Advancements in finite element methods for Newtonian and non-Newtonian flows

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Sciences

---

by
Keith J. Galvin
August 2013

Accepted by:
Dr. Hyesuk Lee, Committee Chair
Dr. Leo Rebholz, Co-Chair
Dr. Chris Cox
Dr. Vincent Ervin

Abstract

This dissertation studies two important problems in the mathematics of computational fluid dynamics. The first problem concerns the accurate and efficient simulation of incompressible, viscous Newtonian flows, described by the Navier-Stokes equations. A direct numerical simulation of these types of flows is, in most cases, not computationally feasible. Hence, the first half of this work studies two separate types of models designed to more accurately and efficient simulate these flows. The second half focuses on the defective boundary problem for non-Newtonian flows. Non-Newtonian flows are generally governed by more complex modeling equations, and the lack of standard Dirichlet or Neumann boundary conditions further complicates these problems. We present two different numerical methods to solve these defective boundary problems for non-Newtonian flows, with application to both generalized-Newtonian and viscoelastic flow models.

Chapter 3 studies a finite element method for the 3D Navier-Stokes equations in velocity-vorticity-helicity formulation, which solves directly for velocity, vorticity, Bernoulli pressure and helical density. The algorithm presented strongly enforces solenoidal constraints on both the velocity (to enforce the physical law for conservation of mass) and vorticity (to enforce the mathematical law that div(curl)= 0). We prove unconditional stability of the velocity, and with the use of a (consistent) penalty term on the difference between the computed vorticity and curl of the computed velocity, we are also able to prove unconditional stability of the vorticity in a weaker norm. Numerical experiments are given that confirm expected convergence rates, and test the method on a benchmark problem.

Chapter 4 focuses on one main issue from the method presented in Chapter 3, which is the question of appropriate (and practical) vorticity boundary conditions. A new, natural vorticity boundary condition is derived directly from the Navier-Stokes equations. We propose a numerical scheme implementing this new boundary condition to evaluate its effectiveness in a numerical

experiment.

Chapter 5 derives a new, reduced order, multiscale deconvolution model. Multiscale deconvolution models are a type of large eddy simulation models, which filter out small energy scales and model their effect on the large scales (which significantly reduces the amount of degrees of freedom necessary for simulations). We present both an efficient and stable numerical method to approximate our new reduced order model, and evaluate its effectiveness on two 3d benchmark flow problems.

In Chapter 6 a numerical method for a generalized-Newtonian fluid with flow rate boundary conditions is considered. The defective boundary condition problem is formulated as a constrained optimal control problem, where a flow balance is forced on the inflow and outflow boundaries using a Neumann control. The control problem is analyzed for an existence result and the Lagrange multiplier rule. A decoupling solution algorithm is presented and numerical experiments are provided to validate robustness of the algorithm.

Finally, this work concludes with Chapter 7, which studies two numerical algorithms for viscoelastic fluid flows with defective boundary conditions, where only flow rates or mean pressures are prescribed on parts of the boundary. As in Chapter 6, the defective boundary condition problem is formulated as a minimization problem, where we seek boundary conditions of the flow equations which yield an optimal functional value. Two different approaches are considered in developing computational algorithms for the constrained optimization problem, and results of numerical experiments are presented to compare performance of the algorithms.

Table of Contents

Title Page . . . . . **i**

Abstract . . . . . **ii**

List of Tables . . . . . **vi**

List of Figures . . . . . **vii**

1 Introduction **1**

2 Preliminaries **10**

3 A Numerical Study for a Velocity-Vorticity-Helicity formulation of the 3D Time-Dependent NSE **16**
3.1 Discrete VVH Formulation 17
3.2 Numerical Results 25

4 Natural vorticity boundary conditions for coupled vorticity equations **30**
4.1 Derivation 30
4.2 Numerical Results 32

5 A New Reduced Order Multiscale Deconvolution Model **35**
5.1 Derivation 35
5.2 The Discrete Setting 37
5.3 Error Analysis 43
5.4 Numerical Results 56

6 Analysis and approximation of the Cross model for quasi-Newtonian flows with defective boundary conditions **62**
6.1 Modeling Equations and Preliminaries 63
6.2 The Optimal Control Problem 65
6.3 The Optimality System 66
6.4 Steepest descent approach 72
6.5 Numerical Results 74

7 Approximation of viscoelastic flows with defective boundary conditions **79**
7.1 Model equations 79
7.2 The Optimality system 81
7.3 Steepest descent approach 83
7.4 Mean pressure boundary condition 85
7.5 Nonlinear least squares approach 86
7.6 Numerical Results 90

8 Conclusions 99

Appendices 101
A deal.II code for 3d vorticity equation 102

Bibliography 151

List of Tables

3.1 Velocity and Vorticity errors and convergence rates using the nodal interpolant of the true vorticity for the vorticity boundary condition. 26
3.2 Velocity and Vorticity errors and convergence rates using the nodal interpolant of the L^2 projection of the curl of the discrete velocity into $\mathbf{V}_h$, for the vorticity boundary condition. 27
3.3 Velocity and Vorticity errors and convergence rates using nodal averages of the curl of the discrete velocity for the vorticity boundary condition. 27

4.1 Velocity errors and convergence rates for the first 3d numerical experiment 34
4.2 Vorticity errors and convergence rates for the first 3d numerical experiment 34

List of Figures

2.1 Barycenter refined tetrahedra and triangle. 13

3.1 Flow domain for the 3d step test problem. 28
3.2 Shown above are (top) speed contours and streamlines, (middle) vorticity magnitude, and (bottom) helical density, from the fine mesh computation at time $t = 10$ at the $x = 5$ mid-slice-plane for the 3d step problem with nodal averaging vorticity boundary condition. 29

5.1 Fine mesh used for the resolved NSE solution and the coarse mesh used for the RMDM approximations. 56
5.2 Fine mesh used for the resolved NSE solution and the coarse mesh used for the RMDM approximations. 59
5.3 Diagram of the contraction domain, along with the fine and coarse meshes used in the computations for the contraction problem. 60
5.4 Speed contour plots of the resolved NSE solution as well as solutions of Algorithm 5.2.4. at $t = 4$. 61

6.1 Domain for the flow problem. Red indicates an inflow boundary. Blue indicates an outflow boundary. 75
6.2 Streamlines and magnitude of the velocity approximation for $r = 1.5$ and $\mathbf{g}_0 = [0.1, 0.1]$. 76
6.3 Inflow and outflow velocity profiles for $r = 1.5$ and $\mathbf{g}_0 = [0.1, 0.1]$. 77
6.4 Streamlines and magnitude of the velocity approximation for $r = 1.5$ and $\mathbf{g}_0 = [10, 10]$. 77
6.5 Inflow and outflow velocity profiles for $r = 1.5$ and $\mathbf{g}_0 = [10, 10]$. 78

7.1 Shown above is the domain for the flow problem. 91
7.2 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using Dirichlet boundary conditions for the velocity and stress. 92
7.3 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$. 93
7.4 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the Gauss-Newton algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$. 94
7.5 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the mean pressure matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$. 96
7.6 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [5, ..., 5]$. 97

7.7 Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the Gauss-Newton algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [5, ..., 5]$. 98

Chapter 1

Introduction

The understanding of fluid flow has been a subject of scientific interest for hundreds of years. More recently, the branch of fluid mechanics known as computational fluid dynamics (CFD) has been an area of intense interest for mathematicians due to the multitude of scientific areas that depend on it. Many industries (e.g. automotive, aerospace, environmental) rely on both accurate and efficient simulations of various types of fluids. However, state of the art models and methods are far from being able to efficiently solve most problems of interest in CFD to a desired degree of precision. Moore's law states (roughly) that the amount of computing power available doubles every two years, and has proven to be a fairly accurate estimate over the last 50 years. Despite the great advances made in computing power in that time period, and even assuming Moore's law for computational speed increase continues, the accurate and timely simulation of most flows will not be achieved in the foreseeable future. Advances in mathematics for CFD have gained far more towards this goal than computing power, by developing robust and efficient algorithms built on solid mathematical and physical grounds.

It is the goal of this work to extend the state of the art in mathematics of CFD for two important problems. The first concerns the accurate and efficient simulation of incompressible, viscous Newtonian fluids. We will present and analyze a new numerical method for approximating solutions to the velocity-vorticity-helicity formulation of the Navier-Stokes equations. The driving force behind this new method is that it offers increased physical fidelity and numerical accuracy, along with a step towards further understanding the important but ill-understood physical quantity helicity. Discussion of this method naturally raises the very difficult question of how to accurately

impose boundary conditions on the vorticity, as well as how to compute with turbulent flows. For the former, we propose a new natural boundary condition for the vorticity equation which increases both the accuracy and physical relevance of our discrete vorticity approximation. For the latter, we consider a new reduced-order multiscale model for simulating Newtonian fluids.

The second main problem we study in this work concerns the robust simulation of non-Newtonian fluids in the absence of standard boundary conditions. This problem often arises when modeling flow in an unbounded domain (e.g. modeling blood flow in a portion of a blood vessel). We consider two different approaches for developing accurate and efficient methods for these "defective-boundary" problems for non-Newtonian flows. The first, a gradient-descent method, is presented and tested for both generalized-Newtonian and viscoelastic flow models, and analyzed in the case of the former. The second, a nonlinear least squares method, is presented and tested on a viscoelastic flow model.

The flow of time-dependent, incompressible, viscous Newtonian flows is modeled by the Navier-Stokes equations (NSE), which may be derived from the continuity equation (describing conservation of mass) and the equation describing conservation of momentum. In dimensionless form, the NSE are formulated as

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f}, \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1.2}$$

where $\mathbf{u}$ and p denote the fluid velocity and pressure, respectively, $\mathbf{f}$ denotes an external body force, and $\nu > 0$ denotes the fluid's kinematic viscosity. The Reynolds number $Re = \nu^{-1}$ is a dimensionless parameter representing the ratio of inertial forces to viscous forces. In laminar flows, which are characterized by low Reynolds number, viscous forces dominate inertial forces, making the flow field smooth. Simulations of laminar flows can often be performed without too much complication. For flows with moderate Reynolds numbers, inertial forces start to play a larger role, resulting in complex flow behaviors, making predictions much more difficult. Turbulent flows, characterized by high Reynolds numbers, present very complex and chaotic flow properties, often requiring special models and methods.

In 2010, a velocity-vorticity-helicity (VVH) formulation of the NSE was presented in [55]. This formulation was derived by taking the curl of mass and momentum equations (1.1)-(1.2), and

applying several vector identities to produce the vorticity-helical density equations

$$\frac{\partial \mathbf{w}}{\partial t} - \nu\Delta\mathbf{w} + 2D(\mathbf{w})\mathbf{u} - \nabla\eta = \nabla \times \mathbf{f}, \tag{1.3}$$

$$\nabla \cdot \mathbf{w} = 0. \tag{1.4}$$

where $\mathbf{w} := \nabla \times \mathbf{u}$ is the fluid vorticity, $\eta := \mathbf{u} \cdot \mathbf{w}$ is the helical density, and $D(\mathbf{w}) := \frac{1}{2}(\nabla\mathbf{w} + (\nabla\mathbf{w})^T)$ is the symmetric part of the vorticity gradient. The dimensionless VVH formulation of the NSE then comes from coupling (1.3)-(1.4) to the NSE via the rotational form of the nonlinearity in the momentum equation

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + \mathbf{w} \times \mathbf{u} + \nabla P = \mathbf{f}, \tag{1.5}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1.6}$$

$$\frac{\partial \mathbf{w}}{\partial t} - \nu\Delta\mathbf{w} + 2D(\mathbf{w})\mathbf{u} - \nabla\eta = \nabla \times \mathbf{f}, \tag{1.7}$$

$$\nabla \cdot \mathbf{w} = 0. \tag{1.8}$$

Here $P := \frac{1}{2}\mathbf{u} \cdot \mathbf{u} + \nabla p$ denotes the Bernoulli pressure, which is needed because of our use of the rotational form of the nonlinearity. Since its original derivation in 2010, VVH has been studied in other applications including numerical methods for solving steady incompresible flow [50], the Boussinesq equations [54], and as a selection criterion for the filtering radius in the NS-ω turbulence model [51], all with excellent results.

The VVH formulation of the NSE has four important characteristics that make it attractive for use in simulations. First, numerical methods based on finding velocity and vorticity tend to be more accurate (usually for an added cost, but not necessarily with VVH) [62, 63, 59, 61, 52], and especially in the boundary layer [17]. Second, it solves directly for the helical density η, which may give insight into the important but ill-understood quantity helicity, $H = \int_\Omega \eta \, d\mathbf{x}$, which is believed to play a fundamental role in turbulence [4, 53, 25, 9, 13, 12, 20, 19]. VVH is the first formulation to directly solve for this helical quantity. Third, the use of $\nabla\eta$ in the vorticity equation enables η to act as a Lagrange multiplier corresponding to the divergence-free constraint for the vorticity, analogous to how the pressure relates to the conservation of mass equation. VVH is the first velocity-vorticity method to naturally enforce incompressibility of the vorticity, which is important since (1.4) is as

much a mathematical constraint as it is a physical one, making its violation inconsistent on multiple levels. Finally, the structure of the VVH system allows for a natural splitting of the system into a two-step linearization, since lagging vorticity in the velocity equation linearizes the equation, and similarly lagging velocity in the vorticity equation linearizes this equation as well. A numerical method based on such a splitting was proposed in [55], and when coupled with a finite element discretization, was shown to be accurate on some simple test problems. Chapter 3 of this work will precisely define and further study this discretization of the VVH formulation of the NSE by providing a rigorous stability analysis (for both velocity and vorticity), and testing the method on a benchmark problem.

Amidst our study of this discretization of the VVH formulation, an important, but difficult question is raised in regards to boundary conditions for the vorticity. Consider the basic vorticity equation, derived by taking the curl of the momentum equation (1.1),

$$\frac{\partial \mathbf{w}}{\partial t} - \nu \Delta \mathbf{w} + \mathbf{u} \cdot \nabla \mathbf{w} - \mathbf{w} \cdot \nabla \mathbf{u} = \nabla \times \mathbf{f}. \tag{1.9}$$

Perhaps the most natural and reasonable boundary condition for the vorticity is

$$\mathbf{w} = \nabla \times \mathbf{u} \text{ on } \partial\Omega. \tag{1.10}$$

Unfortunately, this boundary condition presents some difficulty when employed with finite elements. In general, differentiating the piecewise-polynomial $\mathbf{u}_h$ can often lead to a decrease in convergence order [50]. Recently, various methods for avoiding this loss in accuracy have been proposed. In [62], a finite difference approximation of (1.10) using nodal values of the finite element functions is employed. This method is fairly successful on uniform meshes when second-order accuracy is desired, however, it's implementation on non-uniform meshes and for higher-order elements can be quite complex. In general, in the presence of sharp boundary layers of the velocity (e.g. for flows with moderate or high Re), the use of the vorticity boundary condition (1.10) may require extreme mesh refinement around the boundary to avoid inaccurate vorticity approximations. Other methodologies for implementing vorticity boundary conditions have also been tried, with some success. In [55], the vorticity on the boundary was set to be the L^2 projection of the discontinuous finite element function $\nabla \times u_h$ into the continuous finite element space. This method is one of three implemented

in the numerical testing of our method for the VVH system presented in Chapter 3, providing fairly accurate results on a benchmark flow problem. Other strategies include using the boundary element method [42], or the lattice Boltzmann method [18]. In Chapter 4, we employ a different approach in deriving a new vorticity boundary condition, in hopes of avoiding any unnecessary complication. The proposed method includes natural boundary conditions for a weak formulation of the vorticity equation. The boundary conditions are derived directly from the physical equations and the finite element method, making them simpler to understand than some of the aforementioned strategies. A full derivation of these vorticity boundary conditions will be presented in Chapter 4, along with a numerical scheme to evaluate their effectiveness in a numerical experiment.

Another clear need in the development of the VVH algorithm is for some kind of stabilization/subgrid model to allow us to handle higher Re flows. In Chapter 5 we consider a new reduced order, multiscale, approximate deconvolution model for Newtonian flows. Approximate deconvolution models (ADM) are a form of large eddy simulation (LES) models introduced in [2, 3] for the purpose of simulating large-scale flow strctures at a reduced computational cost compared with direct numerical simulation (DNS). We know from Kolmogorov's 1941 theory that eddies below a critical size ($O(Re^{-3/4})$ for 3d flow) are dominated by viscous forces and disappear very quickly, while those above this critical size are deterministic in nature. Hence, a DNS requires $O(Re^{9/4})$ mesh points in space per time step to accurately simulate eddies in 3d. Even for moderate Re flows, this requirement makes DNS computationally infeasible. ADM models (and LES models in general) aim to avoid this problem by filtering out small scales, while modeling their effect on the large scales. Because only large scales are being solved for, these models require a significantly smaller amount of mesh points than DNS. Recently, a promising new multiscale deconvolution model (MDM) [22] has been proposed which avoids some of the drawbacks of general ADM models, and is given by

$$\mathbf{v}_t + \overline{G_\gamma \mathbf{v} \cdot \nabla G_\gamma \mathbf{v}} + \nabla q - \nu \Delta \mathbf{v} = \overline{\mathbf{f}} \tag{1.11}$$

$$\nabla \cdot \mathbf{v} = 0. \tag{1.12}$$

This formulation makes use of two different Helmholtz filters (associated with two different filtering radii α and γ) and a deconvolution operator G_γ which connects the two filter scales. This formulation and these filters and operators will all be defined in detail in Chapter 5, where we derive (in detail) a new, reduced order MDM, along with an efficient and stable algorithm to approximate it.

The second half of this work is concerned with the accurate and efficient simulation of the defective boundary problem for two types of non-Newtonian fluids. The modeling of flow in an unbounded domain requires the introduction of artificial boundaries. Often, the flow is assumed to satisfy some Dirichlet or Neumann boundary condition on a portion of these artificial boundaries (e.g. inflow or outflow boundaries). However, the amount of boundary data available for a given flow is often very limited, making these types of boundary conditions very hard to impose. In many practical applications the only flow data available are quantitative (e.g. average flow rates, mean pressure values, etc.). In situations like these, it is often more realistic to model the flow using defective boundary conditions. Typically, governing equations are chosen depending on the flow being modeled, and instead of completing these equations with standard Dirichlet or Neumann type boundary conditions, the defective boundary problem consists of only considering information such as flow rates (or mean pressure values) on the inflow or outflow boundaries S_i, i.e.

$$\int_{S_i} \mathbf{u} \cdot \mathbf{n} \, dS = Q_i \text{ for } i = 1, ..., m. \tag{1.13}$$

We note that these boundary conditions are known as "defective" because they are insufficient to close the differential model (i.e. our flow problem is ill-posed) [26]. The goal of the second half of this work is to study this problem in the context of two different types of flows (and hence two different types of modeling equations).

Before we proceed we note that flow problems with defective boundary conditions have been studied in various applications in the past. In [41], the defective boundary problem for the NSE was studied where flow rates are specified on inflow and outflow boundaries. In this work a "do-nothing" approach is presented where the flow rate conditions are implicitly incorporated into the variational formulation through the choice of appropriate boundary conditions and function spaces, resulting in a well-posed variational problem. An alternative approach to the defective boundary problem for the NSE subject to flow rate conditions was presented in [26]. In this study, the flow rate conditions are enforced weakly via the Lagrange multiplier method. In [24] the defective boundary problem for quasi-Newtonian flows subject to flow rate conditions was investigated using the Lagrange multiplier method. Both the continuous and discrete variational formulations of a generalized set of modeling equations were proven to be well-posed, and error analysis of the numerical approximation was also presented. In [27], a new approach to the defective boundary problem for Stokes flow was proposed.

This approach formulates the defective boundary problem as an optimal control problem through the choice of a suitable functional to minimize. This approach proved to be versatile, as the functional to minimize can be altered to match various kinds of defective boundaries (flow rates, mean pressure, etc). In the optimal control formulation, the control was chosen to be a constant normal stress on each of the inflow and outflow boundaries, and appears in the modeling equations through the addition of a boundary integral (often referred to as a "boundary control" [35]).

The study of optimal control problems for Newtonian and non-Newtonian fluids has been istelf an active research area in the recent past, e.g. [35, 36, 37]. One approach to solve these types of optimization problems is based off of solving "sensitivity equations," which are derived through the Frechet derivative of the constraint operator with respect to the control variables [35, 11, 38]. An alternative approach studied in [35, 49] is an adjoint-based optimization method, in which the method of Lagrange multipliers is used to derive an optimality system consisting of constraint equations, adjoint equations, and a necessary condition. In [21] an optimal control problem for the Ladyzhenskaya model for generalized-Newtonian flows was studied. Additionally, a shape optimization problem for blood flow modeled by the Cross model was presented in [1]. In [48] a defective boundary problem for generalized-Newtonian flows was studied. In that work the model problem considered was the three-field power law model subject to flow rate or mean pressure conditions on portions of the boundary. The defective boundary problem was formulated as an optimal control problem which was then transformed into an unconstrained optimization problem via the Lagrange multiplier method. However, analysis of the adjoint problem and the method of Lagrange multipliers was limited, in part due to the choice of modeling equations.

In Chapter 6, we begin by considering the defective boundary problem for generalized-Newtonian fluids governed by the Cross modeling equations [16] (which will be explicitly defined later in this work). Newtonian fluids are characterized by having a shear stress, denoted by $\boldsymbol{\sigma}$, that is directly proportional to its shear rate (given by $D(\mathbf{u})$), i.e.

$$\boldsymbol{\sigma} = 2\nu D(\mathbf{u}), \tag{1.14}$$

where the fluid viscosity ν is constant. On the other hand, generalized-Newtonian flows have the same stress-strain relationship, but with a non-constant fluid viscosity dependent upon the velocity

of the flow

$$\boldsymbol{\sigma} = 2\nu(|D(\mathbf{u})|)D(\mathbf{u}), \tag{1.15}$$

where the viscosity function $\nu(|D(\mathbf{u})|)$ is chosen to reflect the flow being modeled. The Cross model specifies the viscosity function as

$$\nu(|D(\mathbf{u})|) := \nu_\infty + \frac{(\nu_0 - \nu_\infty)}{1 + (\lambda|D(\mathbf{u})|)^{2-r}}, \tag{1.16}$$

where $\lambda > 0$ is a time constant, $1 \leq r \leq 2$ is a dimensionless rate constant, and ν_0 and ν_∞ denote limiting viscosity values at a zero and infinite shear rate, respectively, assumed to satisfy $0 \leq \nu_\infty \leq \nu_0$. We take the approach of [48] to approximate our model problem subject to flow rate and mean pressure conditions. The problem is formulated as an optimal control problem for which we analytically justify the use of the method of Lagrange multipliers to derive an optimality system. We then show that the resulting adjoint system is well-posed. Finally, we consider a complex numerical experiment to test the robustness of an optimization algorithm previously presented in [48].

In Chapter 7, we consider the same defective boundary problem but for viscoelastic fluids governed by the Johnson-Segalman modeling equations. Viscoelastic fluids are a type of non-Newtonian fluid that exhibit both viscous and elastic characteristics when undergoing deformation. This is reflected in the modeling equations by an extra nonlinear constitutive equation, which relates the stress tensor $\boldsymbol{\sigma}$ to the fluid velocity. Some analytical and numerical studies for an optimal control of non-Newtonian flows can be found in [1, 21, 45, 49]. The Johnson-Segalman modeling equations for viscoelastic, creeping flow are given by

$$\boldsymbol{\sigma} + \lambda(\mathbf{u} \cdot \nabla)\boldsymbol{\sigma} + \lambda g_a(\boldsymbol{\sigma}, \nabla\mathbf{u}) - 2\alpha D(\mathbf{u}) = 0, \tag{1.17}$$

$$-\nabla \cdot \boldsymbol{\sigma} - 2(1-\alpha)\nabla \cdot D(\mathbf{u}) + \nabla p = \mathbf{f}, \tag{1.18}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1.19}$$

Here λ denotes the Weissenberg number, defined as the product of relaxation time and a characteristic strain rate of the fluid, α is a number satisfying $0 < \alpha < 1$ which can be considered as the fraction

of viscoelastic viscosity, and $g_a(\boldsymbol{\sigma}, \nabla \mathbf{u})$ is a nonlinear function of $\boldsymbol{\sigma}$ and $\mathbf{u}$ that will be explicitly defined in Chapter 7. We consider the defective boundary problem for viscoelastic fluids governed by these equations. This includes a fully detailed formulation of the problem itself, the minimization problem, and a derivation of the optimality system. The numerical algorithm presented in Chapter 6 will then be used to solve the minimization problem, along with a second, new algorithm. Finally, we consider a numerical test to compare and contrast both algorithms.

This work is arranged as follows. Chapter 2 contains mathematical notation and preliminaries that will be used throughout the following sections. Chapter 3 presents a stability analysis and numerical testing of a finite element method for the VVH formulation. Chapter 4 fully defines a new vorticity boundary condition, and presents a numerical experiment designed to verify its accuracy. Chapter 5 derives and analyzes a new reduced order MDM, and presents two numerical tests to verify its efficiency. Chapter 6 presents the work on generalized-Newtonian flows with defective boundary conditions, and Chapter 7 contains the work on viscoelastic flows with defective boundary conditions. Finally, Chapter 8 contains conclusions from the various works presented herein.

Chapter 2

Preliminaries

Throughout the analysis presented in this work we will assume that the domain Ω denotes a bounded, connected subset of $\mathbb{R}^d$ (with $d = 2$ or 3), with piecewise smooth boundary $\partial\Omega$. We will denote the $L^2(\Omega)$ norm and inner product by $\|\cdot\|$ and $(\cdot,\cdot)$, respectively, while $L^p(\Omega)$ norms will be denoted by $\|\cdot\|_{L^p}$. Sobolev $W_p^k(\Omega)$ norms and seminorms will be indicated by $\|\cdot\|_{W_p^k}$ and $|\cdot|_{W_k^p}$, respectively. We will use the standard notation of $H^k(\Omega)$ to refer to the sobolev space $W_2^k(\Omega)$, with norm $\|\cdot\|_k$. Dual spaces will be denoted $(\cdot)^*$ with duality pairing $\langle\cdot,\cdot\rangle$ and norm $\|\cdot\|^*$. For domains other than Ω we will explicitly indicate the domain in the space and norm notation. For $k \in \mathbb{R}$ the space H_0^k is defined as

$$H_0^k(\Omega) := \{v \in H^k(\Omega) \,|\, v = 0 \text{ on } \partial\Omega\}.$$

The zero-mean subspace of $L^2(\Omega)$ is defined as

$$L_0^2(\Omega) := \{q \in L^2(\Omega) \,|\, \int_\Omega q = 0\}.$$

For functions $v(x,t)$ defined on $\Omega \times (0,T)$ for some positive end time T, we will make use of the norms

$$\|v\|_{n,k} := \left(\int_0^T \|v(\cdot,t)\|_k^n \, dt\right)^{1/n} \text{ and } \|v\|_{\infty,k} := ess \sup_{0<t<T} \|v(\cdot,t)\|_k .$$

For functions of time, we will use the notation $t^n := n\Delta t$ where Δt denotes a chosen time-step. For

continuous functions of time $f(t)$, we use the notation

$$f^n := f(t^n),$$

and

$$f^{n+1/2} := f(t^{n+\frac{1}{2}}) = f\left(\frac{t^{n+1}+t^n}{2}\right).$$

The average of the nth and $(n+1)$st time level of a discrete function v is denoted

$$v^{n+1/2} := \frac{v^{n+1}+v^n}{2}.$$

Our error analysis will require the use of discrete time analogues of the continuous in time norms:

$$|||v|||_{p,k} := \left(\sum_{n=1}^{N_T} \|v^n\|_k^p \, \Delta t\right)^{1/p} \quad \text{and} \quad |||v|||_{\infty,k} := \max_{1\le n\le N_T} \|v^n\|_k$$

We will use bold font to denote vector functions and tensor functions. We will also use bold font to denote vector function spaces, e.g.

$$\mathbf{H}^1(\Omega) := (H^1(\Omega))^d \text{ and } \mathbf{H}_0^1(\Omega) := (H_0^1(\Omega))^d.$$

Throughout our analysis we will frequently employ the following inequality, one result of which is that for $v \in H_0^1(\Omega)$, the seminorm $|v|_1$ is equivalent to $\|v\|_1$.

Lemma 2.0.1 (The Poincare-Friedrichs inequality)**.** *There exists a positive constant $C_{PF} = C_{PF}(\Omega)$ such that*

$$\|v\| \le C_{PF} \|\nabla v\| \;\forall\, v \in H_0^1(\Omega).$$

Proof. A proof of this well known inequality can be found in [28]. □

We will often use the $(H_0^1(\Omega))^* = H^{-1}(\Omega)$ norm, denoted by $\|\cdot\|_{-1}$, to measure the size of

a forcing function. The $H^{-1}(\Omega)$ norm is defined as

$$\|f\|_{-1} := \sup_{v \in H_0^1(\Omega)} \frac{\langle f, v \rangle}{\|\nabla v\|}.$$

We note that the space $H^{-1}(\Omega)$ is the closure of $L^2(\Omega)$ in $\|\cdot\|_{-1}$.

The continuous velocity, pressure, and stress spaces, denoted $\mathbf{X}$, Q, and $\mathbf{\Sigma}$, respectively, will be specified in each chapter. The weakly divergence-free subspace $\mathbf{V}$ of $\mathbf{X}$ is defined as

$$\mathbf{V} := \{\mathbf{v} \in \mathbf{X} \,|\, (\nabla \cdot \mathbf{v}, q) = 0 \,\forall\, q \in Q\}.$$

In the discrete setting, we begin by letting τ_h denote a regular, conforming triangulation or tetrahedralization of Ω. The velocity and pressure finite element spaces defined on τ_h will be denoted as $\mathbf{X}_h$ and Q_h, respectively, and will be specified in each chapter. The divergence-free subspace $\mathbf{V}_h$ of $\mathbf{X}_h$ is defined as

$$\mathbf{V}_h := \{\mathbf{v}_h \in \mathbf{X}_h \,|\, (\nabla \cdot \mathbf{v}_h, q_h) = 0 \,\forall\, q_h \in Q_h\}.$$

We will often make use of the Taylor-Hood (TH) element pair, defined as $(\mathbf{X}_h, Q_h) = ((P_k)^d, P_{k-1})$, i.e.

$$\mathbf{X}_h := \{\mathbf{v}_h \in \mathbf{H}_0^1(\Omega) \,|\, \mathbf{v}_h|_K \in (P_k)^d(K) \,\forall\, K \in \tau_h\}$$
$$Q_h := \{q_h \in L_0^2(\Omega) \cap C^0(\Omega) \,|\, q_h|_K \in P_{k-1}(K) \,\forall\, K \in \tau_h\}.$$

It is a well-known result that for $k \geq 2$ the TH element pair satisfies the discrete inf-sup condition [10, 30].

We will also make use of the Scott-Vogelius (SV) element pair $(\mathbf{X}_h, Q_h) = ((P_k)^d, P_{k-1}^{disc})$ [60], which uses the same velocity approximation space as TH elements, but allows the pressure approximation space to be discontinuous. An immediate consequence of this choice of spaces is that $\nabla \cdot \mathbf{X}_h \subset Q_h$. Hence, with this choice of elements, the discretely div-free subspace $\mathbf{V}_h \subset \mathbf{X}_h$ now becomes

$$\mathbf{V}_h := \{\mathbf{v}_h \in \mathbf{X}_h \,|\, (\nabla \cdot \mathbf{v}_h, q_h) = 0 \,\forall\, q_h \in Q_h\} = \{\mathbf{v}_h \in \mathbf{X}_h \,|\, \nabla \cdot \mathbf{v}_h = 0\}.$$

This makes the SV element pair a natural choice for both velocity-pressure and vorticity-helicity systems as it results in pointwise enforcement of solenoidal constraints for the velocity and vorticity (as opposed to weak enforcement by TH elements). The drawback of using discontinuous elements is that the dimension of Q_h in the SV element pair is significantly larger than in the TH element pair, resulting in a linear system with a greater amount of degrees of freedom when using SV elements.

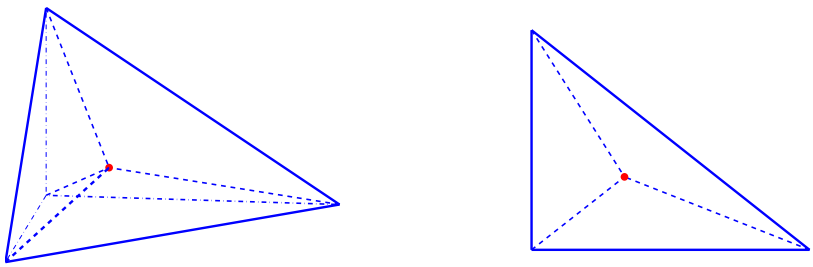


Figure 2.1: Barycenter refined tetrahedra and triangle.

In order for the SV element pair to be discretely inf-sup stable, any of the following conditions on the mesh τ_h are sufficient [57, 66, 65, 67]:

1. In $2d$, $k \geq 4$ and the mesh has no singular vertices
2. In $3d$, $k \geq 6$ on a quasi-uniform tetrahedral mesh
3. In $2d$ or $3d$, when $k \geq d$ and the mesh is generated as a barycenter refinement of a regular, conforming triangular or tetrahedral mesh
4. When the mesh is of Powell-Sabin type and $k = 1$ in $2d$ or $k = 2$ in $3d$

We note that a complete classification of conditions for discrete inf-sup stability of SV elements, including the minimum degree for general meshes without special refinements, is an open question. In

our computations performed with SV elements we will always use condition 3. Figure 2.1 illustrates a barycenter-refined triangle.

For our convergence studies in Chapter 3, 4, and 5 we will assume our choice of finite element spaces satisfies the following well known approximation properties:

$$\inf_{\mathbf{v}\in\mathbf{X}_h} \|\mathbf{u}-\mathbf{v}\| \le Ch^{k+1} \|\mathbf{u}\|_{k+1} \text{ for any } \mathbf{u}\in\mathbf{H}^{k+1}(\Omega)$$

$$\inf_{\mathbf{v}\in\mathbf{X}_h} \|\mathbf{u}-\mathbf{v}\|_1 \le Ch^{k} \|\mathbf{u}\|_{k+1} \text{ for any } \mathbf{u}\in\mathbf{H}^{k+1}(\Omega)$$

$$\inf_{r\in Q_h} \|p-r\| \le Ch^{s+1} \|p\|_{s+1} \text{ for any } p\in H^{s+1}(\Omega).$$

We note that these approximation properties hold for both TH and SV elements.

