approach has advantages in the context of multigrid solution techniques as well. Successful application of multigrid requires a hierarchy of meshes, with the coarsest meshes in the hierarchy consisting of a relatively small number of elements or mesh cells only. Such hierarchical meshes are most easily created by refining an initial coarse mesh. However, for domains with curved boundaries some means of adjusting the refined mesh to fit the domain is required, for which the mesh deformation approach can be easily applied.*

2.6.2.3 Efficient solution of the discrete adjoint equations

One advantage of the discrete adjoint method compared to the continuous adjoint is that preconditioners for the original problem can be utilised for the adjoint problem as well, because the adjoint equations are defined by the transpose of the stiffness matrix of the forward problem. In the context of non-symmetric problems like the incompressible Navier-Stokes equations, it has to be observed that the transpose of a *right* preconditioner C_R^{-1} for the original problem becomes a *left* preconditioner for the adjoint equations and vice versa, simply because

$$\left(KC_R^{-1}\right)^T = C_R^{-T}K^T.$$

Thus in order to define a left F_p preconditioner for the adjoint problem a right preconditioner for the forward problem should be transposed.

The zero mean pressure condition (ZMPC), which for the forward problem has been dealt with by means of the projection approach, see Subsection 2.5.2.3, has to be studied closely again in the formulation and solution of the adjoint problem. Application of the projection approach implies that the ZMPC becomes a "hidden" constraint in the sense that it does not appear in the matrix defining the linear equation system. In our previous publication [75] this issue has been dealt with explicitly by including this constraint into the system matrix before it is transposed. This was done in replacing the last row of the equation system by the ZMPC which takes the form $w^T p = 0$, where w is a weight vector and p the coefficient vector for the discrete pressure approximation, see Subsection 2.5.2.3. In the context of the adjoint equation even more disadvantages of this approach arise in addition to those discussed in Subsection 2.5.2.3. Since the degrees of freedom of the adjoint equations (adjoint variables Ψ) correspond to the individual equations of the forward problem, the variable corresponding to the ZMPC has a special meaning which the F_p preconditioner does not account for in its standard form. To overcome this, in [75] we modified the preconditioner by adding an additional projection step in analogy to the projection approach from Subsection 2.5.2.3. With this modification the adjoint

preconditioner worked with comparable performance as the forward preconditioner.

However, while this approach works satisfactorily, it lacks elegance. The necessity of maintaining separate preconditioner routines for the forward problem and the adjoint problem as well as the lack of a rigorous analysis for the modified adjoint preconditioner provide motivation to seek a unified approach, in which the same preconditioning routines can be used for the forward and the adjoint problem (apart from transposing the matrices used in the preconditioner). It turns out that this is indeed possible.

Following the arguments in Subsection 2.6.1, the adjoint equation (2.6.3a) for a Lagrange multiplier approach to implementing the ZMPC has the structure (2.6.4) and it is known that $\Psi_\lambda = 0$. Equation (2.6.4) imposes the ZMPC on the adjoint state Ψ in exactly the same way as it is imposed on the forward solution ω. Thus it is possible to enforce this constraint in the same way as for the forward problem, i.e. by means of the projection approach of Subsection 2.5.2.3. Even further, it is possible to apply the same solver as for the linearised forward problems to the adjoint problem (2.6.5) by simply transposing the system matrix J. Since the resulting problem has the same structure as the forward problem, the same preconditioning techniques can be applied.

The author has tested this, achieving the same behaviour for the adjoint solves as for the forward solves. Exemplary results are given in Table 2.15 for shape optimisation Example 2.1 (see Section 2.7.1.1). For this test the same accuracy settings have been used for the forward and adjoint solves, i.e. 10^{-5} relative reduction in the preconditioned outer residual and 10^{-6} relative reduction in the first inner solve to set the inner iteration number for the F-block, A_p and M_p solves. The table lists the mesh refinement level ℓ, the Reynolds number Re, the number of degrees of freedom (#dof), the maximal number of F_p preconditioned GMRES iterations required to solve the linear systems ($\max(\#it)$), the maximal number of inner GMRES iterations for the F block per outer GMRES iteration ($\max(\frac{\#\text{inner}_F}{\#\text{outer}})$), the maximal time it took to solve one of the (outer) linear equation systems on the level ($\max(t_{\text{solve}})$). Note that for the forward solve in each case the maximum is over the Newton steps, whereas for the adjoint solve only one solve is performed, thus the maximum is only over one datum. It should also be noted that the F block of the matrix K as well as its coarse grid representations and the pressure space advection-diffusion-reaction operator F_p have to be transposed in order to achieve this.

As this new result supersedes our previous approach in [75], a more detailed discussion of the previous approach is omitted.

	ℓ	Re	#dof	max(#*it*)	max($\frac{\#\text{inner}_F}{\#\text{outer}}$)	max(t_{solve})
solve	6	100.0	112512	46	18	$2.2e+02$
adjoint	6	100.0	112512	40	15	$1.7e+02$
solve	7	100.0	446208	46	18	$8.5e+02$
adjoint	7	100.0	446208	41	15	$7.5e+02$

Table 2.15: Comparison forward and adjoint solves with F_p preconditioner, cavity with obstacle (Example 2.1) at $Re = 100$

2.6.3 The discrete adjoint method applied in the finite volume context

Due to the Cartesian block structured meshes used in the finite volume (FV) discretisation studied in this work, the mesh deformation approach can not be applied to this discretisation. Instead the geometries are restricted to what can be handled by parametric meshes of the block Cartesian type. Note that this limitation does not apply to FV discretisations in general, as this discretisation technique has been developed specifically to allow discretisations on unstructured meshes. For those interested in the issues related to more flexible meshes and geometries we remark that the methods presented for the finite element discretisation are applicable for more flexible FV discretisations as well.

It remains to present a solution procedure for the adjoint equations and to discuss the derivatives of the FV discretisation with respect to the mesh parameters.

For solving the adjoint equation (2.6.3a) an analogous solution procedure to that for the forward problem can be used, i.e. time-stepping in the case of the FV discretisation of Section 2.4. The solution procedure is summarised in Algorithm 5, with the same notation as used in Section 2.4.5. Compared to Algorithm 1, the nonlinear residual $R(\omega, \mathscr{F})$ of the forward problem is replaced by the residual of the adjoint equation $r(\Psi) := J^T\Psi - \partial\tilde{I}/\partial\omega$ and the Jacobian J is transposed. Note that this time-stepping procedure performed more efficiently in our tests than solving the (linear) adjoint equation directly. It appears to be likely that quadratic growth of the cost for the GMRES algorithm with the number of iterations is the source to this phenomenon. The relatively inaccurate solves of the time-step systems $(J^T + D)y = r$ require few GMRES steps. Performing these cheap solves multiple times in order to converge to the solution of the adjoint equation is still more efficient than performing the higher number of GMRES steps to directly solve the adjoint equation accurately.

Like in the finite element case, the author recommends computing the term $\Psi_\omega^T[\partial R/\partial\mathscr{F}]$

Algorithm 5 Adjoint Solution Procedure

$\Psi := 0$ {initialisation}
$r := -\frac{\partial \widetilde{I}}{\partial \omega}$ {initial evaluation of residual}
while $||r|| > \varepsilon$ **do**
  solve $(J^T + D)y = r$
  $\Psi := \Psi - y$ {apply time-step}
  $\Psi := \Psi - e_p(w_p^T \Psi)$ {make sure $w_p^T \Psi = 0$}
  $r := J^T \Psi - \frac{\partial \widetilde{I}}{\partial \omega}$ {update residual}
end while

by summing local contributions of the form

$$\Psi_{\text{loc}}^T \frac{\partial R_{\text{loc}}}{\partial \mathscr{F}},$$

rather than assembling the sparse matrix $\partial R/\partial \mathscr{F}$ first, because $\partial R/\partial \mathscr{F}$ would be used for this one matrix vector product only. This allows considerable savings of time and memory. In the FV case the local contributions are naturally cell interface based, like the computation of the residual $R(\omega)$ itself. For the block Cartesian meshes considered in this work the parameters $\mathscr{F}$ are the coordinates of the corners of each of the mesh blocks. These enter the computation in two ways only: via the measure of the cell interface $|A_j|$ in formula (2.4.7), and via the distance of neighbouring cell centres h_j in formula (2.4.10). Thus evaluation of the local derivatives $\partial R_{\text{loc}}/\partial \mathscr{F}$ is straightforward.

2.7 Numerical examples

In this section the shape optimisation approach developed in this chapter is applied to two example problems. The first example, a driven cavity with an internal obstacle of variable position, uses simple parametric domains for which both discretisations of this work can be applied and compared. The second, more realistic example considers shape optimisation of an obstacle in channel flow. The space of feasible geometries for this problem is less restrictive, so the flexibility of unstructured triangular meshes is used in this case. However, the finite volume discretisation is not applicable to this second example.

Finally Section 2.7.2.1 provides validation of the discretisation techniques of this thesis, their implementation and the corresponding implementations of the discrete adjoint method.

2.7.1 Optimisation examples

2.7.1.1 Example 2.1: Multiply connected cavity

Problem formulation: A modified version of the lid driven cavity problem serves as a first example. A square obstacle of fixed size is placed in the cavity, see Figure 2.16. The average velocity component in the negative x-direction $\overline{V}$ in the area Ω_B, below the obstacle, is used as performance criterion. The aim is to maximise this average velocity by choosing the position for the obstacle appropriately. This may for example be of interest if such a cavity is used in a coating process where the coating material contains very small particles which fall out as sediment if the flow velocity becomes too small.

$$\begin{aligned}
\Omega &:= (0,1)^2 \setminus \left[\frac{1}{4}+\mathscr{F}_1, \frac{3}{4}+\mathscr{F}_1\right] \times \left[\frac{1}{4}+\mathscr{F}_2, \frac{3}{4}+\mathscr{F}_2\right], \\
\Omega_B &:= \left(\frac{1}{4}+\mathscr{F}_1, \frac{3}{4}+\mathscr{F}_1\right) \times \left(0, \frac{1}{4}+\mathscr{F}_2\right), \\
I &:= \overline{V} := \frac{1}{|\Omega_B|} \int_{\Omega_B} -u_1 \, \mathrm{d}\Omega.
\end{aligned} \tag{2.7.1}$$

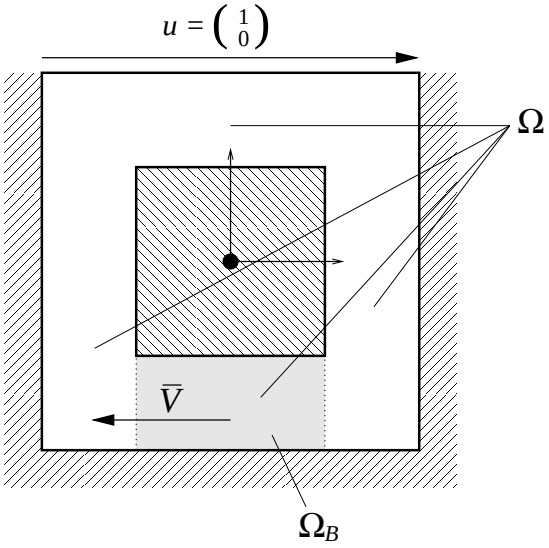


Figure 2.16: Example 2.1: Multiply connected cavity

The boundary conditions for the Navier-Stokes problem are defined by

$$\Gamma_T := \{(x_1, x_2) : x_2 = 1, \quad x_1 \in (0,1)\}, \tag{2.7.2}$$

$$\Gamma_W := \partial\Omega \setminus \Gamma_T, \tag{2.7.3}$$

$$\mathbf{u} = (0,0)^T \qquad \text{on } \Gamma_W, \tag{2.7.4}$$

$$\mathbf{u} = (1,0)^T \qquad \text{on } \Gamma_T. \tag{2.7.5}$$

The parameters $\mathscr{F} = (\mathscr{F}_1, \mathscr{F}_2)^T$ are the deviation of the centre of the obstacle from the centre of the cavity. The position of the obstacle is constrained such that a gap remains between the obstacle and the boundary of the cavity,

$$-0.2 \leq \mathscr{F}_1 \leq 0.2, \tag{2.7.6a}$$

$$-0.2 \leq \mathscr{F}_2 \leq 0.2. \tag{2.7.6b}$$

So the overall problem is:

Example 2.1. *Maximise* (2.7.1)*, subject to* (2.7.6)*.*

Due to the simple rectangular geometry this problem is well suited to be treated with parametric meshes and both the finite element as well as the finite volume discretisation of this work can be applied.

Results and discussion: The optimisation problem has been solved for $Re = 10$, $(\nu = 10^{-1})$, $Re = 20$, $(\nu = 5 \times 10^{-2})$, $Re = 100$, $(\nu = 10^{-2})$ and $Re = 200$, $(\nu = 5 \times 10^{-3})$.

In all four cases the optimisation solver was started with $\mathscr{F} = [0,0]^T$ and converged to a solution in the interior of the parameter space, i.e. the constraints (2.7.6) were not active at the optimal solution. Figure 2.17 shows the velocity profiles along the lines $x = 0.5$ and $y = 0.5$ for both the initial as well as the optimised geometry. An increase in the velocity at the bottom part of the domain can be observed in all four cases and the optimal geometries show little difference between the different Reynolds number regimes.

Tables 2.16 and 2.17 list the corresponding values of the viscosity parameter ν ($\nu = 1/Re$), the initial performance I_{ini}, the optimal performance I_{opt}, the optimal solution $\mathscr{F}_{\text{opt}}$, the number of iterations of the `DONLP2` [84] optimisation solver (#it), the total number of evaluations of the performance function by the optimisation solver (#eval) and the number of degrees of freedom in the discretised Navier-Stokes equations (#DOFs). It can be observed in these tables that the first component of $\mathscr{F}_{\text{opt}}$ shows a clear dependency on ν and that the FE and FV solvers compute optimal parameters that are very close to each other. However, the initial as well as the optimal values of I show significantly different behaviour in both solvers. For this optimisation problem in two variables `DONLP2` converges in relatively few iterations, only 6–9 iterations. However, the number of function evaluations is slightly larger because the line search steps are not counted as iterations. These numbers demonstrate the need for efficient solutions strategies if optimisation is to be considered.

In order to investigate the reason for the differing values of the initial and optimised performance values I between the FE and FV discretisation, tables 2.18 and 2.19 list these quantities again for different levels of mesh refinement. Note that, due to the higher convergence order, the FE discretisation produces very accurate approximations of the functional even on the coarsest mesh, while the first order convergent FV scheme shows significant differences even between the computed performance values on the finer meshes. In order to show the difference between the finite element results for differing refinement levels the results have to be listed to more digits than in the finite volume case. Due to stability limitations ($Pe \leq 1$) the FE discretisation can only be applied to the $\nu = 5 \times 10^{-3}$ case on the finest mesh with 446208 degrees of freedom. Thus no comparison between refinement levels is possible in this case.

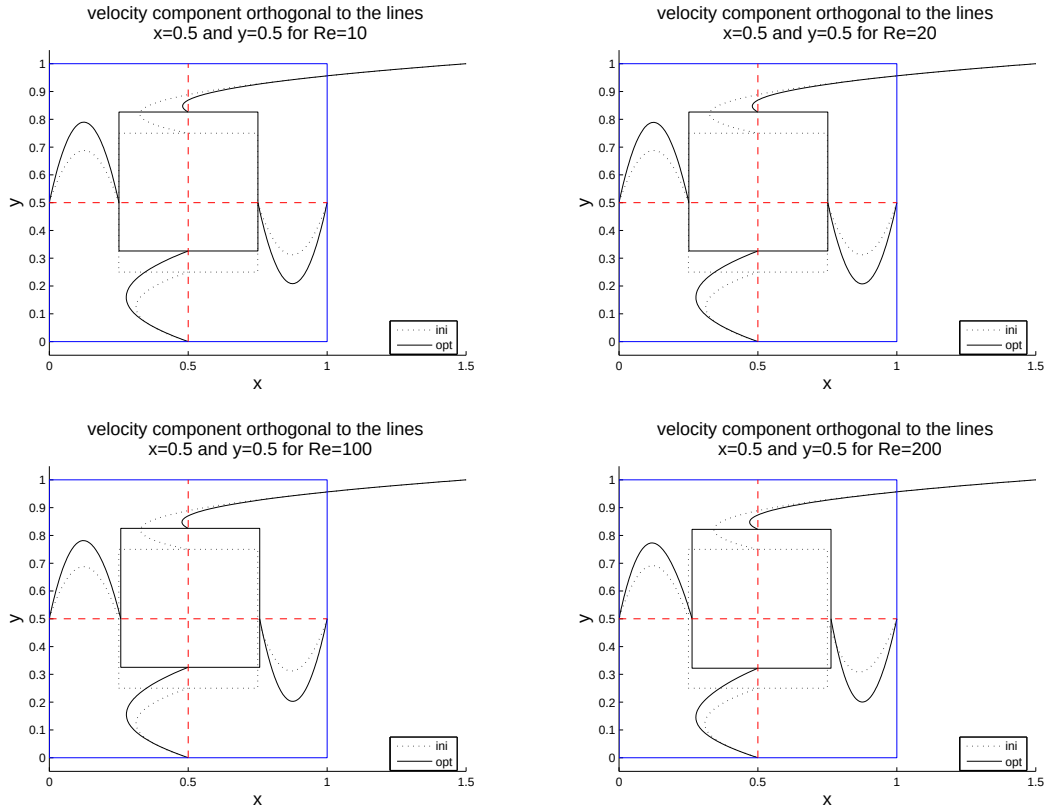


Figure 2.17: Initial ($\mathscr{F} = (0,0)^T$) and optimised velocity profiles for Example 2.1

ν	I_{ini}	I_{opt}	$\mathscr{F}_{\text{opt}}$	#it	#eval	#DOFs
10^{-1}	0.1251	0.1485	$(0.0008, 0.0761)^T$	6	12	112512
5×10^{-2}	0.1251	0.1485	$(0.0015, 0.0761)^T$	9	45	112512
10^{-2}	0.1251	0.1478	$(0.0073, 0.0754)^T$	6	19	112512
5×10^{-3}	0.1272	0.1467	$(0.0131, 0.0720)^T$	8	40	446208

Table 2.16: Example 2.1: FE-solver optimisation results for varying Reynolds numbers

ν	I_{ini}	I_{opt}	$\mathscr{F}_{\text{opt}}$	#it	#eval	#DOFs
10^{-1}	0.1235	0.1465	$(0.0008, 0.0762)^T$	7	25	36864
5×10^{-2}	0.1237	0.1468	$(0.0015, 0.0766)^T$	6	18	36864
10^{-2}	0.1204	0.1427	$(0.0071, 0.0784)^T$	7	24	36864
5×10^{-3}	0.1165	0.1367	$(0.0130, 0.0794)^T$	7	24	36864

Table 2.17: Example 2.1: FV-solver optimisation results for varying Reynolds numbers

ℓ	ν	I_{ini}	I_{opt}	$\mathscr{F}_{\text{opt}}$	#it	#eval	#DOFs	t_{opt}
1	0.1	1.25212e-01	1.48544e-01	$(7.57850\text{e-}04, 7.60550\text{e-}02)^T$	6	12	1968	1.48e+02
2	0.1	1.25213e-01	1.48593e-01	$(7.60563\text{e-}04, 7.60760\text{e-}02)^T$	6	12	7392	6.61e+02
3	0.1	1.25171e-01	1.48570e-01	$(7.62301\text{e-}04, 7.60934\text{e-}02)^T$	6	12	28608	3.03e+03
4	0.1	1.25142e-01	1.48548e-01	$(7.69101\text{e-}04, 7.61024\text{e-}02)^T$	6	12	112512	1.49e+04
5	0.1	1.25127e-01	1.48536e-01	$(7.63659\text{e-}04, 7.61062\text{e-}02)^T$	6	12	446208	6.54e+04

Table 2.18: Example 2.1: FE-solver optimisation results varying mesh refinement

ν	I_{ini}	I_{opt}	$\mathscr{F}_{\text{opt}}$	#it	#eval	#DOFs
10^{-1}	0.1217	0.1443	$(0.0008, 0.0764)^T$	8	26	9216
10^{-1}	0.1235	0.1465	$(0.0008, 0.0762)^T$	7	25	36864
10^{-1}	0.1243	0.1475	$(0.0008, 0.0762)^T$	8	27	147456
10^{-1}	0.1248	0.1481	$(0.0008, 0.0761)^T$	7	25	589824
5×10^{-3}	0.1165	0.1367	$(0.0130, 0.0794)^T$	7	24	36864
5×10^{-3}	0.1213	0.1413	$(0.0131, 0.0760)^T$	7	22	147456
5×10^{-3}	0.1240	0.1438	$(0.0131, 0.0741)^T$	6	21	589824

Table 2.19: Example 2.1: FV-solver optimisation results for varying mesh refinement

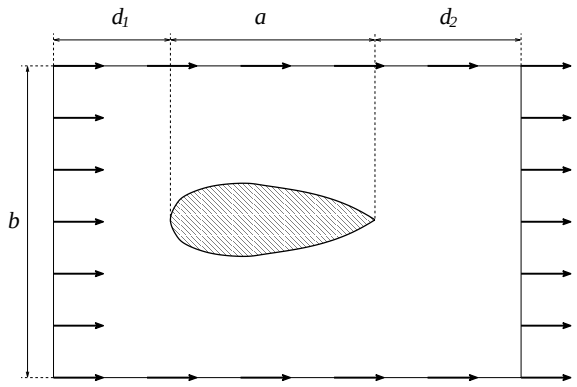


Figure 2.18: Example 2.2: Obstacle in a channel

2.7.1.2 Example 2.2: Obstacle in a channel

Problem formulation: Consider an object Q to be moving with constant velocity V inside a long channel of width b in a viscous fluid. The aim shall be to design the shape of this object such that the forces necessary to move it in the channel become as small as possible while keeping the length of the object a constant and the volume greater or equal to a given value.

Fixing the reference frame at the front tip of the object Q one arrives at the situation depicted in Figure 2.18. For an idealised channel of infinite length, the distances d_1 and d_2 are infinite. To allow finite element calculations, the domain is restricted to a finite domain, with finite values d_1 and d_2 chosen such that their influence on the quantities of interest is (hopefully) negligible. The performance criterion is the x-component of the body force acting on the object Q,

$$F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \int_{\partial Q} \mu(\nabla u + \nabla u^T)\, n \, \mathrm{d}\Gamma - \int_{\partial Q} p\, n \, \mathrm{d}\Gamma, \tag{2.7.7}$$

$$I := F_1, \tag{2.7.8}$$

where n is the outward normal of the fluid domain Ω, which can also be seen as an inward normal of the surface of Q. To keep the y-component of the acting force equal to zero, the shape of the object Q is restricted to be symmetrical about the centre line of the domain.