In Chapter 5, the trilinear operator $b^* : \mathbf{X}\times\mathbf{X}\times\mathbf{X}\to\mathbb{R}$ defined by

$$b^*(\mathbf{u},\mathbf{v},\mathbf{w}) := (\mathbf{u}\cdot\nabla\mathbf{v},\mathbf{w})$$

will be used. The following useful properties of b^* are proven in [46].

Lemma 2.0.2. *If $\nabla\cdot\mathbf{u}=0$, then*

$$b^*(\mathbf{u},\mathbf{v},\mathbf{v}) = 0.$$

Additionally, there exists a constant C dependent on the size of Ω such that

$$|b^*(\mathbf{u},\mathbf{v},\mathbf{w})| \le C\|\mathbf{u}\|^{\frac{1}{2}}\|\nabla\mathbf{u}\|^{\frac{1}{2}}\|\nabla\mathbf{v}\|\|\nabla\mathbf{w}\|,$$

$$|b^*(\mathbf{u},\mathbf{v},\mathbf{w})| \le C\|\nabla\mathbf{u}\|\|\nabla\mathbf{v}\|\|\nabla\mathbf{w}\|.$$

Our error analysis will also use the following discrete version of the Gronwall inequality.

Lemma 2.0.3. *Let k, B and $a_\mu, b_\mu, c_\mu, \gamma_\mu$, for integers $\mu\ge 0$, be nonnegative numbers such that*

$$a_n + k\sum_{\mu=0}^{n} b_\mu \le k\sum_{\mu=0}^{n}\gamma_\mu a_\mu + k\sum_{\mu=0}^{n} c_\mu + B \text{ for } n\ge 0. \tag{2.1}$$

Suppose that $k\gamma_\mu < 1$*, for all* μ*, and set* $\sigma_\mu = (1 - k\gamma_\mu)^{-1}$*. Then,*

$$a_n + k\sum_{\mu=0}^{n} b_\mu \leq \exp\left(k\sum_{\mu=0}^{n}\sigma_\mu\gamma_\mu\right)\left[k\sum_{\mu=0}^{n} c_\mu + B\right] \text{ for } n \geq 0. \tag{2.2}$$

Remark 2.0.4. *If the first sum on the right in* (2.1) *extends only up to* $n-1$*, then estimate* (2.2) *holds for all* $k > 0$*, with* $\sigma_\mu = 1$*.*

Proof. A proof of these results can be found in [40]. □

Chapter 3

A Numerical Study for a Velocity-Vorticity-Helicity formulation of the 3D Time-Dependent NSE

In this chapter we study a finite element method for the 3d NSE in velocity-vorticity-helicity form. For $\Omega \subset \mathbb{R}^3$, recent work [56] has shown that the NSE can be equivalently written in VVH form as: Find $\mathbf{u} : \Omega \times (0,T) \to \mathbb{R}^3$ and $p : \Omega \times (0,T) \to \mathbb{R}$ satisfying

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + \mathbf{w} \times \mathbf{u} + \nabla P = \mathbf{f} \quad \text{in} \quad \Omega \times (0,T), \tag{3.1}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in} \quad \Omega \times (0,T), \tag{3.2}$$

$$\mathbf{u}|_{t=0} = \mathbf{u}_0 \quad \text{in} \quad \Omega, \tag{3.3}$$

$$\mathbf{u} = \boldsymbol{\phi} \quad \text{on} \quad \partial\Omega \times (0,T), \tag{3.4}$$

and find $\mathbf{w} : \Omega \times (0,T) \to \mathbb{R}^3, \eta : \Omega \times (0,T) \to \mathbb{R}$ satisfying

$$\frac{\partial \mathbf{w}}{\partial t} - \nu \Delta \mathbf{w} + 2\mathbf{D}(\mathbf{w})\mathbf{u} - \nabla \eta = \nabla \times \mathbf{f} \quad \text{in} \quad \Omega \times (0,T), \tag{3.5}$$

$$\nabla \cdot \mathbf{w} = 0 \quad \text{in} \quad \Omega \times (0,T), \tag{3.6}$$

$$\mathbf{w}|_{t=0} = \nabla \times \mathbf{u}_0 \quad \text{in} \quad \Omega, \tag{3.7}$$

$$\mathbf{w} = \nabla \times \mathbf{u} \quad \text{on} \quad \partial\Omega \times (0,T), \tag{3.8}$$

where $\boldsymbol{\phi}$ is a Dirichlet boundary condition for velocity satisfying $\int_\Omega \boldsymbol{\phi} \cdot \mathbf{n} = 0 \ \forall t \in (0,T)$. In [55], a numerical algorithm based on a 2-step linearization of the VVH formulation was proposed. In this chapter, we study this discretization of the VVH formulation further by providing a rigorous stability analysis, testing the method on several benchmark problems, and with various vorticity boundary conditions.

3.1 Discrete VVH Formulation

For our finite element discretization of the VVH formulation, we will choose velocity and pressure spaces $(\mathbf{X}_h, Q_h) \subset (\mathbf{H}_0^1(\Omega), L^2(\Omega))$ on our mesh τ_h to be the Scott-Vogelius element pair $(\mathbf{P}_k, P_{k-1}^{disc})$. We will denote the vorticity space by Y_h, where $Y_h \subset \mathbf{H}^1(\Omega)$ is the space $\mathbf{P}_k$. We note that the only difference between the velocity and vorticity finite element spaces is the value of the finite element functions on the boundary $\partial\Omega$. To simplify the analysis, we require the mesh is sufficiently regular so that the inverse inequality holds,

$$\|\nabla \mathbf{u}_h\| \leq C_i h^{-1} \|\mathbf{u}_h\|. \tag{3.9}$$

For the initial conditions for our velocity and vorticity approximations we will use the L^2 projection into $\mathbf{V}_h$. For $\phi \in \mathbf{L}^2(\Omega)$, the L^2 projection of ϕ into $\mathbf{V}_h$, denoted by $P_{\mathbf{V}_h}(\phi)$, satisfies

$$(P_{\mathbf{V}_h}(\phi), \mathbf{v}_h) = (\phi, \mathbf{v}_h) \text{ for any } \mathbf{v}_h \in \mathbf{V}_h.$$

Define the operator $A_h^{-1} : \mathbf{L}^2(\Omega) \to \mathbf{V}_h$ as the solution operator to the discrete Stokes problem:

$$(\nabla A_h^{-1} \boldsymbol{\psi}, \nabla \mathbf{v}_h) = (\boldsymbol{\psi}, \mathbf{v}_h), \ \forall \ \mathbf{v}_h \in \mathbf{V}_h. \tag{3.10}$$

This operator will not be used in computations, but is used in the analysis of the proposed algorithm. The following lemma was proven in [50].

Lemma 3.1.1. *Assume Ω is such that the Stokes problem is H^2-regular. For any $\boldsymbol{\psi} \in \mathbf{L}^2(\Omega)$ it holds*

$$\|A_h^{-1}\boldsymbol{\psi}\|_{L^\infty} + \|\nabla A_h^{-1}\boldsymbol{\psi}\|_{L^3} \leq C_0 \|\boldsymbol{\psi}\|, \tag{3.11}$$

and for any $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $q \in L^2(\Omega)$, and $\boldsymbol{\phi} \in \mathbf{H}^1(\Omega)$

$$|(\mathbf{f}, \nabla \times A_h^{-1}\boldsymbol{\psi})| \leq C(\|\mathbf{f}\|_{-1} + h\|\mathbf{f}\|)\|\boldsymbol{\psi}\|, \tag{3.12}$$

$$|(q, \nabla \cdot A_h^{-1}\boldsymbol{\psi})| \leq C(\|q\|_{-1} + h\|q\|)\|\boldsymbol{\psi}\|, \tag{3.13}$$

$$|(\nabla\boldsymbol{\phi}, \nabla A_h^{-1}\boldsymbol{\psi})| \leq C(\|\boldsymbol{\phi}\| + \|\boldsymbol{\phi}\|_{-\frac{1}{2},\partial\Omega} + h\|\nabla\boldsymbol{\phi}\|)\|\boldsymbol{\psi}\|. \tag{3.14}$$

The chosen time discretization is trapezoidal, and the linearization uses second order extrapolation. The fully discrete 2-step version of (3.1)-(3.8) we study is: find $(\mathbf{u}_h, \mathbf{w}_h, P_h, \eta_h) \in (\mathbf{X}_h, \mathbf{Y}_h, Q_h, Q_h)$ satisfying $\forall(\mathbf{v}_h, \boldsymbol{\chi}_h, q_h, r_h) \in (\mathbf{X}_h, \mathbf{X}_h, Q_h, Q_h)$,

Step 1:

$$\begin{aligned} \frac{1}{\Delta t}(\mathbf{u}_h^{n+1} - \mathbf{u}_h^n, \mathbf{v}_h) + \nu(\nabla \mathbf{u}_h^{n+\frac{1}{2}}, \nabla \mathbf{v}_h) - (P_h^{n+1}, \nabla \cdot \mathbf{v}_h) & \\ +((\frac{3}{2}\mathbf{w}_h^n - \frac{1}{2}\mathbf{w}_h^{n-1}) \times \mathbf{u}_h^{n+\frac{1}{2}}, \mathbf{v}_h) - (\mathbf{f}^{n+\frac{1}{2}}, \mathbf{v}_h) &= 0, \quad (3.15) \\ (\nabla \cdot \mathbf{u}_h^{n+1}, q_h) &= 0, \quad (3.16) \end{aligned}$$

Step 2:

$$\begin{aligned} \frac{1}{\Delta t}(\mathbf{w}_h^{n+1} - \mathbf{w}_h^n, \boldsymbol{\chi}_h) + \nu(\nabla \mathbf{w}_h^{n+\frac{1}{2}}, \nabla \boldsymbol{\chi}_h) & \\ +(\eta_h^{n+1}, \nabla \cdot \boldsymbol{\chi}_h) + \gamma\nu^{-1}((\nabla \times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}}) \times \mathbf{u}_h^{n+\frac{1}{2}}, (\nabla \times \boldsymbol{\chi}_h) \times \mathbf{u}_h^{n+\frac{1}{2}}) & \quad (3.17) \\ +2(\mathbf{D}(\mathbf{w}_h^{n+\frac{1}{2}})\mathbf{u}_h^{n+\frac{1}{2}}, \boldsymbol{\chi}_h) - (\nabla \times \mathbf{f}^{n+\frac{1}{2}}, \boldsymbol{\chi}_h) &= 0, \\ (\nabla \cdot \mathbf{w}_h^{n+1}, r_h) &= 0, \quad (3.18) \\ \mathbf{w}_h^{n+1}|_{\partial\Omega} - I_h(\nabla \times \mathbf{u}^{n+1})|_{\partial\Omega} &= \mathbf{0}, \quad (3.19) \end{aligned}$$

where $\mathbf{u}_h^0 = P_{\mathbf{V}_h}(\mathbf{u}_0)$, $\mathbf{w}_h^0 = P_{\mathbf{V}_h}(\nabla \times \mathbf{u}_0)$, and I_h denotes an appropriate interpolant. As is common practice in trapezoidal schemes for fluid flow, the Lagrange multiplier terms are solved for directly at their $n + 1/2$ time levels, i.e. no splitting into time n and $n + 1$ pieces is necessary, and so P_h^{n+1}

and η_h^{n+1} are approximations to their continuous counterparts at $t = t^{n+1/2}$. Note also that we have assumed a homogeneous Dirichlet boundary condition for velocity, and a Dirichlet condition for vorticity that it be equal to an appropriate interpolant of the curl of the velocity on the boundary. This is the simplest case for analysis, but is still quite formidable. Extension to other common boundary conditions will lead to additional technical details, and need to be considered on case by case basis.

Due to the difficulties associated with any analysis involving the vorticity equation, there are two components in the above scheme that are for the purposes of analysis only. The unconditional stability of the velocity does not depend on either of these components of the numerical scheme, but proving unconditional stability of the vorticity requires both of them.

First, the boundary condition for the discrete vorticity (3.19) is given in terms of the true velocity, which is not practical. In computations, we use instead the condition

$$\mathbf{w}_h^{n+1}|_{\partial\Omega} - I_h(\nabla \times \mathbf{u}_h^{n+1})|_{\partial\Omega} = \mathbf{0}, \tag{3.20}$$

however analyzing the system with such a boundary condition does not appear possible in this particular formulation. Developing improved formulations for which such a vorticity boundary condition does allow analysis is an important open question. We will consider two possibilities of interpolants in our computations: i) a nodal interpolant of the L^2 projection of the curl of the velocity into $\mathbf{V}_h$, and ii) a nodal interpolant of a local averaging of the curl of the velocity. A new vorticity boundary condition, presented in the next section, is also feasible with this discretization.

The second part of the scheme that is not used in computations is the penalty term in (3.17), i.e. we choose $\gamma = 0$ in our computations. In the continuous case this term is consistent for the homogeneous or periodic boundary conditions on a rectangular box: for sufficiently regular

solutions, $\mathbf{w} = \nabla \times \mathbf{u}$, and A^{-1} the continuous Stokes solution operator, since $\nabla \cdot \mathbf{u} = 0$,

$$\begin{aligned}
(\nabla \times A^{-1}\mathbf{w}) \times \mathbf{u} &= (\nabla \times A^{-1}(\nabla \times \mathbf{u})) \times \mathbf{u} \\
&= (A^{-1}(\nabla \times (\nabla \times \mathbf{u}))) \times \mathbf{u} \\
&= (A^{-1}(-\Delta \mathbf{u} - \nabla(\nabla \cdot \mathbf{u})) \times \mathbf{u} \\
&= (A^{-1}(-\Delta \mathbf{u})) \times \mathbf{u} \\
&= (A^{-1}(A\mathbf{u})) \times \mathbf{u} \\
&= \mathbf{0}.
\end{aligned} \tag{3.21}$$

Outside of the periodic case, the differential operators will not commute and thus errors will arise at the boundary from this term; hence the term appears to damp vorticity creation at the boundary, and we do not use it in our computations. However, it does not appear possible to prove a vorticity stability bound without it.

3.1.1 Stability Analysis

Lemma 3.1.2 (Stability)**.** *Assume* $\mathbf{f} \in L^2(0,T;\mathbf{H}^{-1}(\Omega))$ *and* $\mathbf{u}_0 \in \mathbf{L}^2(\Omega)$*. Then velocity solutions to* (3.15)-(3.17) *are unconditionally stable, and satisfy*

$$\left\|\mathbf{u}_h^M\right\|^2 + \Delta t \sum_{n=0}^{M-1} \nu \left\|\nabla \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 \leq \Delta t \sum_{n=0}^{M-1} \nu^{-1} \left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{-1}^2 + C \left\|\mathbf{u}_0\right\|^2 := C_4. \tag{3.22}$$

Proof. Let $\mathbf{v}_h = \mathbf{u}_h^{n+\frac{1}{2}}$ in (3.15) and simplify to get

$$\frac{1}{2\Delta t}(\left\|\mathbf{u}_h^{n+1}\right\|^2 - \left\|\mathbf{u}_h^n\right\|^2) + \nu \left\|\nabla \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 = (\mathbf{f}^{n+\frac{1}{2}}, \mathbf{u}_h^{n+\frac{1}{2}}).$$

Using Cauchy Schwarz, Young's inequality, and simplifying yields

$$\frac{1}{2\Delta t}(\left\|\mathbf{u}_h^{n+1}\right\|^2 - \left\|\mathbf{u}_h^n\right\|^2) + \frac{\nu}{2} \left\|\nabla \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 \leq \frac{\nu^{-1}}{2} \left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{-1}^2.$$

Multiplying by $2\Delta t$ and summing from 0 to $M-1$ then gives

$$\begin{aligned}\left\|\mathbf{u}_h^M\right\|^2+\Delta t\sum_{n=0}^{M-1}\nu\left\|\nabla\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 &\leq \Delta t\sum_{n=0}^{M-1}\nu^{-1}\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{-1}^2+\left\|\mathbf{u}_h^0\right\|^2\\ &\leq \Delta t\sum_{n=0}^{M-1}\nu^{-1}\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{-1}^2+C\left\|\mathbf{u}_0\right\|^2,\end{aligned}$$

which proves the estimate (3.22). □

Remark 3.1.3. We note that the unconditional stability of the velocity solution is independent of both the vorticity boundary condition and the penalty term of the discrete vorticity equation.

Lemma 3.1.4. *Assume* $\mathbf{f}\in L^2(0,T;\mathbf{L}^2(\Omega))$, $\mathbf{u}_0\in\mathbf{H}_0^1(\Omega)$, $\mathbf{u}\in L^\infty(0,T;\mathbf{H}^2(\Omega))$, $\mathbf{u}_t\in L^\infty(0,T;\mathbf{H}^1(\Omega))$, *and* $\mathbf{u}_{tt}\in L^\infty(0,T;\mathbf{H}^1(\Omega))$. *Then vorticity solutions are also stable, in the sense of*

$$\left\|\nabla A_h^{-1}\mathbf{w}_h^M\right\|^2+\Delta t\sum_{n=0}^{M-1}\nu\left\|\mathbf{w}_h^{n+\frac{1}{2}}\right\|^2\leq C(\nu^{-2},C_4,M,T,\mathbf{f},\mathbf{u}):=C_5. \tag{3.23}$$

Remark 3.1.5. *It appears that the penalty parameter* γ *needs to satisfy* $\gamma>\frac{1}{2}$ *for the proof to hold. When* $\gamma=0$, *we are reduced to the non-penalty term case, for which we are unable to prove unconditional stability.*

Proof. For the vorticity bound, let $\mathbf{w}_h^{n*}=I_h(\nabla\times\mathbf{u}^n)$ where I_h is a discretely div-free preserving interpolant. Note $\mathbf{w}_h^{n*}\in\mathbf{V}_h$ and $\mathbf{w}_h^{n*}$ satisfies the vorticity boundary condition (3.19). The vorticity solution can then be decomposed as

$$\mathbf{w}_h^{n+\frac{1}{2}}=\mathbf{w}_h^{n+\frac{1}{2}*}+\overline{\mathbf{w}_h^{n+\frac{1}{2}}}, \tag{3.24}$$

where $\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\in\mathbf{V}_h$. Letting $I_h(\nabla\times\mathbf{u})\leq C_u$ for all t, we have

$$\left\|\mathbf{w}_h^{n+\frac{1}{2}*}\right\|\leq C_u. \tag{3.25}$$

Substituting (3.24) into the vorticity equation (3.17) yields, $\forall \boldsymbol{\chi}_h \in \mathbf{V}_h$,

$$\begin{aligned}\frac{1}{\Delta t}(\overline{\mathbf{w}_h^{n+1}} - \overline{\mathbf{w}_h^n}, \boldsymbol{\chi}_h) + \gamma\nu^{-1}((\nabla \times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \times \mathbf{u}_h^{n+\frac{1}{2}}, (\nabla \times \boldsymbol{\chi}_h) \times \mathbf{u}_h^{n+\frac{1}{2}}) \\ + \nu(\nabla\overline{\mathbf{w}_h^{n+\frac{1}{2}}}, \nabla\boldsymbol{\chi}_h) = (\nabla \times \mathbf{f}^{n+\frac{1}{2}}, \boldsymbol{\chi}_h) - 2(D(\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\mathbf{u}_h^{n+\frac{1}{2}}, \boldsymbol{\chi}_h) \\ - \gamma\nu^{-1}((\nabla \times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*}) \times \mathbf{u}_h^{n+\frac{1}{2}}, (\nabla \times \boldsymbol{\chi}_h) \times \mathbf{u}_h^{n+\frac{1}{2}}) \\ - 2(D(\mathbf{w}_h^{n+\frac{1}{2}^*})\mathbf{u}_h^{n+\frac{1}{2}}, \boldsymbol{\chi}_h) - \nu(\nabla\mathbf{w}_h^{n+\frac{1}{2}^*}, \nabla\boldsymbol{\chi}_h) - \frac{1}{\Delta t}(\mathbf{w}_h^{n+1^*} - \mathbf{w}_h^{n^*}, \boldsymbol{\chi}_h). \quad (3.26)\end{aligned}$$

Let $\boldsymbol{\chi}_h = A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}$ and simplify to get

$$\begin{aligned}\frac{1}{\Delta t}(\overline{\mathbf{w}_h^{n+1}} - \overline{\mathbf{w}_h^n}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) + \gamma\nu^{-1}\left\|(\nabla \times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \times \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 + \nu\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \\ = (\nabla \times \mathbf{f}^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) - 2(D(\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\mathbf{u}_h^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \\ - 2(D(\mathbf{w}_h^{n+\frac{1}{2}^*})\mathbf{u}_h^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) - \nu(\nabla\mathbf{w}_h^{n+\frac{1}{2}^*}, \nabla A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \\ - \gamma\nu^{-1}((\nabla \times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*}) \times \mathbf{u}_h^{n+\frac{1}{2}}, (\nabla \times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \times \mathbf{u}_h^{n+\frac{1}{2}}) \\ - \frac{1}{\Delta t}(\mathbf{w}_h^{n+1^*} - \mathbf{w}_h^{n^*}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}). \quad (3.27)\end{aligned}$$

It is straightforward to show that A_h^{-1} is a symmetric operator on $\mathbf{V}_h$ and thus

$$\begin{aligned}\frac{1}{\Delta t}(\overline{\mathbf{w}_h^{n+1}} - \overline{\mathbf{w}_h^n}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) &= \frac{1}{2\Delta t}[(\overline{\mathbf{w}_h^{n+1}}, A_h^{-1}\overline{\mathbf{w}_h^{n+1}}) + (\overline{\mathbf{w}_h^{n+1}}, A_h^{-1}\overline{\mathbf{w}_h^n}) \\ &\quad -(\overline{\mathbf{w}_h^n}, A_h^{-1}\overline{\mathbf{w}_h^{n+1}}) - (\overline{\mathbf{w}_h^n}, A_h^{-1}\overline{\mathbf{w}_h^n})] \\ &= \frac{1}{2\Delta t}[(\overline{\mathbf{w}_h^{n+1}}, A_h^{-1}\overline{\mathbf{w}_h^{n+1}}) - (\overline{\mathbf{w}_h^n}, A_h^{-1}\overline{\mathbf{w}_h^n})] \\ &= \frac{1}{2\Delta t}(\left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^{n+1}}\right\|^2 - \left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^n}\right\|^2). \quad (3.28)\end{aligned}$$

Using (3.12) on the first RHS term of (3.27) yields

$$\begin{aligned}(\nabla \times \mathbf{f}^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) &= (\mathbf{f}^{n+\frac{1}{2}}, \nabla \times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) \\ &\leq C(\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{V^*} + h\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|)\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\| \quad (3.29) \\ &\leq C(\epsilon)\nu^{-1}(\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{V^*}^2 + h^2\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|^2) + \nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2.\end{aligned}$$

Using vector identities, integration by parts, and that $A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}$ is divergence free, on the first

trilinear term in (3.27) gives

$$
\begin{aligned}
&|2(D(\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\mathbf{u}_h^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})| \\
&\qquad = |(\nabla\times(\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\times\mathbf{u}_h^{n+\frac{1}{2}}) - \nabla(\mathbf{u}_h^{n+\frac{1}{2}}\cdot\mathbf{w}_h^{n+\frac{1}{2}}), A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})| \\
&\qquad = |(\nabla\times(\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\times\mathbf{u}_h^{n+\frac{1}{2}}), A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})| + |(\mathbf{u}_h^{n+\frac{1}{2}}\cdot\mathbf{w}_h^{n+\frac{1}{2}}, \nabla\cdot A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})| \\
&\qquad = |(\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\times\mathbf{u}_h^{n+\frac{1}{2}}, \nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})| \\
&\qquad = |(\overline{\mathbf{w}_h^{n+\frac{1}{2}}}, (\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}})| \qquad (3.30)\\
&\qquad \le \frac{\gamma^{-1}\nu}{2}\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 + \frac{\gamma\nu^{-1}}{2}\left\|(\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2.
\end{aligned}
$$

The part of the penalty term on the right-hand side of (3.27) is majorized as

$$
\begin{aligned}
&-\gamma\nu^{-1}((\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*})\times\mathbf{u}_h^{n+\frac{1}{2}}, (\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}) \qquad (3.31)\\
&\le \gamma\nu^{-1}\left\|(\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|\left\|(\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\| \\
&\le \gamma\nu^{-1}\left\|(\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 + \frac{\gamma\nu^{-1}}{4}\left\|(\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2.
\end{aligned}
$$

The first term on the right hand side of (3.31) can be bounded using Holder's inequality, (3.25) and Lemma 3.1.1:

$$
\begin{aligned}
\left\|(\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 &\le C\left\|\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*}\right\|_{L^3}^2\left\|\mathbf{u}_h^{n+\frac{1}{2}}\right\|_{L^6}^2 \\
&\le CC_0^2\left\|\mathbf{w}_h^{n+\frac{1}{2}^*}\right\|^2\left\|\nabla\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 \qquad (3.32)\\
&\le CC_0^2C_u^2\left\|\nabla\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2. \qquad (3.33)
\end{aligned}
$$

Substituting back into (3.31) we now have

$$
\begin{aligned}
&-\gamma\nu^{-1}((\nabla\times A_h^{-1}\mathbf{w}_h^{n+\frac{1}{2}^*})\times\mathbf{u}_h^{n+\frac{1}{2}}, (\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}) \\
&\le \gamma\nu^{-1}CC_0^2C_u^2\left\|\nabla\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 + \frac{\gamma\nu^{-1}}{4}\left\|(\nabla\times A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}})\times\mathbf{u}_h^{n+\frac{1}{2}}\right\|^2. \qquad (3.34)
\end{aligned}
$$

The second trilinear term in (3.27) can be bounded using Lemma 3.1.1 and 3.1.2 to obtain

$$\begin{aligned}
2(D(\mathbf{w}_h^{n+\frac{1}{2}*})\mathbf{u}_h^{n+\frac{1}{2}}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) &\leq C\left\|\nabla \mathbf{w}_h^{n+\frac{1}{2}*}\right\| \left\|\mathbf{u}_h^{n+\frac{1}{2}}\right\| \left\|A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|_{L^\infty} \\
&\leq C(\epsilon)C_4\nu^{-1}\left\|\nabla \mathbf{w}_h^{n+\frac{1}{2}*}\right\|^2 + \nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \qquad (3.35)\\
&\leq C(\epsilon)C_4C_u^2\nu^{-1} + \nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2. \qquad (3.36)
\end{aligned}$$

Using (3.14) gives

$$\begin{aligned}
\nu(\nabla \mathbf{w}_h^{n+\frac{1}{2}*}, \nabla A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) &\leq C\nu(\left\|\mathbf{w}_h^{n+\frac{1}{2}*}\right\| + \left\|\mathbf{w}_h^{n+\frac{1}{2}*}\right\|_{-\frac{1}{2},\partial\Omega} + h\left\|\nabla \mathbf{w}_h^{n+\frac{1}{2}*}\right\|)\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\| \\
&\leq C(\epsilon)\nu(\left\|\mathbf{w}_h^{n+\frac{1}{2}*}\right\|^2 + \left\|\mathbf{w}_h^{n+\frac{1}{2}*}\right\|^2_{-\frac{1}{2},\partial\Omega} + h^2\left\|\nabla \mathbf{w}_h^{n+\frac{1}{2}*}\right\|^2) \\
&\quad +\nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \\
&\leq C(\epsilon)C_u^2\nu + \nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2. \qquad (3.37)
\end{aligned}$$

Finally, Cauchy Schwarz, Young's inequality, and the definition of $\mathbf{w}_h^{n*}$ yield

$$\begin{aligned}
\frac{1}{\Delta t}(\mathbf{w}_h^{n+1*} - \mathbf{w}_h^{n*}, A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}) &\leq \left\|I_h(\nabla\times(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}))\right\|\left\|A_h^{-1}\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\| \\
&\leq C_0\left\|I_h(\nabla\times(\mathbf{u}_t(t^{n+1}) + \mathbf{u}_{tt}(t^*)))\right\|\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\| \qquad (3.38)\\
&\leq C_0C(\epsilon)\nu^{-1}(\left\|I_h(\nabla\times\mathbf{u}_t(t^{n+1}))\right\|^2 + \left\|I_h(\nabla\times\mathbf{u}_{tt}(t^*))\right\|^2) \\
&\quad +\nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \\
&\leq C_0C(\epsilon)\nu^{-1} + \nu\epsilon\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2.
\end{aligned}$$

Substitute into (3.27) using (3.28)-(3.38) to get

$$\begin{aligned}
\frac{1}{2\Delta t}\left(\left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^{n+1}}\right\|^2 - \left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^{n}}\right\|^2\right) + \nu\left(1 - \frac{1}{2\gamma} - 4\epsilon\right)\left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \\
\leq C\nu^{-1}\left(\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|^2_{V^*} + h^2\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|^2\right) + \gamma\nu^{-1}CC_0C_u^2\left\|\nabla \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2 \\
+ C(\epsilon)C_4\nu^{-1}C_u^2 + C(\epsilon)\nu C_u^2 + C_0C(\epsilon)\nu^{-1}. \qquad (3.39)
\end{aligned}$$

Choosing an arbitrarily small ϵ, the penalty parameter γ satisfying $(1 - \frac{1}{2\gamma} - 4\epsilon) > 0$, multiplying by $2\Delta t$, and summing from 0 to $M-1$ yield

$$\left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^M}\right\|^2 + \Delta t \sum_{n=0}^{M-1} \frac{1}{4}\nu \left\|\overline{\mathbf{w}_h^{n+\frac{1}{2}}}\right\|^2 \leq \nu^{-1}\Delta t C \sum_{n=0}^{M-1} (\left\|\mathbf{f}^{n+\frac{1}{2}}\right\|_{V^*}^2 + h^2 \left\|\mathbf{f}^{n+\frac{1}{2}}\right\|^2)$$
$$+ C(\nu^{-1}, C_4, C_u, \nu) + \left\|\nabla A_h^{-1}\overline{\mathbf{w}_h^0}\right\|^2 + \gamma\nu^{-2}CC_0C_u^2\Delta t \sum_{n=0}^{M-1} \nu \left\|\nabla \mathbf{u}_h^{n+\frac{1}{2}}\right\|^2. \quad (3.40)$$

Using the result for the velocity stability bound on the last sum of (3.40) finishes the proof. □

3.2 Numerical Results

We now present two numerical experiments to test the VVH method studied in this chapter. For all tests, we use $(\mathbf{P}_3, P_2^{disc})$ Scott-Vogelius elements, on barycenter-refined tetrahedral meshes. To solve the linear systems, we use the robust and efficient method proposed in [56] for this element choice. This is the lowest order element pair that is LBB stable on this mesh. The first experiment confirms expected convergence rates, and the second tests the method on 3D channel flow over a step.

All computations use $\gamma = 0$. In the computations, vorticity appears to be stable with this choice, and so it was not necessary to add this (costly) stabilization term. However, proving discrete stability of vorticity does not seem possible in this case, and so its use is believed to cover a gap in the analysis only.

3.2.1 Convergence Rates

Our first experiment is used to test convergence rates for the problem $\Omega = (0,1)^3$, where the true solution is given by

$$\mathbf{u}(x,y,z,t) = \begin{pmatrix} (1+.01t)\cos(2\pi z) \\ (1+.01t)\sin(2\pi z) \\ (1+.01t)\sin(2\pi x) \end{pmatrix} \quad (3.41)$$

For this problem we take $\nu = 1$, and initial condition $\mathbf{u}_h^0 = P_{V_h}(\mathbf{u}_0), \mathbf{w}_h^0 = P_{V_h}(\nabla \times \mathbf{u}_0)$. We compute with end time $T = 1$, and monitor error while decreasing the values of Δt with h. Uniform meshes

are used in the sense that each mesh divides Ω into equal size cubes, then divides each cube into six tetrahedra, and then performs a barycenter refinement of each tetrahedra. In the tables, h denotes the length of a side of a cube. For the velocity boundary condition, we use the nodal interpolant of the true solution on the boundary. For the vorticity boundary condition, we compute three different ways, all using a Dirichlet condition for discrete vorticity: using the nodal interpolant of the true vorticity, using the nodal interpolant of the L^2 projection of the curl of the discrete velocity into $\mathbf{V}_h$, and also using a simple local averaging of the curl of the discrete velocity.

The results are shown in Tables 3.1-3.3, respectively. With our choice of elements and a trapezoidal time discretization, optimal error is $O(\Delta t^2 + h^3)$, and since we tie together the spatial and temporal refinements by cutting Δt in third when h is cut in half, $O(h^3)$ is optimal. All three vorticity boundary conditions provide similar results: suboptimal rates are observed in the $L^2(0,T;\mathbf{H}^1(\Omega))$ norm until the last mesh refinement, when the rate jumps to around 3. We also see that for the velocity in the $L^2(0,T;\mathbf{L}^2(\Omega))$ norm we see optimal convergence rates, where as the vorticity in the $L^2(0,T;\mathbf{L}^2(\Omega))$ norm we do not seem to recover any L^2 lift. Here, while the errors observed using the (more practical) non-exact boundary conditions are expectably larger, the rates of convergence observed do not seem to decrease. A complete convergence theory for the method currently appears impenetrable without several assumptions not needed for usual NSE analysis, but progress on this front will likely lead to answers about boundary-dependence of convergence rates.

h	dof	Δt	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	10,218	1	3.8609e-2	-	8.6168e-2	-
1/4	78,462	1/3	2.3317e-3	4.0495	9.5332e-3	3.1761
1/6	261,474	1/6	4.6873e-4	3.9568	2.7359e-3	3.0787
1/8	615,990	1/9	1.5074e-4	3.9435	1.1360e-3	3.0533
1/10	1,198,746	1/18	5.9864e-5	4.1385	5.4716e-4	3.2738

h	Δt	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	1	4.4557e-1	-	2.1412	-
1/4	1/3	4.6819e-2	3.2505	4.4218e-1	2.2757
1/6	1/6	1.2533e-2	3.2504	1.7694e-1	2.2589
1/8	1/9	5.0199e-3	3.1804	9.4030e-2	2.1976
1/10	1/18	2.4978e-3	3.1280	4.8454e-2	2.9712

Table 3.1: Velocity and Vorticity errors and convergence rates using the nodal interpolant of the true vorticity for the vorticity boundary condition.

h	Δt	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	1/1	3.8609e-2	-	8.6168e-2	-
1/4	1/3	2.3317e-3	4.0495	9.5343e-3	3.1760
1/6	1/6	4.6873e-4	3.9568	2.7361e-3	3.0788
1/8	1/9	1.5074e-4	3.9435	1.1361e-3	3.0552
1/10	1/18	5.9864e-5	4.1385	5.4719e-4	3.2739
h	Δt	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	1/1	7.0753e-1	-	2.7086	-
1/4	1/3	7.3843e-2	3.2603	5.1060e-1	2.4073
1/6	1/6	1.9136e-2	3.3304	1.9562e-1	2.3662
1/8	1/9	7.4270e-3	3.2899	1.0169e-1	2.2742
1/10	1/18	3.4219e-3	3.4728	5.2398e-2	2.9715

Table 3.2: Velocity and Vorticity errors and convergence rates using the nodal interpolant of the L^2 projection of the curl of the discrete velocity into $\mathbf{V}_h$, for the vorticity boundary condition.

h	Δt	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	1/1	3.8609e-2	-	8.6168e-2	-
1/4	1/3	2.3317e-3	4.0495	9.5342e-3	3.1760
1/6	1/6	4.6873e-4	3.9568	2.7361e-3	3.0788
1/8	1/9	1.5074e-4	3.9435	1.1361e-3	3.0552
1/10	1/18	5.9864e-5	4.1385	5.4720e-4	3.2739
h	Δt	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	Rate	$\|\mathbf{w}-\mathbf{w}_h\|_{L^2(0,T;\mathbf{H}^1(\Omega))}$	Rate
1/2	1/1	6.6951e-1	-	2.6394	-
1/4	1/3	8.0245e-2	3.0606	5.0373e-1	2.3895
1/6	1/6	2.0989e-2	3.3075	1.9434e-1	2.3490
1/8	1/9	8.2120e-3	3.2619	1.0115e-1	2.2699
1/10	1/18	3.8818e-3	3.3579	5.2427e-2	2.9451

Table 3.3: Velocity and Vorticity errors and convergence rates using nodal averages of the curl of the discrete velocity for the vorticity boundary condition.

3.2.2 3D Channel Flow Over a Forward-Backward Facing Step

The next experiment tests the scheme on 3d flow over a forward-backward facing step, studied in [43, 15]. In the problem the channel is modeled by a $[0, 10] \times [0, 40] \times [0, 10]$ rectangular box, with a $10 \times 1 \times 1$ step on the bottom of the channel, beginning 5 units into the channel. A diagram of the flow domain is shown in Figure 3.1.

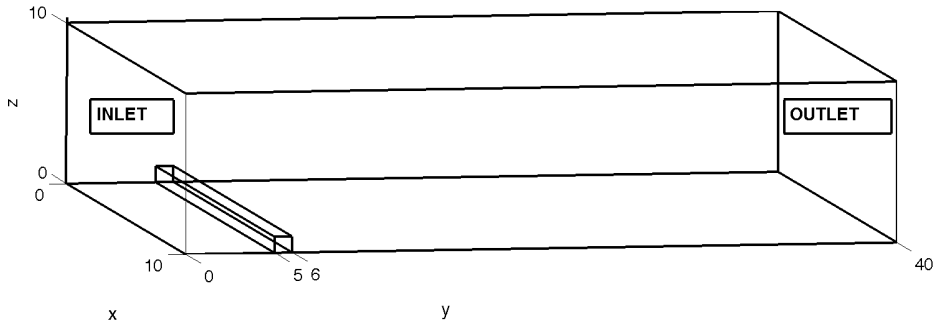


Figure 3.1: Flow domain for the 3d step test problem.

We compute to end-time $T = 10$, $\nu = \frac{1}{200}$, and $\Delta t = .025$. No-slip boundary conditions are used on the top, bottom, and sides of the channel, as well as on the step, and an inflow=outflow condition is employed for both . For the initial condition, we use the $Re = 20$ steady solution. Note this is consistent with [15] but in contrast to [43], where a constant inflow profile ($\mathbf{u}(x, 0, z) =< 0, 1, 0 >$) is used; such a boundary condition is non-physical, but also not usable in a method that solves for vorticity (since it will blow up as $h \to 0$ at the inflow edges). We compute the solution on a barycenter-refined tetrahedral mesh, which provides 1,282,920 total degrees of freedom. For the vorticity boundary condition on the walls and sides, we tried Dirichlet conditions that it be a nodal interpolant of the local average of the curl of the velocity, simply zero, and the projection of the curl of the velocity into $\mathbf{V}_h$. Only for the case of nodal averaging did we see the expected results, shown in Figure 3.2 as a speed contour plot of the sliceplane x=5 with overlaying streamlines, where eddies form behind the step and shed. Plots of vorticity magnitude and helical density are also provided. For the case of zero vorticity boundary condition latter, the simulation did not capture eddy detachment, and for the projection boundary condition, we saw instabilities occur and a bad solution resulted.

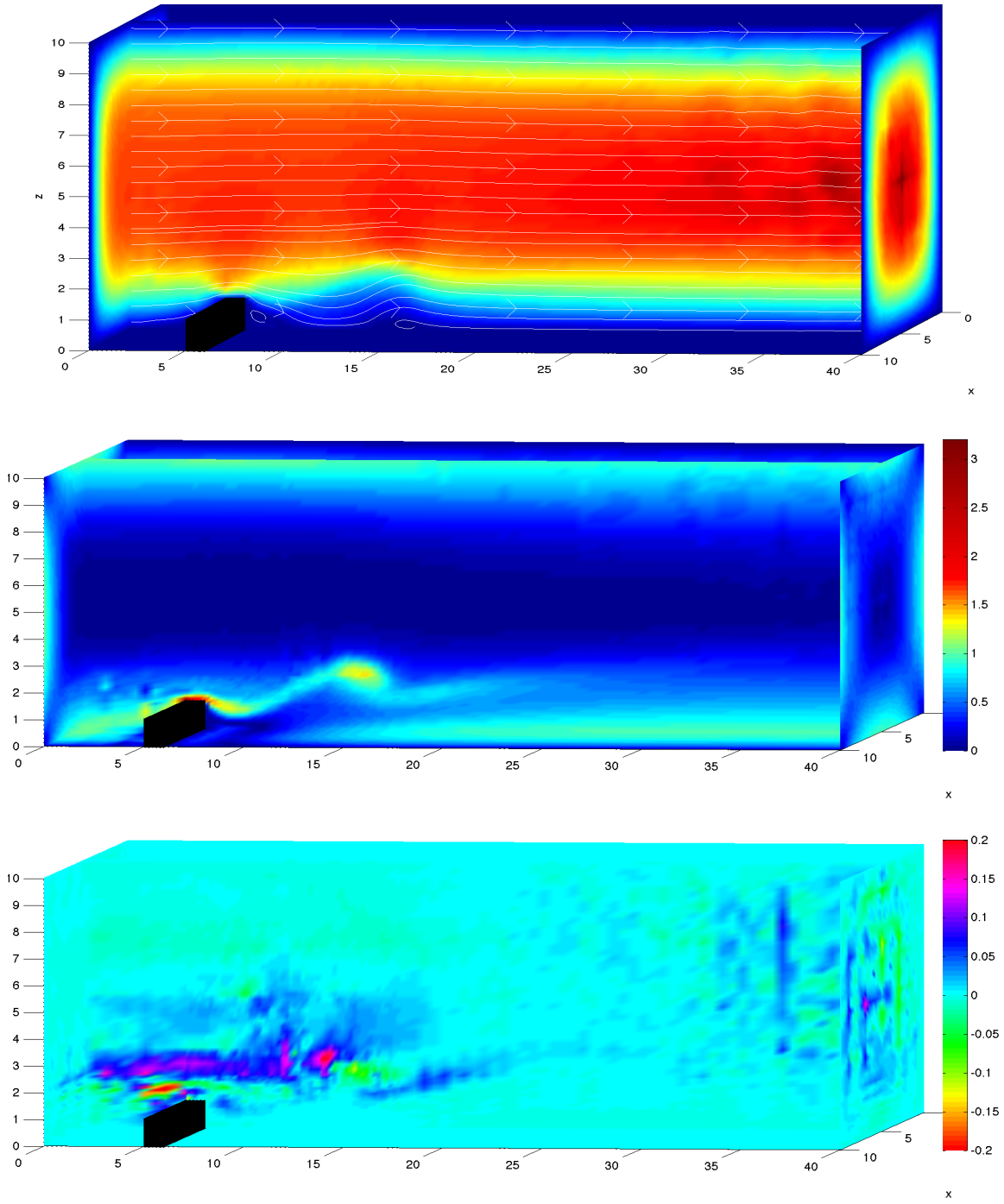


Figure 3.2: Shown above are (top) speed contours and streamlines, (middle) vorticity magnitude, and (bottom) helical density, from the fine mesh computation at time $t = 10$ at the $x = 5$ mid-slice-plane for the 3d step problem with nodal averaging vorticity boundary condition.

Chapter 4

Natural vorticity boundary conditions for coupled vorticity equations

This chapter derives new natural boundary conditions for the vorticity equations that result from the application of the curl operator to the steady NSE momentum equation, given by

$$-\nu\Delta\mathbf{w} + (\mathbf{u}\cdot\nabla)\mathbf{w} - (\mathbf{w}\cdot\nabla)\mathbf{u} = \nabla\times\mathbf{f}. \tag{4.1}$$

A finite element method for solving the 3d vorticity equations is presented to test the accuracy of the proposed boundary conditions, and results from a simple numerical experiment are presented verifying optimal convergence rates are acheived. We note that the vorticity boundary conditions presented herein could also easily be derived for the time-dependent vorticity equations, and would apply equally well to the vorticity-helical density equations studied in the previous chapter.

4.1 Derivation

Suppose we are given some general Dirichlet boundary condition for the velocity in the NSE, i.e. $\mathbf{u} = \mathbf{g}$ on $\partial\Omega$. We are mainly interested in the case where $\partial\Omega$ is a solid wall with no-slip ($\mathbf{g} = \mathbf{0}$)

boundary conditions, and so leaving $\mathbf{g}$ to be general includes this case. Our first vorticity boundary condition easily follows:

$$\mathbf{w} \cdot \mathbf{n} = (\nabla \times \mathbf{g}) \cdot \mathbf{n} \quad \text{on} \ \ \partial\Omega. \tag{4.2}$$

To deduce two more boundary conditions for $\mathbf{w}$, consider the incompressible NSE written in rotational form (see, e.g. [32] for more on rotational form of NSE),

$$\nu \nabla \times \mathbf{w} + \mathbf{w} \times \mathbf{u} + \nabla P = \mathbf{f},$$

where $P = \frac{1}{2}|\mathbf{u}|^2 + p$ is the Bernoulli pressure. Taking the tangential component of both sides of this equation gives

$$\nu(\nabla \times \mathbf{w}) \times \mathbf{n} = (\mathbf{f} - \nabla P - \mathbf{w} \times \mathbf{g}) \times \mathbf{n} \quad \text{on } \partial\Omega, \tag{4.3}$$

which provides two more boundary conditions for $\mathbf{w}$ in terms of the primitive NSE velocity and pressure variables. In velocity-vorticity splitting schemes where the NSE momentum equation is used for the velocity (as in the work of Wong and Baker [62] or the scheme presented in the previous chapter of this work), the NSE velocity and pressure are considered as knowns when solving the vorticity (or vorticity-helical density) equations.

We observe that the boundary condition (4.3) is the natural boundary condition for the following weak formulation of the vorticity equation: Find $\mathbf{w} \in \mathbf{H}_0(\text{div}) \cap \mathbf{H}(\text{curl})$ satisfying

$$\begin{aligned} &\nu(\nabla \times \mathbf{w}, \nabla \times \mathbf{v}) + \nu(\nabla \cdot \mathbf{w}, \nabla \cdot \mathbf{v}) + ((\mathbf{u} \cdot \nabla)\mathbf{w} - (\mathbf{w} \cdot \nabla)\mathbf{u}, \mathbf{v}) \\ &+ \oint_{\partial\Omega} ((\mathbf{w} \times \mathbf{g}) \times \mathbf{n}) \cdot \mathbf{v} \, d\mathbf{s} = (\nabla \times \mathbf{f}, \mathbf{v}) + \oint_{\partial\Omega} ((\mathbf{f} - \nabla P) \times \mathbf{n}) \cdot \mathbf{v} \, d\mathbf{s}, \end{aligned} \tag{4.4}$$

for any $\mathbf{v} \in \mathbf{H}_0(\text{div}) \cap \mathbf{H}(\text{curl})$.

To avoid computing pressure gradient over $\partial\Omega$ we rewrite the last term in (4.4) using integration by parts on $\partial\Omega$. To this end, we use the surface gradient and divergence, defined as:

$$\nabla_\Gamma p = \nabla p - (\mathbf{n} \cdot \nabla p)\mathbf{n}, \ \text{ and } \ \text{div}_\Gamma \mathbf{v} = \text{tr}(\nabla_\Gamma \mathbf{v}),$$

which are intrinsic surface quantities and do not depend on an extension of a scalar function p and a vector quantity $\mathbf{v}$ off a surface. We also need the following identity, proved in [34], for a smooth,

closed surface Γ:

$$\int_\Gamma p\,\mathrm{div}_\Gamma \mathbf{v} + \mathbf{v}\cdot\nabla_\Gamma p\,d\mathbf{s} = \int_\Gamma \kappa(\mathbf{v}\cdot\mathbf{n})p\,d\mathbf{s}, \tag{4.5}$$

where κ is the surface mean curvature.

From the definition of the surface gradient we immediately get the identity:

$$(\nabla P)\times\mathbf{n} = (\nabla_\Gamma P)\times\mathbf{n}.$$

Hence, with (4.5) we see

$$\begin{aligned}\oint_{\partial\Omega}((\nabla P)\times\mathbf{n})\cdot\mathbf{v}\,d\mathbf{s} = \oint_{\partial\Omega}((\nabla_\Gamma P)\times\mathbf{n})\cdot\mathbf{v}\,d\mathbf{s} &= \oint_{\partial\Omega}(\mathbf{v}\times\mathbf{n})\cdot\nabla_\Gamma P\,d\mathbf{s}\\ &= -\oint_{\partial\Omega} P\mathrm{div}_\Gamma(\mathbf{v}\times\mathbf{n})\,d\mathbf{s}.\end{aligned} \tag{4.6}$$

Finally, using $\mathbf{v}\cdot\mathbf{n}=0$ we have

$$\mathrm{div}_\Gamma(\mathbf{v}\times\mathbf{n}) = (\nabla\times\mathbf{v})\cdot\mathbf{n}.$$

Thus, the weak formulation of the vorticity equation now reads: Find $\mathbf{w}\in\mathbf{H}_0(\mathrm{div})\cap\mathbf{H}(\mathrm{curl})$ satisfying

$$\begin{aligned}&\nu(\nabla\times\mathbf{w},\nabla\times\mathbf{v}) + \nu(\nabla\cdot\mathbf{w},\nabla\cdot\mathbf{v}) + ((\mathbf{u}\cdot\nabla)\mathbf{w} - (\mathbf{w}\cdot\nabla)\mathbf{u},\mathbf{v})\\ &\qquad + \oint_{\partial\Omega}((\mathbf{w}\times\mathbf{g})\times\mathbf{n})\cdot\mathbf{v}\,d\mathbf{s} = (\nabla\times\mathbf{f},\mathbf{v}) + \oint_{\partial\Omega}(\mathbf{f}\times\mathbf{n})\cdot\mathbf{v}\,d\mathbf{s} + \oint_{\partial\Omega}P(\nabla\times\mathbf{v})\cdot\mathbf{n}\,d\mathbf{s},\end{aligned} \tag{4.7}$$

for any $\mathbf{v}\in\mathbf{H}_0(\mathrm{div})\cap\mathbf{H}(\mathrm{curl})$. In the case of no slip boundary conditions ($\mathbf{g}=\mathbf{0}$), the system reduces to

$$\begin{aligned}&\nu(\nabla\times\mathbf{w},\nabla\times\mathbf{v}) + \nu(\nabla\cdot\mathbf{w},\nabla\cdot\mathbf{v}) + ((\mathbf{u}\cdot\nabla)\mathbf{w} - (\mathbf{w}\cdot\nabla)\mathbf{u},\mathbf{v})\\ &\qquad = (\nabla\times\mathbf{f},\mathbf{v}) + \oint_{\partial\Omega}(\mathbf{f}\times\mathbf{n})\cdot\mathbf{v}\,d\mathbf{s} + \oint_{\partial\Omega}p(\nabla\times\mathbf{v})\cdot\mathbf{n}\,d\mathbf{s}.\end{aligned} \tag{4.8}$$

4.2 Numerical Results

In this section we consider a basic 3d numerical test designed to evaluate the accuracy of a numerical scheme implementing our new vorticity boundary conditions. Let I denote some

interpolation operator on $\partial\Omega$. The scheme we propose to compute with is given in two steps:
Step 1:

$$\begin{aligned}\nu(\nabla\mathbf{u}_h,\nabla\mathbf{v}_h)+(\mathbf{u}_h\cdot\nabla\mathbf{u}_h,\mathbf{v}_h)-(p_h,\nabla\cdot\mathbf{v}_h)&=(\mathbf{f},\mathbf{v}_h)\\(\nabla\cdot\mathbf{u}_h,q_h)&=0\\\mathbf{u}_h|_{\partial\Omega}&=I(\mathbf{g})\end{aligned}$$

Step 2:

$$\begin{aligned}\nu(\nabla\times\mathbf{w}_h,\nabla\times\mathbf{v}_h)+\nu(\nabla\cdot\mathbf{w}_h,\nabla\cdot\mathbf{v}_h)+((\mathbf{u}_h\cdot\nabla)\mathbf{w}_h-(\mathbf{w}_h\cdot\nabla)\mathbf{u}_h,\mathbf{v}_h)+\int_{\partial\Omega}((\mathbf{w}_h\times\mathbf{g})\times\mathbf{n})\cdot\mathbf{v}_h\,d\mathbf{s}&\\=(\nabla\times\mathbf{f},\mathbf{v}_h)+\int_{\partial\Omega}(\mathbf{f}\times\mathbf{n})\cdot\mathbf{v}_h\,d\mathbf{s}+\int_{\partial\Omega}p(\nabla\times\mathbf{v}_h)\cdot\mathbf{n}\,d\mathbf{s}&\\(\mathbf{w}_h-I(\nabla\times\mathbf{g}))\cdot\mathbf{n}|_{\partial\Omega}=\mathbf{0}.&\end{aligned}$$

Our 3d numerical experiment is designed to test convergence rates for the problem $\Omega=(0,1)^3$, where the true, steady NSE solution is given by

$$\begin{aligned}u_1(x,y,z)&=\sin(2\pi y)\\u_2(x,y,z)&=\cos(2\pi z)\\u_3(x,y,z)&=e^x\\p(x,y,z)&=\sin(2\pi x)+\cos(2\pi y)+\sin(2\pi z),\end{aligned}$$

where p denotes the standard NSE pressure. Because we are given nonhomogenous boundary conditions for the velocity, if we want to enforce vorticity boundary conditions only with boundary integrals, we must include the left hand side term $\int_{\partial\Omega}((\mathbf{w}_h\times\mathbf{g})\times\mathbf{n})\cdot\mathbf{v}_h\,ds$ in the vorticity equation. We compute with Reynolds number $Re=1$ and force field

$$\mathbf{f}=\mathbf{u}\cdot\nabla\mathbf{u}+\nabla p-\nu\Delta\mathbf{u}$$

as determined by the true NSE solution. Velocity and vorticity approximations are computed using $\mathbf{Q}_2$ elements, while Q_1 elements are used for the pressure. All computations were performed using the software *deal.II* [6, 7].

For our choice of finite element spaces optimal convergence in the $H^1(\Omega)$ norm is $O(h^2)$, and hence we compute on a series of uniform hexahedral meshes, each of which is one uniform refinement of the previous mesh (i.e. h is cut in half with each successive mesh). Velocity and vorticity errors are shown in Tables 4.1 and 4.2. Both velocity and vorticity errors are optimal in the energy norm $H^1(\Omega)$.

h	$\|\mathbf{u}-\mathbf{u}_h\|_0$	rate	$\|\mathbf{u}-\mathbf{u}_h\|_1$	rate
1/2	1.31726E-1		2.08727	
1/4	1.80537E-2	2.87	5.66015E-1	1.97
1/8	2.3043E-3	2.97	1.43775E-1	1.98
1/16	2.90438E-4	2.99	3.60712E-2	1.99

Table 4.1: Velocity errors and convergence rates for the first 3d numerical experiment

h	$\|\mathbf{w}-\mathbf{w}_h\|_0$	rate	$\|\mathbf{w}-\mathbf{w}_h\|_1$	rate
1/2	1.08737		14.2486	
1/4	1.59229E-1	2.77	3.79774	1.91
1/8	2.61973E-2	2.60	9.53532E-1	1.99
1/16	5.43578E-3	2.27	2.37986E-1	2.00

Table 4.2: Vorticity errors and convergence rates for the first 3d numerical experiment

Chapter 5

A New Reduced Order Multiscale Deconvolution Model

This chapter proposes a new reduced order multiscale deconvolution model (MDM). This model leads to a natural, efficient numerical scheme that is both unconditionally stable and optimally accurate. Numerical tests are provided that confirm the effectiveness of the scheme.

5.1 Derivation

Recall the NSE, given by

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nu \Delta \mathbf{u} = \mathbf{f} \tag{5.1}$$

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

In this chapter we will only consider homogenous Dirichlet boundary conditions for the velocity $\mathbf{u}$. Our model formulation will use two different incompressible Helmholtz filters, the first of which is

defined by

$$-\alpha^2\Delta\overline{\mathbf{u}} + \alpha^2\nabla\lambda + \overline{\mathbf{u}} = \mathbf{u}, \tag{5.3}$$

$$\nabla\cdot\overline{\mathbf{u}} = 0, \tag{5.4}$$

$$(\mathbf{u} - \overline{\mathbf{u}})|_{\partial\Omega} = 0, \tag{5.5}$$

where α denotes the length scale associated with the filtered velocity $\overline{\mathbf{u}}$. For convenience, we will denote the solution operator to (5.3)-(5.5) by $F_\alpha(\cdot)$, i.e. $F_\alpha\mathbf{u} = \overline{\mathbf{u}}$. The second incompressible Helmholtz filter is given by

$$-\gamma^2\Delta\widetilde{\mathbf{u}} + \gamma^2\nabla\rho + \widetilde{\mathbf{u}} = \mathbf{u}, \tag{5.6}$$

$$\nabla\cdot\widetilde{\mathbf{u}} = 0, \tag{5.7}$$

$$(\mathbf{u} - \widetilde{\mathbf{u}})|_{\partial\Omega} = 0, \tag{5.8}$$

where γ is a second, intermediate length scale associated with the filtered velocity $\widetilde{\mathbf{u}}$, satisfying $0 < \gamma \leq \alpha$. In practice, α is generally chosen to be the size of the smallest flow structures to be resolved, and γ is a parameter determining the modeling error relative to the NSE. In [22] it was shown that $\widetilde{\mathbf{u}}$ has an explicit form depending on $\overline{\mathbf{u}}$ given by

$$\widetilde{\mathbf{u}} = \frac{\alpha^2}{\gamma^2}\overline{\mathbf{u}} - (1 - \frac{\alpha^2}{\gamma^2})\tilde{\overline{\mathbf{u}}},$$

so we will define our multiscale, approximate deconvolution operator G_γ as

$$G_\gamma\mathbf{u} := \frac{\alpha^2}{\gamma^2}\mathbf{u} + (1 - \frac{\alpha^2}{\gamma^2})\widetilde{\mathbf{u}} = \frac{\alpha^2}{\gamma^2}\mathbf{u} - \frac{\alpha^2 - \gamma^2}{\gamma^2}\widetilde{\mathbf{u}}. \tag{5.9}$$

Using this definition we immediately see that we can write the intermediate filtered quantity $\widetilde{\boldsymbol{\phi}}$ explicitly as

$$\mathbf{u} \approx \widetilde{\mathbf{u}} = G_\gamma\overline{\mathbf{u}}.$$

Because F_α is invertible we can write $\mathbf{u} = {F_\alpha}^{-1}\overline{\mathbf{u}}$, and using the definition of the α-filter (5.3) on the time derivative term, the NSE momentum equation (1.1) becomes

$$(-\alpha^2\Delta\overline{\mathbf{u}}_t + \alpha^2\nabla\lambda_t + \overline{\mathbf{u}}_t) + {F_\alpha}^{-1}\overline{\mathbf{u}}\cdot\nabla({F_\alpha}^{-1}\overline{\mathbf{u}}) + \nabla p - \nu\Delta({F_\alpha}^{-1}\overline{\mathbf{u}}) = \mathbf{f}. \tag{5.10}$$

It was shown in [22] that we have the following local error estimate for the accuracy in approximating the operator ${F_\alpha}^{-1}$ by G_γ:

$$\|\mathbf{u} - G_\gamma\overline{\mathbf{u}}\|_{H^k(\Omega)} = \|\mathbf{u} - \widetilde{\mathbf{u}}\|_{H^k(\Omega)} \leq \gamma^2\|\mathbf{u}\|_{H^{k+2}(\Omega)}.$$

This estimate suggests that multiscale, approximate deconvolution can be used to approximate the inverse of the α-filter (i.e. ${F_\alpha}^{-1}\phi \approx G_\gamma\phi$). Thus, denoting $\mathbf{v} := \overline{\mathbf{u}}$ and $q := p + \alpha^2\lambda_t$ in (5.10), we arrive at

$$\mathbf{v}_t - \alpha^2\Delta\mathbf{v}_t + G_\gamma\mathbf{v}\cdot\nabla G_\gamma\mathbf{v} + \nabla q - \nu\Delta G_\gamma\mathbf{v} = \mathbf{f}.$$

Employing the same multiscale, approximate deconvolution approximation in the conservation of mass equation (5.2) yields $\nabla\cdot G_\gamma\overline{\mathbf{u}} = \nabla\cdot G_\gamma\mathbf{v} = 0$. Note that by (5.7) and the definition of the deconvolution operator G_γ, the new conservation of mass constraint is satisfied via enforcing $\nabla\cdot\mathbf{v} = 0$. The reduced order multiscale deconvolution model (RMDM) with incompressible filters for homogenous Dirichlet boundary conditions is then

$$\mathbf{v}_t - \alpha^2\Delta\mathbf{v}_t + G_\gamma\mathbf{v}\cdot\nabla G_\gamma\mathbf{v} + \nabla q - \nu\Delta G_\gamma\mathbf{v} = \mathbf{f} \tag{5.11}$$

$$\nabla\cdot\mathbf{v} = 0 \tag{5.12}$$

$$-\gamma^2\Delta\widetilde{\mathbf{v}} + \gamma^2\nabla\rho + \widetilde{\mathbf{v}} = \mathbf{v} \tag{5.13}$$

$$\nabla\cdot\widetilde{\mathbf{v}} = 0 \tag{5.14}$$

$$(\mathbf{v} - \widetilde{\mathbf{v}})|_{\partial\Omega} = 0. \tag{5.15}$$

5.2 The Discrete Setting

Define finite dimensional spaces $\mathbf{X}_h \subset \mathbf{X}$ and $Q_h \subset Q$ to be the Scott-Vogelius (SV) mixed finite element pair $(\mathbf{X}_h, Q_h) := (\mathbf{P}_k(\tau_h), P_{k-1}^{disc}(\tau_h))$.

The following discrete filter is defined analogously to its continuous counterpart by taking

its variational formulation and restricting to finite dimensional spaces.

Definition 5.2.1. *Given $\boldsymbol{\phi} \in L^2(\Omega)$, define $\widetilde{\boldsymbol{\phi}}^h$ to be the solution of the problem: Find $(\widetilde{\boldsymbol{\phi}}^h, \rho_h) \in (X_h, Q_h)$ satisfying*

$$\gamma^2(\nabla\widetilde{\boldsymbol{\phi}}^h, \nabla\boldsymbol{\chi}_h) + (\widetilde{\boldsymbol{\phi}}^h, \boldsymbol{\chi}_h) - (\rho_h, \nabla\cdot\boldsymbol{\chi}_h) = (\boldsymbol{\phi}, \boldsymbol{\chi}_h) \,\forall\, \boldsymbol{\chi}_h \in \mathbf{X}_h, \tag{5.16}$$

$$(\nabla\cdot\widetilde{\boldsymbol{\phi}}^h, q_h) = 0 \,\forall\, q_h \in Q_h. \tag{5.17}$$

We will denote the solution operator of this discrete filter by F_h, i.e. $F_h\boldsymbol{\phi} := \widetilde{\boldsymbol{\phi}}^h$. The equivalent discretely divergence-free representation of the filter is: Given $\boldsymbol{\phi} \in L^2(\Omega)$, find $\widetilde{\boldsymbol{\phi}}^h \in V_h$ satisfying

$$\gamma^2(\nabla\widetilde{\boldsymbol{\phi}}^h, \nabla\boldsymbol{\chi}_h) + (\widetilde{\boldsymbol{\phi}}^h, \boldsymbol{\chi}_h) = (\boldsymbol{\phi}, \boldsymbol{\chi}_h) \,\forall\, \boldsymbol{\chi}_h \in \mathbf{V}_h.$$

The following lemma, found in [58], contains useful bounds for discretely filtered functions.

Lemma 5.2.2. *For $\boldsymbol{\phi} \in \mathbf{L}^2(\Omega)$,*

$$\left\|\widetilde{\boldsymbol{\phi}}^h\right\| \le \|\boldsymbol{\phi}\|.$$

For $\boldsymbol{\phi} \in \mathbf{X}$, there exists a constant C dependent on the size of Ω such that

$$\left\|\nabla\widetilde{\boldsymbol{\phi}}^h\right\| \le C\,\|\nabla\boldsymbol{\phi}\|.$$

For $\boldsymbol{\phi} \in \mathbf{V}_h$,

$$\left\|\nabla\widetilde{\boldsymbol{\phi}}^h\right\| \le \|\nabla\boldsymbol{\phi}\|.$$

The next lemma provides a bound on the difference between continuously filtered and discretely filtered functions.

Lemma 5.2.3. *For $\boldsymbol{\phi} \in \mathbf{H}^k(\Omega) \cap \mathbf{V}$ we have the bound*

$$\left\|\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h\right\|^2 + \gamma^2\left\|\nabla(\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h)\right\|^2 \le C(\gamma^{-2}h^{2k+2} + h^{2k})|\boldsymbol{\phi}|_{k+1}^2. \tag{5.18}$$

For $k \le 2$, we have the improved bound

$$\left\|\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h\right\|^2 + \gamma^2\left\|\nabla(\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h)\right\|^2 \le C(h^{2k+2} + \gamma^2h^{2k})|\boldsymbol{\phi}|_{k+1}^2. \tag{5.19}$$

Proof. Multiplying the γ-filter equation (5.6) by arbitrary $\boldsymbol{\chi}_h \in \mathbf{V}_h$ and integrating over Ω yields

$$\gamma^2(\nabla\widetilde{\boldsymbol{\phi}}, \nabla\boldsymbol{\chi}_h) + (\widetilde{\boldsymbol{\phi}}, \boldsymbol{\chi}_h) = (\boldsymbol{\phi}, \boldsymbol{\chi}_h).$$

Subtracting the discrete γ-filter equation (5.16) and denoting $\mathbf{e} = \widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h$ gives, for any $\boldsymbol{\chi}_h \in \mathbf{V}_h$,

$$\gamma^2(\nabla\mathbf{e}, \nabla\boldsymbol{\chi}_h) + (\mathbf{e}, \boldsymbol{\chi}_h) = 0.$$

Standard finite element analysis and interpolation estimates produce the bound

$$\left\|\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h\right\|^2 + \gamma^2 \left\|\nabla(\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h)\right\|^2 \leq C(h^{2k+2} + \gamma^2 h^{2k})|\widetilde{\boldsymbol{\phi}}|^2_{k+1}.$$

In [47] it was shown that for $k \geq 3$ we have the estimate $\gamma^2|\widetilde{\boldsymbol{\phi}}|^2_{k+1} \leq C|\boldsymbol{\phi}|^2_{k+1}$, and hence we get the bound

$$\left\|\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h\right\|^2 + \gamma^2 \left\|\nabla(\widetilde{\boldsymbol{\phi}} - \widetilde{\boldsymbol{\phi}}^h)\right\|^2 \leq C(\gamma^{-2}h^{2k+2} + h^{2k})|\boldsymbol{\phi}|^2_{k+1}.$$

Additionally, for $k = 0, 1, 2$ it was shown in [47] that we have the improved estimate $|\widetilde{\boldsymbol{\phi}}|_k \leq |\boldsymbol{\phi}|_k$, which finishes the proof. □

5.2.1 An Unconditionally Stable Algorithm for the RMDM

The fully discrete algorithm we study for the RMDM is backward Euler in time and finite element in space.

Algorithm 5.2.4. *Given two filtering radii $\alpha \geq \gamma > 0$, initial velocity $\mathbf{v}_h^0 \in \mathbf{V}_h$, a forcing function $\mathbf{f} \in L^\infty(0, T; \mathbf{H}^{-1}(\Omega))$, end time $T > 0$, and timestep $\Delta t > 0$, set $M = \frac{T}{\Delta t}$ and compute for $n = 1, 2, ..., M-1$, and for all $\boldsymbol{\chi}_h \in \mathbf{X}_h$ and $r_h \in Q_h$,*

$$\frac{\alpha^2}{\Delta t}(\nabla\mathbf{v}_h^{n+1} - \nabla\mathbf{v}_h^n, \nabla\boldsymbol{\chi}_h) + \frac{1}{\Delta t}(\mathbf{v}_h^{n+1} - \mathbf{v}_h^n, \boldsymbol{\chi}_h) + \nu(\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h), \nabla\boldsymbol{\chi}_h)$$
$$-(q_h^{n+1}, \nabla\cdot\boldsymbol{\chi}_h) + b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^n - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h, \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h, \boldsymbol{\chi}_h\right) = (\mathbf{f}^{n+1}, \boldsymbol{\chi}_h), \quad (5.20)$$

$$(\nabla\cdot\mathbf{v}_h^{n+1}, r_h) = 0. \quad (5.21)$$

In the following stability analysis we will assume that there exists a const $C_0 \geq 1$ such that

$$\alpha = C_0 \gamma, \tag{5.22}$$

i.e. the coarse mesh and fine mesh filtering radii will always be tied together by the constant C_0.