The shape of Q is parameterised by cubic Bezier-splines (B-splines, [44, Appendix B] or [30]) as demonstrated in Figure 2.19. In each spline segment these curves are cubic

polynomials in a parameter $t \in [0,1]$ of the form

$$x(t) = \sum_{i=0}^{3} x_i B_i^3(t),$$

where $x_i \in \mathbb{R}^2$ are so called control points and B_i^n denote the Bernstein polynomials

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}.$$

The curve segments start at their respective control point x_0 and end at x_3. The tangents at the beginning and end points of each segment are defined by the lines $\overline{x_0 x_1}$ and $\overline{x_2 x_3}$ respectively. By defining the control points at a segment interface such that $x_3 - x_2|_{\text{left}} = x_1 - x_0|_{\text{right}}$ it can be achieved that twice continuously differentiable curves (B-splines) are constructed. The red diamonds in Figure 2.19 mark the interface control points between spline segments (x_0 and x_3) and the endpoints of the tangential lines mark the positions of the interior control points (x_1 and x_2) of each segment. Due to the construction as continuously differentiable curves only one of these interior control points can be chosen arbitrarily per segment, apart from the last segment. The free control points are marked by red circles in Figure 2.19.

The shape parameters for the optimisation problem are chosen to be the vertical positions of the control points (red circles and diamonds). The horizontal position of the control points, as well as the vertical position of the two points on the symmetry axis, are fixed. Thus, the parameterisation as illustrated in Figure 2.19 results in ten parameters.

The use of B-splines has several advantages. They allow great control of the resulting curve in terms of geometrical constraints due to the convex hull property [30]. The Bezier curve segments of the spline are guaranteed to be contained in the convex hull of the control points of this specific segment. This allows simple geometric constraints on the resulting curve to be expressed as constraints on the position of the control points of the spline, which in turn form the parameters of the shape. The possibility of self overlapping of the domain, and consequently of the mesh, is avoided by constraining the shape such that it contains the green polygon in Figure 2.19, which is obtained as the convex hull of the front and rear tip of the object, and two points at $0.1\,a$ and $0.9\,a$ horizontally, deviated vertically from the centreline of Q by $0.02\,a$.

The application of parametric meshes for this problem is challenging and, while it may be possible, it would not necessarily provide more insight than the previous example. Instead, the more general approach of using the general mesh-generator `Triangle` [80] in combination with mesh deformation is taken, see sections 2.2.6 and 2.6.2.2. Due to the

non-rectangular geometry of the obstacle the finite volume discretisation of this present work is not applicable, so only results for the finite element discretisation can be provided.

So the overall problem is:

Example 2.2. *Minimise* (2.7.8)*, subject to* $|Q| \geq const$.

Illustration of the adjoint solution and sensitivity: In order to illustrate the character of the solution and the discrete adjoint solution, Figure 2.20 shows both of them in the vicinity of the obstacle for $Re = 10$. It is interesting to observe that the discrete adjoint solution Ψ behaves like a PDE solution, but only in the interior of the domain. Close to the obstacle boundary effects from the definition of the adjoint problem in a discrete sense are visible. The forcing terms $\partial I/\partial\omega$ of the adjoint equation act in a very narrow region around the boundary while effectively homogeneous Dirichlet boundary conditions are in place. Corresponding to the singularity-like behaviour of the adjoint velocity variables Ψ_u near the obstacle boundary, the adjoint pressure Ψ_p does also show extreme peaks in the vicinity of the obstacle. The sensitivity $\mathrm{D}I/\mathrm{D}\mathscr{F}$ is visualised in the bottom part of Figure 2.20 by adding the scaled gradient $\mathrm{D}I/\mathrm{D}\mathscr{F}$ to the current shape parameters. Thus the illustrated shape modification shows a modification increment in the direction of the strongest increase in the performance functional I.

Results: Optimisation has been performed for two Reynolds number regimes, $Re = 10$ and $Re = 20$. For the computations the dimensions of the domain have been set to $b = 4$, $d_1 = 2$, $d_2 = 6$ and $a = 1$. The initial geometry of the obstacle was set to a B-spline interpolation of the NACA0040 profile. Thus the width of the initial geometry is 0.4. The volume of the obstacle is restricted to be greater or equal to the volume of the initial geometry, that is $|Q| \geq 4.2633$. The results of the optimisation are summarised in Table 2.20, where the viscosity parameter ν, the corresponding Reynolds number Re, the initial performance I_{ini}, the optimised performance I_{opt}, the number of iterations of the `DONLP2` optimisation solver (#it), the total number of performance function evaluations (#evals) and the number of degrees of freedom in the discrete Navier-Stokes system (#DOFs) are listed. Note that due to the automatic mesh generation the number of DOFs is not fixed but depends on the current base mesh. The values given specify the range that was observed. The optimised obstacle geometries are shown in Figure 2.19.

Discussion: The optimised shapes in Figure 2.19 may appear surprising, as one may expect a more tear-drop-like shape to result in a lower drag, or maybe with the volume

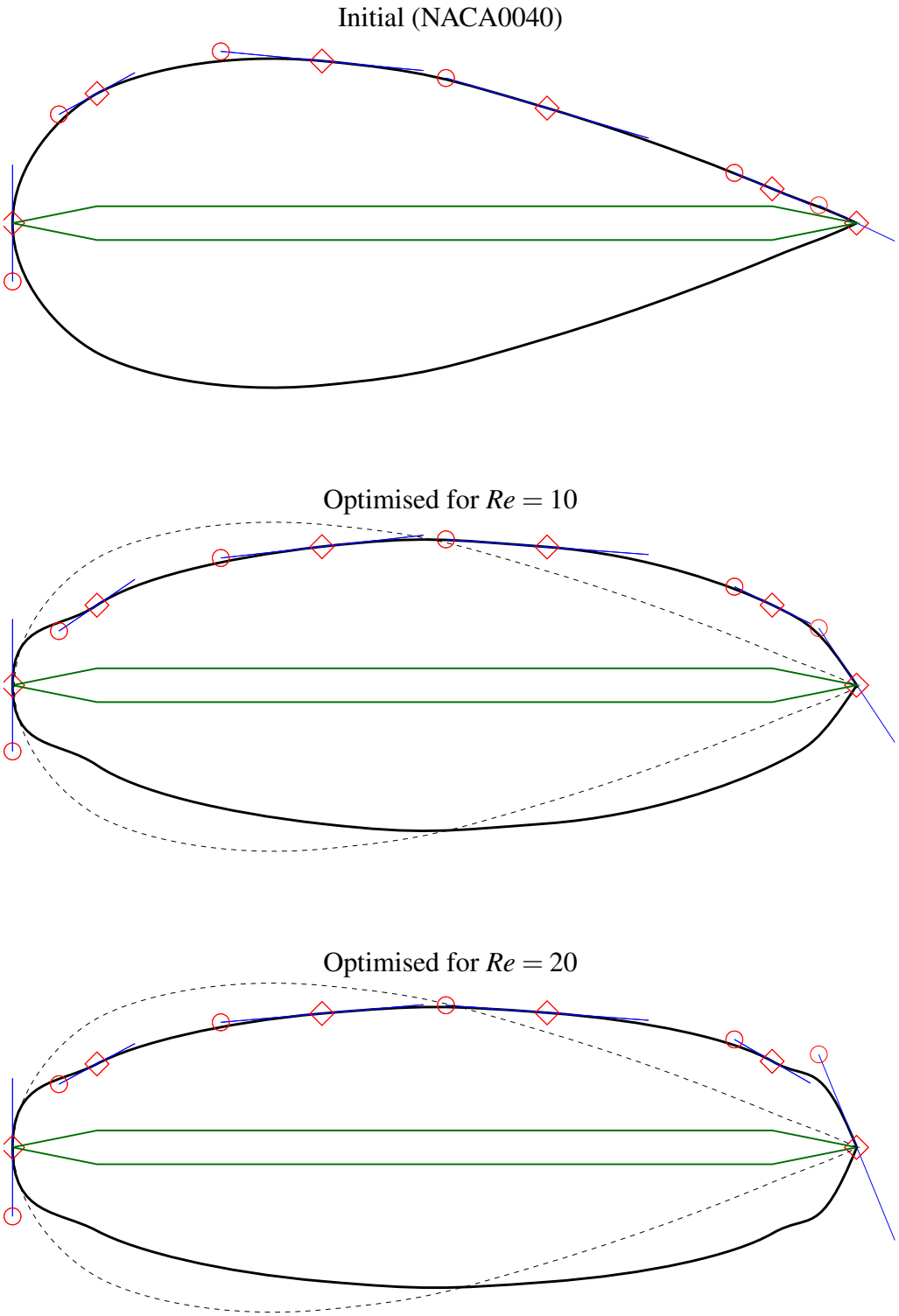
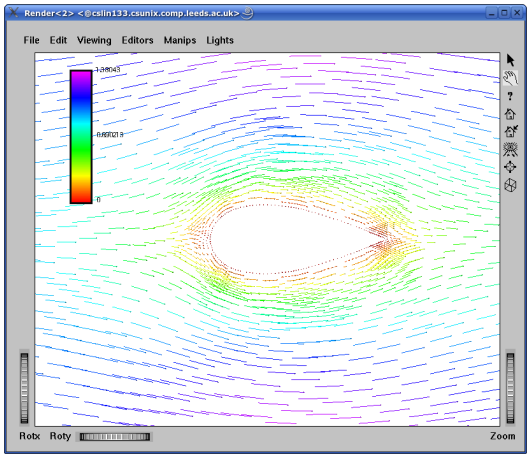
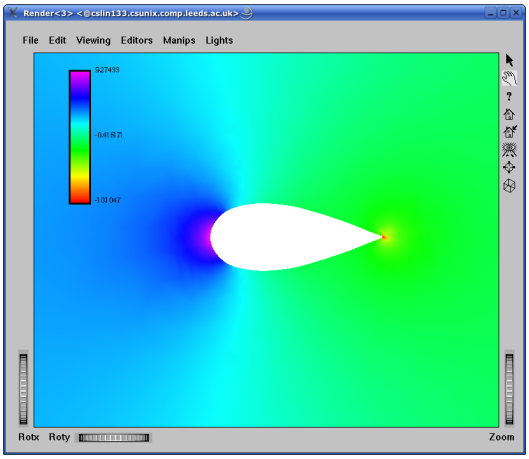


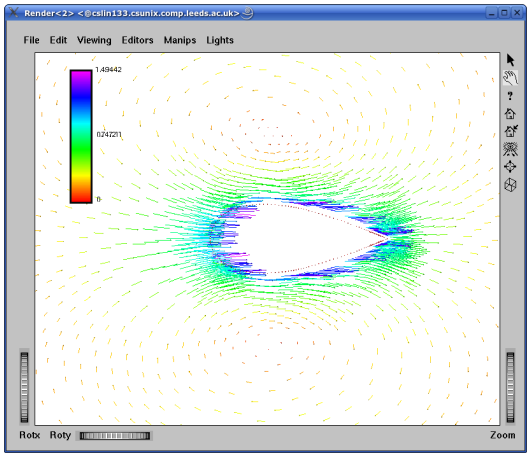
Figure 2.19: Example3: Parameterised obstacle shapes



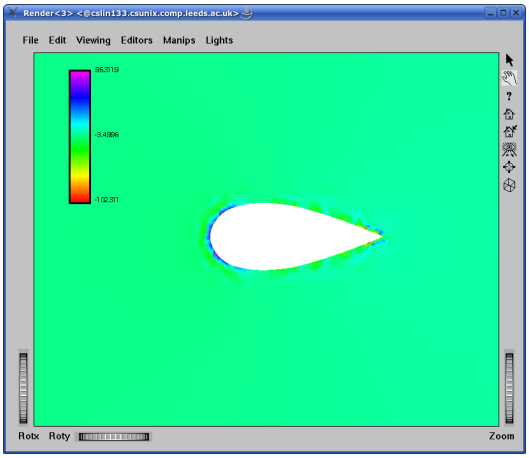
(a) Velocity solution u



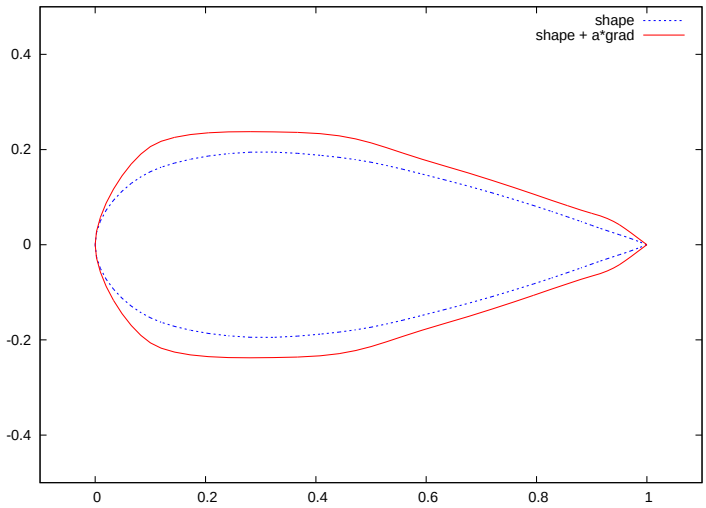
(b) Pressure solution p



(c) Adjoint velocity Ψ_u



(d) Adjoint pressure Ψ_p



(e) Sensitivity $DI/D\mathcal{F}$

Figure 2.20: Example 2.2: Solution, adjoint solution, and (scaled) sensitivity for the obstacle at $Re = 10$

ν	Re	I_{ini}	I_{opt}	#it	#evals	#DOFs (approx)		
0.1	10	1.4056	1.3714	18	69	140,000	–	150,000
0.05	20	0.8743	0.8452	16	60	550,000	–	600,000

Table 2.20: Example 2.2: Optimisation results

and length constraints one may expect a cigar-like shape such as it is used for air ships. However, these (expected) geometries are known for their good low drag performance in high Reynolds number regimes ($Re \approx 10^4$ to 10^6), in which different physical phenomena are of importance than in the regimes for which these geometries have been optimised.

To put the results into perspective let us consider two scenarios where water flow past such an obstacle is characterised by Reynolds number $Re = 10$. The kinematic viscosity ν for water is $\nu_{\text{water}} = 1.1 \times 10^{-6} m^2 s^{-1}$ [65]. The Reynolds number Re can be expressed in terms of ν, characteristic length scale ℓ (length of the obstacle) and characteristic velocity V (inflow velocity),

$$Re = \frac{V\ell}{\nu}.$$

Assuming a length of the obstacle of $1m$, the inflow velocity would have to be $V = 1.1 \times 10^{-5} m\,s^{-1}$ in order to have $Re = 10$. On the other hand, if the inflow velocity is set to $V = 10km/h = 36m\,s^{-1}$, the length of the obstacle would have to be $\ell \approx 3.06 \times 10^{-7} m$. Thus it is probably more realistic to think of the fluid to be a very viscous one, like honey for example.

The reduction in the drag I achieved by the optimisation is relatively mild. This may be attributed to the relatively good initial shape, which makes large improvements difficult.

2.7.2 Validation of the software

2.7.2.1 Validation of the discretisation techniques

In order to verify that the implementation of the discretisations is correct all three discretisation schemes, Galerkin FEM, SDFEM and FVM, are applied to a standard test problem: lid driven cavity flow, see Figure 2.21. The results are compared with those from Ghia et al [35], which are widely acknowledged as a good reference solution. Figures 2.22, 2.23 and 2.24 show velocity profiles along the lines $x = 0.5$ and $y = 0.5$, for $Re = 100$ and $Re = 1000$. The results are presented for a sequence of meshes and compared to those from Ghia et al[15]. Note that for Galerkin FEM $Re = 1000$ would require an extremely

[15]The results in [35] where computed on a mesh with 129×129 nodes, using a second order finite difference scheme with first order upwinding for the convective terms.

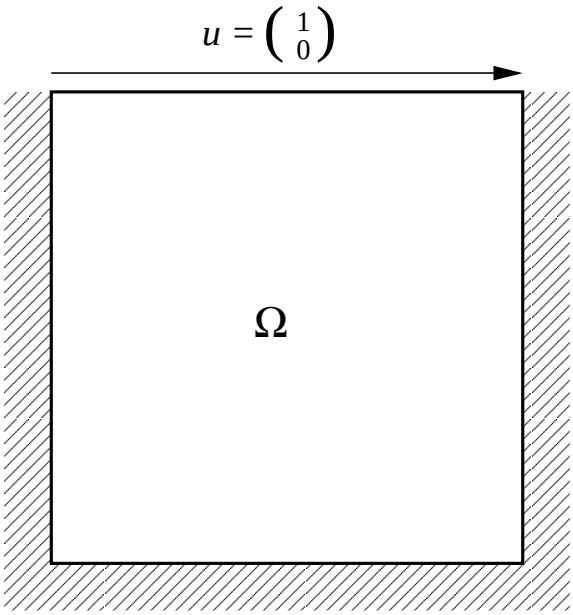


Figure 2.21: Test setup: Lid driven cavity

fine mesh due to the stability requirements, and solving would be rather inefficient due to the degrading of the efficiency of the F_p preconditioner and the F-block multigrid as discussed in Section 2.5.2. Thus results for $Re = 1000$ are not presented for this case but for $Re = 10$ instead, although for $Re = 10$ no values are presented in [35].

In all cases the solutions appear to converge to values close to those from Ghia et al, although with differing convergence rates. The slow convergence of the first order FV scheme is evident in Figure 2.24. Even at $Re = 100$ the first order scheme requires a very fine mesh to get accurate results. At $Re = 1000$ even the finest mesh with more than 16 million cells produces a solution that is far from converged. Nevertheless, this comparison shows the behaviour expected from a first order scheme.

Judging by the distance to the reference solution and the distance between sequentially refined meshes, the SDFEM stabilised finite element discretisation achieves similar accuracy as the FV discretisation with far fewer degrees of freedom, compare, for example, the lines *bcfvs* 256^2 *cells* in Figure 2.24 with *feins* 257^2 *nodes*[16] in Figure 2.23. Note that oscillations remain for the coarse mesh SDFEM solutions, in the $Re = 100$ regime and even more pronounced in the $Re = 1000$ regime. In both regimes the two finest meshes shown in Figure 2.23 show no visible difference, thus the solution can be regarded as well converged, even though converged to slightly different values than the reference solution.

[16] `FEINS` is the name of the FE solver of this work. The name is an abbreviation for Finite Elements for Incompressible Navier Stokes. The FV has been given the name `BCFVS`, Block Cartesian Finite Volume Solver.

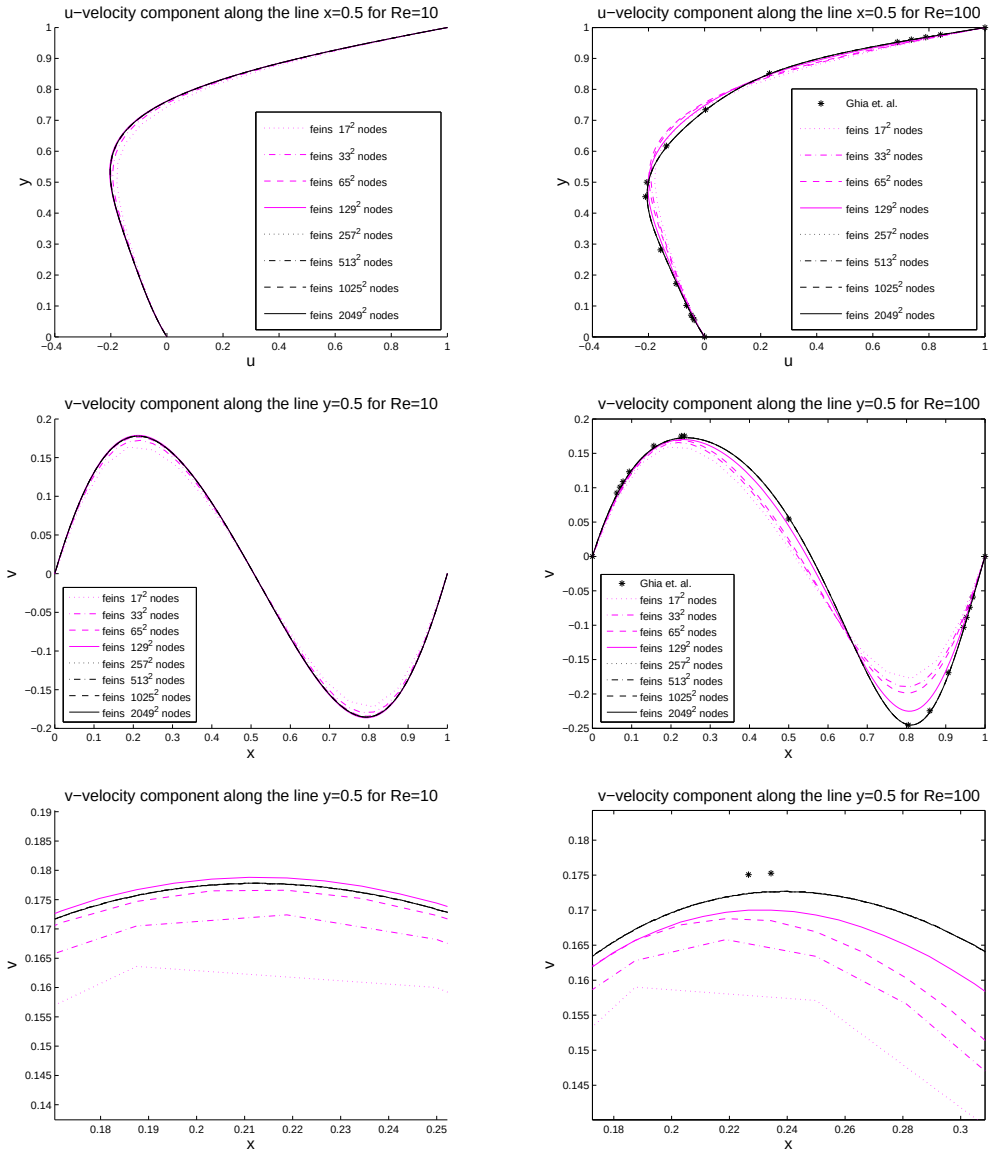


Figure 2.22: Galerkin FEM velocity profiles (denoted by feins) for the lid driven cavity problem along the lines $x = 0.5$ and $y = 0.5$ for $Re = 10$ and $Re = 100$ (and close-up view).