Lemma 5.2.5 (Stability)**.** *Solutions to Algorithm 5.2.4 satisfy*

$$\begin{aligned}
&\alpha^2 \left\|\nabla \mathbf{v}_h^M\right\|^2 + \left\|\mathbf{v}_h^M\right\|^2 + \nu \Delta t \sum_{n=0}^{M-1} \left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 \\
&\qquad \leq (2\nu\Delta t(C_0^2 - 1) + C_0^2\alpha^2) \left\|\nabla \mathbf{v}_h^0\right\|^2 + C_0^2 \left\|\mathbf{v}_h^0\right\|^2 + \nu^{-1}\Delta t \sum_{n=0}^{M-1} \left\|\mathbf{f}^{n+1}\right\|_{-1}^2 \leq C\nu^{-1},
\end{aligned} \tag{5.23}$$

where C depends on data but can be considered independent of $\alpha, \gamma, \Delta t, \nu$, and h.

Proof. Choosing $\boldsymbol{\chi}_h = \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h$ in (5.20) immediately eliminates both the nonlinear and pressure terms, and leaves

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t}\left(\nabla \mathbf{v}_h^{n+1} - \nabla \mathbf{v}_h^n, \nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right) + \frac{1}{\Delta t}\left(\mathbf{v}_h^{n+1} - \mathbf{v}_h^n, \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h\right) \\
&\qquad + \nu \left\|\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right\|^2 = (\mathbf{f}^{n+1}, \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h).
\end{aligned} \tag{5.24}$$

We begin by bounding the right hand side term of (5.24) in the usual manner

$$(\mathbf{f}^{n+1}, \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h) \leq \frac{1}{2\nu}\left\|\mathbf{f}^{n+1}\right\|_{-1}^2 + \frac{\nu}{2}\left\|\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right\|^2. \tag{5.25}$$

Decomposing the first term in (5.24) produces

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t}\left(\nabla \mathbf{v}_h^{n+1} - \nabla \mathbf{v}_h^n, \nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right) \\
&\qquad = \frac{\alpha^4}{\gamma^2\Delta t}(\nabla \mathbf{v}_h^{n+1} - \nabla \mathbf{v}_h^n, \nabla \mathbf{v}_h^{n+1}) - \frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2\Delta t}(\nabla \mathbf{v}_h^{n+1} - \nabla \mathbf{v}_h^n, \nabla \widetilde{\mathbf{v}_h^n}^h).
\end{aligned} \tag{5.26}$$

Using Cauchy Schwarz and Young's inequalities and rearranging terms yields

$$\frac{\alpha^2}{\Delta t}\left(\nabla \mathbf{v}_h^{n+1}-\nabla \mathbf{v}_h^n, \nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right) \geq \frac{\alpha^4}{2\gamma^2\Delta t}(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2-\|\nabla \mathbf{v}_h^n\|^2) - \frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2\Delta t}(\nabla \mathbf{v}_h^{n+1}, \nabla\widetilde{\mathbf{v}_h^n}^h)+\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2\Delta t}(\nabla \mathbf{v}_h^n, \nabla\widetilde{\mathbf{v}_h^n}^h). \tag{5.27}$$

Because the operator F_h is both self-adjoint and positive in the L^2 inner product in $\mathbf{V}_h$, we then get the lower bound

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t}\left(\nabla \mathbf{v}_h^{n+1}-\nabla \mathbf{v}_h^n, \nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right)\\
&\geq \frac{\alpha^4}{2\gamma^2\Delta t}(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2-\|\nabla \mathbf{v}_h^n\|^2)-\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2\Delta t}(\nabla F_h^{1/2}\mathbf{v}_h^{n+1}, \nabla F_h^{1/2}\mathbf{v}_h^n)+\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2\Delta t}\left\|\nabla F_h^{\frac{1}{2}}\mathbf{v}_h^n\right\|^2\\
&\geq \frac{\alpha^4}{2\gamma^2\Delta t}(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2-\|\nabla \mathbf{v}_h^n\|^2)+\frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2\Delta t}\left(\left\|\nabla F_h^{1/2}\mathbf{v}_h^n\right\|^2-\left\|\nabla F_h^{1/2}\mathbf{v}_h^{n+1}\right\|^2\right).
\end{aligned} \tag{5.28}$$

Using the same techniques we can produce the following lower bound for the second term in (5.24)

$$\begin{aligned}
&\frac{1}{\Delta t}\left(\mathbf{v}_h^{n+1}-\mathbf{v}_h^n, \frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h\right)\\
&\qquad\geq \frac{\alpha^2}{2\gamma^2\Delta t}(\left\|\mathbf{v}_h^{n+1}\right\|^2-\|\mathbf{v}_h^n\|^2)-\frac{\alpha^2-\gamma^2}{2\gamma^2\Delta t}\left(\left\|F_h^{1/2}\mathbf{v}_h^n\right\|^2-\left\|F_h^{1/2}\mathbf{v}_h^{n+1}\right\|^2\right).
\end{aligned} \tag{5.29}$$

Expanding the viscous term in (5.24) gives

$$\begin{aligned}
&\nu\left\|\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right\|^2\\
&\qquad=\frac{\nu\alpha^4}{\gamma^4}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2+\frac{\nu(\alpha^2-\gamma^2)^2}{\gamma^4}\left\|\nabla\widetilde{\mathbf{v}_h^n}^h\right\|^2-\frac{2\nu\alpha^2(\alpha^2-\gamma^2)}{\gamma^4}(\nabla \mathbf{v}_h^{n+1}, \nabla\widetilde{\mathbf{v}_h^n}^h).
\end{aligned} \tag{5.30}$$

Using the Cauchy-Schwarz inequality yields

$$\begin{aligned}
\nu\left\|\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right\|^2 \geq\ &\frac{\nu\alpha^4}{\gamma^4}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2+\frac{\nu(\alpha^2-\gamma^2)^2}{\gamma^4}\left\|\nabla\widetilde{\mathbf{v}_h^n}^h\right\|^2\\
&-\frac{2\nu\alpha^2(\alpha^2-\gamma^2)}{\gamma^4}\left(\frac{1}{2}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2+\frac{1}{2}\left\|\nabla\widetilde{\mathbf{v}_h^n}^h\right\|^2\right),
\end{aligned} \tag{5.31}$$

which, after simplifying, leads to the lower bound

$$\nu\left\|\nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h)\right\|^2 \geq \frac{\nu\alpha^2}{\gamma^2}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2-\frac{\nu(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla\widetilde{\mathbf{v}_h^n}^h\right\|^2. \tag{5.32}$$

Using Lemma 5.2.2 then gives

$$\nu \left\| \nabla(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h) \right\|^2 \geq \frac{\nu\alpha^2}{\gamma^2}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 - \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla \mathbf{v}_h^n\right\|^2$$
$$= \nu\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 + \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2}\left(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 - \left\|\nabla \mathbf{v}_h^n\right\|^2\right). \quad (5.33)$$

Applying the bounds in (5.25)-(5.33) to (5.24) yields

$$\frac{\alpha^4}{2\Delta t\gamma^2}(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 - \left\|\nabla \mathbf{v}_h^n\right\|^2) + \frac{\alpha^2(\alpha^2-\gamma^2)}{2\Delta t\gamma^2}\left(\left\|\nabla F_h^{1/2}\mathbf{v}_h^n\right\|^2 - \left\|\nabla F_h^{1/2}\mathbf{v}_h^{n+1}\right\|^2\right)$$
$$+ \frac{\alpha^2}{2\Delta t\gamma^2}(\left\|\mathbf{v}_h^{n+1}\right\|^2 - \left\|\mathbf{v}_h^n\right\|^2) + \frac{\alpha^2-\gamma^2}{2\Delta t\gamma^2}\left(\left\|F_h^{1/2}\mathbf{v}_h^n\right\|^2 - \left\|F_h^{1/2}\mathbf{v}_h^{n+1}\right\|^2\right) + \nu\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2$$
$$+ \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2}(\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 - \left\|\nabla \mathbf{v}_h^n\right\|^2) \leq \frac{1}{2}\nu^{-1}\left\|\mathbf{f}^{n+1}\right\|_{-1}^2. \quad (5.34)$$

Summing from $n=0$ to $M-1$ and multiplying by Δt gives

$$\left(\frac{\alpha^4}{2\gamma^2}\left\|\nabla \mathbf{v}_h^M\right\|^2 - \frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2}\left\|\nabla F_h^{1/2}\mathbf{v}_h^M\right\|^2\right) + \left(\frac{\alpha^2}{2\gamma^2}\left\|\mathbf{v}_h^M\right\|^2 - \frac{\alpha^2-\gamma^2}{2\gamma^2}\left\|F_h^{1/2}\mathbf{v}_h^M\right\|^2\right)$$
$$+ \frac{\nu\Delta t(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla \mathbf{v}_h^M\right\|^2 + \nu\Delta t\sum_{n=0}^{M-1}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2 \leq \frac{\nu\Delta t(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla \mathbf{v}_h^0\right\|^2 + \Delta t\nu^{-1}\sum_{n=0}^{M-1}\left\|\mathbf{f}^{n+1}\right\|_{-1}^2$$
$$+ \left(\frac{\alpha^4}{2\gamma^2}\left\|\nabla \mathbf{v}_h^0\right\|^2 - \frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2}\left\|\nabla F_h^{1/2}\mathbf{v}_h^0\right\|^2\right) + \left(\frac{\alpha^2}{2\gamma^2}\left\|\mathbf{v}_h^0\right\|^2 - \frac{\alpha^2-\gamma^2}{2\gamma^2}\left\|F_h^{1/2}\mathbf{v}_h^0\right\|^2\right). \quad (5.35)$$

Using the Cauchy Schwarz inequality and Lemma 5.2.2 we see that $\left\|F_h^{1/2}\boldsymbol{\phi}\right\|^2 = (\boldsymbol{\phi}, \widetilde{\boldsymbol{\phi}}) \leq \left\|\boldsymbol{\phi}\right\|^2$, and similarly $\left\|\nabla F_h^{1/2}\boldsymbol{\phi}\right\|^2 \leq \left\|\nabla\boldsymbol{\phi}\right\|^2$. Therefore, multiplying by 2 and reducing, we see

$$\alpha^2\left\|\nabla \mathbf{v}_h^M\right\|^2 + \frac{\nu\Delta t(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla \mathbf{v}_h^M\right\|^2 + \left\|\mathbf{v}_h^M\right\|^2 + \nu\Delta t\sum_{n=0}^{M-1}\left\|\nabla \mathbf{v}_h^{n+1}\right\|^2$$
$$\leq \frac{2\nu\Delta t(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla \mathbf{v}_h^0\right\|^2 + \frac{\alpha^4}{\gamma^2}\left\|\nabla \mathbf{v}_h^0\right\|^2 + \frac{\alpha^2}{\gamma^2}\left\|\mathbf{v}_h^0\right\|^2 + \Delta t\nu^{-1}\sum_{n=0}^{M-1}\left\|\mathbf{f}^{n+1}\right\|_{-1}^2. \quad (5.36)$$

Substituting $\gamma = \frac{1}{C_0}\alpha$ and simplifying finishes the proof. □

Remark 5.2.6. *With the current analysis, the assumption $\alpha = C_0\gamma$ is necessary because of the presence of γ^{-2} on the right side of estimate* (5.36).

5.3 Error Analysis

We now prove convergence of Algorithm 5.2.4 to solutions of the model (5.11)-(5.15).

Theorem 5.3.1 (Convergence Estimate)**.** *Let $\alpha \geq \gamma > 0$ denote two fixed filtering radii, and assume v to be the model solution to* (5.11)-(5.15) *satisfying the given problem data of Algorithm 5.2.4. If we assume the model solution satisfies the smoothness conditions*

$$\mathbf{v} \in L^\infty(0,T;\mathbf{H}^{k+1}(\Omega)), \quad \mathbf{v}_t, \mathbf{v}_{tt} \in L^\infty(0,T;\mathbf{H}^1(\Omega))$$

with $k \geq 2$, then for any timestep $\Delta t > 0$, the error in the numerical solution from Algorithm 5.2.4 satisfies

$$\begin{aligned}
&\left\|\mathbf{v}^M - \mathbf{v}_h^M\right\|^2 + \alpha^2 \left\|\nabla(\mathbf{v}^M - \mathbf{v}_h^M)\right\|^2 + \nu\Delta t \sum_{n=0}^{M-1} \left\|\nabla(\mathbf{v}^{n+1} - \mathbf{v}_h^{n+1})\right\|^2 \\
&\leq C\nu^{-1}\exp(\nu^{-2})\left(h^{2k}(\alpha^4 + 1 + \alpha^{-2} + \nu^{-2} + \nu^2 + \nu^2\alpha^{-2})\right. \\
&\qquad\qquad \left. + h^{2k+2}(1 + \alpha^{-4} + \nu^2\alpha^{-4}) + \Delta t^2(\alpha^4 + 1 + \nu^2)\right),
\end{aligned}$$

where C is a constant dependent only on data, independent of $\alpha, \gamma, \Delta t, h,$ and ν.

Proof. Throughout this proof we will use C to represent a generic constant, possibly different at each instance, that is independent of $\nu, h, \alpha, \gamma,$ and Δt. Beginning with (5.11)-(5.12), multiply by $\boldsymbol{\chi}_h \in \mathbf{V}_h$ and integrate over Ω to get

$$\alpha^2(\nabla\mathbf{v}_t, \nabla\boldsymbol{\chi}_h) + (\mathbf{v}_t, \boldsymbol{\chi}_h) + b^*\left(G_\gamma\mathbf{v}, G_\gamma\mathbf{v}, \boldsymbol{\chi}_h\right) + \nu(\nabla G_\gamma\mathbf{v}, \nabla\boldsymbol{\chi}_h) = (\mathbf{f}, \boldsymbol{\chi}_h). \tag{5.37}$$

After adding and subtracting terms, we then have for $n = 0, 1, ..., M-1$ and for any $\boldsymbol{\chi}_h \in \mathbf{V}_h$

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t}(\nabla\mathbf{v}^{n+1} - \nabla\mathbf{v}^n, \nabla\boldsymbol{\chi}_h) + \frac{1}{\Delta t}(\mathbf{v}^{n+1} - \mathbf{v}^n, \boldsymbol{\chi}_h) + \nu\left(\nabla\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \nabla\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \nabla\boldsymbol{\chi}_h\right) \\
&\quad + b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^n - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \boldsymbol{\chi}_h\right) = (\mathbf{f}^{n+1}, \boldsymbol{\chi}_h) + G(\mathbf{v}, \boldsymbol{\chi}_h, n),
\end{aligned} \tag{5.38}$$

where $G(\mathbf{v}, \boldsymbol{\chi}_h, n)$ is defined as

$$\begin{aligned}
G(\mathbf{v}, \boldsymbol{\chi}_h, n) := \alpha^2 &\left(\frac{\nabla \mathbf{v}^{n+1} - \nabla \mathbf{v}^n}{\Delta t} - \nabla \mathbf{v}_t(t^{n+1}), \nabla \boldsymbol{\chi}_h \right) + \left(\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t} - \mathbf{v}_t(t^{n+1}), \boldsymbol{\chi}_h \right) \\
&+ \nu \left[\left(\nabla \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \nabla \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \nabla \boldsymbol{\chi}_h \right) - \left(\nabla G_\gamma \mathbf{v}^{n+1}, \nabla \boldsymbol{\chi}_h \right) \right] \\
&- \left[b^* \left(G_\gamma \mathbf{v}^{n+1}, G_\gamma \mathbf{v}^{n+1}, \boldsymbol{\chi}_h \right) - b^* \left(G_\gamma \mathbf{v}^n, G_\gamma \mathbf{v}^{n+1}, \boldsymbol{\chi}_h \right) \right] \\
&- \left[b^* \left(G_\gamma \mathbf{v}^n, G_\gamma \mathbf{v}^{n+1}, \boldsymbol{\chi}_h \right) - b^* \left(G_\gamma \mathbf{v}^n, \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}, \boldsymbol{\chi}_h \right) \right] \\
&- \left[b^* \left(G_\gamma \mathbf{v}^n, \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}, \boldsymbol{\chi}_h \right) \right. \\
&\quad \left. - b^* \left(\frac{\alpha^2}{\gamma^2} \mathbf{v}^n - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \boldsymbol{\chi}_h \right) \right]. \qquad (5.39)
\end{aligned}$$

From (5.38) subtract (5.20), restricting $\boldsymbol{\chi}_h \in \mathbf{V}_h$, and denote $\mathbf{e}^{n+1} = \mathbf{v}^{n+1} - \mathbf{v}_h^{n+1}$ to get

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t} (\nabla \mathbf{e}^{n+1} - \nabla \mathbf{e}^n, \nabla \boldsymbol{\chi}_h) + b^* \left(\frac{\alpha^2}{\gamma^2} \mathbf{v}^n - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \boldsymbol{\chi}_h \right) \\
&+ \frac{1}{\Delta t} (\mathbf{e}^{n+1} - \mathbf{e}^n, \boldsymbol{\chi}_h) - b^* \left(\frac{\alpha^2}{\gamma^2} \mathbf{v}_h^n - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}_h^n}^h, \frac{\alpha^2}{\gamma^2} \mathbf{v}_h^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}_h^n}^h, \boldsymbol{\chi}_h \right) \\
&+ \nu \left(\nabla \frac{\alpha^2}{\gamma^2} \mathbf{e}^{n+1} - \nabla \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{e}^n}^h, \nabla \boldsymbol{\chi}_h \right) = G(\mathbf{v}, \boldsymbol{\chi}_h, n). \qquad (5.40)
\end{aligned}$$

Decompose $\mathbf{e}^{n+1} = (\mathbf{v}^{n+1} - \mathbf{w}_h^{n+1}) + (\mathbf{w}_h^{n+1} - \mathbf{v}_h^{n+1}) =: \boldsymbol{\eta}^{n+1} + \boldsymbol{\phi}_h^{n+1}$, where $\mathbf{w}_h^{n+1}$ is an arbitrarily chosen element of $\mathbf{V}_h$. Then we can rewrite (5.40) as

$$\begin{aligned}
&\frac{\alpha^2}{\Delta t} (\nabla \boldsymbol{\phi}_h^{n+1} - \nabla \boldsymbol{\phi}_h^n, \nabla \boldsymbol{\chi}_h) + \frac{1}{\Delta t} (\boldsymbol{\phi}_h^{n+1} - \boldsymbol{\phi}_h^n, \boldsymbol{\chi}_h) + \nu \left(\nabla \frac{\alpha^2}{\gamma^2} \boldsymbol{\phi}_h^{n+1} - \nabla \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\boldsymbol{\phi}_h^n}^h, \nabla \boldsymbol{\chi}_h \right) \\
&= -\frac{\alpha^2}{\Delta t} (\nabla \boldsymbol{\eta}^{n+1} - \nabla \boldsymbol{\eta}^n, \nabla \boldsymbol{\chi}_h) - \frac{1}{\Delta t} (\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n, \boldsymbol{\chi}_h) - \nu \left(\nabla \frac{\alpha^2}{\gamma^2} \boldsymbol{\eta}^{n+1} - \nabla \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\boldsymbol{\eta}^n}^h, \nabla \boldsymbol{\chi}_h \right) \\
&+ G(\mathbf{v}, \boldsymbol{\chi}_h, n) - b^* \left(\frac{\alpha^2}{\gamma^2} \mathbf{e}^n - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{e}^n}^h, \frac{\alpha^2}{\gamma^2} \mathbf{v}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}^n}^h, \boldsymbol{\chi}_h \right) \\
&- b^* \left(\frac{\alpha^2}{\gamma^2} \mathbf{v}_h^n - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{v}_h^n}^h, \frac{\alpha^2}{\gamma^2} \mathbf{e}^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\mathbf{e}^n}^h, \boldsymbol{\chi}_h \right). \qquad (5.41)
\end{aligned}$$

Continue by choosing

$$\boldsymbol{\chi}_h = \frac{\alpha^2}{\gamma^2} \boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2 - \gamma^2}{\gamma^2} \widetilde{\boldsymbol{\phi}_h^n}^h.$$

Using similar analysis techniques to those employed in Lemma 5.2.5, we can lower bound the time

derivative terms on the left side of (5.41) as

$$\frac{\alpha^2}{\Delta t}\left(\nabla\boldsymbol{\phi}_h^{n+1}-\nabla\boldsymbol{\phi}_h^n,\nabla\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\nabla\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)$$
$$\geq\frac{\alpha^4}{2\gamma^2\Delta t}\left(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\nabla\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2\Delta t}\left(\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right),\quad(5.42)$$

and,

$$\frac{1}{\Delta t}\left(\boldsymbol{\phi}_h^{n+1}-\boldsymbol{\phi}_h^n,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)$$
$$\geq\frac{\alpha^2}{2\gamma^2\Delta t}\left(\left\|\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2-\gamma^2}{2\gamma^2\Delta t}\left(\left\|F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right).\quad(5.43)$$

Using the bounds (5.42)-(5.43) on (5.41) in conjunction with our choice of $\boldsymbol{\chi}_h=\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h$ gives

$$\frac{\alpha^2}{2\gamma^2\Delta t}\left(\left\|\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2-\gamma^2}{2\gamma^2\Delta t}\left(\left\|F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right)$$
$$+\frac{\alpha^4}{2\gamma^2\Delta t}\left(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\nabla\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2\Delta t}\left(\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right)$$
$$+\nu\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\leq-\frac{\alpha^2}{\Delta t}\left(\nabla\boldsymbol{\eta}^{n+1}-\nabla\boldsymbol{\eta}^n,\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right)$$
$$-\frac{1}{\Delta t}\left(\boldsymbol{\eta}^{n+1}-\boldsymbol{\eta}^n,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)+G(\mathbf{v},\boldsymbol{\chi}_h,n)$$
$$-\nu\left(\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h\right),\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right)$$
$$-b^*\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h,\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)$$
$$-b^*\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h,\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)$$
$$-b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)$$
$$-b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right).\quad(5.44)$$

The first two terms on the right side of (5.44) are majorized as in [23]:

$$\left|\frac{1}{\Delta t}\left(\boldsymbol{\eta}^{n+1}-\boldsymbol{\eta}^{n},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\ \leq \frac{4C_{PF}^2}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\|\boldsymbol{\eta}_t\|^2\,dt+\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2, \tag{5.45}$$

and

$$\left|\frac{\alpha^2}{\Delta t}\left(\nabla\boldsymbol{\eta}^{n+1}-\nabla\boldsymbol{\eta}^{n},\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right)\right| \\ \leq \frac{4\alpha^4}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\|\nabla\boldsymbol{\eta}_t\|^2\,dt+\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2. \tag{5.46}$$

The viscous term on the right side of (5.44) is bounded using Cauchy-Schwarz and Young's inequalities, as well as Lemma 2.2:

$$\begin{aligned}&\left|\nu\left(\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h\right),\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right)\right| \\ &\quad\leq\nu\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\ &\quad\leq\frac{\alpha^2}{\gamma^2}\nu\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\ &\qquad+\frac{\alpha^2-\gamma^2}{\gamma^2}\nu\left\|\nabla\widetilde{\boldsymbol{\eta}^n}^h\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\ &\quad\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu\frac{\alpha^4}{\gamma^4}\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right).\end{aligned} \tag{5.47}$$

Using Lemma 2.0.2, Young's inequality and Lemma 5.2.2, majorize the first nonlinear term in (5.44)

to get

$$
\begin{aligned}
&\left|-b^*\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h,\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\leq C\left(\left\|\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right\|^{1/2}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^{1/2}\right. \\
&\qquad\left.\cdot\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|\right) \\
&\leq 4C\nu^{-1}\left\|\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)\right\|^2 \\
&\qquad+\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2 \\
&\leq C\nu^{-1}\frac{(2\alpha^2-\gamma^2)^2}{\gamma^4}\left\|\nabla\boldsymbol{\phi}_h^n\right\|\left\|\boldsymbol{\phi}_h^n\right\|\left(\frac{\alpha^4}{\gamma^4}\left\|\nabla\mathbf{v}^{n+1}\right\|^2+\frac{(\alpha^2-\gamma^2)^2}{\gamma^4}\left\|\nabla\widetilde{\mathbf{v}^n}^h\right\|^2\right) \\
&\qquad+\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2 \\
&\leq C\nu^{-2}\frac{\alpha^8(2\alpha^2-\gamma^2)^4}{\gamma^{16}}\left\|\boldsymbol{\phi}_h^n\right\|^2\left\|\nabla\mathbf{v}^{n+1}\right\|^4+C\nu^{-2}\frac{(\alpha^2-\gamma^2)^4(2\alpha^2-\gamma^2)^4}{\gamma^{16}}\left\|\boldsymbol{\phi}_h^n\right\|^2\left\|\nabla\mathbf{v}^n\right\|^4 \\
&\qquad+\frac{\nu}{16}\left\|\nabla\boldsymbol{\phi}_h^n\right\|^2+\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2 \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+\frac{\nu}{16}\left\|\nabla\boldsymbol{\phi}_h^n\right\|^2+C\nu^{-2}\frac{\alpha^{16}}{\gamma^{16}}\left\|\boldsymbol{\phi}_h^n\right\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^4+\left\|\nabla\mathbf{v}^n\right\|^4\right).
\end{aligned}
\tag{5.48}
$$

Using Lemma 2.0.2 and Young's inequality on the second nonlinear term in (5.44) gives

$$
\begin{aligned}
&\left|-b^*\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h,\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\le C\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\
&\le C\frac{2\alpha^2-\gamma^2}{\gamma^2}\left\|\nabla\boldsymbol{\eta}^n\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\
&\le C\frac{\alpha^2(2\alpha^2-\gamma^2)}{\gamma^4}\left\|\nabla\boldsymbol{\eta}^n\right\|\left\|\nabla\mathbf{v}^{n+1}\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\
&\qquad + C\frac{(\alpha^2-\gamma^2)(2\alpha^2-\gamma^2)}{\gamma^4}\left\|\nabla\boldsymbol{\eta}^n\right\|\left\|\nabla\mathbf{v}^n\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\
&\le \frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^4(2\alpha^2-\gamma^2)^2}{\gamma^8}\left\|\nabla\boldsymbol{\eta}^n\right\|^2\left\|\nabla\mathbf{v}^{n+1}\right\|^2 \\
&\qquad + C\nu^{-1}\frac{(\alpha^2-\gamma^2)^2(2\alpha^2-\gamma^2)^2}{\gamma^8}\left\|\nabla\boldsymbol{\eta}^n\right\|^2\left\|\nabla\mathbf{v}^n\right\|^2 \\
&\le \frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\left\|\nabla\boldsymbol{\eta}^n\right\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^2+\left\|\nabla\mathbf{v}^n\right\|^2\right). \qquad (5.49)
\end{aligned}
$$

By Lemma 2.0.2, the third nonlinear term in (5.44) vanishes. Finally, the fourth nonlinear term is bounded in the same manner as in (5.49)

$$
\begin{aligned}
&\left|-b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}_h^n-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}_h^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\eta}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\eta}^n}^h,\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\le \frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\left\|\nabla\mathbf{v}_h^n\right\|^2\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\left\|\nabla\boldsymbol{\eta}^n\right\|^2\right). \qquad (5.50)
\end{aligned}
$$

Applying the bounds (5.45)-(5.50) to equation (5.44) yields

$$
\begin{aligned}
&\frac{\alpha^2}{2\gamma^2\Delta t}\left(\left\|\boldsymbol{\phi}_h^{n+1}\right\|^2-\left\|\boldsymbol{\phi}_h^n\right\|^2\right)+\frac{\alpha^2-\gamma^2}{2\gamma^2\Delta t}\left(\left\|F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right) \\
&+\frac{\alpha^4}{2\gamma^2\Delta t}\left(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2-\left\|\nabla\boldsymbol{\phi}_h^n\right\|^2\right)+\frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2\Delta t}\left(\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right) \\
&+\frac{10\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\le\frac{\nu}{16}\left\|\nabla\boldsymbol{\phi}_h^n\right\|^2+\frac{4C_{PF}^2}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\left\|\boldsymbol{\eta}_t\right\|^2\,dt \\
&+\frac{4\alpha^4}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\left\|\nabla\boldsymbol{\eta}_t\right\|^2\,dt+C\nu\frac{\alpha^4}{\gamma^4}\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\left\|\nabla\boldsymbol{\eta}^n\right\|^2\right)+C\nu^{-2}\frac{\alpha^{16}}{\gamma^{16}}\left\|\boldsymbol{\phi}_h^n\right\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^4+\left\|\nabla\mathbf{v}^n\right\|^4\right) \\
&+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\left\|\nabla\boldsymbol{\eta}^n\right\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^2+\left\|\nabla\mathbf{v}^n\right\|^2\right)+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\left\|\nabla\mathbf{v}_h^n\right\|^2\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\left\|\nabla\boldsymbol{\eta}^n\right\|^2\right) \\
&+\left|G\left(\mathbf{v},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h,n\right)\right|. \qquad (5.51)
\end{aligned}
$$

We will now continue by majorizing

$$\left| G\left(\mathbf{v}, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h, n \right) \right|,$$

as defined in (5.39) with $\boldsymbol{\chi}_h = \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h$. The first term of (5.39) is bounded using standard estimates based on Taylor series

$$\begin{aligned}
&\alpha^2 \left(\frac{\nabla \mathbf{v}^{n+1} - \nabla \mathbf{v}^n}{\Delta t} - \nabla \mathbf{v}_t(t^{n+1}), \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right) \\
&\qquad \leq \frac{\nu}{16} \left\| \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right\|^2 + 2\Delta t^2 \alpha^4 \nu^{-1} \left\| \mathbf{v}_{tt} \right\|^2_{L^\infty(t^n, t^{n+1}, \mathbf{H}^1(\Omega))}. \qquad (5.52)
\end{aligned}$$

The second term of G is bounded in the same manner

$$\begin{aligned}
&\left(\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t} - \mathbf{v}_t(t^{n+1}), \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \\
&\qquad \leq \frac{\nu}{16} \left\| \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right\|^2 + 2\Delta t^2 C_{PF}^2 \nu^{-1} \left\| \mathbf{v}_{tt} \right\|^2_{L^\infty(t^n, t^{n+1}, \mathbf{H}^1(\Omega))}. \qquad (5.53)
\end{aligned}$$

Using Cauchy-Schwarz and Young's inequalities on the third term of (5.39) yields

$$\begin{aligned}
&\left| \nu \left(\nabla \left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h \right) - \nabla G_\gamma \mathbf{v}^{n+1}, \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right) \right| \\
&\quad = \left| \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2} \left(\nabla \left(\widetilde{\mathbf{v}^{n+1}} - \widetilde{\mathbf{v}^n}^h \right), \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right) \right| \\
&\quad = \left| \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2} \left(\nabla \left(\widetilde{\mathbf{v}^{n+1}} - \widetilde{\mathbf{v}^n} \right), \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right) \right. \\
&\qquad \left. + \frac{\nu(\alpha^2-\gamma^2)}{\gamma^2} \left(\nabla \left(\widetilde{\mathbf{v}^n} - \widetilde{\mathbf{v}^n}^h \right), \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right) \right| \\
&\quad \leq \frac{\nu}{16} \left\| \nabla \left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h \right) \right\|^2 \\
&\qquad + C\nu \frac{(\alpha^2-\gamma^2)^2}{\gamma^4} \left(\left\| \nabla \left(\widetilde{\mathbf{v}^{n+1}} - \widetilde{\mathbf{v}^n} \right) \right\|^2 + \left\| \nabla \left(\widetilde{\mathbf{v}^n} - \widetilde{\mathbf{v}^n}^h \right) \right\|^2 \right).
\end{aligned}$$

Using estimates based on Taylor series as well as Lemma 5.2.3 to bound the difference between

continuous and discrete filters, we then see

$$\begin{aligned}
&\left|\nu\left(\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)-\nabla G_\gamma\mathbf{v}^{n+1},\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right)\right| \\
&\quad\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu\frac{\alpha^4}{\gamma^4}\Delta t^2\left\|\mathbf{v}_t\right\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))} \\
&\quad\quad+C\nu\frac{\alpha^4}{\gamma^6}(\gamma^{-2}h^{2k+2}+h^{2k})\left\|\mathbf{v}^n\right\|^2_{\mathbf{H}^{k+1}(\Omega)}. \qquad (5.54)
\end{aligned}$$

To bound the fourth term of G, we use Lemma 2.0.2, Young's inequality, and Taylor's theorem to arrive at

$$\begin{aligned}
&\left|b^*\left(G_\gamma\mathbf{v}^{n+1},G_\gamma\mathbf{v}^{n+1},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)-b^*\left(G_\gamma\mathbf{v}^n,G_\gamma\mathbf{v}^{n+1},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&=\left|b^*\left(G_\gamma(\mathbf{v}^{n+1}-\mathbf{v}^n),G_\gamma\mathbf{v}^{n+1},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\leq\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|\left\|\nabla G_\gamma(\mathbf{v}^{n+1}-\mathbf{v}^n)\right\|\left\|\nabla G_\gamma\mathbf{v}^{n+1}\right\| \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\left\|\nabla G_\gamma(\mathbf{v}^{n+1}-\mathbf{v}^n)\right\|^2\left\|\nabla G_\gamma\mathbf{v}^{n+1}\right\|^2 \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\left\|\nabla(\mathbf{v}^{n+1}-\mathbf{v}^n)\right\|^2\left\|\nabla\mathbf{v}^{n+1}\right\|^2 \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\Delta t^2\left\|\mathbf{v}_t\right\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\left\|\nabla\mathbf{v}^{n+1}\right\|^2. \qquad (5.55)
\end{aligned}$$

The same techniques are used to bound the fifth term in (5.39)

$$\begin{aligned}
&\left|b^*\left(G_\gamma\mathbf{v}^n,G_\gamma\mathbf{v}^{n+1},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right. \\
&\quad\left.-b^*\left(G_\gamma\mathbf{v}^n,\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n},\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&=\left|b^*\left(G_\gamma\mathbf{v}^n,\frac{\alpha^2-\gamma^2}{\gamma^2}\left(\widetilde{\mathbf{v}^n}-\widetilde{\mathbf{v}^{n+1}}\right),\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{(\alpha^2-\gamma^2)^2}{\gamma^4}\left\|\nabla\left(\widetilde{\mathbf{v}^n}-\widetilde{\mathbf{v}^{n+1}}\right)\right\|^2\left\|\nabla G_\gamma\mathbf{v}^n\right\|^2 \\
&\leq\frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\Delta t^2\left\|\mathbf{v}_t\right\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\left\|\nabla\mathbf{v}^n\right\|^2. \qquad (5.56)
\end{aligned}$$

For the sixth term in (5.39), we start by using Lemma 2.0.2 and Lemma 5.2.2

$$
\begin{aligned}
&\left| b^*\left(G_\gamma \mathbf{v}^n, \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right. \\
&\quad \left. - b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^n - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&= \left| b^*\left(\frac{\alpha^2-\gamma^2}{\gamma^2}\left(\widetilde{\mathbf{v}^n}^h - \widetilde{\mathbf{v}^n}\right), \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right. \\
&\quad \left. - b^*\left(G_\gamma \mathbf{v}^n, \frac{\alpha^2-\gamma^2}{\gamma^2}\left(\widetilde{\mathbf{v}^n} - \widetilde{\mathbf{v}^n}^h\right), \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\leq C\frac{\alpha^2-\gamma^2}{\gamma^2}\left\|\nabla\left(\widetilde{\mathbf{v}^n}^h - \widetilde{\mathbf{v}^n}\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\| \\
&\quad \cdot \left(\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h\right)\right\| + \|\nabla G_\gamma \mathbf{v}^n\|\right) \\
&\leq C\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^4}\left\|\nabla\left(\widetilde{\mathbf{v}^n}^h - \widetilde{\mathbf{v}^n}\right)\right\|\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|\left(\left\|\nabla\mathbf{v}^{n+1}\right\| + \|\nabla\mathbf{v}^n\|\right).
\end{aligned}
$$

Using Young's inequality and Lemma 5.2.3 we then get

$$
\begin{aligned}
&\left| b^*\left(G_\gamma \mathbf{v}^n, \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right. \\
&\quad \left. - b^*\left(\frac{\alpha^2}{\gamma^2}\mathbf{v}^n - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\mathbf{v}^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\mathbf{v}^n}^h, \frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right| \\
&\leq \frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2 \\
&\quad + C\nu^{-1}\frac{\alpha^4(\alpha^2-\gamma^2)^2}{\gamma^8}\left\|\nabla\left(\widetilde{\mathbf{v}^n}^h - \widetilde{\mathbf{v}^n}\right)\right\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\| + \|\nabla\mathbf{v}^n\|\right)^2 \\
&\leq \frac{\nu}{16}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1} - \frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2 \\
&\quad + C\nu^{-1}\frac{\alpha^8}{\gamma^{10}}\left(\gamma^{-2}h^{2k+2} + h^{2k}\right)\|\mathbf{v}^n\|^2_{\mathbf{H}^{k+1}(\Omega)}\left(\left\|\nabla\mathbf{v}^{n+1}\right\| + \|\nabla\mathbf{v}^n\|\right)^2. \qquad (5.57)
\end{aligned}
$$

Using the bounds (5.52)-(5.57) on equation (5.51), we get

$$
\begin{aligned}
&\frac{\alpha^2}{2\gamma^2\Delta t}\left(\left\|\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2-\gamma^2}{2\gamma^2\Delta t}\left(\left\|F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right)\\
&+\frac{\alpha^4}{2\gamma^2\Delta t}\left(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\nabla\boldsymbol{\phi}_h^n\|^2\right)+\frac{\alpha^2(\alpha^2-\gamma^2)}{2\gamma^2\Delta t}\left(\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^n\right\|^2-\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^{n+1}\right\|^2\right)\\
&+\frac{\nu}{4}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\leq\frac{\nu}{16}\|\nabla\boldsymbol{\phi}_h^n\|^2+\frac{4C_{PF}^2}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\|\boldsymbol{\eta}_t\|^2\,dt\\
&+\frac{4\alpha^4}{\nu\Delta t}\int_{t^n}^{t^{n+1}}\|\nabla\boldsymbol{\eta}_t\|^2\,dt+C\nu\frac{\alpha^4}{\gamma^4}\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)+C\nu^{-2}\frac{\alpha^{16}}{\gamma^{16}}\|\boldsymbol{\phi}_h^n\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^4+\|\nabla\mathbf{v}^n\|^4\right)\\
&+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\|\nabla\boldsymbol{\eta}^n\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^2+\|\nabla\mathbf{v}^n\|^2\right)+C\nu^{-1}\frac{\alpha^8}{\gamma^8}\|\nabla\mathbf{v}_h^n\|^2\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)\\
&+C\nu^{-1}\Delta t^2\alpha^4\|\mathbf{v}_{tt}\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}+C\nu^{-1}\Delta t^2\|\mathbf{v}_{tt}\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\\
&+C\nu\Delta t^2\frac{\alpha^4}{\gamma^4}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}+C\nu\frac{\alpha^4}{\gamma^6}(\gamma^{-2}h^{2k+2}+h^{2k})\|\mathbf{v}^n\|^2_{\mathbf{H}^{k+1}(\Omega)}\\
&+C\nu^{-1}\Delta t^2\frac{\alpha^8}{\gamma^8}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\left\|\nabla\mathbf{v}^{n+1}\right\|^2+C\nu^{-1}\Delta t^2\frac{\alpha^8}{\gamma^8}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\|\nabla\mathbf{v}^n\|^2\\
&+C\nu^{-1}\frac{\alpha^8}{\gamma^{10}}\left(\gamma^{-2}h^{2k+2}+h^{2k}\right)\|\mathbf{v}^n\|^2_{\mathbf{H}^{k+1}(\Omega)}\left(\left\|\nabla\mathbf{v}^{n+1}\right\|+\|\nabla\mathbf{v}^n\|\right)^2. \qquad (5.58)
\end{aligned}
$$

Multiplying through by $2\Delta t$ and summing from $n=0$ to $M-1$ yields

$$
\begin{aligned}
&\frac{\alpha^2}{\gamma^2}\left(\left\|\boldsymbol{\phi}_h^M\right\|^2-\left\|\boldsymbol{\phi}_h^0\right\|\right)+\frac{\alpha^2-\gamma^2}{\gamma^2}\left(\left\|F_h^{1/2}\boldsymbol{\phi}_h^0\right\|^2-\left\|F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2\right)+\frac{\alpha^4}{\gamma^2}\left(\left\|\nabla\boldsymbol{\phi}_h^M\right\|^2-\left\|\nabla\boldsymbol{\phi}_h^0\right\|^2\right)\\
&+\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2}\left(\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^0\right\|^2-\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2\right)+\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}}_h^{n}{}^h\right)\right\|^2\\
&\leq\frac{\nu\Delta t}{8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\phi}_h^n\|^2+C\nu^{-1}\int_0^T\|\boldsymbol{\eta}_t\|^2\,dt+C\nu^{-1}\alpha^4\int_0^T\|\nabla\boldsymbol{\eta}_t\|^2\,dt\\
&+C\nu\Delta t\frac{\alpha^4}{\gamma^4}\sum_{n=0}^{M-1}\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)+C\nu^{-2}\Delta t\frac{\alpha^{16}}{\gamma^{16}}\sum_{n=0}^{M-1}\|\boldsymbol{\phi}_h^n\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^4+\|\nabla\mathbf{v}^n\|^4\right)\\
&+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\eta}^n\|^2\left(\left\|\nabla\mathbf{v}^{n+1}\right\|^2+\|\nabla\mathbf{v}^n\|^2\right)\\
&+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\nabla\mathbf{v}_h^n\|^2\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)\\
&+C\nu^{-1}\Delta t^3\alpha^4\sum_{n=0}^{M-1}\|\mathbf{v}_{tt}\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}+C\nu^{-1}\Delta t^3\sum_{n=0}^{M-1}\|\mathbf{v}_{tt}\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\\
&+C\nu\Delta t^3\frac{\alpha^4}{\gamma^4}\sum_{n=0}^{M-1}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}+C\nu\Delta t\frac{\alpha^4}{\gamma^6}(\gamma^{-2}h^{2k+2}+h^{2k})\sum_{n=0}^{M-1}\|\mathbf{v}^n\|^2_{\mathbf{H}^{k+1}(\Omega)}\\
&+C\nu^{-1}\Delta t^3\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\left\|\nabla\mathbf{v}^{n+1}\right\|^2\\
&+C\nu^{-1}\Delta t^3\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\mathbf{v}_t\|^2_{L^\infty(t^n,t^{n+1},\mathbf{H}^1(\Omega))}\|\nabla\mathbf{v}^n\|^2\\
&+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^{10}}(\gamma^{-2}h^{2k+2}+h^{2k})\sum_{n=0}^{M-1}\|\mathbf{v}^n\|^2_{\mathbf{H}^{k+1}(\Omega)}\left(\left\|\nabla\mathbf{v}^{n+1}\right\|+\|\nabla\mathbf{v}^n\|\right)^2. \qquad (5.59)
\end{aligned}
$$

Note that our assumption of $\mathbf{v}^0\in\mathbf{V}_h$ implies $\boldsymbol{\phi}_h^0=0$. Using this and assumptions on the regularity

of the model solution, (5.59) becomes

$$\frac{\alpha^2}{\gamma^2}\left\|\boldsymbol{\phi}_h^M\right\|^2-\frac{\alpha^2-\gamma^2}{\gamma^2}\left\|F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2+\frac{\alpha^4}{\gamma^2}\left\|\nabla\boldsymbol{\phi}_h^M\right\|^2-\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2$$
$$+\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\leq\frac{\nu\Delta t}{8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\phi}_h^n\|^2+C\nu^{-1}\int_0^T\|\boldsymbol{\eta}_t\|^2\,dt$$
$$+C\nu^{-1}\alpha^4\int_0^T\|\nabla\boldsymbol{\eta}_t\|^2\,dt+C\nu\Delta t\frac{\alpha^4}{\gamma^4}\sum_{n=0}^{M-1}\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)+C\nu^{-2}\Delta t\frac{\alpha^{16}}{\gamma^{16}}\sum_{n=0}^{M-1}\|\boldsymbol{\phi}_h^n\|^2$$
$$+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\eta}^n\|^2+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^8}\sum_{n=0}^{M-1}\|\nabla\mathbf{v}_h^n\|^2\left(\left\|\nabla\boldsymbol{\eta}^{n+1}\right\|^2+\|\nabla\boldsymbol{\eta}^n\|^2\right)+C\nu^{-1}T\Delta t^2\alpha^4$$
$$+C\nu^{-1}T\Delta t^2+C\nu T\Delta t^2\frac{\alpha^4}{\gamma^4}+C\nu T\frac{\alpha^4}{\gamma^6}\left(\gamma^{-2}h^{2k+2}+h^{2k}\right)+C\nu^{-1}T\Delta t^2\frac{\alpha^8}{\gamma^8}$$
$$+C\nu^{-1}T\frac{\alpha^8}{\gamma^{10}}\left(\gamma^{-2}h^{2k+2}+h^{2k}\right). \tag{5.60}$$

With regularity assumptions and interpolation estimates, (5.60) becomes

$$\frac{\alpha^2}{\gamma^2}\left\|\boldsymbol{\phi}_h^M\right\|^2-\frac{\alpha^2-\gamma^2}{\gamma^2}\left\|F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2+\frac{\alpha^4}{\gamma^2}\left\|\nabla\boldsymbol{\phi}_h^M\right\|^2-\frac{\alpha^2(\alpha^2-\gamma^2)}{\gamma^2}\left\|\nabla F_h^{1/2}\boldsymbol{\phi}_h^M\right\|^2$$
$$+\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\leq\frac{\nu\Delta t}{8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\phi}_h^n\|^2+C\nu^{-2}\Delta t\frac{\alpha^{16}}{\gamma^{16}}\sum_{n=0}^{M-1}\|\boldsymbol{\phi}_h^n\|^2$$
$$+C\nu^{-1}T\left(h^{2k+2}+\alpha^4h^{2k}+\frac{\alpha^8}{\gamma^8}h^{2k}+\Delta t^2\alpha^4+\Delta t^2+\Delta t^2\frac{\alpha^8}{\gamma^8}+\frac{\alpha^8}{\gamma^{10}}(\gamma^{-2}h^{2k+2}+h^{2k})\right)$$
$$+C\nu T\left(\frac{\alpha^4}{\gamma^4}h^{2k}+\Delta t^2\frac{\alpha^4}{\gamma^4}+\frac{\alpha^4}{\gamma^6}(\gamma^{-2}h^{2k+2}+h^{2k})\right)+C\nu^{-1}\Delta t\frac{\alpha^8}{\gamma^8}h^{2k}\sum_{n=0}^{M-1}\|\nabla\mathbf{v}_h^n\|^2. \tag{5.61}$$

We can lower bound the viscous term in the same fashion as in Lemma 5.2.5 to get

$$\begin{aligned}&\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\left(\frac{\alpha^2}{\gamma^2}\boldsymbol{\phi}_h^{n+1}-\frac{\alpha^2-\gamma^2}{\gamma^2}\widetilde{\boldsymbol{\phi}_h^n}^h\right)\right\|^2\\&\quad\geq\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2+\frac{\alpha^2-\gamma^2}{\gamma^2}(\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2-\|\nabla\boldsymbol{\phi}_h^n\|^2)\right)\\&\quad=\frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2+\frac{\nu\Delta t(\alpha^2-\gamma^2)}{2\gamma^4}\left\|\nabla\boldsymbol{\phi}_h^M\right\|^2.\end{aligned} \tag{5.62}$$

Combining (5.61) and (5.62), and using Lemma 5.2.5 and $(\overline{\boldsymbol{\phi}}^h, \boldsymbol{\phi}) \leq \|\boldsymbol{\phi}_h\|^2$ gives

$$\begin{aligned}
&\left\|\boldsymbol{\phi}_h^M\right\|^2 + \alpha^2 \left\|\nabla\boldsymbol{\phi}_h^M\right\|^2 + \frac{\nu\Delta t}{2}\sum_{n=0}^{M-1}\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2 \leq \frac{\nu\Delta t}{8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\phi}_h^n\|^2 + C\nu^{-2}\Delta t\frac{\alpha^{16}}{\gamma^{16}}\sum_{n=0}^{M-1}\|\boldsymbol{\phi}_h^n\|^2 \\
&+ C\nu^{-1}\left(h^{2k}\left(\alpha^4 + \frac{\alpha^8}{\gamma^8} + \frac{\alpha^8}{\gamma^{10}} + \nu^{-2}\frac{\alpha^8}{\gamma^8}\right) + h^{2k+2}\left(1 + \frac{\alpha^8}{\gamma^{12}}\right) + \Delta t^2\left(\alpha^4 + 1 + \frac{\alpha^8}{\gamma^8}\right)\right) \\
&+ C\nu\left(h^{2k}\left(\frac{\alpha^4}{\gamma^4} + \frac{\alpha^4}{\gamma^6}\right) + \frac{\alpha^4}{\gamma^8}h^{2k+2} + \frac{\alpha^4}{\gamma^4}\Delta t^2\right). \qquad (5.63)
\end{aligned}$$

Subtracting $\frac{\nu\Delta t}{8}\sum_{n=0}^{M-1}\|\nabla\boldsymbol{\phi}_h^n\|^2$ from both sides yields

$$\begin{aligned}
&\left\|\boldsymbol{\phi}_h^M\right\|^2 + \alpha^2 \left\|\nabla\boldsymbol{\phi}_h^M\right\|^2 + \frac{3\nu\Delta t}{8}\sum_{n=0}^{M-1}\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2 \leq C\nu^{-2}\Delta t\frac{\alpha^{16}}{\gamma^{16}}\sum_{n=0}^{M-1}\|\boldsymbol{\phi}_h^n\|^2 \\
&+ C\nu^{-1}\left(h^{2k}\left(\alpha^4 + \frac{\alpha^8}{\gamma^8} + \frac{\alpha^8}{\gamma^{10}} + \nu^{-2}\frac{\alpha^8}{\gamma^8}\right) + h^{2k+2}\left(1 + \frac{\alpha^8}{\gamma^{12}}\right) + \Delta t^2\left(\alpha^4 + 1 + \frac{\alpha^8}{\gamma^8}\right)\right) \\
&+ C\nu\left(h^{2k}\left(\frac{\alpha^4}{\gamma^4} + \frac{\alpha^4}{\gamma^6}\right) + \frac{\alpha^4}{\gamma^8}h^{2k+2} + \frac{\alpha^4}{\gamma^4}\Delta t^2\right). \qquad (5.64)
\end{aligned}$$

Applying the discrete Gronwall lemma we see, for any $\Delta t > 0$,

$$\begin{aligned}
&\left\|\boldsymbol{\phi}_h^M\right\|^2 + \alpha^2 \left\|\nabla\boldsymbol{\phi}_h^M\right\|^2 + \nu\Delta t\sum_{n=0}^{M-1}\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2 \\
&\quad\leq C\nu^{-1}\exp\left(\nu^{-2}\frac{\alpha^{16}}{\gamma^{16}}\right)\left[h^{2k}\left(\alpha^4 + \frac{\alpha^8}{\gamma^8} + \frac{\alpha^8}{\gamma^{10}} + \nu^{-2}\frac{\alpha^8}{\gamma^8} + \nu^2\frac{\alpha^4}{\gamma^4} + \nu^2\frac{\alpha^4}{\gamma^6}\right)\right. \\
&\qquad\left. + h^{2k+2}\left(1 + \frac{\alpha^8}{\gamma^{12}} + \nu^2\frac{\alpha^4}{\gamma^8}\right) + \Delta t^2\left(\alpha^4 + 1 + \frac{\alpha^8}{\gamma^8} + \nu^2\frac{\alpha^4}{\gamma^4}\right)\right]. \qquad (5.65)
\end{aligned}$$

Finally, using our assumption that $\alpha = C_0\gamma$ gives us the bound

$$\begin{aligned}
&\left\|\boldsymbol{\phi}_h^M\right\|^2 + \alpha^2 \left\|\nabla\boldsymbol{\phi}_h^M\right\|^2 + \nu\Delta t\sum_{n=0}^{M-1}\left\|\nabla\boldsymbol{\phi}_h^{n+1}\right\|^2 \qquad (5.66) \\
&\quad\leq C\nu^{-1}\exp(\nu^{-2})\left(h^{2k}(\alpha^4 + 1 + \alpha^{-2} + \nu^{-2} + \nu^2 + \nu^2\alpha^{-2})\right. \\
&\qquad\left. + h^{2k+2}(1 + \alpha^{-4} + \nu^2\alpha^{-4}) + \Delta t^2(\alpha^4 + 1 + \nu^2)\right).
\end{aligned}$$

Using the triangle inequality now finishes the proof. □

5.4 Numerical Results

This section presents two benchmark numerical experiments chosen to evaluate the accuracy of Algorithm 5.2.4 on flow problems with complex behaviours. In both experiments, a resolved solution is generated by computing the NSE directly on a fine triangular mesh, using the fully implicit Crank-Nicolson finite element method (as in [46]). Solutions are then generated for Algorithm 5.2.4 using various values for α and γ. We hope to see an increase in accuracy in RMDM solutions generated with $\alpha > \gamma$ as opposed to those generated with $\alpha = \gamma$ (i.e. those with no deconvolution). All computations were performed using the open-source software *FreeFem++* [39].

5.4.1 2D Channel flow over a backward-facing step

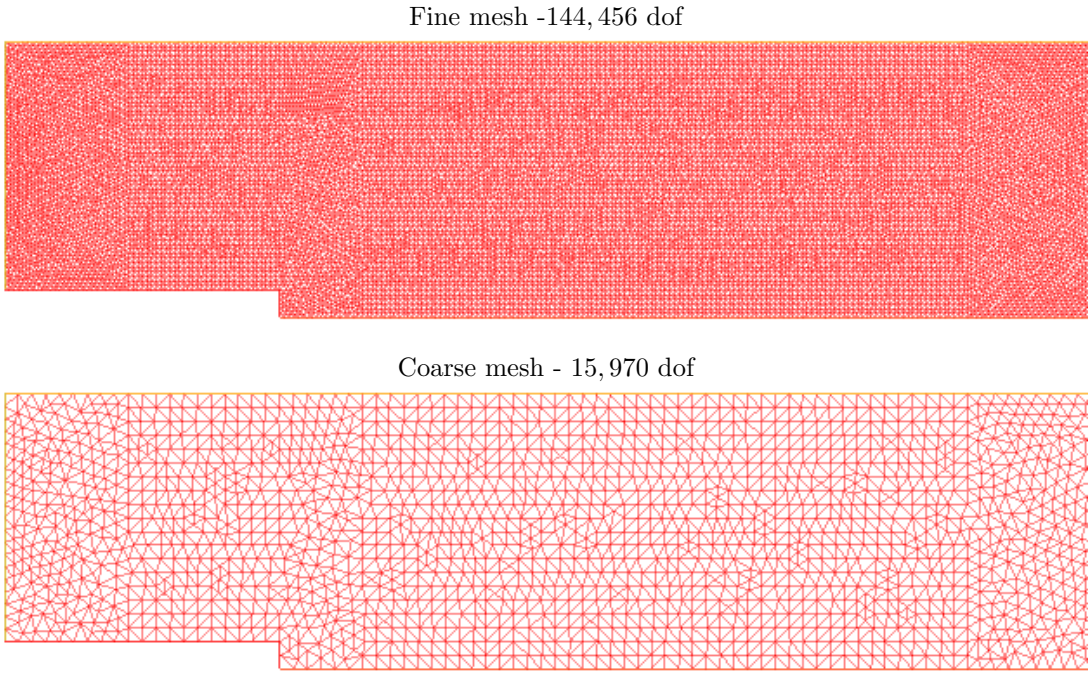


Figure 5.1: Fine mesh used for the resolved NSE solution and the coarse mesh used for the RMDM approximations.

The first experiment we present is a well-known benchmark flow problem consisting of 2D channel flow over a backward-facing step. Once the flow passes over the step, the shear-layer separates from the bottom wall, causing complex flow behaviour behind the step. It has been well documented that the subsequent length until the flow reattaches to the wall is dependent upon both

the Reynolds number and the expansion ratio of the channel (see e.g. [5] and [44]). The domain Ω consists of a $40h \times 10h$ channel with a step of height h and length $10h$ running along the bottom of the channel, as seen in Figure 5.1. Our choice of channel parameterization yields an expansion ratio of 10/9. For our experiment we employed a step height of $h = 1$. Flow entering the channel on the left is assumed to satisfy the parabolic Dirichlet boundary condition

$$\mathbf{u} = \begin{pmatrix} 4(y-1)(10-y)/81 \\ 0 \end{pmatrix},$$

while zero-traction boundary conditions are enforced on flow exiting the channel on the right. No-slip boundary conditions are assumed on all other boundaries. All computations were performed using Reynolds number $\mathrm{Re} = 1000$, a time-step of $\Delta t = 0.01$, and with the flow started from rest at $T = 0$.

The reference NSE solution was computed using TH $(\mathbf{P}_2, P_1)$ elements on a mesh providing $144,456$ combined degrees of freedom for the velocity and pressure (see Figure 5.1). The computed reference solution reaches a steady state by $T = 200$, with a steady-state reattachment length of approximately 17.5 units, which agrees with the experimental results found in [5]. A plot of the velocity streamlines and speed contours of the solution at $T = 200$ can be seen in Figure 5.2.

Solutions to Algorithm 5.2.4 were computed using TH elements on a coarse mesh providing $15,970$ combined degrees of freedom for the velocity and pressure (see Figure 5.1). We began by computing Algorithm 5.2.4 with $\alpha = \gamma = 0.125$, i.e. with no deconvolution. A plot of the streamlines and speed contours of the solution at $T = 200$ is shown in Figure 5.2. It is clear that the RMDM without deconvolution produces a solution that is incorrect. The three eddies present behind the step have yet to merge into one large eddy. However, when γ is lowered to 0.06, the RMDM solution looks very close to the reference solution. From Figure 5.2 we can clearly see that both the number of eddies present behind the step and the reattachment length both seem to be correct. Hence, by reducing γ we can produce a substantially more accurate approximation.

5.4.2 2D Channel flow with a contraction and two outlets

The second benchmark experiment we present is channel flow with a contraction, one inlet on the left of the channel, and two outlets at the top and the right of the channel (as seen in Figure

5.3). This problem was first studied by Heywood et al. [41]. Flow entering the channel satisfies the parabolic profile

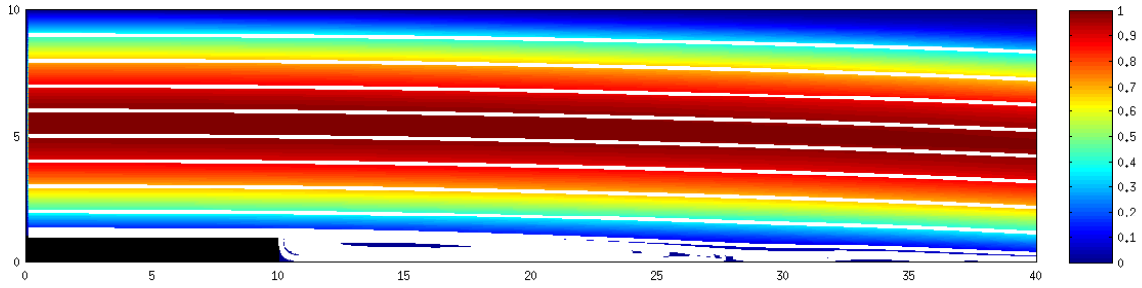
$$\mathbf{u} = \begin{pmatrix} 4y(1-y) \\ 0 \end{pmatrix}.$$

Zero-traction boundary conditions are enforced on both outflow boundaries, and no-slip boundary conditions are enforced on all wall boundaries. All computations were performed with $\mathrm{Re} = 1000$, flow starting at rest at $T = 0$, and an end time of $T = 4$.

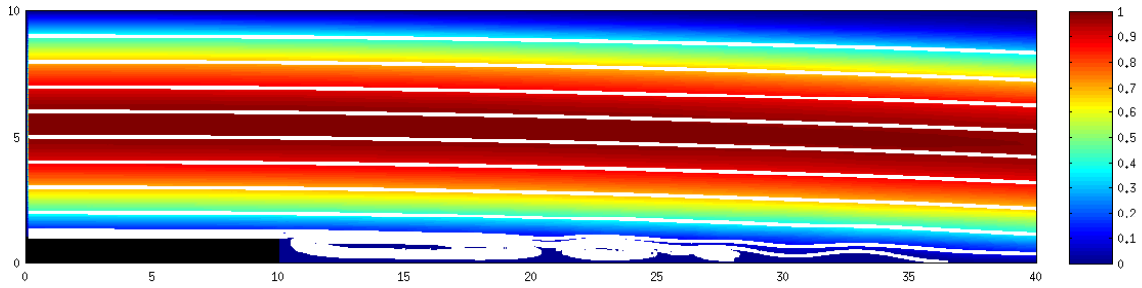
The reference NSE solution was computed using TH $(\mathbf{P}_2, P_1)$ elements on a fine mesh providing $260,378$ combined degrees of freedom for the velocity and the pressure (see Figure 5.3), and with a time-step of $\Delta t = 0.005$. Speed contours of the resolved solution are shown in Figure 5.4. Note that the flow speeds up in the contraction, and seems to oscillate up and down on the right side of the channel. Additionally, the flow seems to remain in a single stream once it passes the contraction, although by $T = 4$, we also see the appearance of smaller flow structures near the right outflow boundary.

Solutions to Algorithm 5.2.4 were computed using TH elements with a time-step of $\Delta t = 0.01$ on the coarse mesh pictured in 5.4, providing $10,820$ degrees of freedom for the velocity and pressure. Figure 5.4 shows a plot of the speed contours of the solution of our proposed method for the RMDM with $\alpha = \gamma = 0.05$ (no deconvolution). It is clear from the plots that this velocity solution does not accurately capture the shape of the flow after the contraction. We also see flow leaving the channel through the top outlet, which is not present in the resolved NSE solution plot. However, when we recompute with $\alpha = 0.05$ and $\gamma = 0.03$, the velocity solution (pictured in Figure 5.4) much more accurately captures the shape of the flow after the contraction. Additionally, we see very little fluid exiting the channel through the top outlet, which agrees with our "true" resolved solution. Both solutions seem to partially capture the formation of smaller flow structures near the right outlet. Again, we see that the solution to Algorithm 5.2.4 with $\alpha > \gamma$ is much closer to the resolved NSE solution than that computed with no deconvolution.

Resolved NSE solution



RMDM $\alpha = 0.125, \gamma = 0.125$



RMDM $\alpha = 0.125, \gamma = 0.06$

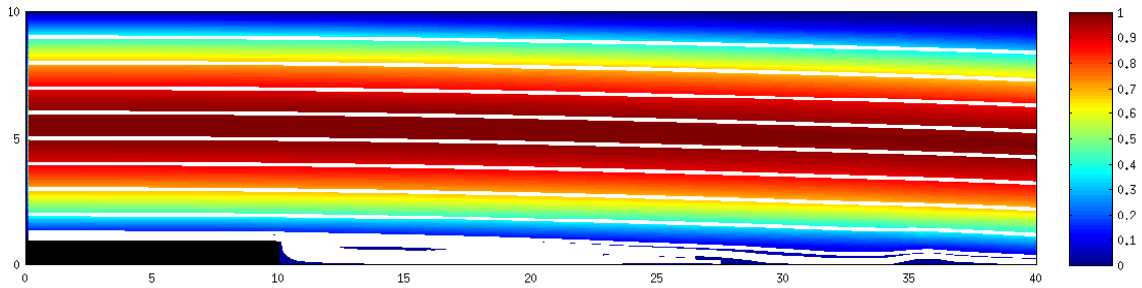
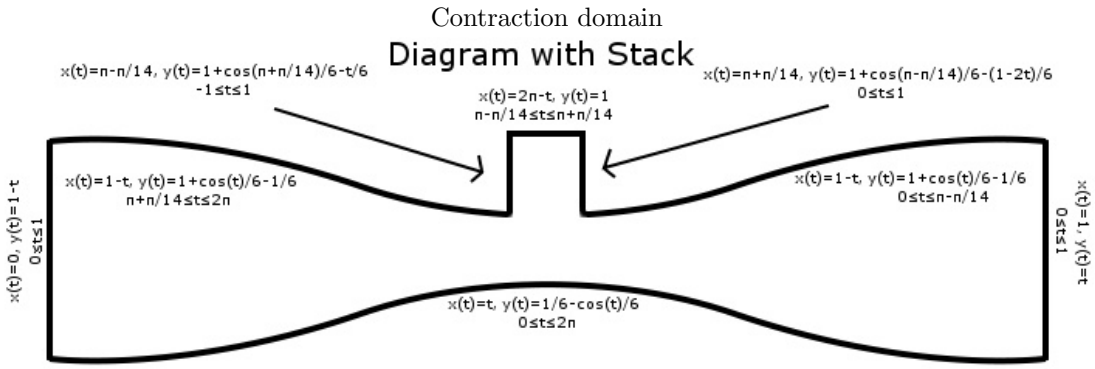
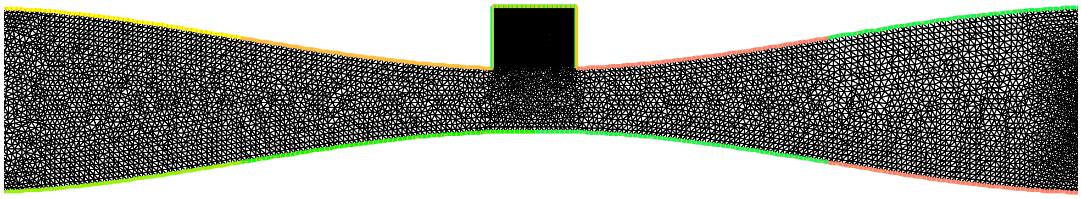


Figure 5.2: Fine mesh used for the resolved NSE solution and the coarse mesh used for the RMDM approximations.



Fine mesh - $260,378$ dof



Coarse Mesh - $10,820$ dof

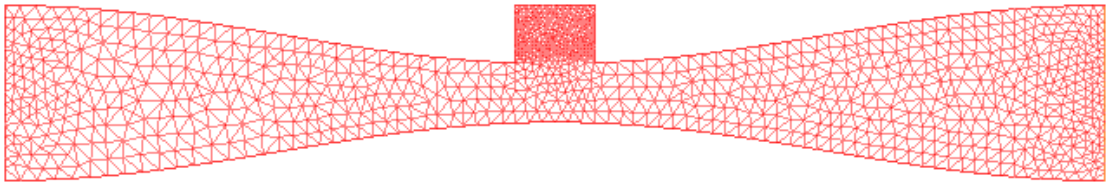


Figure 5.3: Diagram of the contraction domain, along with the fine and coarse meshes used in the computations for the contraction problem.