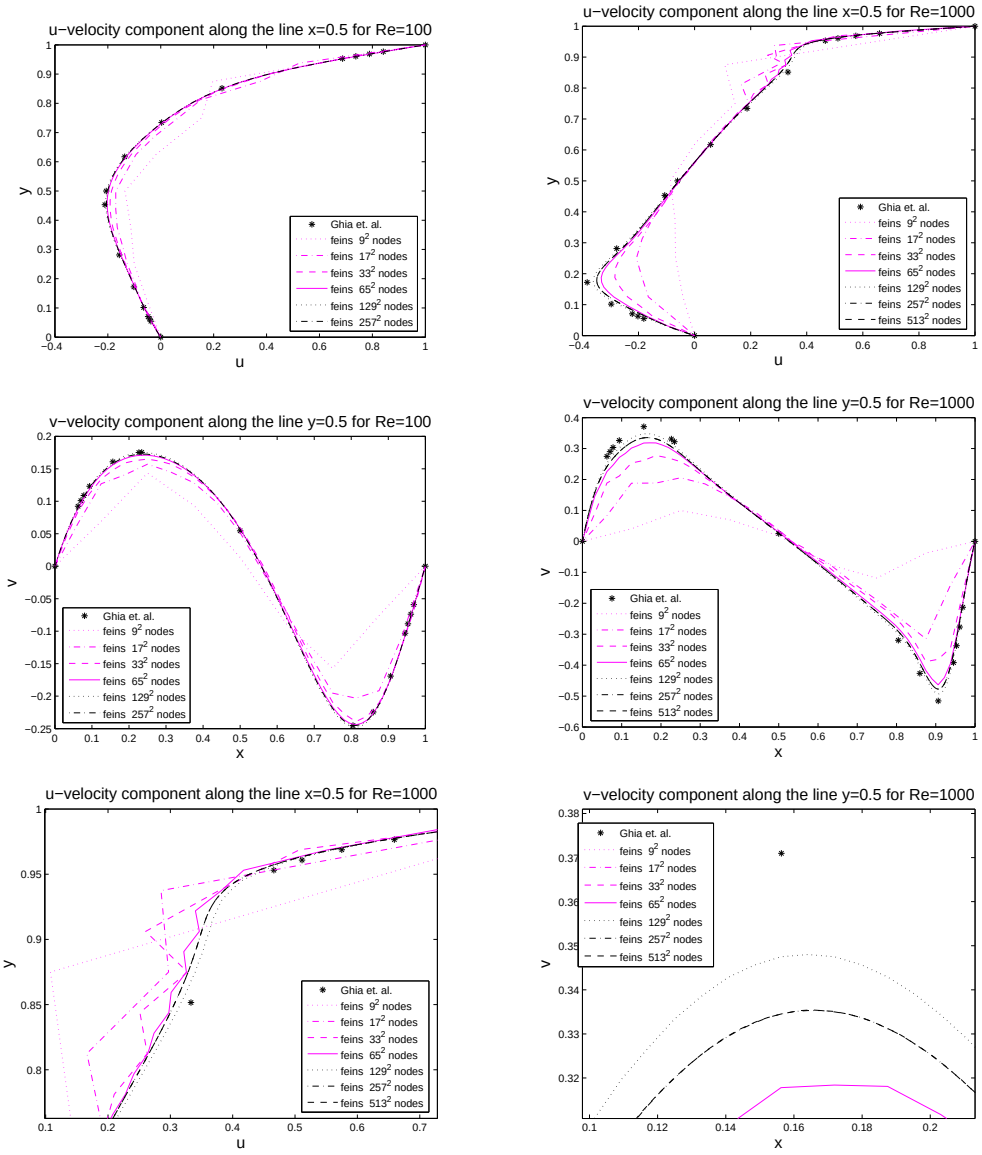


Figure 2.23: SDFEM velocity profiles (denoted by feins) for the lid driven cavity problem along the lines $x = 0.5$ and $y = 0.5$ for $Re = 100$ and $Re = 1000$ (and close-up view).

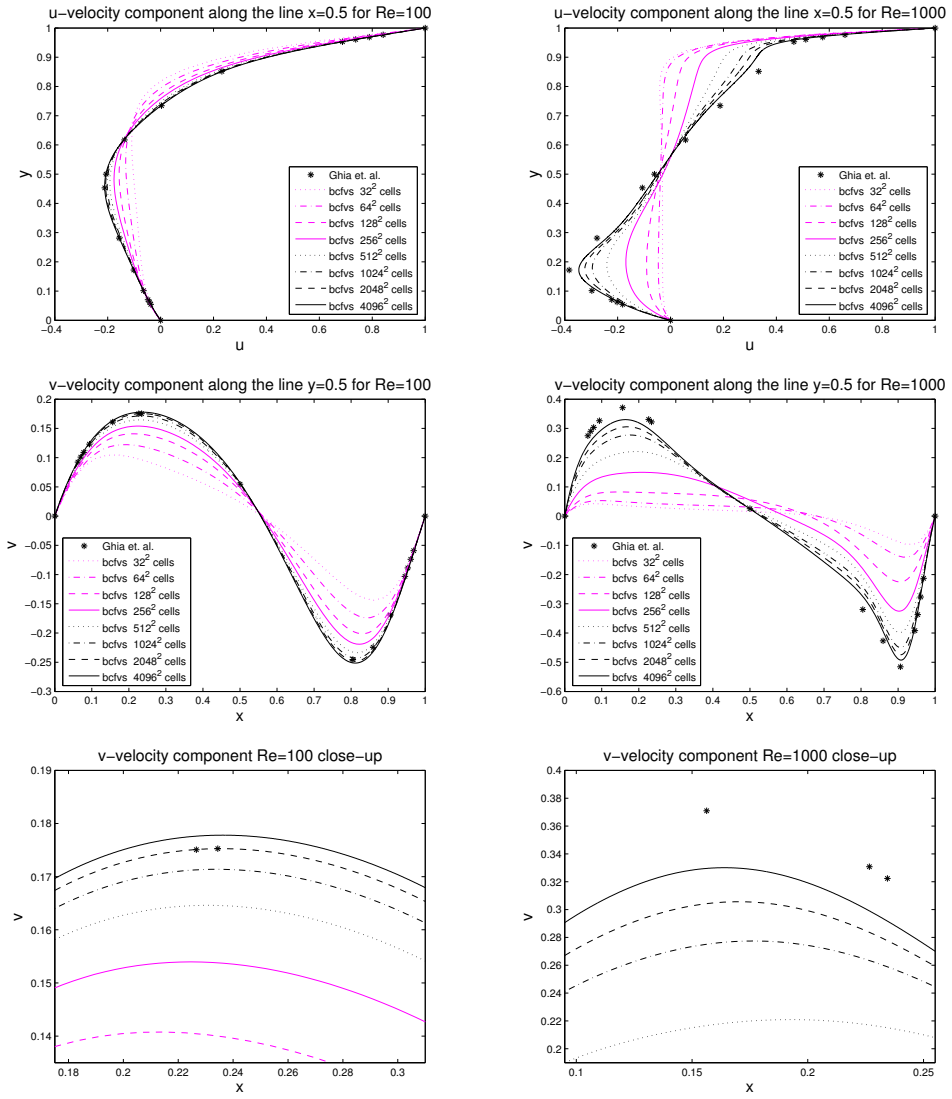


Figure 2.24: FV velocity profiles for the lid driven cavity problem along the lines $x = 0.5$ and $y = 0.5$ for $Re = 100$ and $Re = 1000$ (and close-up view).

In the case of the Galerkin finite element discretisation fast convergence is observed as well, see Figure 2.22. Note that in both Reynolds number regimes there is no visible difference between the four finest meshes, not even in the closeup views at the bottom of the figure.

2.7.2.2 Validation of the derivatives

For both shape optimisation examples the derivative values $\mathrm{D}I/\mathrm{D}\mathscr{F}$ computed by the adjoint approach have been verified by comparing them to those computed by the finite difference approximation

$$\frac{\mathrm{D}I}{\mathrm{D}\mathscr{F}_i} \approx \frac{I(\mathscr{F}_i + h) - I(\mathscr{F}_i - h)}{2h}$$

(central difference) with $h = 10^{-5}$. Note that smaller h led to unreliable results for the finite difference approximation due to cancellation effects and the inexact solves of the discretised Navier-Stokes systems. Tables 2.21 and 2.22 lists the derivative values for the finite element discretisation and finite volume discretisation as computed by the adjoint method, finite differences (FD) and the relative difference between them. The tests have been performed at $Re = 20$ for the initial geometries used in the optimisation runs for Example 2.1 and Example 2.2, on meshes producing the number of degrees of freedom as listed in the tables ($\#DOFs$). The derivative values compare well and the most significant relative errors occur only for those components of the gradient with small absolute value. In those cases the relative error of the finite difference approximation is largest due to cancellation effects. Table 2.21 does also show the total time taken by either method. In Example 2.2 where the parameter vector is of dimension ten, the advantage of the adjoint method is clearly visible.

Example 2.1		
adjoint	FD	(adj-FD)/|FD|
-1.6259e-02	-1.6261e-02	1.5167e-04
-5.1754e-01	-5.1754e-01	9.1737e-06
$Re = 20$ ($\nu = 0.05$), $\#DOFs = 112512$		
$t_{\text{adj}} = 5.2e+02$	$t_{\text{FD}} = 1.7e+03$	

Example 2.2		
adjoint	FD	(adj-FD)/|FD|
7.6111e+00	7.6111e+00	9.3887e-07
2.8666e+00	2.8666e+00	-8.6047e-08
1.1410e+01	1.1410e+01	4.8489e-07
9.5121e+00	9.5121e+00	6.3536e-07
-1.6896e+00	-1.6896e+00	-3.2079e-07
-1.7203e-02	-1.7195e-02	-4.1383e-04
-1.7018e+00	-1.7018e+00	4.2783e-06
1.3484e+00	1.3484e+00	1.0735e-06
1.7395e+00	1.7395e+00	-2.2238e-07
2.0839e-01	2.0840e-01	-3.0649e-05
$Re = 20$ ($\nu = 0.05$), $\#DOFs = 588736$		
$t_{\text{adj}} = 2.9e+01$	$t_{\text{FD}} = 3.5e+02$	

Table 2.21: Verification of adjoint derivative evaluation in the FE discretisation by comparison to finite difference values

Example 2.1		
adjoint	FD	(adj-FD)/|FD|
-1.5851e-02	-1.5851e-02	-4.9369e-07
-5.0701e-01	-5.0701e-01	7.4693e-09
$Re = 20$ ($\nu = 0.05$), $\#DOFs = 147456$		

Table 2.22: Verification of adjoint derivative evaluation in the FV discretisation by comparison to finite difference values

2.8 Conclusions

In this chapter the application of the discrete adjoint method to shape optimisation for fluid dynamics problems has been discussed. The method has been successfully applied for two discretisation techniques, namely finite elements and finite volumes. Since shape optimisation invariably requires repetitive evaluation of the performance criterion and its derivatives, particular emphasis has been given to the need for efficient solution techniques for the linear systems arising in both the forward and the adjoint problems. It has been demonstrated that efficient solution strategies for the forward problem lead to efficient solution methods for the adjoint problem in a natural way, requiring only a few modifications. Various aspects arising in the application of the discrete adjoint method have been discussed in Section 2.6, including the computation of partial derivatives of the discretisation with respect to the mesh. The utility of the presented approaches has been demonstrated by successfully applying them to two shape optimisation example problems.

A more detailed discussion of conclusions regarding this chapter, including a discussion of opportunities for future research, will be given in Chapter 4 together with the conclusions for the second major part of this thesis. This second part deals with an entirely different application of the discrete adjoint method: adaptive mesh design.

Chapter 3

Adaptive mesh design

In this chapter we consider adaptive mesh design as a topic of its own, not limited to problems from fluid dynamics. Thus, the focus is on PDEs in general, using a model problem and the finite element discretisation to develop a (hopefully) general approach. The discussion is meant to be independent of the previous chapter, apart from Section 3.4.2.2, where some results from Chapter 2 regarding the application of the discrete adjoint method in the finite element discretisation are used.

3.1 Introduction

The use of *a posteriori* error estimation in order to guide local mesh refinement is now common in the finite element (FE) solution of PDEs [3, 13, 74, 107]. Such techniques not only provide a reliable indication as to the overall accuracy of a computed solution, but also provide a reliable indication as to which regions of the computational domain contribute most (and least) greatly to the overall error in a given solution. Recently, this approach has been augmented by the development of *a posteriori* error estimation techniques for quantities of interest which are derived from the solution of the PDE. Typical examples are described in [11, 37]. This development is significant since it is frequently the case that quantities such as drag, lift, local fluxes, etc., are of more interest to the user than the overall solution. Hence the numerical solution procedure should seek to approximate these chosen quantities as efficiently as possible.

For the purposes of developing efficient adaptive finite element software reliable error estimation, whilst necessary, is not sufficient. Some mechanism is also required for using this information in order to construct an improved trial space from which to seek a new solution. One possibility is to locally enrich the polynomial order of the FE trial space (p-refinement) in regions of the domain where the error is largest and the solution is judged to be sufficiently smooth [2, 9]. Alternatively, isotropic local refinement (h-refinement) of the FE mesh may be used [70, 83] and recently significant research has focused on the efficient combination of these two [4, 78]. In this work our focus is only on techniques for improving the trial space based upon adapting the FE mesh, rather than enriching the polynomial degree, which we take to be piecewise linear throughout this chapter.

For many problems isotropic local refinement of the current mesh provides a perfectly satisfactory mechanism for the adaptive procedure. However, this is not always the case. A significant number of practical problems have solutions which possess features that are highly anisotropic (shocks or boundary layers for example). In such situations, unless one starts from a mesh designed using prior knowledge concerning the location and orientation of such features, regular mesh refinement is generally far from optimal. Numerous authors have considered mechanisms for addressing this problem through the development of techniques for anisotropic refinement. Examples include the approach of [32], which makes use of a solution-dependent metric based upon an approximation to the Hessian matrix, or the technique described in [81], which is suitable for tensor product meshes. See also the work of Kunert (e.g. [53, 54]) on *a posteriori* error estimation for anisotropic meshes or the experimental results of [7].

This chapter investigates an alternative approach to the problem of automatically adapting a mesh, or meshes, based upon *a posteriori* error estimates for problems whose solutions exhibit anisotropic behaviour. This is based upon not only computing an error estimate, but also calculating the sensitivity of this estimate to the positions of the nodes in the current mesh, which can be done efficiently with the discrete adjoint method.

This sensitivity information may then be used to improve the quality (in the sense of reducing the estimated error) of the existing mesh without increasing the dimension of the FE trial space. The ultimate goal would be to adjust the mesh in order to position the nodes in locally optimal locations, by employing techniques from mathematical optimisation for example. However, this is still an extremely demanding and computationally expensive task, even when the sensitivity information can be computed inexpensively. Furthermore, it is essential to ensure that the error estimate itself remains reliable on the meshes produced and so a number of constraints are proposed to reflect this requirement. An approach has been developed therefore that seeks to balance the goal of reducing the

estimated error for a given mesh connectivity with the need to maintain control over both the computational cost and the quality of the error estimate.

The developed approach falls into the class of *r*-refinement methods (e.g. [10, 12, 48, 56, 99]), which generally covers adaptivity by node redistribution (hence *r*-refinement). However, the use of the sensitivity of *a posteriori* error estimates distinguishes it from previous approaches, which either construct the mesh movement as the solution of a non-linear PDE (e.g. [10, 48, 56]) or by solving sequences of local optimisation problems only (e.g. [12, 99] solve small optimisation problems for each node in a Gauß-Seidel like fashion). We refer to [10] for a review of *r*-refinement approaches.

The new approach has been implemented and tested for the solution of a class of singularly-perturbed reaction-diffusion equations. These problems were selected because *a priori* analysis is available to guide the design of anisotropic meshes, for example using so-called Shishkin meshes (see e.g. [8] and the survey article [71]), thus allowing comparisons to be made against our new, more general *a posteriori* approach. The extension to wider classes of problems is also discussed in the concluding remarks in Section 4.2.

Note that the material in this chapter is in large parts a reproduction of our previous work [76].

3.2 Error estimation in the finite element method

Since the finite element method is a method for approximating the solution u of a PDE by a discrete solution u_h, it is natural that this approximation produces a certain error

$$e = u - u_h. \tag{3.2.1}$$

Error estimation generally refers to attempts to estimate this quantity, usually by seeking bounds on localised (e.g. to an element) or global norms. There are two main motivations to do this. The first is to provide the user with some indication of the accuracy of the computed approximation. The second is to provide information as to how the approximation may be improved.

Generally one distinguishes two approaches to error estimation: *a priori* and *a posteriori*. The first, *a priori* error estimation, utilises properties of the differential operator, the domain and the approximation method, but not the approximated solution, in order to derive estimates (e.g. [21]). These estimates are of an asymptotic nature, describing the behaviour of the error in terms of discretisation parameters such as the mesh size parameter h. The constants appearing in these estimates are generally unknown, thus the

estimates provide only qualitative information of the error.

The second approach, *a posteriori* error estimation, utilises the approximated solution itself in order to extract information on the error (e.g. [3, 11, 98]). This typically allows quantitative results in various forms. The basic residual error estimators (e.g. [98, Section 1.2], [3, Section 2.2]), for example, provide local lower and global upper bounds on the error, a necessary requirement for efficient adaptive mesh refinement algorithms. However, these bounds are asymptotic, with generally unknown constants. Thus, these estimators provide valuable information on the distribution of the error, but not a direct quantitative characterisation of the error. Other, computationally more expensive, approaches are required for this purpose, which provide an asymptotically exact quantitative estimate of the error. One family of such estimators are local problem estimators (e.g. [98, Section 1.3], [3, Section 3]). For functionals of the solution, for example the viscous drag in fluid dynamics, an efficient quantitative characterisation of the error is even possible to an accuracy such that it can be used to compute improved approximations of the functional, e.g. [11].

The anisotropic mesh refinement approach presented in this chapter requires *a posteriori* estimates for which the local contributions to the error estimate are continuously differentiable with respect to the positions of the nodes in the finite element mesh. This requirement is not fulfilled by all error estimation techniques.

In the remainder of this section we will first discuss this issue of suitability for the present approach to mesh refinement (Subsection 3.2.1) and then introduce the error estimation technique (Subsection 3.2.2) which is used in Section 3.4 to demonstrate the feasibility of the new automatic anisotropic refinement approach.

3.2.1 Suitability criteria for the error estimates

As already mentioned in the introduction to this section, there are two important criteria for the error estimate. It has to be an *a posteriori* estimate in order to allow automatic mesh adaption to the solution of a given PDE problem. And, to allow application of efficient techniques from mathematical optimisation, the error estimate which is used as the performance criterion for the optimisation has to be differentiable with respect to the node positions in the mesh.

In addition to these criteria the usual requirement of robustness of the error estimate is of special importance in this approach, because it will be used as the primary performance criterion for the mesh optimisation. In order to arrive at anisotropically refined meshes highly irregular meshes have to be allowed, including high aspect ratios and the implied

very small minimal angles in the triangular elements. If, for example, an error estimator grossly underestimates the error for certain deformations of the mesh, the optimisation is likely to steer towards meshes which show these deformations. Thus, the results would be useless in such a case.

Robust energy norm error estimation techniques for anisotropic meshes have, for example, been introduced in [54]. Unfortunately this estimate fails the criterion of differentiability. It describes element dimensions in terms of the longest edge and the dimension perpendicular to it. These terms are obviously not continuously differentiable with respect to the node positions. The estimator is a generalisation of the basic residual estimator [3, Section 2.2] to anisotropic meshes. The estimation of the local error contributions in terms of characteristic element dimensions h_T in general is unlikely to be a good choice for optimisation by means of node movement, because this choice discards directional information on the element and on the error behaviour. However, this directional information is crucial for anisotropic problems.

In this sense an error estimate which is computed by using the element geometry directly, without using artificial quantities such as h_T, is more desirable[1]. One class of such energy norm error estimates is defined by the local problem error estimators, e.g. [98, Section 1.3] and [3, Section 3]. For the construction of these estimators the mesh is subdivided into small patches (which may even be the individual elements). The estimator requires the solution of auxiliary PDE problems on these patches with enriched finite element spaces in order to get an approximation of the local error. Since these local problems have a very similar structure to the original finite element approximation the differentiability is preserved, and even more, the same techniques for the computation of the derivatives can be used, e.g. the discrete adjoint method. Unfortunately the analysis of this estimator in [98], as well as in [3], is based on assumptions incompatible with anisotropic meshes. Thus, its robustness with respect to mesh deformations is not guaranteed. However, this does not imply that the estimation approach is unsuitable for anisotropic meshes. It is quite possible that a different analysis could establish its robustness under assumptions compatible with anisotropic meshes.

We will not take consideration of this error estimation technique further in this thesis, but let it remain as an example of an energy norm error estimator which is sufficiently smooth to be used as an optimisation criterion for the optimisation techniques envisaged in this work.

[1] This only concerns the formulation of the estimate, i.e. the formula defining it. Analysis of the estimate in terms of characteristic length-scales is of course still possible.

3.2.2 The Dual Weighted Residual method

For the numerical experiments undertaken in this chapter the Dual Weighted Residual (DWR) approach has been selected, because it allows the estimation of the error in the approximation of a functional depending on the solution of the PDE. Such quantities, e.g. the lift or drag in fluid dynamics applications, and the error in their approximation, are of direct concern to engineers using the finite element method, while the energy norm of the error is more of a theoretical tool. Further, the use of this error estimation technique allows construction of meshes adapted to approximate the quantity of interest in an even more efficient way than on meshes adapted to produce minimal error in the energy norm.