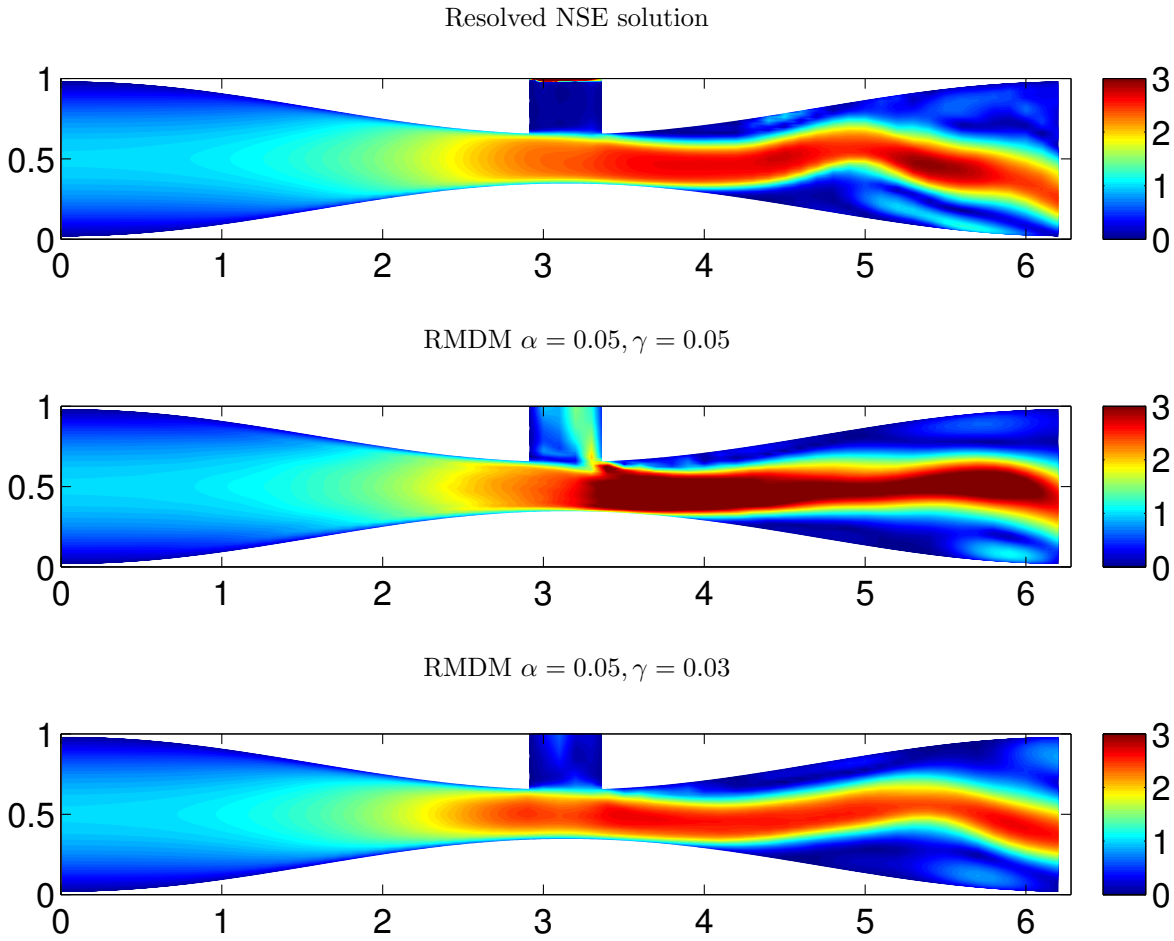


Figure 5.4: Speed contour plots of the resolved NSE solution as well as solutions of Algorithm 5.2.4. at $t = 4$.

Chapter 6

Analysis and approximation of the Cross model for quasi-Newtonian flows with defective boundary conditions

In this chapter we study a numerical method for the Cross model for generalized-Newtonian fluids with flow rate boundary conditions. The defective boundary problem is formulated as a constrained optimal control problem, where a flow balance is enforced on the inflow and outflow boundaries using a Neumann control. The control problem is analyzed for an existence result and the Lagrange multiplier rule. Finally, a decoupling solution algorithm is presented and numerical experiments are provided to validate robustness of the algorithm.

6.1 Modeling Equations and Preliminaries

The Cross model [16] for generalized-Newtonian fluids characterizes viscosity ν as a function of shear rate $|D(\mathbf{u})| = (D(\mathbf{u}) : D(\mathbf{u}))^{\frac{1}{2}}$,

$$\nu(|D(\mathbf{u})|) := \nu_\infty + \frac{(\nu_0 - \nu_\infty)}{1 + (\lambda|D(\mathbf{u})|)^{2-r}}, \tag{6.1}$$

where $\lambda > 0$ is a time constant, $1 \leq r \leq 2$ is a dimensionless rate constant and ν_∞ and ν_0 denote limiting viscosity values at an infinite and zero shear rate, respectively, assumed to satisfy $0 \leq \nu_\infty \leq \nu_0$. Throughout this chapter we will restrict our focus to the case where $\lambda = 1$.

We will suppose the boundary of Ω consists of inflow and outflow boundaries S_i, $i = 1, 2, ..., m$, and a wall boundary $\Gamma = \partial\Omega \setminus S$, where $S = \cup_{i=1}^m S_i$. Consider, as a model problem for steady, incompressible generalized-Newtonian flow, the two-field Cross model given by

$$\begin{aligned} -\nabla \cdot [(\nu_\infty + (\nu_0 - \nu_\infty)(1 + |D(\mathbf{u})|^{2-r})^{-1})D(\mathbf{u})] + \nabla p &= \mathbf{f} \text{ in } \Omega, & (6.2)\\ \nabla \cdot \mathbf{u} &= 0 \text{ in } \Omega, & (6.3)\\ \mathbf{u} &= \mathbf{0} \text{ on } \Gamma, & (6.4) \end{aligned}$$

We will consider the defective boundary problem where flow rates are specified on the inflow and outflow boundaries

$$S_i : \int_{S_i} \mathbf{u} \cdot \mathbf{n} \, dS = Q_i \text{ for } i = 1, 2, ..., m, \tag{6.5}$$

where $\sum_{i=1}^{m} Q_i = 0$ to satisfy the incompressibility condition.

For our choice of velocity space we will choose the subspace

$$\mathbf{X} := \{\mathbf{v} \in \mathbf{H}^1(\Omega) \,|\, \mathbf{v} = \mathbf{0} \text{ on } \Gamma\},$$

and for our pressure space we will take $Q := L^2(\Omega)$. Let $\boldsymbol{\sigma}$ denote the shear stress (tensor) of the fluid. If the flow rate boundary conditions (6.5) are replaced by the well defined Neumann boundary conditions

$$(\boldsymbol{\sigma} - pI) \cdot \mathbf{n} = \mathbf{g}_i \text{ on } S_i \text{ for } i = 1, 2, ..., m, \tag{6.6}$$

then the variational formulation of the problem is: Find $(\mathbf{u}, p) \in \mathbf{X} \times Q$ satisfying, for any $(\mathbf{v}, q) \in \mathbf{X} \times Q$,

$$\nu_\infty(D(\mathbf{u}), D(\mathbf{v})) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{u}), D(\mathbf{v}))$$
$$-(p, \nabla \cdot \mathbf{v}) = (\mathbf{f}, \mathbf{v}) + (\mathbf{g}, \mathbf{v})_S, \tag{6.7}$$
$$(\nabla \cdot \mathbf{u}, q) = 0. \tag{6.8}$$

Define the mapping $A : \mathbf{X} \to \mathbf{X}^*$ as follows: For any $\mathbf{u} \in \mathbf{X}$, $A\mathbf{u}$ satisfies

$$\langle A\mathbf{u}, \mathbf{v} \rangle = \nu_\infty(D(\mathbf{u}), D(\mathbf{v})) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{u}), D(\mathbf{v})) \quad \forall\, \mathbf{v} \in \mathbf{X}. \tag{6.9}$$

In [8] it was shown that for any $\mathbf{u} \in \mathbf{X}$, $A\mathbf{u}$ is strongly monotone and Lipschitz continuous, i.e. there exist constants $m, M > 0$ such that $A\mathbf{u}$ satisfies

$$\langle A\mathbf{u} - A\mathbf{v}, \mathbf{u} - \mathbf{v} \rangle \geq m \|\mathbf{u} - \mathbf{v}\|_1^2 \quad \forall\, \mathbf{v} \in \mathbf{X}, \tag{6.10}$$

and

$$\|A\mathbf{u} - A\mathbf{v}\|^* \leq M \|\mathbf{u} - \mathbf{v}\|_1 \quad \forall\, \mathbf{v} \in \mathbf{X}. \tag{6.11}$$

Note also that our choice of velocity and pressure spaces satisfy the inf-sup condition:

$$\inf_{q \in Q} \sup_{\mathbf{v} \in \mathbf{X}} \frac{(q, \nabla \cdot \mathbf{v})}{\|\mathbf{v}\|_1 \|q\|} \geq C > 0. \tag{6.12}$$

Then the existence of a unique solution to (6.7)-(6.8) follows from the strong monotonicity (6.10) and continuity (6.11) of A, as well as the inf-sup condition. Additionally, for some $C > 0$ we have the following *a priori* bound on solutions of the variational problem (6.7)-(6.8) [14]:

$$\|\mathbf{u}\|_1 + \|p\| \leq C(\|\mathbf{f}\|^* + \|\mathbf{g}\|_{L^2(S)}). \tag{6.13}$$

6.2 The Optimal Control Problem

6.2.1 Problem formulation

We continue by formulating the defective boundary problem (6.2)-(6.5) as an optimal control problem for flow rate matching. We choose as our control the normal component of the total stress on the defective boundaries

$$\mathbf{g} := \mathbf{g}_i := (\boldsymbol{\sigma} - pI) \cdot \mathbf{n} \text{ on } S_i,\ i = 1, 2, ..., m, \tag{6.14}$$

where $\mathbf{g} \in \mathbf{G} := \mathbf{L}^2(S)$. We want to minimize the penalized functional

$$\mathcal{J}(\mathbf{u}, p, \mathbf{g}) := \frac{1}{2} \sum_{i=1}^{m} \left(\int_{S_i} \mathbf{u} \cdot \mathbf{n}\, dS_i - Q_i \right)^2 + \frac{\epsilon}{2} \int_S |\mathbf{g}|^2 dS, \tag{6.15}$$

where the penalty parameter ϵ is a positive constant that measures the relative importance of the last term in (6.15). If we define the admissibility set as

$$U_{ad} := \{(\mathbf{u}, p, \mathbf{g}) \in \mathbf{X} \times Q \times \mathbf{G} : \mathcal{J}(\mathbf{u}, p, \mathbf{g}) < \infty\}, \tag{6.16}$$

then the defective boundary problem (6.2) - (6.5) can be reformulated as the following optimal control problem:

Find $(\mathbf{u}, p, \mathbf{g}) \in U_{ad}$ such that the functional (6.15) is minimized subject to (6.7) $-$ (6.8). (6.17)

Note that the problem of finding a solution to (6.17) is a constrained optimization problem. We will use the method of Lagrange multipliers to transform this constrained optimization problem into an unconstrained one. However, before we proceed, we will now show the existence of a solution to (6.17).

6.2.2 Existence of an optimal control solution

Existence of an optimal solution is established using standard arguments by the following theorem.

Theorem 6.2.1. *Given* $\mathbf{f} \in \mathbf{X}^*$*, there exists a solution* $(\mathbf{u}, p, \mathbf{g}) \in \mathbf{X} \times Q \times \mathbf{G}$ *of the optimal control*

problem (6.17).

Proof. We begin by noting that $(\mathbf{u}, p, \mathbf{0}) \in U_{ad}$ and hence the admissible set is clearly not empty. Assume that $\{(\mathbf{u}_n, p_n, \mathbf{g}_n)\}$ is a minimizing sequence in U_{ad}, i.e.

$$\lim_{n\to\infty} \mathcal{J}(\mathbf{u}_n, p_n, \mathbf{g}_n) = \inf_{(\mathbf{u},p,\mathbf{g})\in U_{ad}} \mathcal{J}(\mathbf{u}, p, \mathbf{g}).$$

By definition of the admissibility set (6.16) we have that $(\mathbf{u}_n, p_n, \mathbf{g}_n)$ satisfy (6.7)-(6.8) for any n, and that $\{\mathbf{g}_n\}$ is uniformly bounded in $\mathbf{L}^2(S)$. We also have that $\{(\mathbf{u}_n, p_n, \mathbf{g}_n)\}$ is uniformly bounded in $\mathbf{X} \times Q \times \mathbf{G}$ by the estimate (6.13). Thus, we can find subsequences, which we will denote $\{(\mathbf{u}_n, p_n, \mathbf{g}_n)\}$, such that

$$\mathbf{u}_n \rightharpoonup \tilde{\mathbf{u}} \text{ in } \mathbf{X},$$
$$p_n \rightharpoonup \tilde{p} \text{ in } Q,$$
$$\mathbf{g}_n \rightharpoonup \tilde{\mathbf{g}} \text{ in } \mathbf{G},$$

for some $(\tilde{\mathbf{u}}, \tilde{p}, \tilde{\mathbf{g}}) \in \mathbf{X} \times Q \times \mathbf{G}$. We note that because $A\mathbf{u}(\cdot)$ is a monotone operator, $A\mathbf{u}(\cdot)$ is sequential weak continuous, and thus

$$\lim_{n\to\infty} (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u}_n)|^{2-r})^{-1} D(\mathbf{u}_n), D(\mathbf{v})) = (\nu_0 - \nu_\infty)((1 + |D(\tilde{\mathbf{u}})|^{2-r})^{-1} D(\tilde{\mathbf{u}}), D(\mathbf{v})).$$

Therefore we may pass through the limit to see that $(\tilde{\mathbf{u}}, \tilde{p}, \tilde{\mathbf{g}})$ satisfies (6.7)-(6.8). Finally, by the weak lower semi-continuity of $\mathcal{J}$ we have that $(\tilde{\mathbf{u}}, \tilde{p}, \tilde{\mathbf{g}})$ is an optimal solution, and thus we have shown the existence of an optimal solution belonging to U_{ad}. □

6.3 The Optimality System

We use the Lagrange multiplier rule to derive the optimality system from which a solution to the optimal control problem (6.17) is obtained.

6.3.1 Existence of Lagrange multipliers

To show the existence of Lagrange multipliers we will follow the same approach as that used in [21]. We begin by stating the following abstract theorem on the existence of Lagrange multipliers for smooth constrained minimization problems on Banach spaces, which we will then tailor to our own specific optimization problem.

Lemma 6.3.1. *Let V and Y be two real Banach spaces, $\mathcal{J}$ a functional on V, and M a mapping from V to Y. Assume u is a solution of the following constrained minimization problem:*

$$find\ u \in V \text{ such that } \mathcal{J}(u) = \inf\{\mathcal{J}(v) \mid v \in V,\ M(v) = y_0\},$$

where y_0 is some fixed element of Y. Additionally, assume the following three conditions are satisfied:

- *M is Frechét differentiable in an open neighborhood of u and its Frechét derivative M' is continuous at u*
- *$\mathcal{J} : Nbhd(u) \subset V \to \mathbb{R}$ is Frechét differentiable at u with Frechét derivative $\mathcal{J}'$*
- *$M'(u)$ maps onto Y*

Then there exists a $\mu \in Y^*$ satisfying

$$-\mathcal{J}'(u) \cdot w + \langle \mu, M'(u) \cdot w) \rangle = 0 \ \ \forall\ w \in V.$$

Proof. See [64]. □

Define $V = \mathbf{X} \times Q \times \mathbf{G}$ and $Y = \mathbf{X}^* \times Q^*$, and let $M : V \to Y$ to be the (generalized) constraint equations, i.e., $M(\mathbf{u}, p, \mathbf{g}) = (\mathbf{f}, \phi)$ for all $(\mathbf{u}, p, \mathbf{g}) \in V$ and $(\mathbf{f}, \phi) \in Y$, if and only if,

$$\widetilde{A}(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) + c(\mathbf{v}, \mathbf{g}) = (\mathbf{f}, \mathbf{v}) \ \ \forall \mathbf{v} \in \mathbf{X}, \tag{6.18}$$

$$b(\mathbf{u}, q) = (\phi, q) \ \ \forall q \in Q, \tag{6.19}$$

where $\widetilde{A} : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$, $b : \mathbf{X} \times Q \to \mathbb{R}$ and $c : \mathbf{X} \times \mathbf{G} \to \mathbb{R}$ are defined as

$$\begin{aligned} \widetilde{A}(\mathbf{u}, \mathbf{v}) &:= \nu_\infty(D(\mathbf{u}), D(\mathbf{v})) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{u}), D(\mathbf{v})), && (6.20) \\ b(\mathbf{v}, \xi) &:= -(\xi, \nabla \cdot \mathbf{v}), && (6.21) \\ c(\mathbf{v}, \mathbf{g}) &:= -(\mathbf{g}, \mathbf{v})_S. && (6.22) \end{aligned}$$

Also define the bilinear form $\overline{A} : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ by

$$\begin{aligned} \overline{A}(\mathbf{w}, \mathbf{v}) &:= \nu_\infty(D(\mathbf{w}), D(\mathbf{v})) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{w}), D(\mathbf{v})) && (6.23) \\ &+ (\nu_0 - \nu_\infty)(r-2)((1 + |D(\mathbf{u})|^{2-r})^{-2} |D(\mathbf{u})|^{-r} (D(\mathbf{u}) : D(\mathbf{w})) D(\mathbf{u}), D(\mathbf{v})). \end{aligned}$$

We first show that $\overline{A}(\cdot, \cdot)$ is coercive and continuous.

Lemma 6.3.2. *Let $\overline{A}(\cdot, \cdot)$ be defined as in* (6.23)*. Then there exist constants $C_1, C_2 > 0$ such that $\overline{A}(\cdot, \cdot)$ satisfies, for any $\mathbf{w}, \mathbf{v} \in \mathbf{X}$,*

$$\overline{A}(\mathbf{w}, \mathbf{v}) \leq C_1 \|\mathbf{w}\|_1 \|\mathbf{v}\|_1, \tag{6.24}$$

$$\overline{A}(\mathbf{w}, \mathbf{w}) \geq C_2 \|\mathbf{w}\|_1^2. \tag{6.25}$$

Proof. Since $r - 2 \leq 0$ for $1 \leq r \leq 2$, we can use Holder's inequality to get

$$\begin{aligned} \overline{A}(\mathbf{w}, \mathbf{w}) &= \nu_0 \|D(\mathbf{w})\|^2 + (\nu_0 - \nu_\infty) \int_\Omega (1 + |D(\mathbf{u})|^{2-r})^{-1} |D(\mathbf{w})|^2 \, d\Omega && (6.26) \\ &\quad + (\nu_0 - \nu_\infty)(r-2) \int_\Omega (1 + |D(\mathbf{u})|^{2-r})^{-2} |D(\mathbf{u})|^{-r} (D(\mathbf{u}) : D(\mathbf{w}))^2 \, d\Omega \\ &\geq \nu_0 \|D(\mathbf{w})\|^2 + (\nu_0 - \nu_\infty) \int_\Omega (1 + |D(\mathbf{u})|^{2-r})^{-1} |D(\mathbf{w})|^2 \, d\Omega \\ &\quad + (\nu_0 - \nu_\infty)(r-2) \int_\Omega (1 + |D(\mathbf{u})|^{2-r})^{-2} |D(\mathbf{u})|^{2-r} |D(\mathbf{w})|^2 \, d\Omega \\ &\geq \nu_0 \|D(\mathbf{w})\|^2 + (\nu_0 - \nu_\infty)(r-1) \int_\Omega (1 + |D(\mathbf{u})|^{2-r})^{-1} |D(\mathbf{w})|^2 \, d\Omega. && (6.27) \end{aligned}$$

We know that $\nu_0 > \nu_\infty$, hence, since the integrand in (6.26) is positive we then see that there exists a $C_1 > 0$ satisfying

$$\overline{A}(\mathbf{w}, \mathbf{w}) \geq \nu_0 \|D(\mathbf{w})\|^2 \geq C_1 \|\mathbf{w}\|_1^2.$$

For continuity, begin by using Holder's inequality

$$\begin{aligned}
|\overline{A}(\mathbf{w},\mathbf{v})| &\leq \nu_\infty \|D(\mathbf{w})\| \|D(\mathbf{v})\| + (\nu_0-\nu_\infty)\int_\Omega (1+|D(\mathbf{u})|^{2-r})^{-1}|D(\mathbf{w})||D(\mathbf{v})|\, d\Omega \\
&\quad +(\nu_0-\nu_\infty)(2-r)\int_\Omega (1+|D(\mathbf{u})|^{2-r})^{-2}|D(\mathbf{u})|^{2-r}|D(\mathbf{w})||D(\mathbf{v})|\, d\Omega \\
&\leq \nu_\infty \|D(\mathbf{w})\| \|D(\mathbf{v})\| + (\nu_0-\nu_\infty)\int_\Omega (1+|D(\mathbf{u})|^{2-r})^{-1}|D(\mathbf{w})||D(\mathbf{v})|\, d\Omega \\
&\quad +(\nu_0-\nu_\infty)(2-r)\int_\Omega (1+|D(\mathbf{u})|^{2-r})^{-1}|D(\mathbf{w})||D(\mathbf{v})|\, d\Omega \\
&= \nu_\infty \|D(\mathbf{w})\| \|D(\mathbf{v})\| \\
&\quad +(\nu_0-\nu_\infty)(3-r)\int_\Omega (1+|D(\mathbf{u})|^{2-r})^{-1}|D(\mathbf{w})||D(\mathbf{v})|\, d\Omega. \qquad (6.28)
\end{aligned}$$

Since the first term in the integrand in (6.28) is smaller than one, we can drop it and use Holder's inequality to see there exists a $C_2 > 0$ such that

$$\overline{A}(\mathbf{w},\mathbf{v}) \leq C_2 \|\mathbf{w}\|_1 \|\mathbf{v}\|_1 .$$

□

Lemma 6.3.3. *Let M be defined as in* (6.18)-(6.19) *and let $(\mathbf{u},p,\mathbf{g})$ denote an optimal solution to* (6.17)*. Then M', the Frechét derivative of M, exists in an open neighborhood of $(\mathbf{u},p,\mathbf{g})$ and is defined by $M'(\mathbf{u},p,\mathbf{g})\cdot(\mathbf{w},\xi,\mathbf{h}) = (\overline{\mathbf{f}},\overline{\phi})$ for all $(\mathbf{w},\xi,\mathbf{h}) \in V$ and $(\overline{\mathbf{f}},\overline{\phi}) \in Y$, if and only if,*

$$\overline{A}(\mathbf{w},\mathbf{v}) + b(\mathbf{v},\xi) + c(\mathbf{v},\mathbf{h}) = (\overline{\mathbf{f}},\mathbf{v}) \ \ \forall \mathbf{v} \in \mathbf{X}, \qquad (6.29)$$

$$b(\mathbf{w},q) = (\overline{\phi},q) \ \ \forall q \in Q. \qquad (6.30)$$

Moreover, M' is continuous at $(\mathbf{u},p,\mathbf{g})$.

Proof. We will show that for $\epsilon > 0$ there exsits $\delta > 0$ such that

$$\|M(\mathbf{u}_1,p_1,\mathbf{g}_1) - M(\mathbf{u}_2,p_2,\mathbf{g}_2) - M'(\mathbf{u},p,\mathbf{g})\cdot(\mathbf{u}_1-\mathbf{u}_2,p_1-p_2,\mathbf{g}_1-\mathbf{g}_2)\|_Y \ < \ \epsilon, \qquad (6.31)$$

if $\|\mathbf{u}_1-\mathbf{u}_2,p_1-p_2,\mathbf{g}_1-\mathbf{g}_2\|_V < \delta$.

Using (6.11) and inequalities shown in (6.28),

$$
\begin{aligned}
&(M(\mathbf{u}_1,p_1,\mathbf{g}_1)-M(\mathbf{u}_2,p_2,\mathbf{g}_2)-M'(\mathbf{u},p,\mathbf{g})\cdot(\mathbf{u}_1-\mathbf{u}_2,p_1-p_2,\mathbf{g}_1-\mathbf{g}_2),(\mathbf{v},q))\\
&=\widetilde{A}(\mathbf{u}_1,\mathbf{v})-\widetilde{A}(\mathbf{u}_2,\mathbf{v})+b(\mathbf{v},p_1-p_2)+c(\mathbf{v},\mathbf{g}_1-\mathbf{g}_2)+b(\mathbf{u}_1-\mathbf{u}_2,q)\\
&\qquad -\overline{A}(\mathbf{u}_1-\mathbf{u}_2,\mathbf{v})-b(\mathbf{v},p_1-p_2)-c(\mathbf{v},\mathbf{g}_1-\mathbf{g}_2)-b(\mathbf{u}_1-\mathbf{u}_2,q)\\
&=\nu_\infty(D(\mathbf{u}_1),D(\mathbf{v}))+(\nu_0-\nu_\infty)((1+|D(\mathbf{u}_1)|^{2-r})^{-1}D(\mathbf{u}_1),D(\mathbf{v}))\\
&\quad -\nu_\infty(D(\mathbf{u}_2),D(\mathbf{v}))+(\nu_0-\nu_\infty)((1+|D(\mathbf{u}_2)|^{2-r})^{-1}D(\mathbf{u}_2),D(\mathbf{v}))\\
&\quad -\big[\nu_\infty(D(\mathbf{u}_1-\mathbf{u}_2),D(\mathbf{v}))+(\nu_0-\nu_\infty)((1+|D(\mathbf{u})|^{2-r})^{-1}D(\mathbf{u}_1-\mathbf{u}_2),D(\mathbf{v}))\\
&\quad +(\nu_0-\nu_\infty)(r-2)((1+|D(\mathbf{u})|^{2-r})^{-2}|D(\mathbf{u})|^{-r}(D(\mathbf{u}):D(\mathbf{u}_1-\mathbf{u}_2))D(\mathbf{u}),D(\mathbf{v}))\big]\\
&\le M\|\mathbf{u}_1-\mathbf{u}_2\|_1\|\mathbf{v}\|_1+\nu_\infty\|D(\mathbf{u}_1-\mathbf{u}_2)\|_0\|D(\mathbf{v}))\|_0\\
&\quad +(\nu_0-\nu_\infty)(3-r)\int_\Omega(1+|D(\mathbf{u})|^{2-r})^{-1}|D(\mathbf{u}_1-\mathbf{u}_2)||D(\mathbf{v})|\,d\Omega\\
&\le (M+\nu_\infty+(\nu_0-\nu_\infty)(3-r))\|\mathbf{u}_1-\mathbf{u}_2\|_1\|\mathbf{v}\|_1. \qquad (6.32)
\end{aligned}
$$

The choice of $\delta=\frac{\epsilon}{M+\nu_\infty+(\nu_0-\nu_\infty)(3-r)}$ then implies (6.31). Continuity of M' follows from Lemma 6.3.2. □

It is straight-forward to show that $\mathcal{J}$ as defined in (6.15) is Frechét differentiable at $(\mathbf{u},p,\mathbf{g})$. We now show that $M'(\mathbf{u},p,\mathbf{g})$ maps onto Y.

Lemma 6.3.4. *The operator $M'(\mathbf{u},p,\mathbf{g})$ is onto Y.*

Proof. First, note that for any $(\overline{\mathbf{f}},\overline{\phi})\in Y$, the problem (6.29) - (6.30) is well-posed. Existence and uniqueness of its solution follow from Lemma 6.3.2, the inf-sup condition (6.12) and the Lax-Milgram theorem. Thus, we are able to find a $(\mathbf{w},\xi)\in\mathbf{X}\times Q$ satisfying the equations

$$
\begin{aligned}
&\nu_\infty(D(\mathbf{w}),D(\mathbf{v}))+(\nu_0-\nu_\infty)((1+|D(\mathbf{u})|^{2-r})^{-1}D(\mathbf{w}),D(\mathbf{v}))-(\xi,\nabla\cdot\mathbf{v})\\
&+(\nu_0-\nu_\infty)(r-2)((1+|D(\mathbf{u})|^{2-r})^{-2}|D(\mathbf{u})|^{-r}(D(\mathbf{u}):D(\mathbf{w}))D(\mathbf{u}),D(\mathbf{v}))=(\overline{\mathbf{f}},\mathbf{v})\ \forall\ \mathbf{v}\in\mathbf{X},\\
&(\nabla\cdot\mathbf{w},q)=(\overline{\phi},q)\ \forall\ q\in Q.
\end{aligned}
$$

Now, choose $\mathbf{h}=\mathbf{0}\in\mathbf{G}$. Then clearly $M'(\mathbf{u},p,\mathbf{g})\cdot(\mathbf{w},\xi,\mathbf{h})=(\overline{\mathbf{f}},\overline{\phi})$ and hence $M'(\mathbf{u},p,\mathbf{g})$ is onto Y. □

Theorem 6.3.5. *Let* $(\mathbf{u}, p, \mathbf{g}) \in V$ *denote an optimal solution of* (6.17). *Then there exists a nonzero Lagrange multiplier* $(\boldsymbol{\gamma}, \sigma) \in \mathbf{X} \times Q$ *satisfying*

$$-\mathcal{J}'(\mathbf{u}, p, \mathbf{g}) \cdot (\mathbf{v}, q, \mathbf{h}) + \langle M'(\mathbf{u}, p, \mathbf{g}) \cdot (\mathbf{v}, q, \mathbf{h}), (\boldsymbol{\gamma}, \sigma) \rangle = 0 \quad \forall\, (\mathbf{v}, q, \mathbf{h}) \in V. \tag{6.33}$$

Proof. Lemmas 6.3.3 and 6.3.4, along with the fact that $\mathcal{J}$ is Frechét differentiable at $(\mathbf{u}, p, \mathbf{g})$ allow us to use Lemma 6.3.1 to show the existence of a $(\boldsymbol{\gamma}, \sigma) \in \mathbf{X} \times Q$ satisfying (6.33). □

6.3.2 The Lagrange multiplier rule

We now derive the optimality system based on the Lagrange multiplier rule. Introduce the Lagrange multipliers $\mathbf{w} \in \mathbf{X}$ and $\xi \in Q$ and define the Lagrangian

$$\begin{aligned} \mathcal{L}(\mathbf{u}, p, \mathbf{g}, \mathbf{w}, \xi) = \mathcal{J}(\mathbf{u}, p, \mathbf{g}) + \nu_\infty (D(\mathbf{u}), D(\mathbf{w})) - (p, \nabla \cdot \mathbf{w}) + (\nabla \cdot \mathbf{u}, \xi) \\ + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{u}), D(\mathbf{w})) - (\mathbf{f}, \mathbf{w}) - (\mathbf{g}, \mathbf{w})_S, \end{aligned} \tag{6.34}$$

for any $(\mathbf{u}, p, \mathbf{g}, \mathbf{w}, \xi) \in \mathbf{X} \times Q \times \mathbf{G} \times \mathbf{X} \times Q$. We now want to find stationary points of $\mathcal{L}(\mathbf{u}, p, \mathbf{g}, \mathbf{w}, \xi)$ over the product space $\mathbf{X} \times Q \times \mathbf{G} \times \mathbf{X} \times Q$. Variations in the Lagrange multipliers $\mathbf{w}, \xi$ yields the state equations

$$\begin{aligned} \nu_\infty (D(\mathbf{u}), D(\mathbf{v})) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{u}), D(\mathbf{v})) \\ -(p, \nabla \cdot \mathbf{v}) = (\mathbf{f}, \mathbf{v}) + (\mathbf{g}, \mathbf{v})_S, \end{aligned} \tag{6.35}$$

$$(\nabla \cdot \mathbf{u}, q) = 0. \tag{6.36}$$

for any $(\mathbf{v}, q) \in \mathbf{X} \times Q$. Variations in the state variables $\mathbf{u}, p$ yield the adjoint equations

$$\begin{aligned} \nu_\infty (D(\mathbf{w}), D(\mathbf{v})) - (\xi, \nabla \cdot \mathbf{v}) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u})|^{2-r})^{-1} D(\mathbf{w}), D(\mathbf{v})) \\ + (\nu_0 - \nu_\infty)(r-2)((1 + |D(\mathbf{u})|^{2-r})^{-2} |D(\mathbf{u})|^{-r} (D(\mathbf{u}) : D(\mathbf{w})) D(\mathbf{u}), D(\mathbf{v})) \\ = -\sum_{i=1}^{m} (\int_{S_i} \mathbf{u} \cdot \mathbf{n}\, dS_i - Q_i)(\int_{S_i} \mathbf{v} \cdot \mathbf{n}\, dS_i)\ \forall \mathbf{v} \in \mathbf{X}, \end{aligned} \tag{6.37}$$

$$(\nabla \cdot \mathbf{w}, q) = 0\ \forall q \in Q, \tag{6.38}$$

and variations in the control $\mathbf{g}$ yield the necessary condition

$$(\mathbf{g}, \mathbf{h})_S = \frac{1}{\epsilon}(\mathbf{w}, \mathbf{h})_S \ \forall \mathbf{h} \in \mathbf{G}. \tag{6.39}$$

Thus, an optimal solution to (6.17) must satisfy the optimality system formed by (6.35)-(6.39). The adjoint problem (6.37)-(6.38) can be rewritten as

$$\overline{A}(\mathbf{w}, \mathbf{v}) + b(\mathbf{v}, \xi) = F(\mathbf{v}) \, \forall \, \mathbf{v} \in \mathbf{X}, \tag{6.40}$$

$$b(\mathbf{w}, q) = 0 \quad \forall \, q \in Q, \tag{6.41}$$

where $F : \mathbf{X} \to \mathbb{R}$ is defined by

$$F(\mathbf{v}) = -\sum_{i=1}^{m} (\int_{S_i} \mathbf{u} \cdot \mathbf{n} \, dS_i - Q_i)(\int_{S_i} \mathbf{v} \cdot \mathbf{n} \, dS_i), \tag{6.42}$$

and it is well-posed.

Note that using (6.29)-(6.30), the equation (6.33) in theorem 6.3.5 can be equivalently rewritten as the adjoint equations (6.37)-(6.38) and necessary condition (6.39). Since $(\mathbf{u}, p, \mathbf{g})$ are an optimal solution of (6.17), they necessarily satisfy the state equations (6.35)-(6.36), and thus we see that the optimality system is satisfied.

6.4 Steepest descent approach

6.4.1 Finite element approximation

We continue by defining an approximate optimality system using finite element methods. Suppose τ_h is a triangulation of Ω such that $\overline{\Omega} = \{\cup K : K \in \tau_h\}$. Let $(\mathbf{X}_h, Q_h) \subset (\mathbf{X}, Q)$ be LBB stable finite element spaces defined on τ_h. The finite element approximation of the state equations (6.35)-(6.36) is as follows: find $(\mathbf{u}_h, p_h) \in (\mathbf{X}_h, Q_h)$ satisfying, for all $(\mathbf{v}_h, q_h) \in (\mathbf{X}_h, Q_h)$,

$$\nu_\infty(D(\mathbf{u}_h), D(\mathbf{v}_h)) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u}_h)|^{2-r})^{-1} D(\mathbf{u}_h), D(\mathbf{v}_h))$$

$$-(p_h, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}, \mathbf{v}_h) + (\mathbf{g}_h, \mathbf{v}_h)_S, \tag{6.43}$$

$$(\nabla \cdot \mathbf{u}_h, q_h) = 0. \tag{6.44}$$

The finite element approximation of the adjoint equations (6.37)-(6.38) is: find $(\mathbf{w}_h, \xi_h) \in (\mathbf{X}_h, Q_h)$ satisfying, for all $(\mathbf{v}_h, q_h) \in (\mathbf{X}_h, Q_h)$,

$$\begin{aligned}
&\nu_\infty(D(\mathbf{w}_h), D(\mathbf{v}_h)) - (\xi, \nabla \cdot \mathbf{v}_h) + (\nu_0 - \nu_\infty)((1 + |D(\mathbf{u}_h)|^{2-r})^{-1} D(\mathbf{w}_h), D(\mathbf{v}_h)) \\
&+(\nu_0 - \nu_\infty)(r-2)((1 + |D(\mathbf{u}_h)|^{2-r})^{-2} |D(\mathbf{u}_h)|^{-r} (D(\mathbf{u}_h) : D(\mathbf{w}_h)) D(\mathbf{u}_h), D(\mathbf{v}_h)) \\
&\qquad = -\sum_{i=1}^{m} \left(\int_{S_i} \mathbf{u}_h \cdot \mathbf{n} \, dS_i - Q_i \right) \left(\int_{S_i} \mathbf{v}_h \cdot \mathbf{n} \, dS_i \right), \qquad (6.45) \\
&\qquad (\nabla \cdot \mathbf{w}_h, q_h) = 0. \qquad (6.46)
\end{aligned}$$

Finally, let $\mathbf{G}_h$ be a finite dimensional subspace of $\mathbf{G}$. Then the finite element approximation of the necessary condition is

$$(\mathbf{g}_h, \mathbf{h}_h)_S = \frac{1}{\epsilon} (\mathbf{w}_h, \mathbf{h}_h)_S \; \forall \, \mathbf{h}_h \in \mathbf{G}_h. \qquad (6.47)$$

6.4.2 Steepest descent algorithm

In practice, the size of the optimality system is very large, and therefore the state and adjoint systems must be decoupled. To do so in this study, we will implement an optimization algorithm presented in [48]. This optimization algorithm uses a gradient method for minimizing the functional $\mathcal{M}(\mathbf{g}) := \mathcal{J}(\mathbf{u}(\mathbf{g}), p(\mathbf{g}), \mathbf{g})$ where $\mathcal{J}$ is defined as in (6.15). This method is given by

$$\mathbf{g}_{k+1} = \mathbf{g}_k - \rho_k \frac{d\mathcal{M}}{d\mathbf{g}_k}, \qquad (6.48)$$

where ρ_k is a step-size chosen in an appropriate fashion. By the optimality condition (6.39), the gradient $\frac{d\mathcal{M}}{d\mathbf{g}_k}$ can be determined by a solution of the adjoint system:

$$\frac{d\mathcal{M}}{d\mathbf{g}_k} = \epsilon \mathbf{g}_k - \mathbf{w}_k|_S. \qquad (6.49)$$

Assuming the step size is dependent on the penalty parameter ϵ, i.e., $\rho_k = \frac{\alpha_k}{\epsilon}$ and using (6.49), the steepest descent algorithm for $\mathbf{g}_k$ reads

$$\mathbf{g}_{k+1} = (1 - \alpha_k)\mathbf{g}_k + \frac{\alpha_k}{\epsilon} \mathbf{w}_k|_S. \qquad (6.50)$$

Algorithm 6.4.1. *(Steepest descent algorithm)*

Choose an initial control $\mathbf{g}_0$.

For $k = 0, 1, \ldots$

1. *Solve (6.43)-(6.44) for* $(\mathbf{u}_k^h, p_k^h)$.
2. *Solve (6.45)-(6.46) for* $(\mathbf{w}_k^h, \xi_k^h)$.
3. *Update the control by (6.50).*

6.5 Numerical Results

In this section we present a model flow problem subject to specified flow rates on both inflow and outflow boundaries. The problem that we present here is a more complicated version of a numerical experiment used in [48, 29]. The domain of the test problem consists of a square box $(1, 6) \times (0, 5)$ connected to two inlet and two outlet channels, each of which has length and width equal to one (see Figure 6.5). The boundary of the domain contains two inflow boundaries

$$S_1 := \{(x, y) : x = 0,\ 1 < y < 2\},$$
$$S_2 := \{(x, y) : x = 0,\ 3 < y < 4\},$$

and two outflow boundaries

$$S_3 := \{(x, y) : x = 7,\ 2 < y < 3\},$$
$$S_4 := \{(x, y) : y = 6,\ 3 < x < 4\},$$

on which only flow rates are specified, as well as a wall boundary on which no-slip boundary conditions are enforced. The flow rate used for the inflow boundaries S_1 and S_2 were specified to be 2 and 1, respectively. The flow rate for the right outflow boundary S_3 was set to $5/2$, and hence the flow rate for the top outflow boundary S_4 was specified to $1/2$.

As a choice of finite element spaces Taylor-Hood elements (P2 elements for the velocity, P1 elements for the pressure) were employed. All of the computations were performed on a triangular mesh that provided $27,186$ degrees of freedom for the velocity and $3,469$ degrees of freedom for the

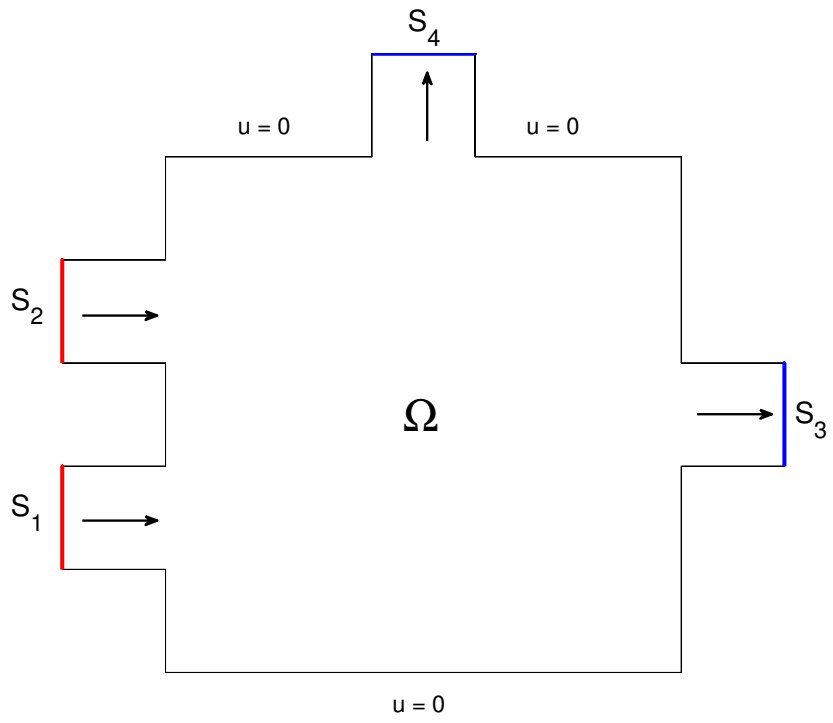


Figure 6.1: Domain for the flow problem. Red indicates an inflow boundary. Blue indicates an outflow boundary.

pressure. In the finite element approximation of the optimal control system, the limiting viscosity values ν_0 and ν_∞ were set to 20 and 1, respectively. The steepest descent algorithm was terminated once a tolerance of $\mathcal{J}(\mathbf{u}_h, p_h, \mathbf{g}_h) < 10^{-6}$ was reached. Finally, all of the computations were performed using the software package *FreeFem++* [39].

We began by selecting $r = 1.5$, corresponding to a shear thinning fluid. The steepest descent algorithm was performed using an initial guess of $\mathbf{g}_0 = [0.1, 0.1]$. Figure 6.2 displays streamlines of the approximation on top of a contour plot of the magnitude of the velocity. Figure 6.3 shows the velocity profile on each of the inflow and outflow boundaries. We note that even for a more complex flow domain with multiple inflow and outflow boundaries, the steepest descent algorithm produces a smooth solution to the optimization problem. It is clear from Figure 6.3 that the relative size of the velocity profiles of the approximation on the inflow and outflow boundaries correctly match the relative sizes of the specified flow rates on those same defective boundaries. We then recomputed using all of the same parameters but changing $r = 2.0$ to simulate a Newtonian fluid. While not shown here, the approximation closely resembled that which was produced when using $r = 1.5$. That

said, there were subtle differences in the flow approximation, most notably the size and shape of the four eddies in the corners of the square portion of the domain.

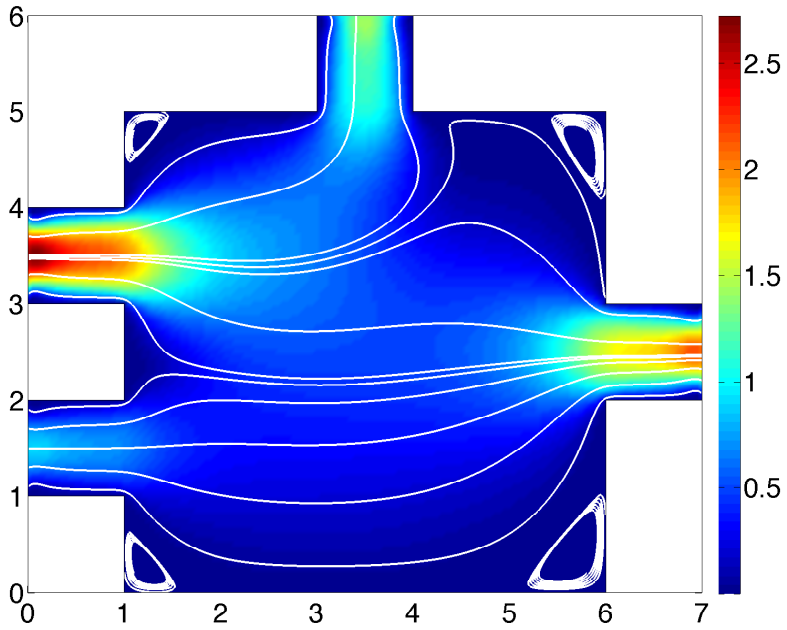


Figure 6.2: Streamlines and magnitude of the velocity approximation for $r = 1.5$ and $\mathbf{g}_0 = [0.1, 0.1]$.

We then recomputed approximations of the defective boundary problem using an initial guess of $\mathbf{g}_0 = [10, 10]$ and $r = 1.5$. Figure 6.4 contains plots of both the streamlines and velocity contours, and Figure 6.5 depicts the flow profile on each of the defective boundaries. From both the streamlines and the velocity profiles on the inflow and outflow boundaries it is clear that by using a different initial guess we have reached a different solution to the optimal control problem than that shown above.

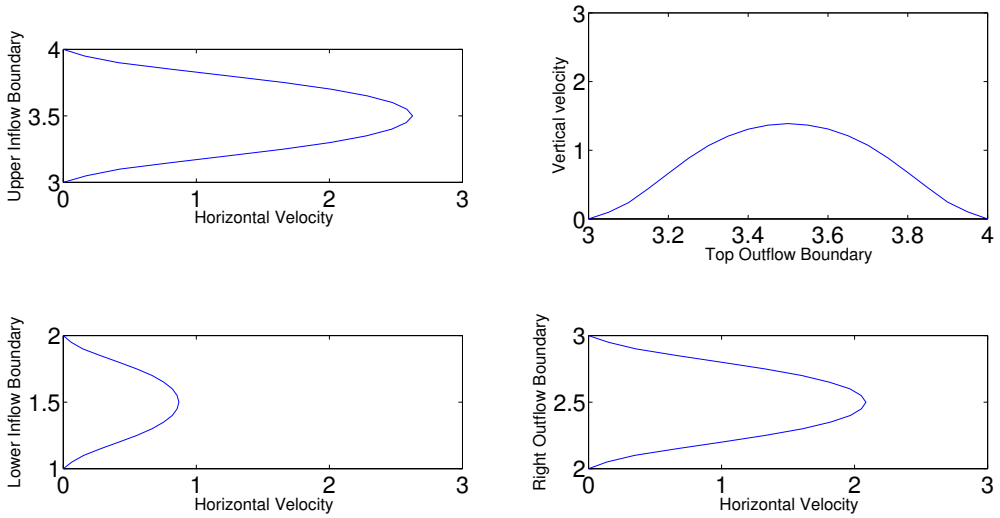


Figure 6.3: Inflow and outflow velocity profiles for $r = 1.5$ and $\mathbf{g}_0 = [0.1, 0.1]$.

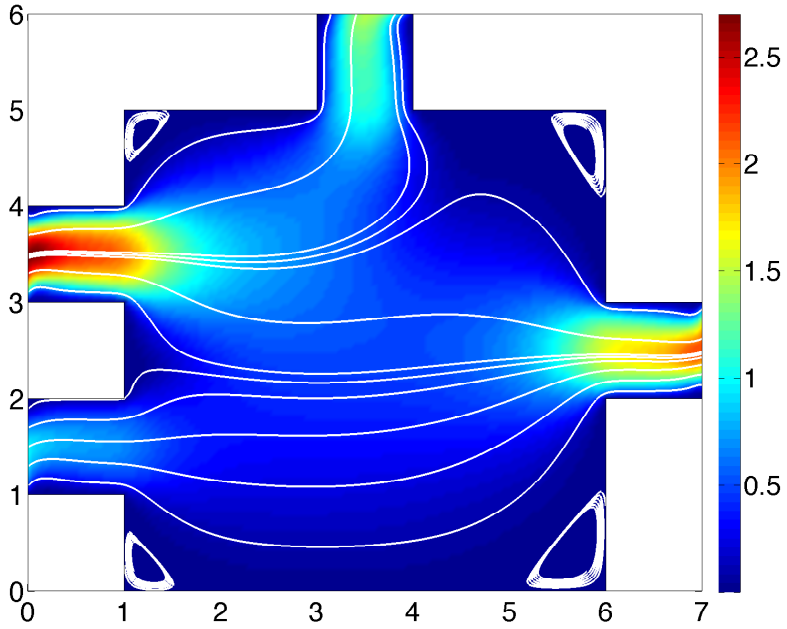


Figure 6.4: Streamlines and magnitude of the velocity approximation for $r = 1.5$ and $\mathbf{g}_0 = [10, 10]$.

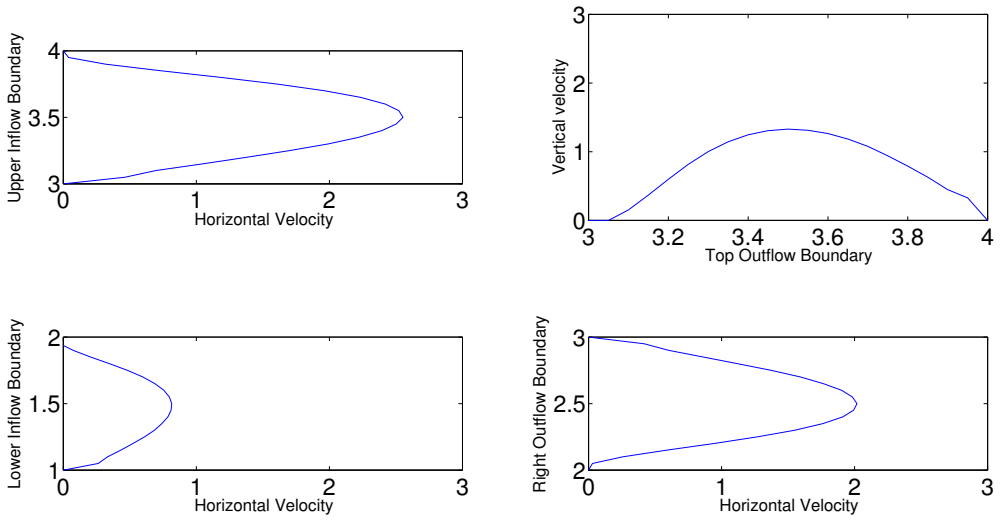


Figure 6.5: Inflow and outflow velocity profiles for $r = 1.5$ and $\mathbf{g}_0 = [10, 10]$.

Chapter 7

Approximation of viscoelastic flows with defective boundary conditions

In this chapter we investigate numerical algorithms for viscoelastic fluid flows with defective boundary conditions, where only flow rates or mean pressures are prescribed on parts of the boundary. The defective boundary condition problem is formulated as a minimization problem, where we seek boundary conditions of the flow equations which yield an optimal functional value. Two different approaches are considered in developing computational algorithms for the constrained optimization problem, and results of numerical experiments are presented to compare performance of the algorithms.

7.1 Model equations

The Johnson-Segalman modeling equations for viscoelastic flows are given by

$$\boldsymbol{\sigma} + \lambda(\mathbf{u}\cdot\nabla)\boldsymbol{\sigma} + \lambda g_a(\boldsymbol{\sigma}, \nabla\mathbf{u}) - 2\,\alpha\, D(\mathbf{u}) = \mathbf{0} \quad \text{in } \Omega\,, \tag{7.1}$$

$$-\nabla\cdot\boldsymbol{\sigma} - 2(1-\alpha)\,\nabla\cdot D(\mathbf{u}) + \nabla p = \mathbf{f} \quad \text{in } \Omega\,, \tag{7.2}$$

$$\nabla\cdot\ \mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{7.3}$$

where $\boldsymbol{\sigma}$ denotes the shear stress (tensor) and λ is the Weissenberg number, defined as the product of the relaxation time and a characteristic strain rate. Assume that p has zero mean value over Ω. In (7.1) and (7.2), α is a number such that $0 < \alpha < 1$ which may be considered as the fraction of viscoelastic viscosity. In (7.1), $g_a(\boldsymbol{\sigma}, \nabla \mathbf{u})$ is defined by

$$g_a(\boldsymbol{\sigma}, \nabla \mathbf{u}) := \frac{1-a}{2}(\boldsymbol{\sigma}\nabla \mathbf{u} + \nabla \mathbf{u}^T \boldsymbol{\sigma}) - \frac{1+a}{2}(\nabla \mathbf{u}\, \boldsymbol{\sigma} + \boldsymbol{\sigma}\nabla \mathbf{u}^T) \tag{7.4}$$

for $a \in [-1, 1]$.

Without loss of generality, let S_i for $i = 1, 2, \ldots, j$ $(j < m)$ be inflow boundaries and $S_{in} := \cup_{i=1}^{j} S_i$. In the standard case the governing equations are completed with the boundary conditions of Dirichlet or Neumann type on inflow, outflow boundaries such as

$$\mathbf{u} = \mathbf{u}_{BC} \text{ on } S, \qquad \boldsymbol{\sigma} = \boldsymbol{\sigma}_{BC} \text{ on } S_{in},$$

or

$$(\boldsymbol{\sigma} + 2(1-\alpha)\, D(\mathbf{u}) - pI)\, \mathbf{n} = \mathbf{h}_{BC} \text{ on } S, \qquad \boldsymbol{\sigma} = \boldsymbol{\sigma}_{BC} \text{ on } S_{in}.$$

We first consider the flow problem with the flow rate conditions only on S:

$$\int_{S_i} \mathbf{u} \cdot \mathbf{n}\, dS = Q_i \text{ for } i = 1, \ldots, m, \tag{7.5}$$

where $\sum_{i=1}^{m} Q_i = 0$ in order to satisfy the incompressibility condition.

The defective boundary condition problem (7.1)-(7.5) can be formulated as a minimization problem for flow rate matching. Choose the force acting on the fluid

$$\mathbf{g}_N := \mathbf{g}_{N\,i} := (\boldsymbol{\sigma} + 2(1-\alpha)\, D(\mathbf{u}) - pI)\mathbf{n} \text{ on } S_i, \quad i = 1, 2, \ldots, m \tag{7.6}$$

and the stress condition

$$\mathbf{g}_D := \mathbf{g}_{D\,i} := \boldsymbol{\sigma} \text{ on } S_i, \quad i = 1, 2, \ldots, j \tag{7.7}$$

as controls.

For the flow rate conditions (7.5), consider minimizing the penalized functional

$$\mathcal{J}(\mathbf{u},p,\boldsymbol{\sigma},\mathbf{g}) := \frac{1}{2}\sum_{i=1}^{m}\left(\int_{S_i}\mathbf{u}\cdot\mathbf{n}\,dS_i - Q_i\right)^2 + \frac{\epsilon_1}{2}\int_S |\mathbf{g}_N|^2 dS + \frac{\epsilon_2}{2}\int_{S_{in}} |\mathbf{g}_D|^2 dS, \tag{7.8}$$

where $\mathbf{g}_N$, $\mathbf{g}_D$ are controls chosen and ϵ_1, ϵ_2 are penalty parameters.

Let $\mathbf{X}$, Q, $\boldsymbol{\Sigma}$ denote the function spaces defined in Ω for $\mathbf{u}$, p, and $\boldsymbol{\sigma}$, respectively. Also let G_N, G_D denote the suitably chosen control spaces for $\mathbf{g}_N$, $\mathbf{g}_D$, respectively. The weak formulation for (7.1)-(7.3), (7.6)-(7.7) is then given by, $\forall\boldsymbol{\tau}\in\boldsymbol{\Sigma}$, $\mathbf{v}\in\mathbf{X}$, and $q\in Q$,

$$\begin{aligned}
(\boldsymbol{\sigma},\boldsymbol{\tau}) + \lambda((\mathbf{u}\cdot\nabla)\boldsymbol{\sigma},\boldsymbol{\tau}) + \lambda(g_a(\boldsymbol{\sigma},\nabla\mathbf{u}),\boldsymbol{\tau}) - 2\alpha(D(\mathbf{u}),\boldsymbol{\tau}) + (\boldsymbol{\sigma},\boldsymbol{\tau})_{S_{in}} &= (\mathbf{g}_D,\boldsymbol{\tau})_{S_{in}}, &(7.9)\\
(\boldsymbol{\sigma},D(\mathbf{v})) + 2(1-\alpha)(D(\mathbf{u}),D(\mathbf{v})) - (p,\nabla\cdot\mathbf{v}) - (\mathbf{f},\mathbf{v}) &= (\mathbf{g}_N,\mathbf{v})_S, &(7.10)\\
(q,\nabla\cdot\mathbf{u}) &= 0. &(7.11)
\end{aligned}$$

Now the defective boundary condition problem is formulated as the following optimization problem:

$$\begin{gathered}
\text{find } (\mathbf{u},p,\boldsymbol{\sigma}) \text{ and } (\mathbf{g}_N,\mathbf{g}_D) \text{ such that the functional (7.8) is minimized}\\
\text{subject to (7.9)-(7.11).}
\end{gathered} \tag{7.12}$$

Remark 7.1.1. *In this chapter we will not consider any analytical results for the optimization problem (7.12) in specified function spaces. All algorithms will be derived under the assumption that an optimal solution and Lagrange multipliers exist in appropriately chosen function spaces.*

7.2 The Optimality system

We use the Lagrange multiplier rule to derive the optimality system and the first-order necessary condition that the optimal solution must satisfy. Define the Lagrangian

$$\begin{aligned}
\mathcal{L}(\mathbf{u},p,\boldsymbol{\sigma},\mathbf{g}_N,\mathbf{g}_D,\mathbf{w},\xi,\boldsymbol{\eta}) &= \mathcal{J}(\mathbf{u},p,\boldsymbol{\sigma},\mathbf{g}) + (\boldsymbol{\sigma},\boldsymbol{\eta}) + \lambda((\mathbf{u}\cdot\nabla)\boldsymbol{\sigma},\boldsymbol{\eta})\\
&+\lambda(g_a(\boldsymbol{\sigma},\nabla\mathbf{u}),\boldsymbol{\eta}) - 2\alpha(D(\mathbf{u}),\boldsymbol{\eta}) + (\boldsymbol{\sigma},\boldsymbol{\eta})_{S_{in}} - (\mathbf{g}_D,\boldsymbol{\eta})_{S_{in}} + (\boldsymbol{\sigma},D(\mathbf{w}))\\
&+2(1-\alpha)(D(\mathbf{u}),D(\mathbf{w})) - (p,\nabla\cdot\mathbf{w}) - (\mathbf{f},\mathbf{w}) - (\mathbf{g}_N,\mathbf{w})_S - (\xi,\nabla\cdot\mathbf{u}),
\end{aligned} \tag{7.13}$$

where $(\mathbf{w}, \xi, \boldsymbol{\eta}) \in \mathbf{X} \times Q \times \boldsymbol{\Sigma}$ are adjoint velocity, pressure and stress, respectively. The adjoint momentum equation is derived by $\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = 0$, as

$$\begin{aligned} -2\alpha(\boldsymbol{\eta}, D(\mathbf{v})) + \lambda\left(((\mathbf{v}\cdot\nabla)\boldsymbol{\sigma}, \boldsymbol{\eta}) + \phi((\boldsymbol{\sigma}, \boldsymbol{\eta}), \mathbf{v})\right) + 2(1-\alpha)(D(\mathbf{w}), D(\mathbf{v})) \\ -(\xi, \nabla\cdot\mathbf{v}) = -\sum_{i=1}^{m}\left(\int_{S_i}\mathbf{u}\cdot\mathbf{n}\, dS_i - Q_i\right)\int_{S_i}\mathbf{v}\cdot\mathbf{n}\, dS_i \quad \forall \mathbf{v} \in \mathbf{X}\,, \end{aligned} \tag{7.14}$$

where

$$(\phi(\boldsymbol{\sigma}, \boldsymbol{\eta}), \mathbf{v}) := \frac{1-a}{2}\left[(\boldsymbol{\sigma}^T\boldsymbol{\eta}, \nabla\mathbf{v}) + (\boldsymbol{\eta}\boldsymbol{\sigma}^T, (\nabla\mathbf{v})^T)\right] - \frac{1+a}{2}\left[(\boldsymbol{\eta}\boldsymbol{\sigma}^T, \nabla\mathbf{v}) + (\boldsymbol{\sigma}^T\boldsymbol{\eta}, (\nabla\mathbf{v})^T)\right]. \tag{7.15}$$

Note that by integrating by parts, the ϕ term may be written as

$$\begin{aligned} (\phi(\boldsymbol{\sigma}, \boldsymbol{\eta}), \mathbf{v}) \;=\; & \frac{1-a}{2}\left[-(\nabla\cdot(\boldsymbol{\eta}^T\boldsymbol{\sigma}), \mathbf{v}) - (\nabla\cdot(\boldsymbol{\eta}\boldsymbol{\sigma}^T), \mathbf{v}) + ((\boldsymbol{\eta}^T\boldsymbol{\sigma})\mathbf{n}, \mathbf{v})_S + ((\boldsymbol{\eta}\boldsymbol{\sigma}^T)\mathbf{n}, \mathbf{v})_S\right] \\ & -\frac{1+a}{2}\left[(-(\nabla\cdot(\boldsymbol{\sigma}\boldsymbol{\eta}^T), \mathbf{v}) - (\nabla\cdot(\boldsymbol{\sigma}^T\boldsymbol{\eta}), \mathbf{v}) + ((\boldsymbol{\sigma}\boldsymbol{\eta}^T)\mathbf{n}, \mathbf{v})_S + ((\boldsymbol{\sigma}^T\boldsymbol{\eta})\mathbf{n}, \mathbf{v})_S\right] \\ =\; & (a-1)\left[(\nabla\cdot(\boldsymbol{\eta}^T\boldsymbol{\sigma}), \mathbf{v}) + ((\boldsymbol{\eta}^T\boldsymbol{\sigma})\mathbf{n}, \mathbf{v})_S\right] \\ & +(1+a)\left[((\nabla\cdot(\boldsymbol{\sigma}\boldsymbol{\eta}^T), \mathbf{v}) + ((\boldsymbol{\sigma}\boldsymbol{\eta}^T)\mathbf{n}, \mathbf{v})_S\right]. \end{aligned} \tag{7.16}$$

Also, we can write the second term in (7.14) as

$$((\mathbf{v}\cdot\nabla)\boldsymbol{\sigma}, \boldsymbol{\eta}) = \left(\begin{bmatrix} \boldsymbol{\sigma} : \frac{\partial}{\partial x}\boldsymbol{\eta} \\ \boldsymbol{\sigma} : \frac{\partial}{\partial y}\boldsymbol{\eta} \end{bmatrix}, \mathbf{v}\right) - ((\nabla\cdot\mathbf{v})\boldsymbol{\sigma}, \boldsymbol{\eta}) + ((\mathbf{v}\cdot\mathbf{n})\boldsymbol{\sigma}, \boldsymbol{\eta})_S. \tag{7.17}$$

The condition $\frac{\partial \mathcal{L}}{\partial p} = 0$ implies the adjoint mass equation

$$(\nabla\cdot\mathbf{w}, q) = 0 \qquad \forall q \in Q. \tag{7.18}$$

The adjoint constitutive equation is derived by $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} = 0$:

$$\begin{aligned} (\boldsymbol{\eta}, \boldsymbol{\tau}) + \lambda\left(((-\mathbf{u}\cdot\nabla)\boldsymbol{\eta}, \boldsymbol{\tau}) + (\psi(\boldsymbol{\eta}, \mathbf{u}), \boldsymbol{\tau})\right) + (D(\mathbf{w}), \boldsymbol{\tau}) \\ +(\boldsymbol{\eta}, \boldsymbol{\tau})_{S_{in}} + \lambda((\mathbf{u}\cdot\mathbf{n})\boldsymbol{\eta}, \boldsymbol{\tau})_S = \mathbf{0} \quad \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}\,, \end{aligned} \tag{7.19}$$

where $\psi(\boldsymbol{\eta}, \mathbf{u})$ is defined by

$$\psi(\boldsymbol{\eta}, \mathbf{u}) := \frac{1-a}{2}(\boldsymbol{\eta}(\nabla \mathbf{u})^T + (\nabla \mathbf{u})\boldsymbol{\eta}) - \frac{1+a}{2}((\nabla \mathbf{u})^T \boldsymbol{\eta} + \boldsymbol{\eta}\nabla \mathbf{u})\,. \tag{7.20}$$

Finally, $\frac{\partial \mathcal{L}}{\partial \mathbf{g}} = 0$, $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{g}}} = 0$ imply the first order necessary conditions:

$$(\mathbf{g}_N, \mathbf{h})_S = \frac{1}{\epsilon_1}(\mathbf{w}, \mathbf{h})_S \qquad \forall \mathbf{h} \in \mathbf{G}_N, \tag{7.21}$$

$$(\mathbf{g}_D, \tilde{\mathbf{h}})_{S_{in}} = \frac{1}{\epsilon_2}(\boldsymbol{\eta}, \tilde{\mathbf{h}})_{S_{in}} \qquad \forall \tilde{\mathbf{h}} \in \mathbf{G}_D. \tag{7.22}$$

7.3 Steepest descent approach

We present in this section the methodology used for the steepest descent approach. Due to the hyperbolic nature of the constitutive equation, a stabilization technique is needed for finite element simulation of viscoelastic flows. The discontinuous Galerkin method is used for approximating the stress.