The DWR method is introduced in this section for the case of a linear functional and a linear PDE. Further details may be found in [11] and the references therein. Consider the derived quantity

$$J(u) := \int_{\Omega} gu \, \mathrm{d}\Omega, \tag{3.2.2}$$

where g is a kernel function and $u \in \mathbb{H}^1_{g_D,\Gamma_D}$ is the unknown solution to a PDE whose weak form is

$$a(u,v) = b(v) \qquad \forall v \in \mathbb{H}^1_{0,\Gamma_D}. \tag{3.2.3}$$

Here $\mathbb{H}^1_{f,\Gamma_D}$ denotes $\left\{u \in \mathbb{H}^1(\Omega) : \quad u = f \text{ on } \Gamma_D\right\}$, where Γ_D is the (non-empty) Dirichlet part of the boundary of the domain Ω. Let u_h be defined as the solution of the FE discretisation of the weak form (3.2.3), hence $u_h \in \mathbb{V}_{g_D,h}$ such that

$$a(u_h, v_h) = b(v_h) \qquad \forall v_h \in \mathbb{V}_{0,h}, \tag{3.2.4}$$

where $\mathbb{V}_{f,h}$ denotes the FE function space $\mathbb{V}_{f,h} \subset \mathbb{H}^1_{f,\Gamma_D}$, and let the discretisation error e be defined as in (3.2.1). The *dual* in the Dual Weighted Residual method comes from its use of the solution z of the dual problem, find $z \in \mathbb{H}^1_{0,\Gamma_D}$ for which

$$a(\varphi, z) = J(\varphi) \qquad \forall \varphi \in \mathbb{H}^1_{0,\Gamma_D}, \tag{3.2.5}$$

which is well defined since J is a bounded linear functional and we assume the forward problem (3.2.3) to be well defined. Furthermore the linearity of J implies that the error in the quantity of interest, $J(u) - J(u_h) = J(e)$, and from (3.2.5)

$$\begin{aligned} J(e) &= a(e,z) \\ &= a(u,z) - a(u_h,z) \\ &= b(z) - a(u_h,z). \end{aligned} \tag{3.2.6}$$

Note that the use of any approximation $z_h \in \mathbb{V}_{0,h}$ for z in (3.2.6) does not provide any information on $J(e)$ since $b(z_h) - a(u_h, z_h) = 0 \; \forall z_h \in \mathbb{V}_{0,h}$ by Galerkin orthogonality. This implies that the FE approximation to the dual solution is not sufficient to provide a successful error estimator, at least so long as it is computed from the same space as u_h is. Of course computing an exact solution to the dual problem (3.2.5) is just as difficult as computing an exact solution to the original problem (3.2.3) and therefore it is necessary to consider an approximate solution for z, which implies that instead of the true error $J(e)$ it is only possible to obtain an estimate for it. To emphasise that this approximation is different from z_h it will, from here on, be denoted by z_{app}, and the corresponding approximation shall be denoted by

$$J_{\text{est}} := b(z_{\text{app}}) - a(u_h, z_{\text{app}}). \tag{3.2.7}$$

Several ways of defining z_{app} in order to obtain such an estimate J_{est} have been proposed, e.g.

1. solve the dual problem with a higher order method on the same mesh,
2. solve the dual problem on a uniformly refined mesh, or
3. use the same trial space to solve the dual problem, but use a higher order interpolant of z_h as z_{app}.

Solving the dual problem with a higher order method or on a uniformly refined mesh means that the error estimation will be more expensive to obtain than solving the original discretised equations. In the former case the overhead results from more degrees of freedom and larger matrix stencils, as well as the need for more accurate integration formulae. In the latter case the number of unknowns in the discrete problem will be increased by a factor of four or eight (in 2d respectively 3d). However, it does have the advantage of using the same order method, hence significant parts of the original code should be directly reusable. Furthermore, the estimated error J_{est} can be used to get an improved approximation to J, thus effectively delivering the accuracy that one would get if one solved the problem on the finer mesh.

The third alternative of using a higher order interpolant of z_h as z_{app} is described as a strong competitor in [11]. However, the quality of this approximation J_{est} relies on a super-convergence property which in turn relies on uniform meshes and smoothness of the dual solution z. As this work concentrates on anisotropic meshes this approach is less suitable, as demonstrated by the results in Section 3.4.3 of this thesis, where the second and third methods are compared for an anisotropic problem for which the meshes are

stretched accordingly. It is found that the robustness of the second method justifies its additional expense.

Error estimates such as the DWR estimate are commonly used to guide local mesh refinement in regions which contribute most to the estimated error. This may be achieved, for example, by evaluating the right-hand side of (3.2.7) separately on each element and then refining those elements with the largest contributions. This refinement is normally done by subdividing elements into smaller ones of the same or similar shape. In two dimensions this typically involves division in two [14] or four [70] sub-triangles and for tetrahedra in three dimensions either two or eight sub-elements (as in [74] or [83] respectively).

For the remainder of this chapter attention is restricted to problems in two dimensions and *h*-refinement based upon sub-division of triangles into four child elements. This method of local refinement is referred to as *isotropic local mesh refinement* in order to distinguish it from other approaches where an element may be refined differently in different spatial directions. Treatment of hanging nodes, which come about when an element is refined but a neighbouring element is not, is briefly discussed in Subsection 3.4.2.1.

3.3 Application of the discrete adjoint method

3.3.1 Overview

The approach discussed in this section may be summarised as seeking to utilise the derivatives, with respect to the positions of the nodes of the finite element mesh, of an *a posteriori* error estimate in order to guide anisotropic local mesh refinement. These derivatives may be computed at relatively low cost via the discrete adjoint technique described in Section 1.1. Numerous possibilities exist for using this derivative information, but this work concentrates on moving nodes of an existing mesh in order to reduce the estimated error (and therefore, one hopes, the actual error) in the quantity of interest. Naturally one must impose certain constraints on this node movement in order to ensure that the mesh remains suitable for finite element calculations (e.g. tangling of the elements should be avoided). However, restrictions are also necessary to ensure that the error estimate itself remains a reliable approximation of the error.

Of course the use of node movement alone can only redistribute a constant number of degrees of freedom and so for most practical problems it will be necessary to combine this with isotropic local mesh refinement in some way. Such a hybrid approach will not generally improve the asymptotic convergence properties of the underlying FE discretisation,

however realistic goals are to yield a better constant for this asymptotic behaviour and/or to reach the asymptotic regime more quickly (i.e. with fewer degrees of freedom). The remainder of this section considers a number of important issues that must be addressed in the design of this type of hybrid algorithm. These include consideration of: how the node movement may be defined, what constitutes a good quality measure in order to drive the refinement, what restrictions on the mesh are required, the implementation of appropriate restrictions on the node movement, and techniques for combining node relocation with isotropic local refinement.

3.3.2 Defining the node movement

Consider a given finite element mesh with a corresponding finite element solution and a prescribed quantity of interest that is derived from this solution. An ideal goal would be to find a set of node positions which minimise the error in this quantity of interest, maintaining the same mesh connectivity. Since this error is not generally available it is necessary to use an approximation J_{est}, based on the dual weighted residual method of Section 3.2.2 for example. As will be discussed in Section 3.3.3, J_{est} itself may not be a particularly good choice as objective function, and it may be better to use a quantity derived from it. For now let $J_{e,\text{est}}$ denote a mesh quality function based on J_{est} which is suitable as an objective function for minimisation. Furthermore, as already discussed, it is necessary to place some constraints on this minimisation so as to guarantee that the resulting mesh is appropriate for finite element calculations. These constraints are summarised in the following statement of the optimisation problem.

Problem 1:

Minimise $J_{e,\text{est}}(s)$, with respect to the node positions s, subject to:

1. the mesh approximates the boundaries of the domain,
2. the mesh is non-self-overlapping (NSO),
3. interior angles of the triangles stay bounded well below π, and
4. the aspect ratio of triangles varies smoothly (i.e. changes in the aspect ratio of neighbouring elements must be bounded).

Constraints 1, 2 and 3 are standard geometric restrictions. Note that bounding the angles from below is not appropriate since this prevents the possibility of strongly anisotropic meshes. Condition 4 is introduced as a safeguard to ensure the reliability of the DWR error estimate, as will be explained in Section 3.3.5. The realisation of these constraints and a suitable choice of $J_{e,\text{est}}$ are discussed in detail in the following subsections.

It should be noted from the outset that, regardless of the precise quantitative realisation of the four constraints or the precise definition of $J_{e,\text{est}}$, Problem 1 is highly nonlinear and its solution is extremely challenging for all but the most trivial finite element meshes. For a practical adaptive algorithm it is therefore essential to limit the computational effort that goes into attempting to find a solution, if one even exists.

3.3.3 Choice of the mesh-quality measure $J_{e,\text{est}}$

As in Section 3.2.2, the choice of J is restricted to linear functionals here for simplicity. Further, although the DWR method allows an *a posteriori* error estimate J_{est} of $J(e) = J(u_h) - J(u)$ to be computed, this quantity may well be negative and so does not make a good objective function for minimisation. The alternative choice $|J_{\text{est}}|$ is non-differentiable which suggests that an estimate $|J_{\text{est}}|^2$ may be more appropriate. Keeping in mind that this estimate is to be used as the basis for isotropic local refinement as well as node movement however leads to the following proposal

$$J_{e,\text{est}}(s) = \sum_{T \in \mathscr{T}} |J_{e,T}|^2. \tag{3.3.1}$$

Here $J_{e,T}$ represents the right-hand side of (3.2.7) evaluated on element T, of triangulation $\mathscr{T}$, only. This choice of $J_{e,\text{est}}$

- is differentiable and easy to compute,
- gives an upper bound on $|J_{\text{est}}|^2$, and
- leads to an approximate equidistribution of the local error contributions in (3.2.7).

Equidistribution of the error would imply that the resulting mesh would form a good basis for further uniform mesh refinement. However, since true equidistribution is unlikely to be achieved, further adaptive isotropic local refinement, as discussed in Subsection 3.3.6, will be a better choice.

3.3.4 Choice of the optimisation method

Gradient-based optimisation methods are generally considered to be superior to gradient-free methods, but require the availability of the gradient and sufficient smoothness of the objective function. The choice of (3.3.1) is designed to take care of the smoothness whilst the discrete adjoint technique, outlined in Section 1.1, may be used to obtain the

gradient in an efficient manner. Moreover, since Problem 1 inevitably involves inequality constraints, a method that accounts for these is clearly required.

For such problems sequential quadratic programming (SQP) is established as a reliable and efficient technique [61, 85]. In an iterative manner a sequence of quadratic optimisation problems (QP) is defined, using a quadratic model of the objective function and a linear approximation of the inequality constraints around the current iterate of the SQP algorithm. The solution of the QP defines a direction for a line search which in turn defines a new iterate for the SQP algorithm. The gradient at the new iterate is then used to update the quadratic model of the function using the BFGS update formula. This iteration becomes stationary at a solution of the problem. Algorithms for solving QP are available and well studied. In this work an interior point method is used for the QP, which has lower asymptotic cost for large problems than the classical active set strategies. A more detailed explanation can be found in [61] for example.

3.3.5 Constraints

The approach of moving nodes in order to reduce a particular error estimate requires a high level of reliability from that error estimate. Specifically, it must work for any mesh that arises during the adaptive process. If this were not the case then it would be possible to reach situations where the optimisation process was driving down the estimate of the error but the quality of this estimate was deteriorating in such a way that the true error would be allowed to go up rather than down. There has been a significant amount of research on the topic of reliable *a posteriori* error estimation for highly anisotropic problems in recent years (e.g. [54, 55]) and typical results demonstrate the quality of such estimates provided the mesh is well aligned with the solution and rapid changes in the discretisation step sizes do not occur inside the layers. Similarly, in this work we have found that provided the aspect ratio of neighbouring elements is not permitted to change too quickly, the estimate (3.2.7) is also very reliable (as we will illustrate in Section 3.4.3). Consequently, this proviso is imposed as Constraint 4 in Problem 1. This constraint may be realised by bounding the ratio of the areas of elements sharing a common edge. Specifically, for each edge in the interior of the FE mesh, let A_1 and A_2 be the areas of the two elements that share that edge. Then it is necessary that

$$c_4 A_1 - A_2 \geq 0 \qquad \text{and} \qquad c_4 A_2 - A_1 \geq 0$$

for some predetermined constant $c_4 > 0$. For the examples described in this thesis the constant is typically chosen to be $c_4 := 5$.

The angle constraint, Constraint 3 in Problem 1, is most easily implemented as one inequality for each interior angle of each of the triangular elements. To reduce computational effort and nonlinearity, it is the cosine of the angle rather than the angle itself, which is bounded. Let α_i be such an angle and let $\cos_{\max} := \cos(\alpha_{\max})$, where $\alpha_{\max}$ is the maximum permissible angle (e.g. $\alpha_{\max} = 100°$). Then the required constraint is simply

$$\cos(\alpha_i) - \cos_{\max} \geq 0,$$

which is easily evaluated from the vertex locations of the triangular element.

Since SQP is used as an iterative solver for the optimisation problem, it is possible to aid the treatment of Constraint 2, the non-self-overlapping (NSO) condition, by applying a trust region approach. Such approaches define at each iteration a region of trust for the local search and it is only this region of the search space that is explored. This trust region may be defined for each node individually: allowing the local search to move each node only within a small region as illustrated in Figure 3.1. For this purpose an additional parameter c_{trust} is defined and the trust region for each node is taken to be the polygon that is defined by a set of linear inequality constraints, where for each edge opposing the node in an element an inequality is defined such that the node may travel only up to c_{trust} of its distance orthogonal to the straight line defined by the edge. The resulting linear inequality constraints are enforced for the QP. This trust region approach may not be sufficient to guarantee NSO on its own, but allows the exclusion of a large proportion of NSO violating node movements from the outset. Additionally the NSO condition has to be verified for all iterations, and in case of violating movements these have to be scaled down such that they satisfy NSO. If each of the iterations is non-self-overlapping Constraint 2 is guaranteed for the computed solution.

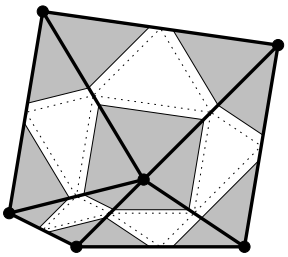


Figure 3.1: Trust region defined for each node

Realisation of Constraint 1, approximation of the domain boundaries, can easily be achieved for polygonal domains by parameterising the boundary points of the mesh with only one parameter, allowing them to move along the straight portions of the boundary, and fixing the position of those points which mark the corners of the polygon altogether.

This approach allows for generalisation to more general domains with curved boundaries, although this would be computationally more expensive and has not been undertaken here.

3.3.6 Combination with adaptive isotropic refinement

In this section two different algorithms are considered for combining mesh optimisation (or, more accurately, mesh improvement since it is far too costly to continue the optimisation algorithm to full convergence and relatively little benefit is to be derived from doing so) with isotropic local mesh refinement. In the first algorithm, referred to as `opt-adapt`, a relatively coarse initial mesh has the positions of its nodes improved by applying a number of steps of the optimisation algorithm. The resulting mesh is then assumed to be a good starting point for adaptive isotropic refinement. This is applied repeatedly until the error estimate indicates a satisfactory accuracy. Figure 3.2 gives a schematic overview of this approach.

The second algorithm also starts from a relatively coarse mesh. This time the adaptivity is performed by first refining the regions with the largest error contributions uniformly, and then relocating some or all of the newly created nodes so as to further reduce the estimated error. These two phases of the adaptive procedure are repeated alternately until the desired accuracy is achieved. Figure 3.3 gives a schematic overview of this approach which is referred to as `adapt-opt`.

Each of these hybrid approaches has its potential strengths and weaknesses. The `opt-adapt` scheme is relatively cheap, since all of the node movement is done on a relatively coarse mesh. However, it is clearly limited by the reliability of the approximation on the coarse mesh which could lead to a sub-optimal distribution of the node points at the initial stage, which would be impossible to recover from in the later stages. Also the maximum aspect ratio is determined by the connectivity of the initial mesh and the constraints that are imposed. These disadvantages would not be expected for the `adapt-opt` scheme due to the repeated introduction of new degrees of freedom by the adaptive isotropic refinement step. However this approach is more expensive since node movement must be performed on relatively fine meshes. The main practical aspects of these two schemes are discussed in Subsection 3.4.2.

3.4 Numerical Examples

In order to assess the anisotropic refinement approach proposed in this chapter a single PDE is considered. This PDE is selected because exact analytical solutions are known in

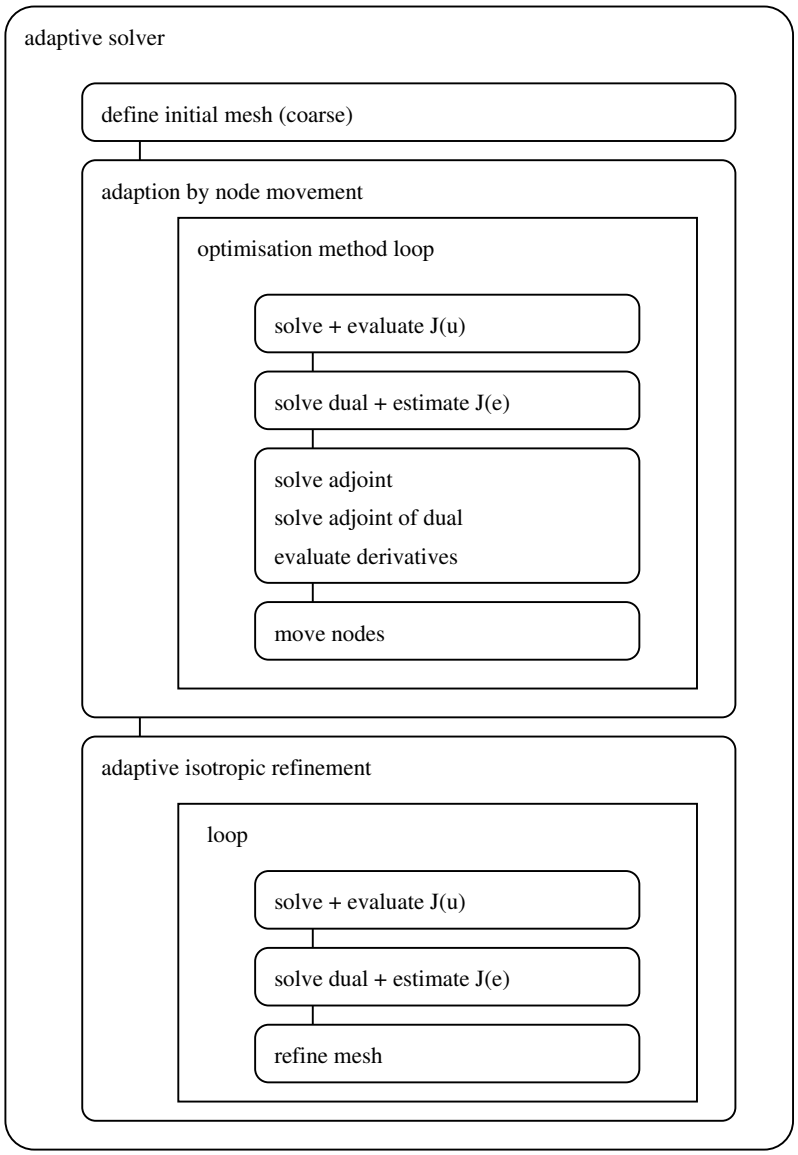


Figure 3.2: Combined adaption strategy `opt-adapt`

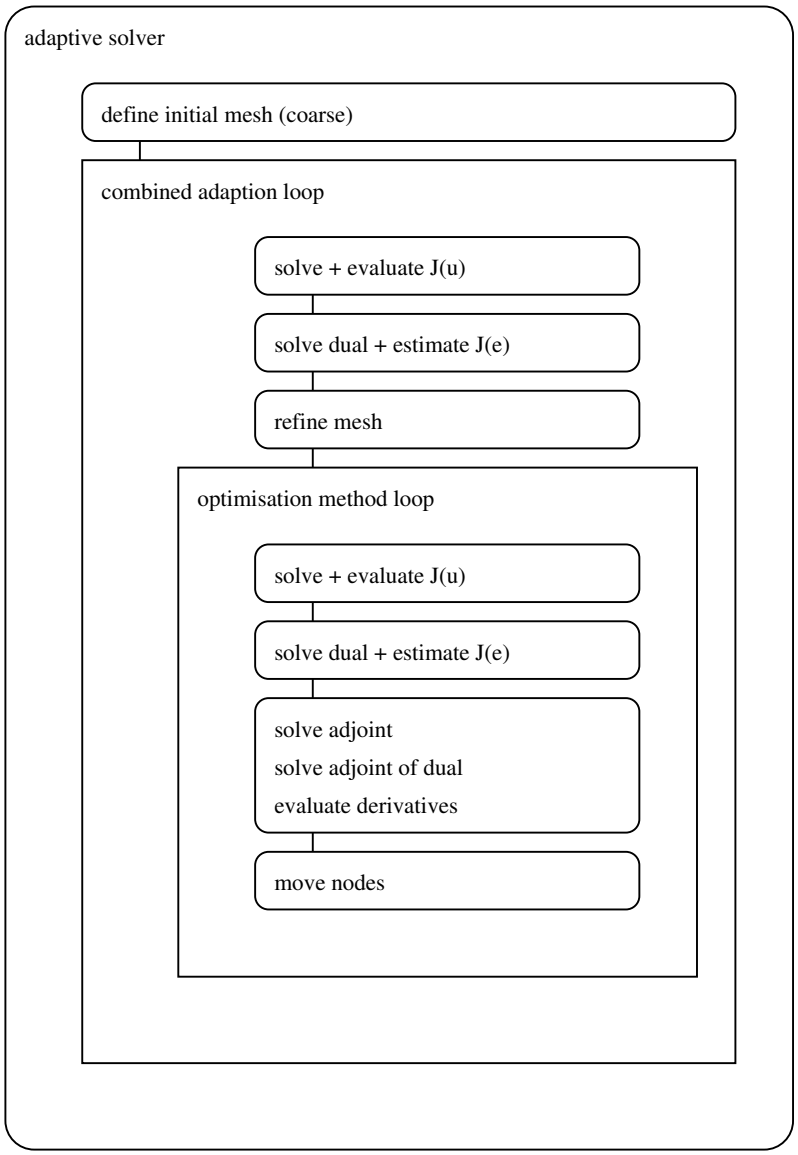


Figure 3.3: Combined adaption strategy `adapt-opt`

certain cases whilst *a priori* information regarding the solution behaviour is available in other cases. This allows the quality of the `opt-adapt` and `adapt-opt` algorithms to be contrasted against well established *a priori* results. It is envisaged that the *a posteriori* approach used here will be much more generally applicable than the *a priori* theory however.

3.4.1 Definition of the example problem

Consider the following reaction-diffusion equation:

$$\left.\begin{array}{rrcll} & -\Delta u + \frac{1}{\varepsilon^2}u &=& \frac{1}{\varepsilon^2}f & \text{in} \quad \Omega \\ \text{subject to} & u &=& 0 & \text{on} \quad \Gamma_D \\ & \frac{\partial u}{\partial n} &=& g_N & \text{on} \quad \Gamma_N, \end{array}\right\} \tag{3.4.1}$$

where $\varepsilon > 0$, $\Gamma_D \cup \Gamma_N$ is the boundary $\partial\Omega$ of the domain Ω, $|\Gamma_D \cap \Gamma_N| = 0$ and $|\Gamma_D| > 0$. As quantity of interest $J(u)$ consider the integral of the normal derivative of u over the Dirichlet boundary,

$$J(u) := \int_{\Gamma_D} \frac{\partial u}{\partial n}\, \mathrm{d}\Gamma. \tag{3.4.2}$$

Numerous alternative choices could have been made but this is selected partly for its simplicity and partly for its similarity to the viscous drag terms that are frequently of interest in fluid dynamics problems, e.g. Example 2.2 from the Chapter 2. The weak form of problem (3.4.1) is to find $u \in \mathbb{H}^1_{0,\Gamma_D}$ such that

$$a(u,v) = b(v) \qquad \forall v \in \mathbb{H}^1_{0,\Gamma_D},$$

where

$$\begin{aligned} a(u,v) &:= \int_\Omega \left(\nabla u.\nabla v + \frac{1}{\varepsilon^2} uv \right) \mathrm{d}\Omega, \\ b(v) &:= \int_\Omega \frac{1}{\varepsilon^2} vf \, \mathrm{d}\Omega + \int_{\Gamma_N} v g_N \, \mathrm{d}\Gamma. \end{aligned}$$

Furthermore it is possible to derive an alternative representation of $J(u)$ by using the fact that

$$\int_{\Omega} \left(\nabla u.\nabla v + \frac{1}{\varepsilon^2} uv \right) \, \mathrm{d}\Omega = \int_{\Omega} \frac{1}{\varepsilon^2} vf \, \mathrm{d}\Omega + \int_{\partial\Omega} v \frac{\partial u}{\partial n} \, \mathrm{d}\Gamma$$

holds for the solution u of (3.4.1) for all $v \in \mathbb{H}^1$. Re-arranging the above equation yields

$$J(u) = \int_{\Gamma_D} v \frac{\partial u}{\partial n} \, \mathrm{d}\Gamma = \int_{\Omega} \left(\nabla u.\nabla v + \frac{1}{\varepsilon^2} uv \right) \, \mathrm{d}\Omega - \int_{\Omega} \frac{1}{\varepsilon^2} vf \, \mathrm{d}\Omega - \int_{\Gamma_N} v \frac{\partial u}{\partial n} \, \mathrm{d}\Gamma,$$

and selecting $v \equiv 1$ gives two different expressions for $J(u)$. Whilst these are equivalent for the exact solution, the second expression provides a better order of convergence for the FE solution and is therefore used from here on. This can conveniently be formulated as

$$J(u) = a(u,1) - b(1). \tag{3.4.3}$$

At this point it should be noted that (3.4.3) is not a linear functional in u, but an affine one. This does not present any significant difficulties however since, following [11, Chapter 6] for example, the dual problem that must be solved for the error estimation simply becomes

$$a(\varphi, z) = a(\varphi, 1) \qquad \forall \varphi \in \mathbb{H}^1_{0,\Gamma_D}. \tag{3.4.4}$$

That is, it utilises the linear part of the functional only. Otherwise the application of the DWR method is as described in Section 3.2.2.

For the purposes of this numerical assessment three example cases are selected with different choices of Ω, f, Γ_D, Γ_N and g_N.

Example 3.1.

Domain: *Unit square,* $\Omega = (0,1)^2$.

Boundary conditions:

$$u = 0 \qquad \forall (x,y) : x = 1$$

$$\frac{\partial u}{\partial n} = 0 \qquad \forall (x,y) : y = 1 \vee y = 0$$

$$\frac{\partial u}{\partial n} = \tfrac{1}{\varepsilon} e^{-1/\varepsilon} \qquad \forall (x,y) : x = 0$$

RHS function:

$$f = 1$$

This problem has the exact solution

$$u(x,y) = 1 - e^{(x-1)/\varepsilon}$$

which, when ε is small, involves a steep boundary layer next to the $x = 1$ boundary. Furthermore, using the representation (3.4.2), it is easy to show that $J(u) = -1/\varepsilon$ for this example. The solution for this example problem is illustrated in Figure 3.4 (left) for the case $\varepsilon = 1/10$.

Example 3.2.

Domain: *Unit square,* $\Omega = (0,1)^2$.

Boundary conditions:

$$u = 0 \qquad \forall (x,y) : x = 1 \vee y = 1$$

$$\frac{\partial u}{\partial n} = \frac{1}{\sqrt{2}\varepsilon} e^{\frac{-1}{\sqrt{2}\varepsilon}} e^{\frac{(y-1)}{\sqrt{2}\varepsilon}} \qquad \forall (x,y) : x = 0$$

$$\frac{\partial u}{\partial n} = \frac{1}{\sqrt{2}\varepsilon} e^{\frac{(x-1)}{\sqrt{2}\varepsilon}} e^{\frac{-1}{\sqrt{2}\varepsilon}} \qquad \forall (x,y) : y = 0$$

RHS function:

$$f(x,y) = \frac{1}{2}\left(1 - e^{\frac{(x-1)}{\sqrt{2}\varepsilon}} + 1 - e^{\frac{(y-1)}{\sqrt{2}\varepsilon}}\right)$$

This example is more challenging than the first, with a truly two dimensional solution.

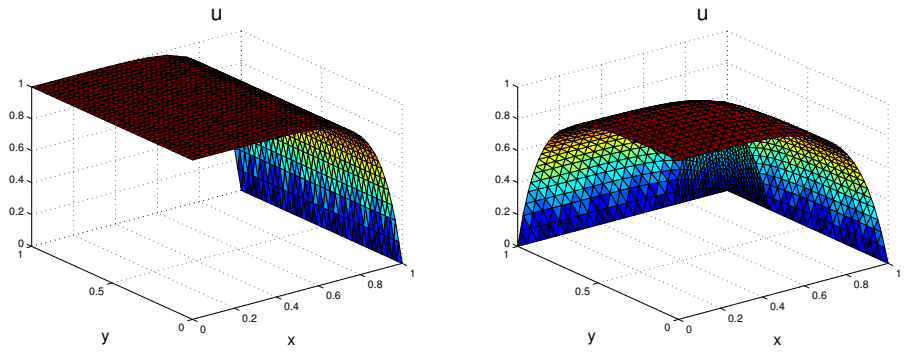


Figure 3.4: Solutions for Example 3.1 (left) and Example 3.2 (right) both for $\varepsilon = 1/10$

However it is still simple enough to permit a known exact solution:

$$\begin{aligned} u(x,y) &= \left(1 - e^{\frac{(x-1)}{\sqrt{2}\varepsilon}}\right)\left(1 - e^{\frac{(y-1)}{\sqrt{2}\varepsilon}}\right), \\ J(u) &= 2\left(1 - e^{\frac{-1}{\sqrt{2}\varepsilon}} - \frac{1}{\sqrt{2}\varepsilon}\right). \end{aligned}$$

The solutions for Example 3.2 is illustrated in Figure 3.4 (right) for the case $\varepsilon = 1/10$.

Example 3.3.

Domain: *Square with a square hole,*

$$\Omega = (-1,1)^2 \setminus (-\frac{1}{5},\frac{1}{5})^2.$$

Boundary conditions:

$$u = 0 \quad \forall (x,y) \in \left[-\frac{1}{5},\frac{1}{5}\right]^2$$

$$\frac{\partial u}{\partial n} = 0 \quad \forall (x,y) : (x = \pm 1) \vee (y = \pm 1)$$

RHS function:

$$f = 1$$

This is a problem for which an exact solution is not known, however to illustrate the domain and the character of the solution Figure 3.5 shows a mesh and a computed solution for $\varepsilon = 1/10$.

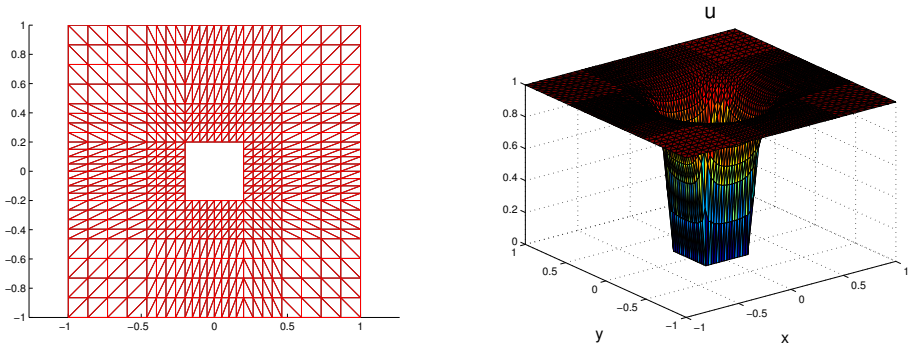


Figure 3.5: Example mesh and corresponding solution for Example 3.3 ($\varepsilon = 1/10$)

3.4.2 Practical aspects

Although the main theoretical issues associated with the optimisation problem, Problem 1, are addressed in sections 1.1, 3.2 and 3.3 a number of practical, implementation, issues are also worthy of discussion.

3.4.2.1 Hanging nodes

A difficulty related to local mesh refinement is the introduction of hanging nodes which come about when an element is refined but a neighbouring element is not. In order to ensure the resulting function space is conforming three different approaches are commonly used. The first uses temporary closure elements, as in [83, 91] for example, the second constructs special basis functions for elements with hanging nodes, e.g. [92, 100], whilst the third simply restricts the values at the hanging nodes so as to ensure a conforming solution, e.g. [69, 72]. In this case care needs to be taken to ensure that no element consists of hanging nodes only, since such a refinement would not enhance the solution quality. In any case the number of hanging nodes per element needs to be controlled in order to ensure non-degrading of the mesh or function space. For the results described in this chapter the common rule is adopted that the refinement level of neighbouring elements must not differ by more than one, otherwise the coarser element is refined. A variant of the third approach similar to that in [69] is employed, whereby the solution value at each hanging node is constrained to be the linear interpolant of the values at the nodes that lie at the end of the edge from which it hangs. This is achieved here by explicitly modifying the finite element system to restrict it to the conforming subspace of the non-conforming FE function space. However an alternative would be to follow [57], where the local element matrices are assembled without regard for the hanging nodes and then the linear interpolation constraints are imposed as part of the iterative solution procedure (through

the introduction of an additional projection step).

3.4.2.2 Adjoint equations for derivatives of the error estimate

This section provides an outline of how the discrete adjoint technique may be applied to evaluate the derivative of the mesh-quality measure $J_{e,\mathrm{est}}(s)$ with respect to the node positions. Here $J_{e,\mathrm{est}}$ takes the role of $\widetilde{I}$ in (1.1.1) and it is considered to be a function of $\underline{u}$, the coefficient vector of the primal finite element solution, and $\underline{z}$, the coefficient vector of the finite element solution of the dual problem. These quantities $\underline{u}$ and $\underline{z}$ are themselves dependent upon the vector s of node coordinates for the underlying FE mesh. To complete the notation of Section 1.1 therefore, let

$$\omega := \begin{bmatrix} \underline{u} \\ \underline{z} \end{bmatrix} \tag{3.4.5}$$

$$\text{and} \qquad R(\omega, s) := \begin{bmatrix} K(s) & \\ & K_{\mathrm{dual}}(s) \end{bmatrix} \omega - \begin{bmatrix} \underline{b}(s) \\ \underline{b}_{\mathrm{dual}}(s) \end{bmatrix}, \tag{3.4.6}$$

where $K(s)$ and $\underline{b}(s)$ are the stiffness matrix and right-hand side from the FE discretisation of the primal problem and $K_{\mathrm{dual}}(s)$ and $\underline{b}_{\mathrm{dual}}(s)$ are those of the dual problem utilised in the error estimate. The adjoint equation (1.1.6) therefore becomes

$$\begin{bmatrix} K^T & \\ & K_{\mathrm{dual}}^T \end{bmatrix} \Psi = \frac{\partial J_{e,\mathrm{est}}}{\partial \omega} \tag{3.4.7}$$

and the total derivatives $\mathrm{D}J_{e,\mathrm{est}}/\mathrm{D}s$ can be evaluated according to (1.1.7) as

$$\frac{\mathrm{D}J_{e,\mathrm{est}}}{\mathrm{D}s} = \frac{\partial J_{e,\mathrm{est}}}{\partial s} - \Psi^T \begin{bmatrix} \frac{\partial (K\underline{u} - \underline{b})}{\partial s} \\ \frac{\partial (K_{\mathrm{dual}}\underline{z} - \underline{b}_{\mathrm{dual}})}{\partial s} \end{bmatrix}. \tag{3.4.8}$$

The partial derivatives with respect to the node positions s, $\partial J_{e,\mathrm{est}}/\partial s$,$\partial(K\underline{u} - \underline{b})/\partial s$ and $\partial(K_{\mathrm{dual}}\underline{z} - \underline{b}_{\mathrm{dual}})/\partial s$, can be evaluated using the approach from Section 2.6.2.1, i.e. by formulating the terms as sums according to quadrature formulas, applying the chain rule of differentiation to reduce the problem to the evaluation of simpler terms and using explicit formulae for these simpler terms as given in Proposition 1. As the integrals consist exclusively of terms which have been present in the model problem in Section 2.6.2.1, a more detailed derivation is omitted at this point. Again, the Jacobians $\partial(K\underline{u} - \underline{b})/\partial s$ and $\partial(K_{\mathrm{dual}}\underline{z} - \underline{b}_{\mathrm{dual}})/\partial s$ form sparse matrices which are only used for one matrix-vector product in (3.4.8). Thus, time and memory may be saved by computing this post-product

with Ψ^T directly. For this purpose the adjoint solution vector Ψ can be treated the same way as ω, as a concatenation of $\underline{u}$ and $\underline{z}$ components:

$$\Psi = \begin{bmatrix} \underline{\Psi}_u \\ \underline{\Psi}_z \end{bmatrix}.$$

As a final remark on the adjoint equations themselves note that the location of hanging nodes is determined by the position of their parents and so should not be included in the vector s. Furthermore, any implementation of the expression (3.4.8) may have its correctness verified by comparing the values of $\mathrm{D}J_{e,\mathrm{est}}/\mathrm{D}s$ to those computed, at much greater expense, through the use of finite differences. For the implementation in this thesis this has been done, confirming the correct implementation of the discrete adjoint method.

3.4.2.3 Optimisation

As already noted in Section 3.3.2 to solve Problem 1 precisely would be prohibitively expensive, and possibly too demanding for even the most robust optimisation software implementations. Nevertheless, application of optimisation techniques, such as SQP for example, does provide an excellent way of improving the mesh quality measure whilst maintaining a mesh that satisfies the desired constraints. In most cases it should be sufficient to apply a fixed number of SQP steps to find a satisfactorily improved mesh. However, utilisation of the trust region approach described in Subsection 3.3.5 implies that the distance which nodes can be moved per step is restricted by the size of the surrounding elements. Hence, for a finer mesh more steps will be required to concentrate a certain proportion of the mesh into a solution feature such as a boundary layer. This must be reflected in the choice of the number of steps allowed.

Furthermore, in the `adapt-opt` variant of the anisotropic refinement it is sensible to restrict the node movement to an even greater extent by keeping the number of optimisation steps very small since this will be performed on a sequence of increasingly fine meshes. For the implementation of this approach therefore the node movement is restricted only to the initial trust region. This still allows the aspect ratio to almost double with each refinement level, but also prevents excessive stretching of elements to improve the general reliability. Additionally, by restricting the node movement to these areas a very small number of SQP steps should suffice to give an appropriate reduction in $J_{e,\mathrm{est}}$, with further improvements being left to the next isotropic refinement step.

3.4.3 Assessment of the quality of the error estimate

As discussed in Section 3.2.2 the calculation of the dual solution z, as part of the DWR error estimation process, requires the use of a different trial space to that used to approximate the primal solution u. In this section two alternative methods for obtaining a representation of z are considered. The first method uses a global refinement of the primal mesh in order to calculate z whereas the second method calculates z on the same mesh as u, which is itself a global refinement of some coarser mesh, but then uses the nodal values for z to represent z as a piecewise quadratic on the coarser mesh. Figure 3.6 illustrates the performance of the resulting error estimate J_{est} and optimisation objective function $J_{e,\text{est}}$ for these two methods when solving Example 3.1 (with $\varepsilon = 10^{-2}$) on two different families of anisotropic meshes. The left-hand side of the figure shows results for "Shishkin-like" meshes which are split into two regions: a boundary layer of thickness $1-a$ and an interior region of thickness a. The mesh in both regions is uniform and contains an equal number of elements. An illustrative mesh is shown at the top and results as the parameter a is varied are shown for the two methods. The right-hand side of the figure shows corresponding results for a slightly different family of meshes: again half the elements are in a boundary layer of thickness $1-a$, but now the interior region is meshed anisotropically as well, so as to prevent the aspect ratio of neighbouring elements from differing by more than a factor of five.

It appears from the results shown in Figure 3.6 that, at least for this artificial example, both methods provide reliable error estimates even when the aspect ratio within the boundary layer is large. Furthermore, both estimates are reliable for even larger aspect ratio elements provided this aspect ratio is not permitted to change too suddenly (the second family of meshes). Overall the first of the two methods of representing the dual solution (global uniform refinement) appears to be superior in the sense that the optimal value for a, as identified by J_{est}, is closer to the optimal value for $J(e)$. Consequently, the results which follow are all based upon this method.

Of course the calculation of the contributions to J_{est} from each element may also be used to drive adaptive isotropic refinement. In doing this here the *fixed-error-reduction* approach is followed. This is described in [11, Chapter 4] for example and comprises of sorting the absolute value of all element error contributions and then, starting from the largest contribution, marking all elements until they make up a fixed proportion $\gamma = 0.3$ of the sum of all contributions. In contrast to [11] the implementation here does not perform coarsening so as to avoid the more complicated data structures which would be needed to support this. An example of a mesh refined this way can be seen in Figure 3.7, where adaptive isotropic refinement was utilised to find a solution to Example 3.3 for $\varepsilon = 10^{-3}$.

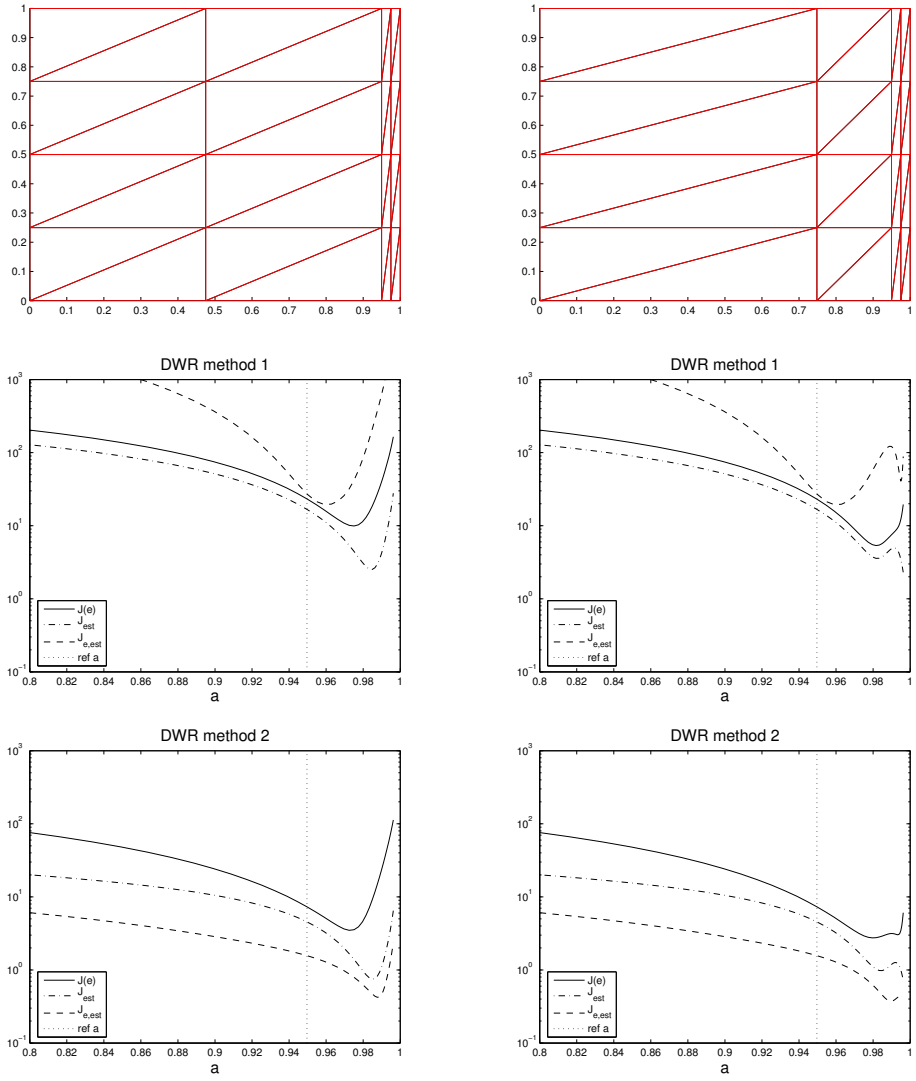


Figure 3.6: Error estimates on parametric meshes for Example 3.1, $\varepsilon = 10^{-2}$

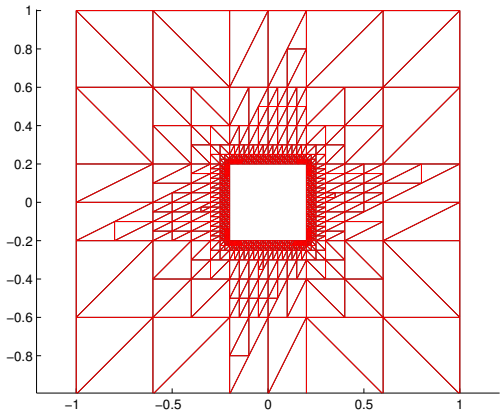


Figure 3.7: Example of an adaptively isotropically refined mesh

3.4.4 Results

Figures 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 illustrate the performance of the `opt-adapt` and `adapt-opt` algorithms for the three example problems considered. These are contrasted with the performance of global and local isotropic mesh refinement and with Shishkin meshes that are based upon *a priori* analysis (e.g. [8, 71] and references therein). For each example four cases are considered, with ε ranging from 10^{-1} to 10^{-4}, and in each case the relative error $|J_{\text{est}}|/|J(u)|$ is plotted against the number of nodes in the primal mesh. In the first two examples, for which the exact solution is known, $|J(e)|/|J(u)|$ is also plotted.

One of the most significant observations concerning these results is that the estimated error and the actual error follow each other closely for all of the meshes used in the production of figures 3.8, 3.9, 3.10 and 3.11. It should also be noted that all of the methods perform in a similar, optimal, manner in the mildly anisotropic regime for $\varepsilon = 10^{-1}$. Furthermore, for smaller values of ε, denoting increasingly anisotropic behaviour, both `opt-adapt` and `adapt-opt` provide significant advantage over the adaptive isotropic refinement approaches and this advantage becomes more pronounced as ε gets smaller. Indeed, both approaches achieve the desired goal of reaching the asymptotic regime more quickly (with fewer degrees of freedom) and with a better constant. Of the two new approaches `opt-adapt` is superior for these examples. This would become even more clear if the comparison was made with respect to CPU time, since `adapt-opt` requires a relatively large total number of SQP steps for increasingly large problems. The cost for this approach therefore significantly exceeds that of `opt-adapt`. It should be noted however that the implementations used to produce these results shown here are based

parameter	`opt-adapt`	`adapt-opt`
c_5	5	%
$\alpha_{\max}$	100°	100°
SQP-steps	40	5
c_{trust}	0.95	0.95

Table 3.1: Parameters for presented examples

upon non-optimised Matlab code and so it is not appropriate, or particularly meaningful, to include any detailed timing information.

A final reference to figures 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 shows that neither of the adaptive approaches considered performs as well as the use of Shishkin meshes when ε is very small. This is unsurprising since the Shishkin approach requires *a priori* analysis, which will not be feasible for more general classes of problem. The results of the computational experiments do indicate however that the fully automated *a posteriori* approaches can deliver anisotropic meshes that are a significant step towards this optimal behaviour. To conclude this section Table 3.1 summarises the numerical values used for various parameters in the computational experiments reported, whilst Figures 3.14 and 3.15 show initial and final meshes obtained for Example 3.3. Note that the final meshes and solutions do not appear to be particularly sensitive to the precise choice of parameters.

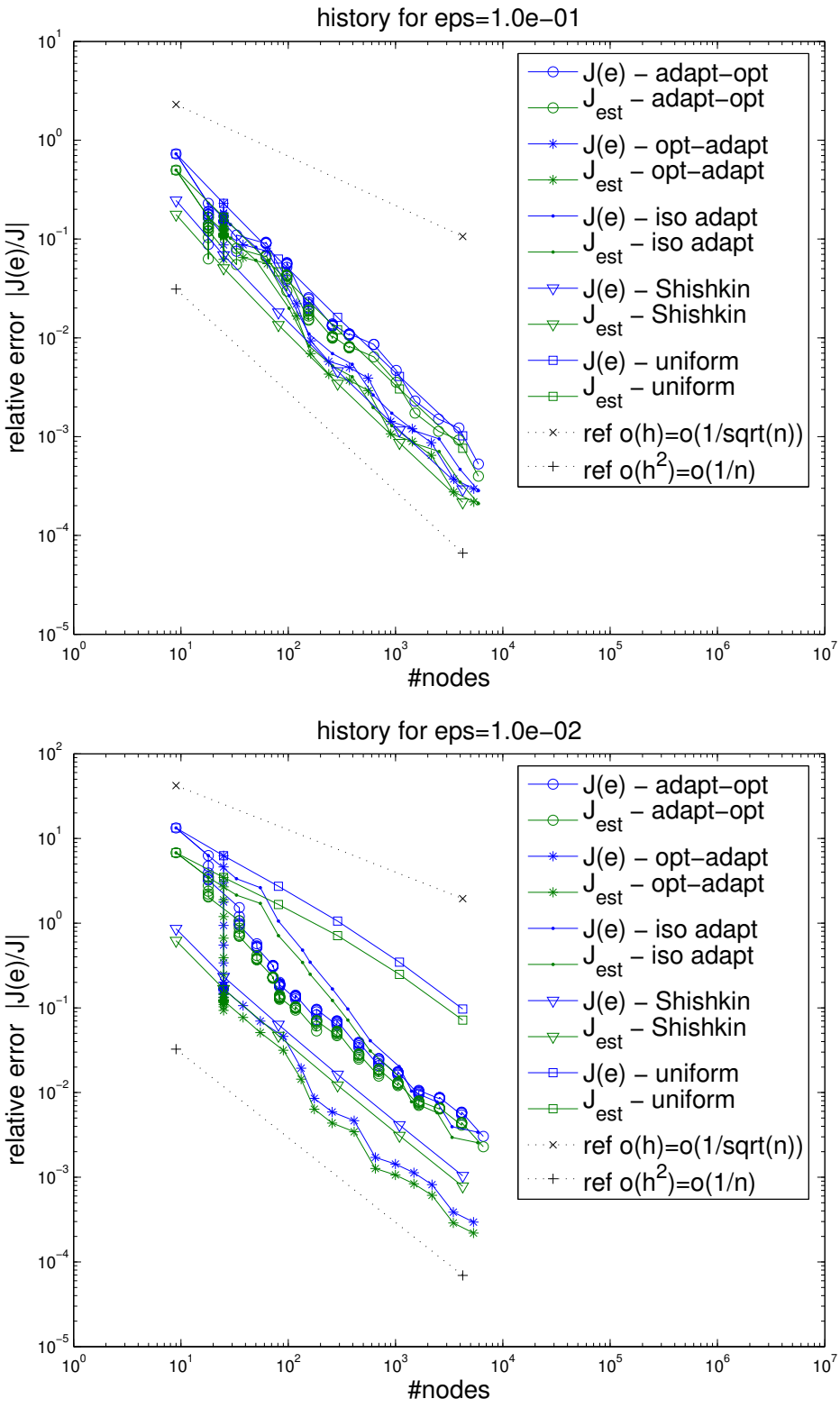


Figure 3.8: Convergence histories for Example 3.1, $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-2}$

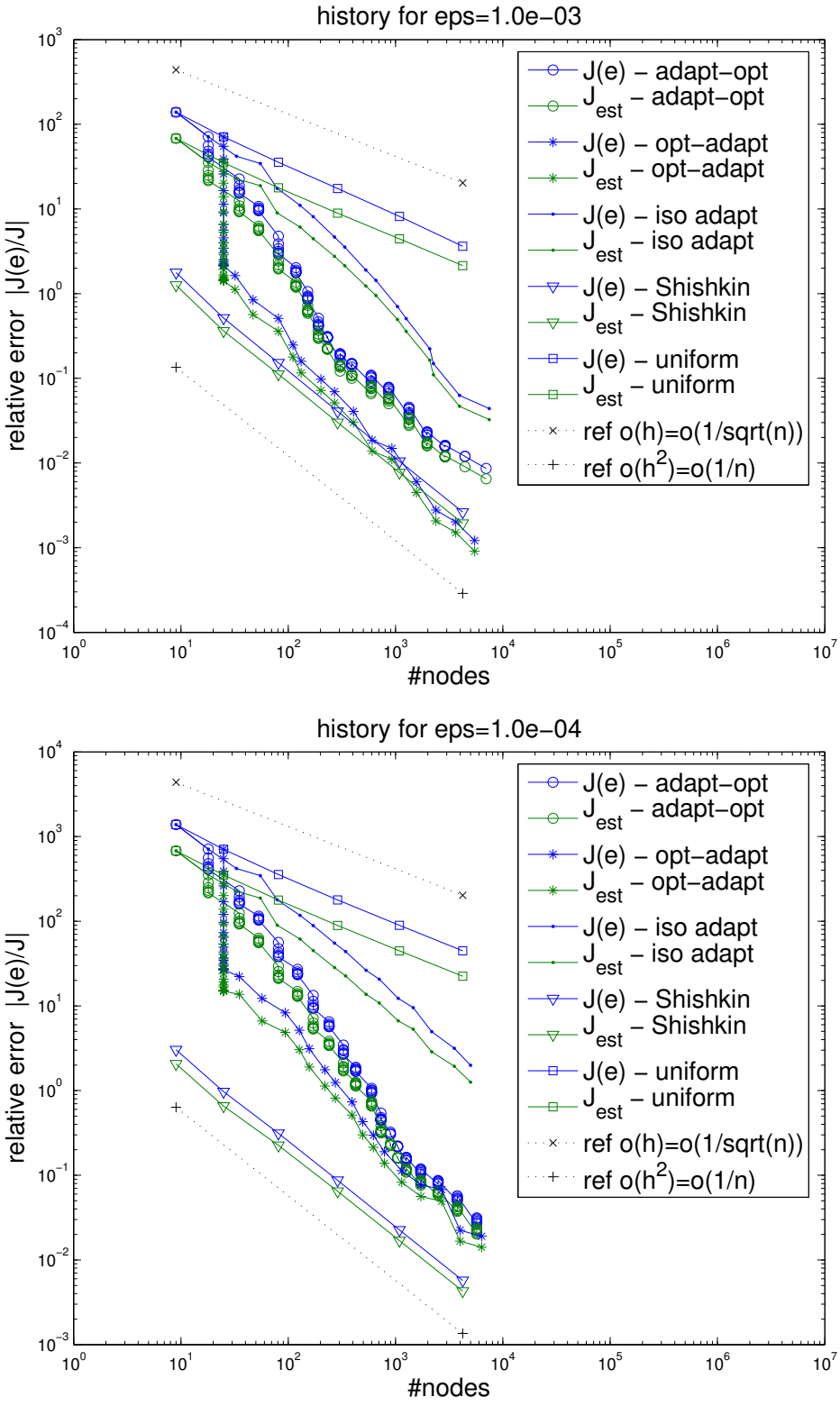


Figure 3.9: Convergence histories for Example 3.1, $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-4}$

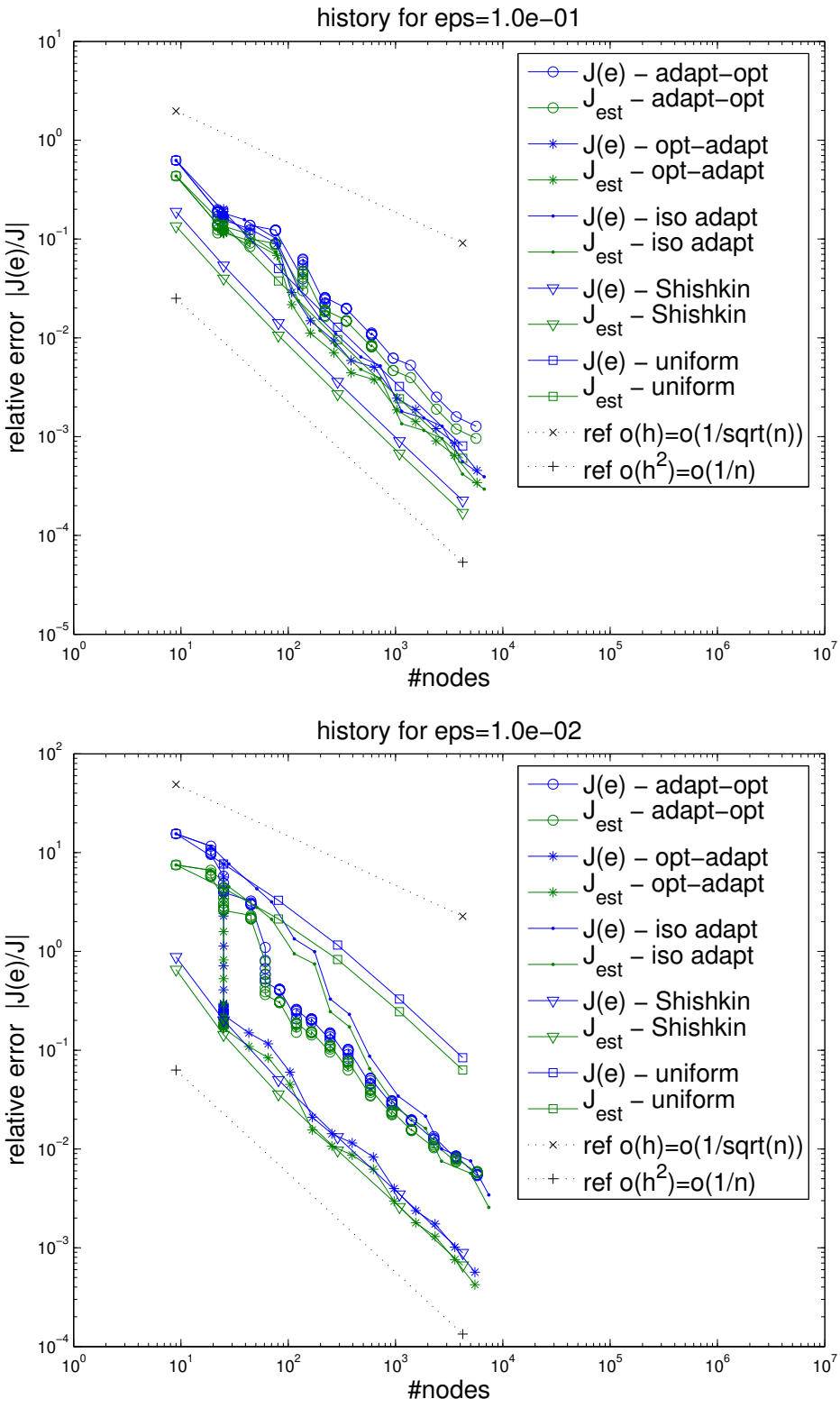


Figure 3.10: Convergence histories for Example 3.2, $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-2}$

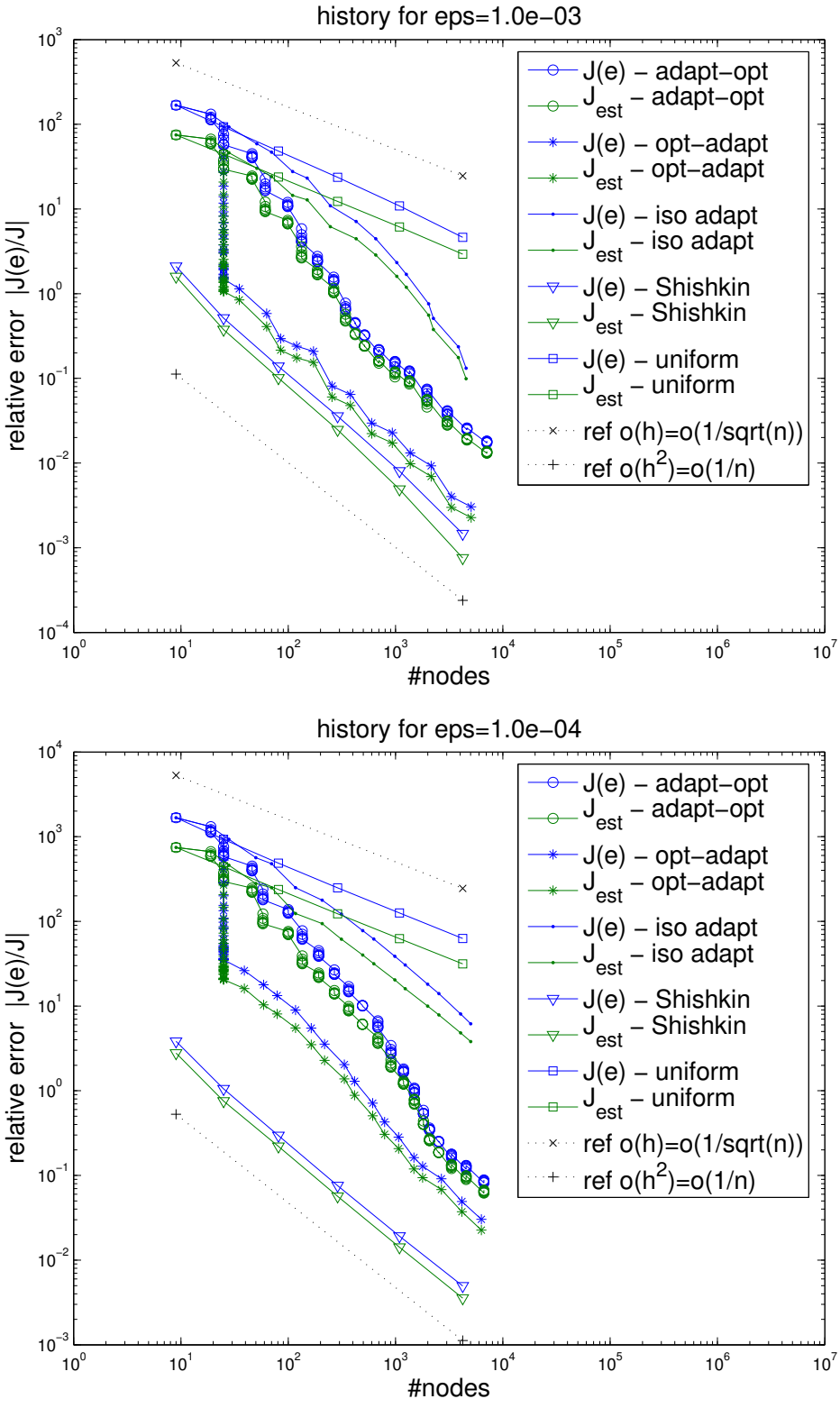


Figure 3.11: Convergence histories for Example 3.2, $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-4}$

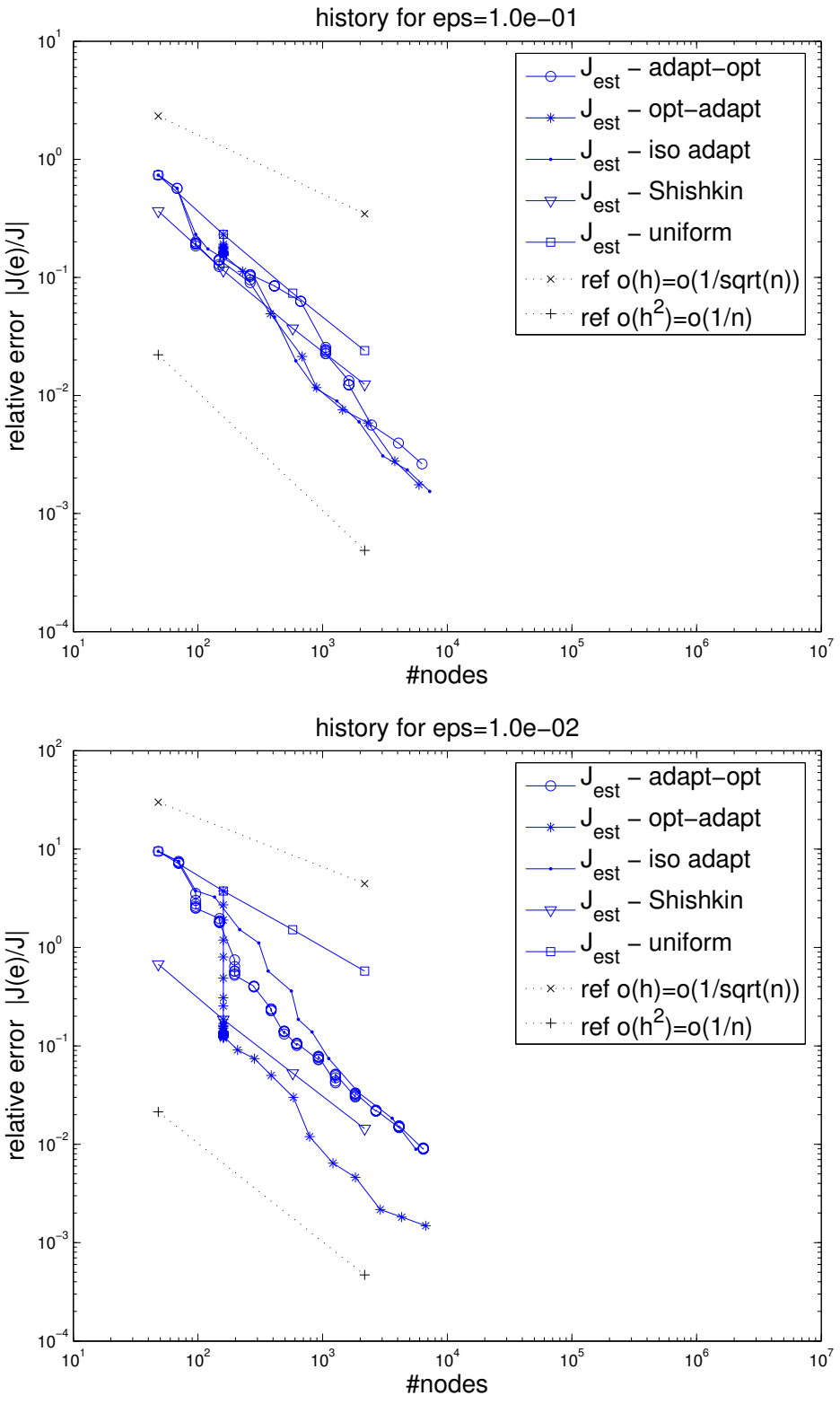


Figure 3.12: Convergence histories for Example 3.3, $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-2}$

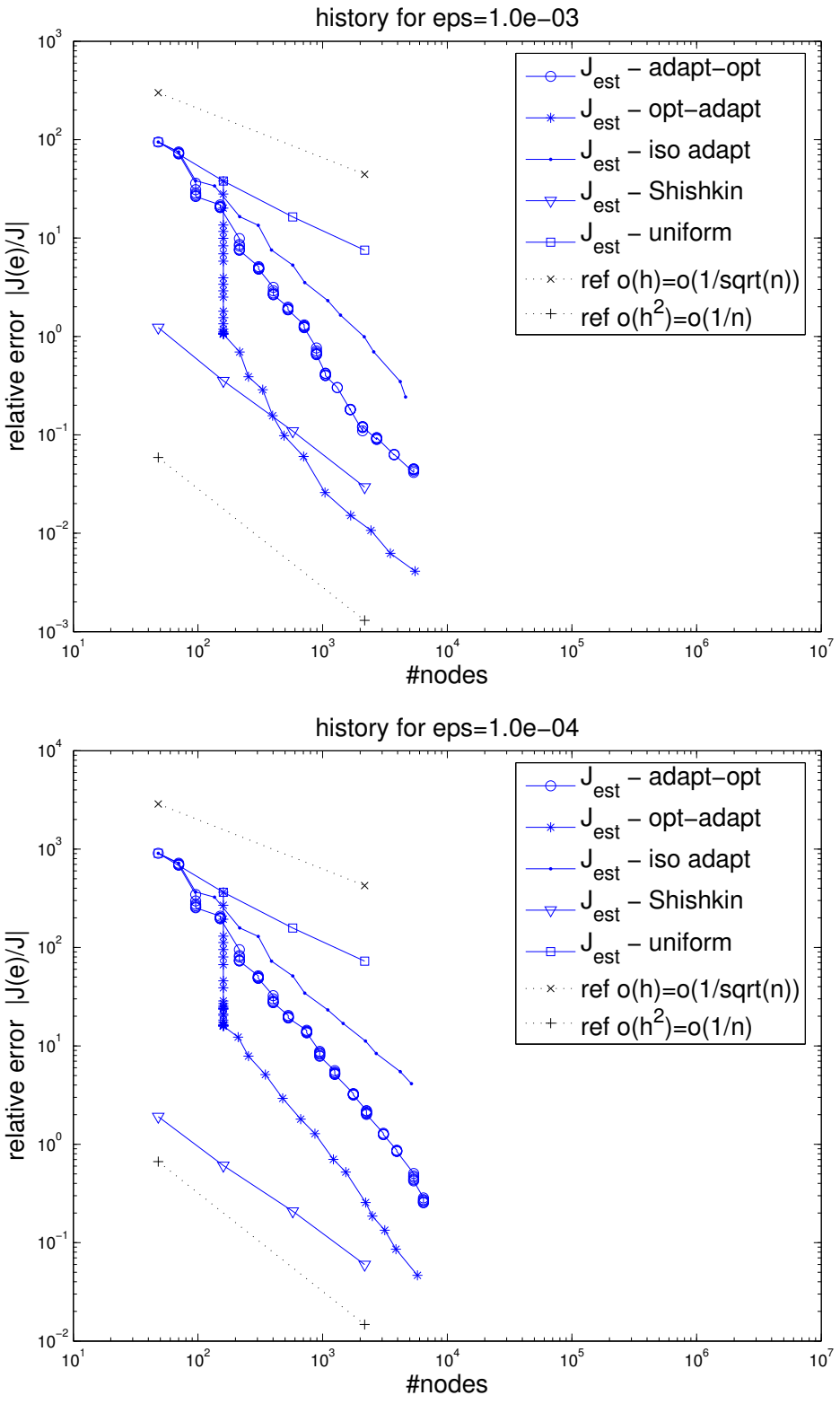


Figure 3.13: Convergence histories for Example 3.3, $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-4}$

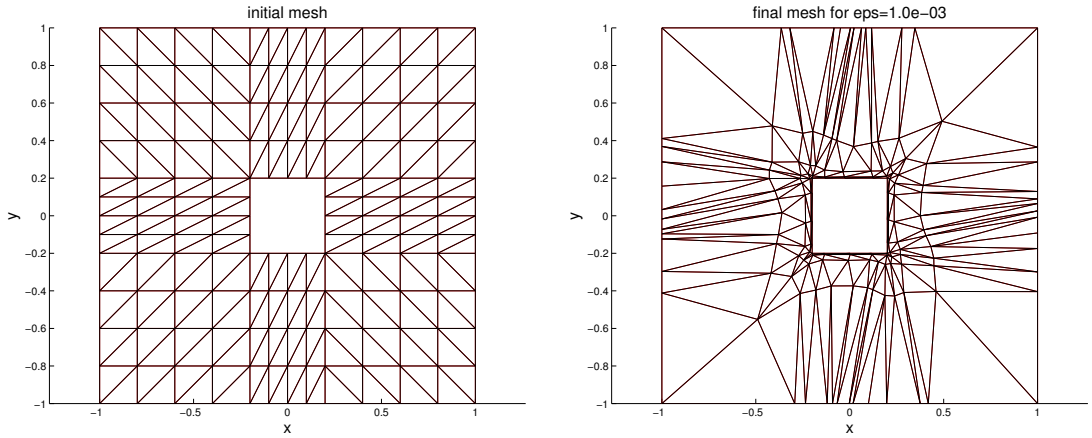


Figure 3.14: Initial and optimised coarse mesh for Example 3.3

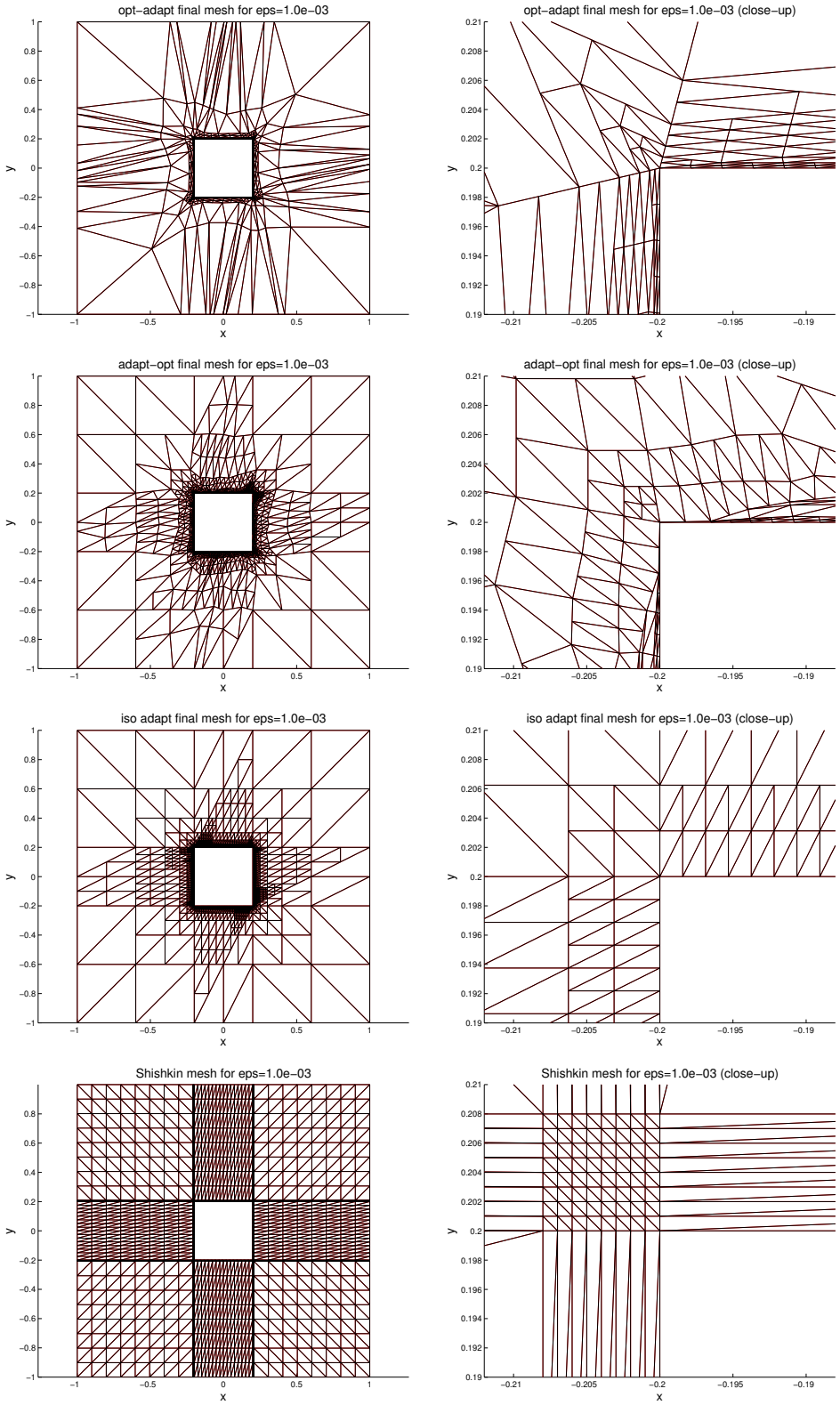


Figure 3.15: Example final meshes for Example 3.3, $\varepsilon = 10^{-3}$

Chapter 4

Conclusions

The application of the discrete adjoint method has been the main topic of this thesis. This technique allows efficient evaluation of the derivative of a function $I(s)$ with respect to parameters s in situations where I depends on s indirectly, via an intermediate variable $\omega(s)$, which is computationally expensive to evaluate. The method has been applied in the context of shape optimisation for fluid dynamics systems and in adaptive mesh design. In both cases the utility of the approach has been demonstrated by numerical experiments.

Both presented applications of this technique require the evaluation of partial derivatives of the residuals of the discretised PDEs with respect to parameters of the discretisation, i.e. the underlying mesh. In the case of the finite volume discretisation considered in this work (Section 2.4), this turned out to be trivial due to the simple structure of the meshes considered, see Section 2.6.3. For the case of finite element discretisations using isoparametric elements of arbitrary degree on unstructured meshes a general approach to evaluating these derivatives has been introduced in Section 2.6.2.1, improving in generality on previously published evaluation approaches (e.g. [43, 49]).

In Section 2.5 the need for efficient solution techniques for the linear systems arising from the PDE discretisation has been emphasised and for the problems considered such techniques have been discussed. In Section 2.6 it was then demonstrated that efficient solution methods for the forward problem can be translated into efficient solution methods for the discrete adjoint equations.

The pure Dirichlet problem of the stationary incompressible Navier-Stokes equations allows for non-uniqueness of the pressure solution. The influence of the regularisation of

the resulting singularity of the Navier-Stokes system in the context of the discrete adjoint method has been analysed in Section 2.6.1. A significant contribution of this work is that it has been demonstrated that the adjoint systems possess the same properties as the original systems, leading to the conclusion that the same techniques used in the treatment of the singularities in the original problem can be used for the adjoint equations.

Two particularly important aspects which have been given consideration in this thesis are the efficient solution of the discrete linearised Navier-Stokes systems and the adaptive mesh design. Conclusions regarding these topics are given in the following two short sections. Finally opportunities for future research, resulting from the exposition in this thesis are summarised in Section 4.3.

4.1 Conclusions regarding efficient solvers for the linear systems

In Section 2.5 two different preconditioning techniques for the linear systems arising in the discretisation of the stationary incompressible Navier-Stokes equations have been considered, the F_p preconditioner and geometric multigrid based on the box-smoother. The performance of both approaches is dependent on the discretisation underlying the linear systems. For the finite element discretisation the F_p preconditioner was found to work efficiently and reliably, without the need to tune any parameters, albeit its performance deteriorates as the Reynolds number is increased. However, for the finite volume discretisation employed in this work, we showed that this preconditioner does not produce mesh size independent conditioning of the linear systems.

The preconditioner using geometric multigrid with the box smoother worked reliably and efficiently in the tests for the finite volume discretisation, while in the finite element case these qualities could not be observed. In those cases where it produced satisfactory results these were found to be very sensitive to parameter choices, i.e. the solver failed for parameter choices in the vicinity of parameter values working satisfactorily.

One step in the application of the F_p preconditioner is to solve linear systems with a discretisation of an advection-diffusion-reaction operator (the so-called F-block), where the reaction terms are only present in the systems arising from Newton linearisation. For these systems multigrid schemes with different stabilisation approaches for the coarse grid discretisation have been considered in a largely experimental way. The outcome can be described as a partial success, with essentially mesh independent behaviour even for the more challenging systems from the Newton linearisation, but the results deteriorating

quickly for increasing Reynolds number.

A notable contribution of this work is the application of the projection approach to the implementation of Dirichlet boundary conditions for multigrid preconditioned non-symmetric systems and for the implementation of the zero-mean-pressure condition, see Section 2.5.2.3. Further, to the knowledge of the author, application of the F_p preconditioner to solve the adjoint problems has first been considered in our work [75] and now in this present thesis. Finally, this present thesis is the first work considering application of the F_p-preconditioning technique to a finite volume technique, unfortunately with a negative result, demonstrating failure of the preconditioning technique to deliver mesh independent condition numbers. With the benefit of hindsight an explanation for this negative result has been put forward.

4.2 Conclusions regarding adaptive mesh design

The application of the discrete adjoint technique to calculate the sensitivity of an *a posteriori* error estimate to the positions of the node points in the underlying finite element mesh is a further significant contribution of this work. This allows the *a posteriori* error estimate to be used not only for isotropic local refinement but also as a means of relocating the nodes in the mesh, in a manner not considered elsewhere. Two alternatives have been considered for how to combine these forms of adaptivity (refinement and movement) both of which make use of nonlinear optimisation techniques including SQP and the trust-region concept. The main conclusion of this part of the work is that it is indeed feasible to both efficiently calculate and make use of the sensitivity of the error estimate within an automatic adaptive algorithm. Furthermore, it has been demonstrated for a number of computational examples of anisotropic reaction-diffusion problems that this approach can deliver improved convergence properties, reaching the asymptotic regime with significantly fewer unknowns than is possible with standard isotropic local refinement alone.

4.3 Opportunities for future research

Regarding efficient solution strategies for the linearised discrete Navier-Stokes systems there remain a number of open questions. A universal solution strategy with optimal computational costs (linear in the number of DOF) which is independent of the Reynolds number, the linearisation and the discretisation remains a high goal. The strong degradation of the performance of the F_p-preconditioner for the Newton linearisation as the Reynolds number is increased, resulting from degradation of the F_p preconditioner itself

and decreased multigrid efficiency for the F-block inner solves, represents one open subproblem. As a possible approach to improve on this, it would be interesting to see if algebraic multigrid approaches [106] or $\mathscr{H}$-matrix approximate LU decompositions [15] constitute more efficient preconditioners for the F-block systems in the F_p preconditioner, as remarked in Section 2.5.2.2. More efficient geometric multigrid techniques may also be possible. Further, as the failure of the F_p-preconditioner in the finite volume discretisation appears to result from the first order stabilisation terms, it would be interesting to apply this preconditioner to a second order accurate finite volume discretisation.

For the second candidate approach, using geometric multigrid directly for the whole linearised Navier-Stokes system, the construction of robust, efficient, discretisation independent smoothers remains an open problem. In particular, a good smoother for the Taylor-Hood finite element discretisation is still unknown to the author. The discussion in [50] indicates that the box-smoother performs best for lowest order discretisations. Thus, one possible approach might be to construct such a low order discretisation for the sole purpose of the smoother, i.e. using a lowest order smoother for the higher order discretisations on all levels of the mesh hierarchy.

In the context of the automatic anisotropic mesh adaption approach introduced in this work, which has been demonstrated to be feasible for a relatively small class of test problems, it is now necessary to extend the work to larger classes of PDE and a wider range of *a posteriori* error estimates. It has already been demonstrated here that a key factor is to ensure the reliability of the error estimate so as to ensure that, as the mesh becomes more anisotropic, this does not deteriorate and lead to optimisation of a meaningless quantity. Other problem classes that may be considered include convection-diffusion problems and the Navier-Stokes equations, whose solutions may involve very steep layers. In terms of error estimation techniques, the local problem estimator (e.g. [98, Section 1.3], [3, Section 3]) appears to be a good alternative. The use of more complex geometries or richer finite element spaces may be considered as well. In principle however all of the techniques developed here can be extended to these situations.

Application of the most efficient adaptive mesh discretisations is of course also of interest in the context of shape optimisation in order to keep the costs for individual performance functional evaluations as low as possible. Thus, the refinement approach from Chapter 3 could in theory also be applied for the problems in Chapter 2. Error estimation for the incompressible Navier-Stokes equations using the DWR-approach has already been developed, see for example [11, Chapter 11], so one step for the application of the presented approach to this situation is already done. However, if adaptive meshes are used, the consistency of the shape gradient (evaluated by means of the discrete adjoint)

and the discrete performance functional may be lost, see the discussion in Section 2.2.5. Thus, a further topic for future research may be to see if the approaches can be combined in such a way that this consistency is retained.

Redistribution of the mesh alone as a means of adaptive refinement has its limitations, as discussed in Section 3.3. In addition to the combination with adaptive locally uniform refinement (Section 3.3.6), changing the mesh connectivity (e.g. edge swapping) may also provide a means to improve the quality of the solution approximation, see [56, 99] for example. Further, if connectivity changes are considered, the quality of the deformed meshes may also be improved by the removal of nodes (coarsening) in regions of small errors. However, both these extensions have not been considered in this thesis and remain as topics for future research.

Appendix A

Convergence of the discrete optimal shape solution

This appendix is concerned with the following question: under which conditions can it be guaranteed that the solutions of the discrete optimal shape problems converge to solutions of the continuous optimal shape problem? In the following a key result from [44, Section 2.4] is reproduced. The exposition is meant to provide an approach to obtaining an answer to this question rather than giving a final answer itself. Thus not all details are present, but an overview of the approach is given. The interested reader is referred to [44] for a detailed discussion of the topic.

From the outset it is clear that solutions to the discrete optimisation problems may approximate different continuous solutions, if the continuous problem possesses more than one locally optimal solution. Indeed, an optimisation algorithm for the discrete problems may easily converge to a different local optimum after refinement of the discretisation. This complicates the definition of meaningful convergence rates, and instead the theory reproduced here only states that for a sequence of increasingly fine discrete solutions there exists a subsequence converging to a solution of the continuous problem.

Let us introduce some notation required for this section and let us give a clear statement of the discrete optimisation problem. Let a family of shape parameterisations with parameter $\varkappa > 0$ be defined, such that the number of shape parameters $\dim(\mathscr{F}_\varkappa)$ is uniquely defined by $\varkappa$. Let $\mathscr{D}_\varkappa$ denote the set of admissible discrete shapes, where $\mathscr{D}_\varkappa \subset \widetilde{\mathscr{D}}$ for some superset $\widetilde{\mathscr{D}} \supset \mathscr{D}$, for all $\varkappa > 0$, but it is not necessary that $\mathscr{D}_\varkappa \subset \mathscr{D}$. Let every $\mathscr{F}_\varkappa \in \mathscr{D}_\varkappa$ uniquely define a computational domain $\Omega_{\varkappa h}$, where the domain discretisation parameter $h > 0$ is a monotone function of $\varkappa$ such that

$$h \to +0 \quad \Longleftrightarrow \quad \varkappa \to +0. \tag{A.0.1}$$

A distinction between discrete shape and computational domain is made to allow for, for example, B-spline definitions of the discrete shape while the computational domain may be polygonal with a triangulation of discretisation length-scale h. Let $\mathbb{V}(\Omega)$ denote the function space associated with the weak formulation of the PDE (2.1.1b), (2.1.1c) on domain Ω and let $\mathbb{V}_h(\Omega_{\varkappa h})$ denote a finite dimensional function space (finite element space) such that $\mathbb{V}_h(\Omega_{\varkappa h}) \subset \mathbb{V}(\Omega_{\varkappa h})$. Then let $u_h := u_h(\mathscr{F}_\varkappa) := u_h(\Omega_{\varkappa h}) \in \mathbb{V}_h(\Omega_{\varkappa h})$ denote the uniquely defined discrete approximation to the unique solution of the PDE (2.1.1b), (2.1.1c) on the computational domain $\Omega_{\varkappa h}$ corresponding to discrete shape $\mathscr{F}_\varkappa$. Thus, for a given discrete shape $\mathscr{F}_\varkappa \in \mathscr{D}_\varkappa$ the following chain of mappings defines the approximated performance,

$$\mathscr{F}_\varkappa \mapsto \Omega_{\varkappa h} \mapsto u_h \mapsto I(u_h(\mathscr{F}_\varkappa), \mathscr{F}_\varkappa). \tag{A.0.2}$$

This results in the discrete approximation to Problem (2.1.1) as

$$\min_{\mathscr{F}_\varkappa \in \mathscr{D}_\varkappa} I(u_h(\mathscr{F}_\varkappa), \mathscr{F}_\varkappa), \tag{A.0.3}$$

which in turn defines (locally) optimal discrete solutions $\mathscr{F}^*_\varkappa$, as well as the corresponding $\Omega^*_{\varkappa h}$ and $u^*_h := u_h(\mathscr{F}^*_\varkappa)$.

In order to analyse convergence properties of (A.0.3), the following assumptions are introduced.

Assumption 1 (Density of discrete shapes)**.**
For any shape $\mathscr{F} \in \mathscr{D}$ and corresponding domain Ω there exists a sequence of discrete shapes $\{\mathscr{F}_\varkappa\}$ and corresponding computational domains $\{\Omega_{\varkappa h}\}$, such that

$$\Omega_{\varkappa h} \longrightarrow \Omega \quad \text{as} \quad \varkappa, h \to +0.$$

Assumption 2 (Compactness)**.**
For any sequence of discrete shapes $\{\mathscr{F}_\varkappa\}$, $\mathscr{F}_\varkappa \in \mathscr{D}_\varkappa$, with corresponding computational domains $\{\Omega_{\varkappa h}\}$, there exist subsequences $\{\mathscr{F}_{\varkappa_j}\}$ with corresponding $\{\Omega_{\varkappa_j h_j}\}$ and an element $\mathscr{F} \in \mathscr{D}$ with corresponding $\Omega := \Omega(\mathscr{F})$, such that

$$\Omega_{\varkappa_j h_j} \longrightarrow \Omega$$
$$\text{and} \qquad u_{h_j}(\Omega_{\varkappa_j h_j}) \longrightarrow u(\Omega) \qquad \text{as} \quad j \to \infty.$$

The assumption above contains a convergence of functions from different function spaces, i.e. $\mathbb{V}(\widehat{\Omega})$ for different domains $\widehat{\Omega}$. The authors in [44] define such a convergence

by means of embedding $\mathbb{V}(\widehat{\Omega})$ into a function space $\mathbb{V}(\widetilde{\Omega})$, where the domain $\widetilde{\Omega}$ is chosen such that it is a superset to all possible domains,

$$\Omega(\mathscr{F}) \subset \widetilde{\Omega} \quad \forall \mathscr{F} \in \widetilde{\mathscr{D}}.$$

Convergence of functions on different domains is then defined as convergence of the extension of the functions to $\mathbb{V}(\widetilde{\Omega})$, which can be uniquely defined.

Assumption 3 (Continuity of I)**.**
Let

$$\left.\begin{array}{rcll} \Omega_{\varkappa h}(\mathscr{F}_\varkappa) & \longrightarrow & \Omega(\mathscr{F}), & \mathscr{F}_\varkappa \in \mathscr{D}_\varkappa, \quad \mathscr{F} \in \mathscr{D} \\ u_h(\Omega_{\varkappa h}(\mathscr{F}_\varkappa)) & \longrightarrow & u(\Omega(\mathscr{F})) & \end{array}\right\}$$

imply that

$$I(u_h(\mathscr{F}_\varkappa), \mathscr{F}_\varkappa) \longrightarrow I(u(\Omega), \mathscr{F}) \qquad \text{for} \quad \varkappa, h \to +0.$$

This allows us to present the following result [44, Theorem 2.11]:

Theorem 1. *Let assumptions 1, 2 and 3 be satisfied. Then for any sequence* $\{(\mathscr{F}^*_\varkappa, u_h(\mathscr{F}^*_\varkappa))\}$ *of optimal pairs of* (A.0.3)*,* $\varkappa \to +0$*, there exists its subsequence* $\{(\mathscr{F}^*_{\varkappa_j}, u_h(\mathscr{F}^*_{\varkappa_j}))\}$ *such that*

$$\left\{\begin{array}{rcl} \Omega_{\varkappa_j h_j}(\mathscr{F}^*_{\varkappa_j}) & \longrightarrow & \Omega(\mathscr{F}^*), \\ u_{h_j}(\Omega_{\varkappa_j h_j}(\mathscr{F}^*_{\varkappa_j})) & \longrightarrow & u(\Omega(\mathscr{F}^*)) \end{array}\right. \tag{A.0.4}$$

for $j \to \infty$*. In addition,* $(\mathscr{F}^*, u(\mathscr{F}^*))$ *is an optimal pair of* (2.1.1)*. Any accumulation point of* $\{(\mathscr{F}^*_\varkappa, u_h(\mathscr{F}^*_\varkappa))\}$ *in the sense of* (A.0.4) *possesses this property.*

Proof. See [44, Section 2.4]. □

Discussion. In order to apply this theory to the shape optimisation problem for the stationary incompressible Navier-Stokes (INS) equations it has to be verified that the assumptions made are indeed fulfilled for this case. Before we comment on assumptions 1, 2 and 3, the question of existence and uniqueness of the solution to the INS equations should be considered. Sufficient criteria for existence and uniqueness of solutions are presented, for example, in [93, Section 10] for the continuous problem and [40] may be consulted for the discretised (FEM) problems. Even though these conditions are not explicitly verified here, they give a good indication that, for the relatively low Reynolds number regimes considered in this work, unique solutions should exist.

Assumption 1, density of the discrete shapes, is clearly fulfilled if the shape parameterisations are chosen appropriately. For sufficiently smooth variable boundary sections,

spline approximations allow arbitrarily close approximation, as does the subsequent interpolation by the piecewise linears which results from triangulation of the domain. The continuity of I, Assumption 3, is a natural requirement on the performance functional I. If it was not fulfilled, optimal solutions would be practically useless, as even the slightest perturbation of the parameters, e.g. due to inaccuracies of a manufacturing process, could result in dramatically different performance. All the performance criteria used in this thesis fulfil this property.

Verifying Assumption 2, compactness, is rather technical. In [44, Section 2.5] this is demonstrated for the Poisson equation with various boundary conditions and also for a stream-function formulation of the Stokes equations.

It is worth noting that the above theory requires that the PDE discretisation parameter h goes to zero as the shape discretisation parameter $\varkappa$ goes to zero. An obvious requirement is that the triangulations of the domains $\Omega(\mathscr{F}_\varkappa)$ are sufficiently fine that changes in the shape parameters $\mathscr{F}_\varkappa$ do actually affect the computational domains $\Omega_{\varkappa h}$. Even more, the meshes should be sufficiently fine such that the effected changes in the continuous solution $u(\Omega(\mathscr{F}_\varkappa))$ are reasonably well resolved by the discrete solutions $u_h(\Omega_{\varkappa h})$.

Of course taking the limit $\varkappa \to +0$ is mainly a theoretical approach and in practical situations bounds on computational resources mean that only a certain finest $\varkappa_0 > 0$ can be achieved at reasonable cost. However, the result indicates that good approximations to solutions may be obtained if a sufficiently small discretisation parameter $\varkappa$ can be used.

Appendix B

Equivalence of the linear Navier-Stokes systems

In this appendix we demonstrate that the system that arises when the last row of (2.5.2) is replaced by the zero mean pressure condition (ZMPC) is equivalent to the augmented system that is obtained when the ZMPC is applied using a Lagrange multiplier approach (2.5.21). This is expressed as the following proposition where, for clarity, we assume $\sum_{i=1}^{m} w_i = 1$ (which is always achievable with appropriate scaling) and that Dirichlet boundary conditions are incorporated explicitly (see below).

Proposition 2. *Provided the Dirichlet boundary conditions are consistent with mass conservation, i.e.* $\int_{\partial\Omega} n^T \boldsymbol{u} \, \mathrm{d}\Gamma = 0$, *the linear system*

$$\begin{bmatrix} F_{1,1int} & F_{1,1bc} & F_{1,2int} & F_{1,2bc} & B_{1int}^T \\ 0 & I_{1bc} & 0 & 0 & 0 \\ F_{2,1int} & F_{2,1bc} & F_{2,2int} & F_{2,2bc} & B_{2int}^T \\ 0 & 0 & 0 & I_{2bc} & 0 \\ \widetilde{B}_{1int} & \widetilde{B}_{1bc} & \widetilde{B}_{2int} & \widetilde{B}_{2bc} & 0 \\ 0 & 0 & 0 & 0 & w^T \end{bmatrix} \begin{bmatrix} u_{1int} \\ u_{1bc} \\ u_{2int} \\ u_{2bc} \\ p \end{bmatrix} = \begin{bmatrix} f_{1int} \\ f_{1bc} \\ f_{2int} \\ f_{2bc} \\ 0 \\ 0 \end{bmatrix} \tag{B.0.1}$$

is equivalent to the system

$$\begin{bmatrix} F_{1,1int} & F_{1,1bc} & F_{1,2int} & F_{1,2bc} & B_{1int}^T & 0 \\ 0 & I_{1bc} & 0 & 0 & 0 & 0 \\ F_{2,1int} & F_{2,1bc} & F_{2,2int} & F_{2,2bc} & B_{2int}^T & 0 \\ 0 & 0 & 0 & I_{2bc} & 0 & 0 \\ B_{1int} & B_{1bc} & B_{2int} & B_{2bc} & 0 & w \\ 0 & 0 & 0 & 0 & w^T & 0 \end{bmatrix} \begin{bmatrix} \widetilde{u}_{1int} \\ \widetilde{u}_{1bc} \\ \widetilde{u}_{2int} \\ \widetilde{u}_{2bc} \\ \widetilde{p} \\ \lambda \end{bmatrix} = \begin{bmatrix} f_{1int} \\ f_{1bc} \\ f_{2int} \\ f_{2bc} \\ 0 \\ 0 \end{bmatrix}, \tag{B.0.2}$$

where the subscript $()_{*int}$ denotes a block corresponding to interior nodes, $()_{*bc}$ a block corresponding to Dirichlet nodes, u_{1*} the expansion coefficient of the first velocity component with respect to the FE ansatz functions and u_{2*} the second component. The equations are ordered analogously to the degrees of freedom. The vectors f_{1bc} and f_{2bc} store the Dirichlet values for the first and second components of $\boldsymbol{u}$ on the boundary.

Proof. Suppose (B.0.1) holds. Define

$$\widetilde{u}_{1int} = u_{1int}; \quad \widetilde{u}_{1bc} = u_{1bc}; \quad \widetilde{u}_{2int} = u_{2int}; \quad \widetilde{u}_{2bc} = u_{2bc}; \quad \widetilde{p} = p; \quad \lambda = 0. \tag{B.0.3}$$

Then the first four and the last block rows of (B.0.2) are clearly satisfied. Furthermore, the first $N_p - 1$ equations of the fifth block row of (B.0.2) are also immediately satisfied, where N_p denotes the number of (linear) pressure ansatz functions. Hence we need only consider the N_p-th equation in this block. The left-hand side of this is

$$\int_\Omega q_{N_p} (\nabla \cdot \mathbf{u}_h) \, \mathrm{d}\Omega.$$

However, we know from the other equations of this block that

$$\int_\Omega q_i (\nabla \cdot \mathbf{u}_h) \, \mathrm{d}\Omega = 0 \qquad \forall i = 1, \ldots, N_p - 1.$$

Hence

$$\begin{aligned}
\int_\Omega q_{N_p}(\nabla\cdot\mathbf{u}_h)\,\mathrm{d}\Omega &= \sum_{i=1}^{N_p}\int_\Omega q_i(\nabla\cdot\mathbf{u}_h)\,\mathrm{d}\Omega &&= \int_\Omega\left(\sum_{i=1}^{N_p} q_i\right)(\nabla\cdot\mathbf{u}_h)\,\mathrm{d}\Omega\\
&= \int_\Omega 1(\nabla\cdot\mathbf{u}_h)\,\mathrm{d}\Omega &&= \sum_{T\in\mathscr{T}}\int_T \nabla\cdot\mathbf{u}_h\,\mathrm{d}\Omega\\
&= \sum_{T\in\mathscr{T}}\int_{\partial T} n^T\cdot\mathbf{u}_h\,\mathrm{d}\Omega &&= \int_{\partial\Omega} n^T\cdot\mathbf{u}_h\,\mathrm{d}\Omega\\
&= 0
\end{aligned}$$

(due to the Dirichlet boundary conditions). All the equations in (B.0.2) are therefore satisfied by (B.0.3).

Next suppose that (B.0.2) holds. Then the i-th equation in the fifth block row of (B.0.2) is

$$\int_\Omega q_i(\nabla\cdot\mathbf{u}_h)\,\mathrm{d}\Omega + \lambda w_i = 0 \qquad \forall i = 1,\ldots,N_p.$$

Summing these equations gives

$$\int_\Omega \nabla\cdot\mathbf{u}_h\,\mathrm{d}\Omega + \lambda = 0,$$

but, using the same argument as above, the Dirichlet boundary conditions imply that $\lambda = 0$. Given that $\lambda = 0$ it follows trivially that

$$u_{1int} = \widetilde{u}_{1int};\quad u_{1bc} = \widetilde{u}_{1bc};\quad u_{2int} = \widetilde{u}_{2int};\quad u_{2bc} = \widetilde{u}_{2bc};\quad p = \widetilde{p}$$

is a solution of (B.0.1). Hence the two systems are equivalent. □

Bibliography

[1] D.J. Acheson. *Elementary Fluid Dynamics*. Clarendon Press, 1990.

[2] M. Ainsworth and D. Kay. The approximation theory for the p-version finite element method and application to nonlinear elliptic PDEs. *Numerische Mathematik*, 82(3):351–388, 1999.

[3] M. Ainsworth and J.T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. Wiley, 2000.

[4] M. Ainsworth and W. Senior. An adaptive refinement strategy for h-p finite element computations. *Applied Numerical Mathematics*, 26:165–178, 1997.

[5] I. Ammara and C. Masson. Development of a fully coupled control-volume finite element method for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 44:621–644, 2004.

[6] W.K. Anderson, R.D. Rausch, and D.L. Bonhaus. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. *Journal of Computational Physics*, 128(2):391–408, 1996.

[7] T. Apel, S. Grosman, P.K. Jimack, and A. Meyer. A new methodology for anisotropic mesh refinement based upon error gradients. *Applied Numerical Mathematics*, 50:329–341, 2004.

[8] T. Apel and G. Lube. Anisotropic mesh refinement for a singularly perturbed reaction diffusion model problem. *Applied Numerical Mathematics*, 26:415–433, 1998.

[9] I. Babuska, B.A. Szabo, and I.N. Katz. The p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 18:515–545, 1981.

[10] M.J. Baines. Grid adaptation via node movement. *Applied Numerical Mathematics*, 26:77–96, 1998.

[11] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser Verlag, 2003.

[12] R.E. Bank and R.K. Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, 1997.

[13] R.E. Bank and A. Weiser. Some a posteriori error estimates for elliptic partial differential equations. *Mathematics of Computation*, 44:283–301, 1985.

[14] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *Impact of Computing in Science and Engineering*, 3:181–191, 1991.

[15] M. Bebendorf. On the numerical solution of convection-dominated problems using hierarchical matrices. Technical Report Preprint 78/2005, Max-Planck-Institut MiS, Leipzig, 2005. Submitted to Numerische Mathematik. Available at http://www.math.uni-leipzig.de/MI/bebendorf/Downloads/Artikel/mb_convdom.pdf.

[16] L.T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, editors. *Large-Scale PDE-Constrained Optimization*. Springer, 2003.

[17] H. Blank, M. Rudgyard, and A. Wathen. Stabilised finite element methods for steady incompressible flow. *Computer Methods in Applied Mechanics and Engineering*, 174:91–105, 1999.

[18] D. Braess. *Finite Elemente – Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer Lehrbuch, Berlin, 1997.

[19] D. Braess. *Finite Elemente – Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer Lehrbuch, Berlin, 3rd edition, 2003.

[20] J.H. Bramble, J.E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22, July 1990.

[21] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Classics in Applied Mathematics. SIAM, Philadelphia, 2002.

[22] A.L. Codd, T.A. Manteuffel, and S.F. McCormick. Multilevel first-order system least squares for nonlinear elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 46(6):2197–2209, 2003.

[23] R. Codina. On stabilized finite element methods for linear systems of convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 188:61–82, 2000.

[24] A.R. Conn, K. Scheinberg, and P.L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79:397–414, 1997.

[25] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester, 2001.

[26] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li, and J.W.H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999. Software available at http://crd.lbl.gov/~xiaoye/SuperLU.

[27] H.C. Elman. Preconditioning for the steady-state Navier-Stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20:1299–1316, 1999.

[28] H.C. Elman. Preconditioning strategies for models of incompressible flow. Technical Report UMCP-CSD:CS-TR-4543, UMIACS, University of Maryland, 2003. Available at http://www.cs.umd.edu/Library/TRs/CS-TR-4543/CS-TR-4543.ps.

[29] H.C. Elman, D. Loghin, and A.J. Wathen. Preconditioning techniques for Newton's method for the incompressible Navier-Stokes equations. *BIT*, 43(5):961–974, 2003.

[30] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1988.

[31] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, 1996.

[32] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation in computational fluid dynamics: Application to the advection-diffusion-reaction and the Stokes problems. *Applied Numerical Mathematics*, 51(4):511–533, December 2004.

[33] L.P. Franka and F. Valentin. On an improved unusual stabilized finite element method for the advective-reactive-diffusive equation. *Computer Methods in Applied Mechanics and Engineering*, 190:1785–1800, 2000.

[34] T. Gelhard, G. Lube, M.A. Olshanskii, and J.-H. Starcke. Stabilized finite element schemes with LBB-stable elements for incompressible flows. *Journal of Computational and Applied Mathematics*, 177:243–267, 2005.

[35] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[36] M.B. Giles and N.A. Pierce. An introduction to the adjoint approach to design. *Flow Turbulence and Combustion*, 65(3–4):393–415, 2000.

[37] M.B. Giles, N.A. Pierce, and E. Süli. Progress in adjoint error correction for integral functionals. *Computing and Visualisation in Science*, 6:113–121, 2004.

[38] P.M. Gresho, R.L. Sani, and M.S. Engelman. *Incompressible Flow and the Finite Element Method: Advection-Diffusion and Isothermal Laminar Flow*. Wiley, 1998.

[39] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. SIAM, 2000.

[40] M.D. Gunzburger. *Finite Element Methods for Viscous Incompressible Flows*. Academic Press, 1989.

[41] M.D. Gunzburger. *Perspectives in Flow Control and Optimization*. SIAM, 2003.

[42] I. Gustafsson. Stability and rate of convergence of modified incomplete Cholesky factorization methods. Technical report, Department of Computer Science, Chalmers University of Technology, Göteborg, Sweden, 1979.

[43] J. Hämäläinen, R.A.E. Mäkinen, and P. Tarvainen. Optimal design of paper machine headboxes. *International Journal for Numerical Methods in Fluids*, 34:685–700, 2000.

[44] J. Haslinger and R.A.E. Mäkinen. *Introduction to Shape Optimization*. SIAM, 2003.

[45] G. Hauke. A simple subgrid scale stabilized method for the advection-diffusion-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 191:2925–2947, 2002.

[46] M. Hinze, A. Walther, and J. Sternberg. An optimal memory-reduced procedure for calculating adjoints of the instationary Navier-Stokes equations. *Optimal Control Applications and Methods*, 27(1):19–40, 2006.

[47] T.J.R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127:387–401, 1995.

[48] P.K. Jimack. A best approximation property of the moving finite element method. *SIAM Journal on Numerical Analysis*, 33(6):2286–2302, December 1996.

[49] P.K. Jimack and A.J. Wathen. Temporal derivatives in the finite-element method on continuously deforming grids. *SIAM Journal on Numerical Analysis*, 28(4):990–1003, August 1991.

[50] V. John and G. Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids*, 37:885–903, 2001.

[51] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, 1987.

[52] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal Scientific Computing*, 24(1):237–256, 2002.

[53] G. Kunert. Toward anisotropic mesh construction and error estimation in the finite element method. *Numerical Methods for Partial Differential Equations*, 18(5):625–648, 2002.

[54] G. Kunert. A posteriori H^1 error estimation for a singularly perturbed reaction diffusion problem on anisotropic meshes. *IMA Journal of Numerical Analysis*, 25(2):408–428, 2005.

[55] G. Kunert and R. Verfürth. Edge residuals dominate a posteriori error estimates for linear finite element methods on anisotropic triangular and tetrahedral meshes. *Numerische Mathematik*, 86:283–303, 2000.

[56] J. Lang, W. Cao, W. Huang, and R.D. Russel. A two-dimensional moving finite element method with local refinement based on a posteriori error estimates. *Applied Numerical Mathematics*, 46:75–94, 2003.

[57] A. Meyer. Projection techniques embedded in the PCGM for handling hanging nodes and boundary restrictions. In B.H.V. Topping and Z. Bittnar, editors, *Engineering Computational Technology*, pages 147–165. Saxe-Cobourg Publications, 2002.

[58] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[59] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

[60] Netlib. *Linear Algebra PACKage (LAPACK)*. Software available at http://www.netlib.org/lapack/.

[61] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[62] E. Oñate. Derivation of stabilized equations for numerical solution of advective-diffusive transport and fluid flow problems. *Computer Methods in Applied Mechanics and Engineering*, 151:233–265, 1998.

[63] E. Oñate. Multiscale computational analysis in mechanics using finite calculus: an introduction. *Computer Methods in Applied Mechanics and Engineering*, 192:3043–3059, 2003.

[64] S.V. Patankar and D.B. Spalding. Calculation procedure for heat, mass and momentum-transfer in 3-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[65] A.R. Paterson. *A First Course in Fluid Dynamics*. Cambridge University Press, 1983.

[66] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer-Verlag, 1984.

[67] M.J.D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92:555–582, 2002.

[68] A. Ramage. A multigrid preconditioner for stabilised discretisations of advection-diffusion problems. *Journal of Computational and Applied Mathematics*, 110:187–203, 1999.

[69] W.C. Rheinboldt and C.K. Mesztenyi. On a data structure for adaptive finite element mesh refinements. *ACM Transactions on Mathematical Software*, 6(2):166–187, 1980.

[70] M.-C. Rivara. A grid generator based on 4-triangles conforming mesh-refinement algorithms. *International Journal for Numerical Methods in Engineering*, 24:1343–1354, 1987.

[71] H.G. Roos. Layer-adapted grids for singular perturbation problems. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 78(5):291–309, 1998.

[72] S. Beuchler and A. Meyer. SPC-PM3AdH v1.0 - Programmer's Manual. Technical Report Preprint SFB393/01-08, TU Chemnitz, Chemnitz, 2001. Available at http://www.tu-chemnitz.de/sfb393/.

[73] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM (Society for Industrial and Applied Mathematics), 2003.

[74] A. Schmidt and K.G. Siebert. Albert - software for scientific computations and applications. *Acta Mathematica Universitatis Comenianae*, 70:105–122, 2001.

[75] R. Schneider and P.K. Jimack. Efficient preconditioning of the discrete adjoint equations for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 47:1277–1283, 2005.

[76] R. Schneider and P.K. Jimack. Toward anisotropic mesh adaption based upon sensitivity of a posteriori estimates. School of Computing Research Report Series 2005.03, University of Leeds, 2005. Available at http://www.comp.leeds.ac.uk/research/pubs/reports/2005/2005_03.pdf.

[77] J. Schöberl. NETGEN - an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1:41–52, 1997. Software available at http://www.hpfem.jku.at/netgen/.

[78] C. Schwab and M. Suri. The p and hp versions of the finite element method for problems with boundary layers. *Mathematics of Computation*, 65:1402–1429, 1996.

[79] J. Schöberl and W. Zulehner. On Schwarz-type smoothers for saddle point problems. *Numerische Mathematik*, 95:377–399, 2003.

[80] J.R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In M.C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996. Article and software available at http://www.cs.cmu.edu/~quake/triangle.html.

[81] K. Siebert. An a posteriori error estimator for anisotropic refinement. *Numerische Mathematik*, 73:373–398, 1996.

[82] V. Simoncini and D.B. Szyld. On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Review*, 47(2):247–272, 2005.

[83] W. Speares and M. Berzins. A 3-d unstructured mesh adaptation algorithm for time-dependent shock dominated problems. *International Journal for Numerical Methods in Fluids*, 25:81–104, 1997.

[84] P. Spellucci. *DONLP2 users guide*. Technical University at Darmstadt, Department of Mathematics, 64289 Darmstadt, Germany. Software available at http://www.mathematik.tu-darmstadt.de/ags/ag8/Mitglieder/spellucci_en.html.

[85] P. Spellucci. *Numerische Verfahren der nichtlinearen Optimierung*. Birkhäuser, Basel, 1993.

[86] L.G. Stanley and D.L. Stewart. *Design Sensitivity Analysis*. SIAM, 2002.

[87] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Journal of Applied Mechanics*, 70:58–63, 2003.

[88] G. Strang and G.J. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, 1973.

[89] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole, 1989.

[90] M. Stynes and E. O'Riordan. A uniformly convergent Galerkin method on a Shishkin mesh for a convection-diffusion problem. *Journal of Mathematical Analysis and Applications*, 214:36–54, 1997.

[91] T. Apel, F. Milde, and U. Reichel. SPC-PM Po 3D v4.0 - Programmer's Manual II. Technical Report Preprint SFB393/99-37, TU Chemnitz, Chemnitz, 1999. Available at http://www.tu-chemnitz.de/sfb393/.

[92] A. Tabarraei and N. Sukumar. Adaptive computations on conforming quadtree meshes. *Finite Elements in Analysis and Design*, 41:686–702, 2005.

[93] R. Temam. *Navier-Stokes Equations and Nonlinear Functional Analysis*. SIAM, second edition edition, 1995.

[94] L. Tobiska and R. Verfürth. Analysis of a streamline diffusion finite element method for the Stokes and Navier-Stokes equations. *SIAM Journal on Numerical Analysis*, 33(1):107–127, 1996.

[95] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[96] S. Turek. *Efficient Solvers for Incompressible Flow Problems, An Algorithmic and Computational Approach*. Lecture Notes in Computational Science and Engineering. Springer, 1999.

[97] S.P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65:138–158, 1986.

[98] R. Verfürth. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley, 1996.

[99] M. Walkley, P.K. Jimack, and M. Berzins. Anisotropic adaptivity for finite element solutions of 3-d convection-dominated problems. *International Journal for Numerical Methods in Fluids*, 40:551–559, 2002.

[100] W. Wang. Special bilinear quadrilateral elements for locally refined finite element grids. *SIAM Journal on Scientific Computing*, 22(6):2029–2050, 2001.

[101] A. Wathen, D. Loghin, D. Kay, H.C. Elman, and D. Silvester. A preconditioner for the 3D Oseen equations. Technical Report NA-02/04, Oxford University Computing Laboratory, February 2002. Available at http://web.comlab.ox.ac.uk/oucl/publications/natr/na-02-04.html.

[102] A. Wathen and D. Sylvester. Fast iterative solution of stabilised Stokes systems, part I: Using simple diagonal preconditioners. *SIAM Journal on Numerical Analysis*, 30(3):630 – 649, June 1993.

[103] A. Wathen and D. Sylvester. Fast iterative solution of stabilised Stokes systems, part II: Using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352 – 1367, October 1994.

[104] A.J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA Journal of Numerical Analysis*, 7:449–457, 1987.

[105] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer Series in Computational Mathematics. Springer, 2001.

[106] C.T. Wu and H.C. Elman. Analysis and comparison of geometric and algebraic multigrid for convection-diffusion equations. Technical report, University of Maryland, 2004. Available at http://www.cs.umd.edu/~elman/papers/mg-amg-paper.pdf.

[107] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24:337–357, 1987.

Index