We will choose the Taylor-Hood element pair $((P_2)^d, P_1)$ for our choice of velocity-pressure finite element space. The stress $\boldsymbol{\sigma}$ is approximated in the discontinuous finite element space of piecewise linears:

$$\boldsymbol{\Sigma}^h := \{\boldsymbol{\tau} \in \boldsymbol{\Sigma} : \boldsymbol{\tau}|_K \in P_1(K)^{d\times d},\ \forall K \in T_h\}\,.$$

Below we introduce some notation to be used for approximting stress by the discontinuous Galerkin method. We define

$$\partial K^-(\mathbf{u}) := \{\mathbf{x} \in \partial K,\ \mathbf{u}\cdot\mathbf{n} < 0\}\,,$$

where ∂K is the boundary of K and $\mathbf{n}$ is outward unit normal,

$$\boldsymbol{\tau}^{\pm}(\mathbf{u}) := \lim_{\epsilon \to 0^{\pm}} \boldsymbol{\tau}(\mathbf{x} + \epsilon \mathbf{u}(\mathbf{x}))\,,$$

and

$$< \boldsymbol{\sigma}^{\pm}, \boldsymbol{\tau}^{\pm} >_{h,\mathbf{u}} := \sum_{K\in T_h} \int_{\partial K^-(\mathbf{u})} (\boldsymbol{\sigma}^{\pm}(\mathbf{u}), \boldsymbol{\tau}^{\pm}(\mathbf{u}))|\mathbf{n}\cdot\mathbf{u}|\, ds\,.$$

We introduce the operator B^h on $\mathbf{X}^h \times \mathbf{\Sigma}^h \times \mathbf{\Sigma}^h$ defined by

$$B^h(\mathbf{u}^h, \boldsymbol{\sigma}^h, \boldsymbol{\tau}^h) := ((\mathbf{u}^h \cdot \nabla)\boldsymbol{\sigma}^h, \boldsymbol{\tau}^h) + < {\boldsymbol{\sigma}^h}^+ - {\boldsymbol{\sigma}^h}^-, {\boldsymbol{\tau}^h}^+ >_{h,\mathbf{u}^h} . \tag{7.1}$$

The Galerkin finite element approximation of the state and adjoint equations is then as follows: find $(\mathbf{u}^h, p^h, \boldsymbol{\sigma}^h) \in \mathbf{X}^h \times S^h \times \mathbf{\Sigma}^h$ and $(\mathbf{w}^h, \xi^h, \boldsymbol{\eta}^h) \in \mathbf{X}^h \times Q^h \times \mathbf{\Sigma}^h$ such that

$$\begin{aligned}
&(\boldsymbol{\sigma}^h, \boldsymbol{\tau}^h) + \lambda B^h(\mathbf{u}^h, \boldsymbol{\sigma}^h, \boldsymbol{\tau}^h) + \lambda(g_a(\boldsymbol{\sigma}^h, \nabla \mathbf{u}^h), \boldsymbol{\tau}^h) \\
&\qquad -2\alpha(D(\mathbf{u}^h), \boldsymbol{\tau}^h) + (\boldsymbol{\eta}^h, \boldsymbol{\tau}^h)_{S_{in}} = ({\mathbf{g}_D}^h, \boldsymbol{\tau})_{S_{in}} \quad \forall \boldsymbol{\tau}^h \in \mathbf{\Sigma}^h , \qquad (7.2)\\
&(\boldsymbol{\sigma}^h, d(\mathbf{v}^h)) + 2(1-\alpha)\,(d(\mathbf{u}^h), d(\mathbf{v}^h)) - (p^h, \nabla \cdot \mathbf{v}^h) \\
&\qquad = (\mathbf{f}, \mathbf{v}^h) + (\mathbf{g}_N^h, \mathbf{v}^h)_S \quad \forall \mathbf{v}^h \in \mathbf{X}^h , \qquad (7.3)\\
&(q^h, \nabla \cdot \mathbf{u}^h) = 0 \quad \forall q^h \in Q^h , \qquad (7.4)
\end{aligned}$$

$$\begin{aligned}
&(\boldsymbol{\eta}^h, \boldsymbol{\tau}^h) + \lambda \left[B^h(-\mathbf{u}^h, \boldsymbol{\eta}^h, \boldsymbol{\tau}^h) + (\psi(\boldsymbol{\eta}^h, \mathbf{u}^h), \boldsymbol{\tau}^h)\right] + (D(\mathbf{w}^h), \boldsymbol{\tau}^h) \\
&\qquad + (\boldsymbol{\eta}^h, \boldsymbol{\tau}^h)_{S_{in}} + \lambda((\mathbf{u}^h \cdot \mathbf{n})\boldsymbol{\eta}^h, \boldsymbol{\tau}^h)_S = \mathbf{0}\, \forall \boldsymbol{\tau} \in \mathbf{\Sigma}^h , \qquad (7.5)\\
&-2\alpha(\boldsymbol{\eta}^h, D(\mathbf{v}^h)) + \lambda \left[\left(\begin{bmatrix} \boldsymbol{\sigma}^h : \frac{\partial}{\partial x}\boldsymbol{\eta}^h \\ \boldsymbol{\sigma}^h : \frac{\partial}{\partial y}\boldsymbol{\eta}^h \end{bmatrix}, \mathbf{v}^h \right) - ((\nabla \cdot \mathbf{v}^h)\boldsymbol{\sigma}^h, \boldsymbol{\eta}^h) + \phi((\boldsymbol{\sigma}^h, \boldsymbol{\eta}^h), \mathbf{v}^h) \right] \\
&\qquad + 2(1-\alpha)(D(\mathbf{w}^h), D(\mathbf{v}^h)) - (\xi^h, \nabla \cdot \mathbf{v}^h) + \lambda((\mathbf{v}^h \cdot \mathbf{n})\boldsymbol{\sigma}^h, \boldsymbol{\eta}^h)_S \\
&\qquad = -\sum_{i=1}^{m} \left(\int_{S_i} \mathbf{u}^h \cdot \mathbf{n}\, dS_i - Q_i \right) \int_{S_i} \mathbf{v}^h \cdot \mathbf{n}\, dS_i \qquad \forall \mathbf{v} \in \mathbf{X}^h , \qquad (7.6)\\
&(q^h, \nabla \cdot \mathbf{w}^h) = 0 \quad \forall q^h \in Q^h . \qquad (7.7)
\end{aligned}$$

The optimality system is a coupled system whose solution yields a solution of the optimization problem. In practice, the size of the system is huge, and therefore the state and adjoint systems need to be decoupled. One way of accomplishing this is through a gradient type method. The gradient method for minimizing the functional $\mathcal{M}(\mathbf{g}) := \mathcal{J}(\mathbf{u}(\mathbf{g}), p(\mathbf{g}), \boldsymbol{\sigma}(\mathbf{g}), \mathbf{g})$ is given in the form of

$$\mathbf{g}_{(k+1)} = \mathbf{g}_{(k)} - \rho_k \frac{d\mathcal{M}}{d\mathbf{g}_k} , \tag{7.8}$$

where ρ_k is a step size (in our computations chosen using the golden section search algorithm). The gradient of the function $\frac{d\mathcal{M}}{d\mathbf{g}_{N\,k}}$ can be determined by a solution of the adjoint system by the

optimality conditions (7.21), (7.22):

$$\frac{d\mathcal{M}}{d\mathbf{g}_{N\,k}} = \epsilon_1\ \mathbf{g}_{N\,k} - \mathbf{w}_k|_S\,. \tag{7.9}$$

Assuming the step size is dependent on the penalty parameter ϵ_1, i.e., $\rho_k = \alpha_k/\epsilon_1$ and using (7.9), the steepest decent algorithm for $\mathbf{g}_{N_k}$ is written as

$$\mathbf{g}_{N\,k+1} = (1-\alpha_k)\mathbf{g}_{N\,k} + \frac{\alpha_k}{\epsilon_1}\mathbf{w}_k|_S\,. \tag{7.10}$$

Similarly, the algorithm for $\mathbf{g}_{D\,k}$ is given by

$$\mathbf{g}_{D\,k+1} = (1-\beta_k)\mathbf{g}_{D\,k} + \frac{\beta_k}{\epsilon_2}\boldsymbol{\eta}_k|_{S_{in}}\,. \tag{7.11}$$

Algorithm 7.3.1. *(Steepest descent algorithm)*

Choose the initial controls $\mathbf{g}_{N\,0}$, $\mathbf{g}_{D\,0}$.

For $k = 0, 1, \ldots$

1. *Solve (7.2)-(7.4) for* $(\mathbf{u}_k^h, p_k^h, \boldsymbol{\sigma}_k^h)$.
2. *Solve (7.5)-(7.7) for* $(\mathbf{w}_k^h, \xi_k^h, \boldsymbol{\eta}_k^h)$.
3. *Update the controls by (7.10) and (7.11).*

7.4 Mean pressure boundary condition

The minimization approach discussed in previous sections can be extended for other types of defective boundary condition, e.g., mean pressure boundary condition. Suppose we have mean pressure specified on defective boundaries:

$$\frac{1}{|S_i|}\int_{S_i} p\ dS = P_i \quad \text{for } i = 1, \ldots, m. \tag{7.12}$$

Since pressure is unique up to a constant, we may set $\mathbf{g}_{N_1} = \mathbf{0}$, and p is shifted appropriately so that the mean pressure condition on S_1 is satisfied. Define the functional

$$\mathcal{J}(\mathbf{u}, p, \boldsymbol{\sigma}, \mathbf{g}) := \frac{1}{2}\sum_{i=1}^{m}\left(\frac{1}{|S_i|}\int_{S_i} p\, dS_i - P_i\right)^2 + \frac{\epsilon_1}{2}\int_S |\mathbf{g}_N|^2 dS + + \frac{\epsilon_2}{2}\int_{S_{in}} |\mathbf{g}_D|^2 dS, \tag{7.13}$$

where $\mathbf{g}_N$ and $\mathbf{g}_D$ are defined as before. Using a similar approach as shown in previous sections, we can obtain the optimality system

$$\begin{aligned}(\boldsymbol{\sigma}, \boldsymbol{\tau}) + \lambda((\mathbf{u}\cdot\nabla)\boldsymbol{\sigma}, \boldsymbol{\tau}) + \lambda(g_a(\boldsymbol{\sigma}, \nabla\mathbf{u}), \boldsymbol{\tau}) - 2\alpha(D(\mathbf{u}), \boldsymbol{\tau}) + (\boldsymbol{\sigma}, \boldsymbol{\tau})_{S_{in}} \\ = (\mathbf{g}_D, \boldsymbol{\tau})_{S_{in}} \quad \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}\,,\end{aligned} \tag{7.14}$$

$$(\boldsymbol{\sigma}, D(\mathbf{v})) + 2(1-\alpha)(D(\mathbf{u}), D(\mathbf{v})) - (p, \nabla\cdot\mathbf{v}) = (\mathbf{f}, \mathbf{v}) + (\mathbf{g}_N, \mathbf{v})_S \quad \forall \mathbf{v} \in \mathbf{X}\,, \tag{7.15}$$

$$(q, \nabla\cdot\mathbf{u}) = 0 \quad \forall q \in Q,\,. \tag{7.16}$$

and the adjoint equations

$$\begin{aligned}(\boldsymbol{\eta}, \boldsymbol{\tau}) + \lambda\left(((-\mathbf{u}\cdot\nabla)\boldsymbol{\eta}, \boldsymbol{\tau}) + (\psi(\boldsymbol{\eta}, \mathbf{u}), \boldsymbol{\tau})\right) + (D(\mathbf{w}), \boldsymbol{\tau}) \\ + (\boldsymbol{\eta}, \boldsymbol{\tau})_{S_{in}} + \lambda((\mathbf{u}\cdot\mathbf{n})\boldsymbol{\eta}, \boldsymbol{\tau})_S = \mathbf{0} \quad \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}\,,\end{aligned} \tag{7.17}$$

$$\begin{aligned}-2\alpha(\boldsymbol{\eta}, D(\mathbf{v})) + \lambda\left(((\mathbf{v}\cdot\nabla)\boldsymbol{\sigma}, \boldsymbol{\eta}) + \phi((\boldsymbol{\sigma}, \boldsymbol{\eta}), \mathbf{v})\right) + 2(1-\alpha)(D(\mathbf{w}), D(\mathbf{v})) \\ -(\xi, \nabla\cdot\mathbf{v}) = 0 \qquad \forall \mathbf{v} \in \mathbf{X}\,,\end{aligned} \tag{7.18}$$

$$(q, \nabla\cdot\mathbf{w}) = \sum_{i=1}^{m}\left(\frac{1}{|S_i|}\int_{S_i} p\, dS_i - P_i\right)\int_{S_i} q\, dS_i \quad \forall q \in P\,, \tag{7.19}$$

where $\phi(\cdot,\cdot)$, $\psi(\cdot,\cdot)$ are defined by (7.15) and (7.20), respectively. The condition for $\mathbf{g}_N$, $\mathbf{g}_D$ are same as (7.21) and (7.22), and the steepest decent algorithm can be used for the mean pressure condition problem.

7.5 Nonlinear least squares approach

In this section we reconsider the minimization problem (7.12) from a nonlinear least squares viewpoint. In this approach, motivated by the definition of the penalty terms in (7.8), we specify the control spaces as $\mathbf{G}_N \times \mathbf{G}_D := \mathbf{L}^2(S) \times \mathbf{L}^2(S_{in})$ to clearly describe the definitions of operators

and to present Algorithm 7.5.1, where calculating of function norms in suitable spaces is essential for the success of the algorithm.

Define the nonlinear operator $N : \mathbf{G}_N \times \mathbf{G}_D \to \mathbb{R}^m \times \mathbf{G}_N \times \mathbf{G}_D$ by

$$N(\mathbf{g}) = \begin{pmatrix} \int_{S_1} \mathbf{u} \cdot \mathbf{n} \, dS - Q_1 \\ \vdots \\ \int_{S_m} \mathbf{u} \cdot \mathbf{n} \, dS - Q_m \\ \sqrt{\epsilon_1}\, \mathbf{g}_N \\ \sqrt{\epsilon_2}\, \mathbf{g}_D \end{pmatrix},$$

where $\mathbf{g} := (\mathbf{G}_N, \mathbf{G}_D)$ and $\mathbf{u}$ is the fluid velocity satisfying (7.9)-(7.11) when g_N, g_D are boundary condition in the equations. Then, (7.8) can be written as

$$\mathcal{J}(\mathbf{g}) = \frac{1}{2} \|N(\mathbf{g})\|^2_{R^m \times \mathbf{G}_N \times \mathbf{G}_D} \tag{7.20}$$

and the nonlinear least squares problem we consider is to

$$\text{seek } \mathbf{g} \in \mathbf{G}_N \times \mathbf{G}_D \text{ such that (7.20) is minimized.} \tag{7.21}$$

We can linearize $N(\mathbf{g})$ using the Fréchet derivative of $N(\cdot)$ at $\overline{\mathbf{g}}$, $N'(\overline{\mathbf{g}})$, by

$$N(\mathbf{g}) = N(\overline{\mathbf{g}}) + N'(\overline{\mathbf{g}})(\mathbf{g} - \overline{\mathbf{g}}) + O(\|\mathbf{g} - \overline{\mathbf{g}}\|^2_{\mathbf{G}_N \times \mathbf{G}_D})$$

so that solutions of the nonlinear least squares problem can be obtained by repeatedly solving the linear least squares problem

$$\min_{\mathbf{h} \in \mathbf{G}_N \times \mathbf{G}_D} \frac{1}{2} \|N(\overline{\mathbf{g}}) + N'(\overline{\mathbf{g}})\mathbf{h}\|^2_{R^m \times \mathbf{G}_N \times \mathbf{G}_D}, \tag{7.22}$$

where $\mathbf{h} = \mathbf{g} - \overline{\mathbf{g}}$. Hence, starting with arbitrary $\mathbf{g}^{(0)}$, we can find a sequence $\{\mathbf{g}^{(k)}\}$ obtained by $\mathbf{g}^{(k)} = \mathbf{g}^{(k-1)} + \mathbf{h}^{(k)}$, where $\mathbf{h}^{(k)}$ is a solution of the linear least squares problem (7.22).

For $\overline{\mathbf{g}} \in \mathbf{G}_N \times \mathbf{G}_D$, the Fréchet derivative $N'(\overline{\mathbf{g}})(\cdot) : \mathbf{G}_N \times \mathbf{G}_D \to \mathbb{R}^m \times \mathbf{G}_N \times \mathbf{G}_D$ is defined

by

$$N'(\overline{\mathbf{g}})(\mathbf{h}) = \begin{pmatrix} \int_{S_1} \mathbf{w}\cdot\mathbf{n}\, dS \\ \vdots \\ \int_{S_m} \mathbf{w}\cdot\mathbf{n}\, dS \\ \sqrt{\epsilon_1}\,\mathbf{h}_N \\ \sqrt{\epsilon_2}\,\mathbf{h}_D \end{pmatrix},$$

where $\mathbf{h} := (\mathbf{h}_N, \mathbf{h}_D)$ and $(\mathbf{w}, \xi, \boldsymbol{\eta})$ is the solution of linearized problem

$$(\boldsymbol{\eta},\boldsymbol{\tau}) + \lambda\left[((\overline{\mathbf{u}}\cdot\nabla)\boldsymbol{\eta},\boldsymbol{\tau}) + ((\mathbf{w}\cdot\nabla)\overline{\boldsymbol{\sigma}},\boldsymbol{\tau}) + (g_a(\boldsymbol{\eta},\nabla\overline{\mathbf{u}}),\boldsymbol{\tau}) + (g_a(\overline{\boldsymbol{\sigma}},\nabla\mathbf{w}),\boldsymbol{\tau})\right]$$
$$-2\alpha(D(\mathbf{w}),\boldsymbol{\tau}) + (\boldsymbol{\eta},\boldsymbol{\tau})_{S_{in}} = (\mathbf{h}_D,\boldsymbol{\tau})_{S_{in}}\,, \tag{7.23}$$

$$(\boldsymbol{\eta}, D(\mathbf{v})) + 2(1-\alpha)(D(\mathbf{w}), D(\mathbf{v})) - (\xi, \nabla\cdot\mathbf{v}) = (\mathbf{h}_N,\mathbf{v})_S \qquad \forall\mathbf{v}\in\mathbf{X}\,, \tag{7.24}$$

$$(q, \nabla\cdot\mathbf{w}) = 0 \quad \forall q \in S\,. \tag{7.25}$$

In the above equations $(\overline{\mathbf{u}}, \overline{\boldsymbol{\sigma}})$ is the solution of (7.9)–(7.11) with $\mathbf{g} = (\mathbf{g}_N, \mathbf{g}_D)$ replaced by $\overline{\mathbf{g}} = (\overline{\mathbf{g}}_N, \overline{\mathbf{g}}_D)$.

It is necessary to define the adjoint operator of $N'(\overline{\mathbf{g}})$ in order to solve the linear least squares problem (7.22). We define $(N'(\overline{\mathbf{g}}))^*(\cdot) : \mathbb{R}^m \times \mathbf{G}_N \times \mathbf{G}_D \to \mathbf{G}_N \times \mathbf{G}_D$ by

$$(N'(\overline{\mathbf{g}}))^* \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{z}|_S + \sqrt{\epsilon_1}\,\boldsymbol{\gamma} \\ \sqrt{\epsilon_2}\boldsymbol{\beta} + \mathbf{t}|_{S_{in}} \end{pmatrix},$$

where $(\mathbf{z}, \varphi, \mathbf{t}) \in \mathbf{X}\times Q\times\boldsymbol{\Sigma}$ is the solution of

$$(\mathbf{t},\boldsymbol{\tau}) + \lambda\left[((-\overline{\mathbf{u}}\cdot\nabla)\mathbf{t},\boldsymbol{\tau}) + (\psi(\mathbf{t},\overline{\mathbf{u}}),\boldsymbol{\tau})\right] + (D(\mathbf{z}),\boldsymbol{\tau})$$
$$+\lambda((\overline{\mathbf{u}}\cdot\mathbf{n})\mathbf{t},\boldsymbol{\tau})_S + (\mathbf{t},\boldsymbol{\tau})_{S_{in}} = 0 \qquad \forall\boldsymbol{\tau}\in\boldsymbol{\Sigma}\,, \tag{7.26}$$

$$- 2\alpha(\mathbf{t}, D(\mathbf{v})) + \lambda \left[\left(\begin{bmatrix} \overline{\boldsymbol{\sigma}} : \frac{\partial}{\partial x}\mathbf{t} \\ \overline{\boldsymbol{\sigma}} : \frac{\partial}{\partial y}\mathbf{t} \end{bmatrix}, \mathbf{v} \right) - ((\nabla \cdot \mathbf{v})\overline{\boldsymbol{\sigma}}, \mathbf{t}) + \phi((\overline{\boldsymbol{\sigma}}, \mathbf{t}), \mathbf{v}) \right] - (\varphi, \nabla \cdot \mathbf{v})$$

$$+ 2(1-\alpha)(D(\mathbf{z}), D(\mathbf{v})) + \lambda((\mathbf{v} \cdot \mathbf{n})\overline{\boldsymbol{\sigma}}, \mathbf{t})_S = \sum_{i=1}^{m} y_i \int_{S_i} \mathbf{v} \cdot \mathbf{n} \, dS \quad \forall \mathbf{v} \in \mathbf{X}\,, \quad (7.27)$$

$$(q, \nabla \cdot \mathbf{z}) = 0 \quad \forall q \in Q\,. \quad (7.28)$$

Again, $(\overline{\mathbf{u}}, \overline{\boldsymbol{\sigma}})$ in (7.26)-(7.28) is the solution of (7.9)–(7.11) with $\mathbf{g}$ replaced by $\overline{\mathbf{g}}$. Note that taking $(\mathbf{v}, q, \boldsymbol{\tau}) = (\mathbf{z}, \varphi, \mathbf{t})$ in (7.23)-(7.25), $(\mathbf{v}, q, \boldsymbol{\tau}) = (\mathbf{w}, \xi, \boldsymbol{\eta})$ in (7.26)-(7.28) and using integration by parts, (7.15), (7.17) and (7.20), we can show

$$\sum_{i=1}^{m} y_i \int_{S_i} \mathbf{w} \cdot \mathbf{n} \, dS = (\mathbf{h}_N, \mathbf{z})_S + (\mathbf{h}_D, \mathbf{t})_{S_{in}},$$

therefore

$$\begin{aligned} \left(N'(\overline{\mathbf{g}}) \begin{pmatrix} \mathbf{h}_N \\ \mathbf{h}_D \end{pmatrix}, \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} \right) &= \sum_{i=1}^{m} y_i \int_{S_i} \mathbf{w} \cdot \mathbf{n} \, dS + \sqrt{\epsilon_1}(\mathbf{h}_N, \boldsymbol{\gamma})_S + \sqrt{\epsilon_2}(\mathbf{h}_D, \boldsymbol{\beta})_{S_{in}} \\ &= (\mathbf{h}_N, \mathbf{z})_S + (\mathbf{h}_D, \mathbf{t})_{S_{in}} + \sqrt{\epsilon_1}(\mathbf{h}_N, \boldsymbol{\gamma})_S + \sqrt{\epsilon_2}(\mathbf{h}_D, \boldsymbol{\beta})_{S_{in}} \\ &= \left(\begin{pmatrix} \mathbf{h}_N \\ \mathbf{h}_D \end{pmatrix}, (N'(\overline{\mathbf{g}}))^* \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} \right). \end{aligned} \quad (7.29)$$

We adopt the following basic conjugate gradient algorithm for the linear least squares problem (7.22), which can be found in many references. For example, see [31] or [33]. For the algorithm, we adopt the notation $A = N'(\overline{\mathbf{g}})$, $b = -N(\overline{\mathbf{g}})$, and $\mathbf{x} = \mathbf{h}$.

Algorithm 7.5.1. *(Conjugate Gradient Method for the Least Squares Problem)*

Given A, b, and $x^{(0)}$,

1. *Set* $r^{(0)} = b - Ax^{(0)}$,
   $p^{(0)} = A^* r^{(0)}$.
2. *For* $n = 0, 1, 2, \cdots$,

a. *if* $\|A^* r^{(n)}\|_{\mathbf{G}_N \times \mathbf{G}_D} < \epsilon$ *stop,*

b. $\sigma^{(n)} = \|A^* r^{(n)}\|^2_{\mathbf{G}_N \times \mathbf{G}_D} / \|Ap^{(n)}\|^2_{R^m \times \mathbf{G}_N \times \mathbf{G}_D}$,

c. $x^{(n+1)} = x^{(n)} + \sigma^{(n)} p^{(n)}$,

d. $r^{(n+1)} = r^{(n)} - \sigma^{(n)} Ap^{(n)}$,

e. $\tau^{(n)} = \|A^* r^{(n+1)}\|^2_{\mathbf{G}_N \times \mathbf{G}_D} / \|A^* r^{(n)}\|^2_{\mathbf{G}_N \times \mathbf{G}_D}$,

f. $p^{(n+1)} = A^* r^{(n+1)} + \tau^{(n)} p^{(n)}$.

Thus, the nonlinear least squares problem (7.21) can be solved using the following Gauss-Newton algorithm.

Algorithm 7.5.2.

1. *Choose* $\mathbf{g}^{(0)}$.

2. *For* $n = 1, 2, 3, \ldots,$

   a. *compute* $\mathbf{h}^{(n)}$ *by the conjugate gradient algorithm* 7.5.1 *with* $A = N'(\mathbf{g}^{(n-1)})$, $b = -N(\mathbf{g}^{(n-1)})$, *and* $x = \mathbf{h}^{(n)}$;

   b. *set* $\mathbf{g}^{(n)} = \mathbf{g}^{(n-1)} + \mathbf{h}^{(n)}$.

7.6 Numerical Results

In this section we consider a model flow problem subject to specified flow rate or mean pressure conditions on defective boundaries. Consider the problem of flow in a square domain, $\Omega = (0, 5) \times (0, 5)$, containing three defective boundaries $S_1 = \{(x, y) : x = 0, 1 < y < 2\}, S_2 = \{(x, y) : x = 0, 3 < y < 4\}$, and $S_3 = \{(x, y) : x = 5, 2 < y < 3\}$ (as seen in Figure 7.1). In the model equations (7.1)-(7.3) we take the parameters $\lambda = 0.5, \alpha = 0.5$, and $a = 0$. In the steepest descent algorithm the golden section search method was used to determine the optimal step sizes in both the flow rate and mean pressure matching problems. Computations were performed on both 30×30 and 50×50 triangular meshes, providing 29,058 and 79,856 total degrees of freedom, respectively. In all tests, the results on the different meshes provided identical plots. All computations were performed using the software *FreeFem++* [39] on a Macbook Pro with 2.26 GHz Intel Core 2 Duo CPU and 2GB of 1066MHz DDR3 SDRAM.

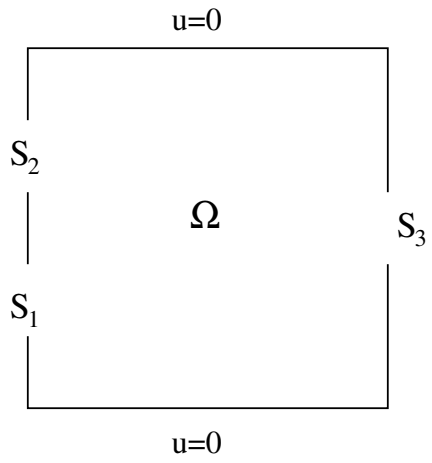


Figure 7.1: Shown above is the domain for the flow problem.

Initially, for comparison purposes, we computed velocity, pressure, and stress solutions using the following parabolic profiles

$$\mathbf{u}|_{S_1} = \begin{bmatrix} -8(y-1)(y-2) \\ 0 \end{bmatrix} \quad \mathbf{u}|_{S_2} = \begin{bmatrix} -4(y-3)(y-4) \\ 0 \end{bmatrix} \quad \mathbf{u}|_{S_3} = \begin{bmatrix} -12(y-2)(y-3) \\ 0 \end{bmatrix}$$

as Dirichlet boundary conditions for the velocity on S_1, S_2, and S_3. Figure 7.2 depicts the magnitude of the velocity and streamlines, the horizontal velocity and pressure profile on each defective boundary, and a contour plot of each stress component of the solution generated using the Dirichelt boundary conditions.

7.6.1 Flow Rate Boundary Condition

The chosen Dirichlet boundary conditions yield flow rates of $Q_1 = -\frac{4}{3}, Q_2 = -\frac{2}{3}$, and $Q_3 = 2$ which were then used as flow rate boundary conditions for the flow matching problem. Both algorithms used the constant vector $\mathbf{g} = [0.1, ..., 0.1]$ as an initial guess for both Dirichlet and Neumann controls. On both meshes, the steepest descent algorithm converged in 13 iterations, and the Gauss-Newton algorithm converged in three iterations. The results from the steepest descent and Gauss-Newton algorithms displayed in Figures 7.3 and 7.4, respectively, as velocity streamlines, inlet and outlet velocity and pressure profiles, and stress contours. The results from these algorithms

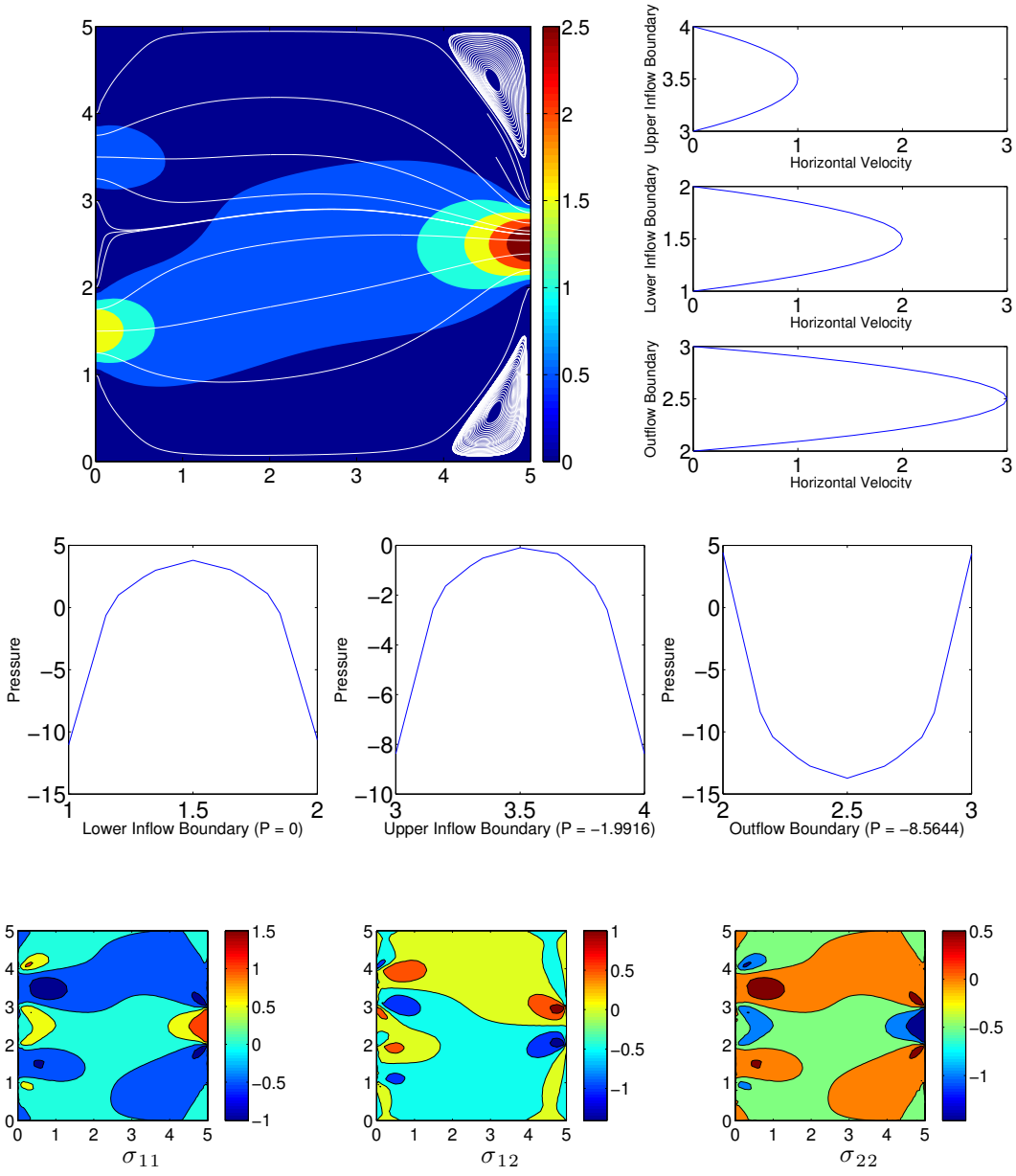


Figure 7.2: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using Dirichlet boundary conditions for the velocity and stress.

agree well with each other, and also seem to agree with those generated using Dirichlet boundary conditions for the velocity and stress.

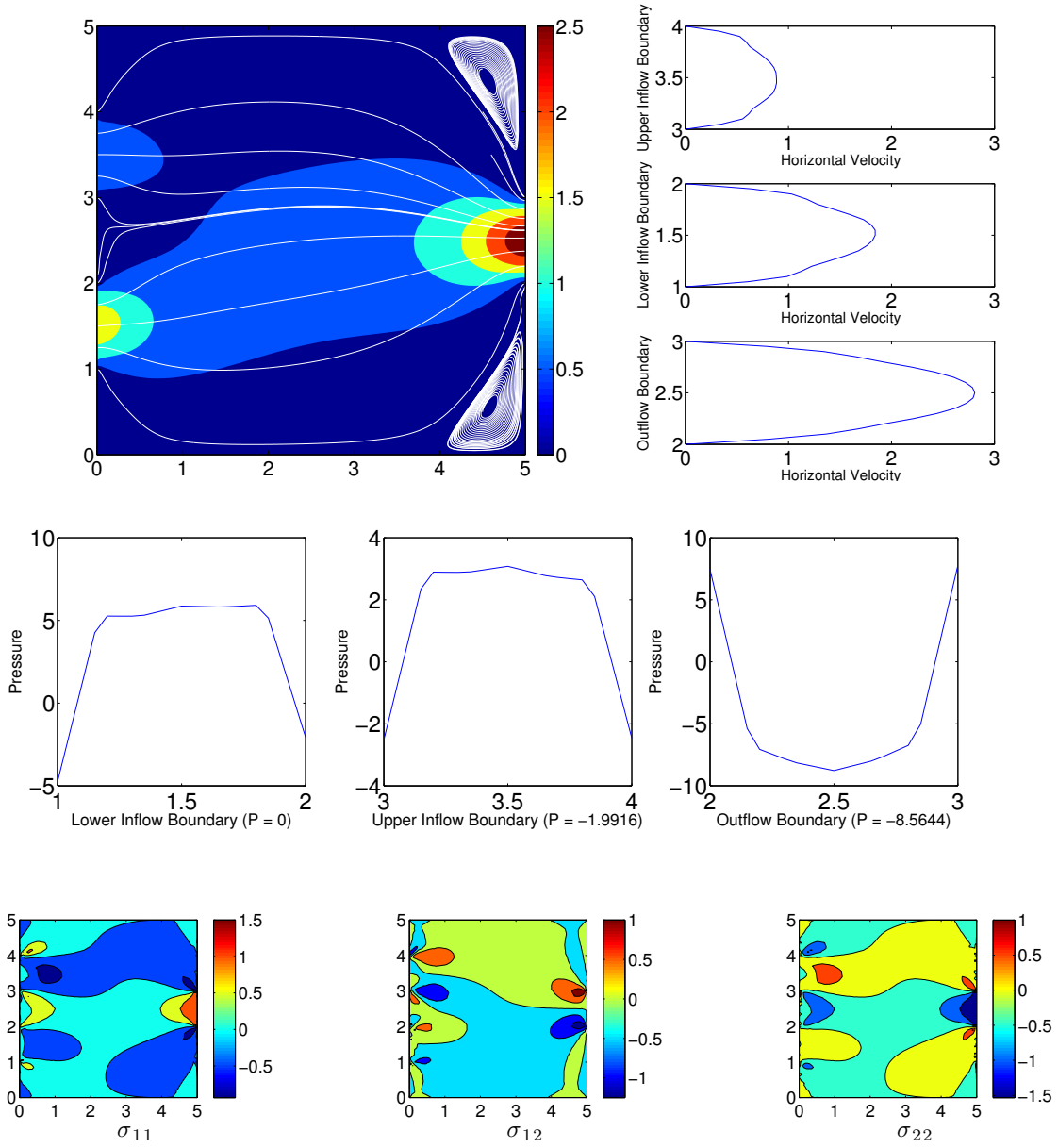


Figure 7.3: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$.

7.6.2 Mean Pressure Boundary Condition

Using the Dirichlet boundary conditions we computed the mean pressure on all of the defective boundaries, and then shifted the numerical solution so that the mean pressure on S_1, P_1, is zero.

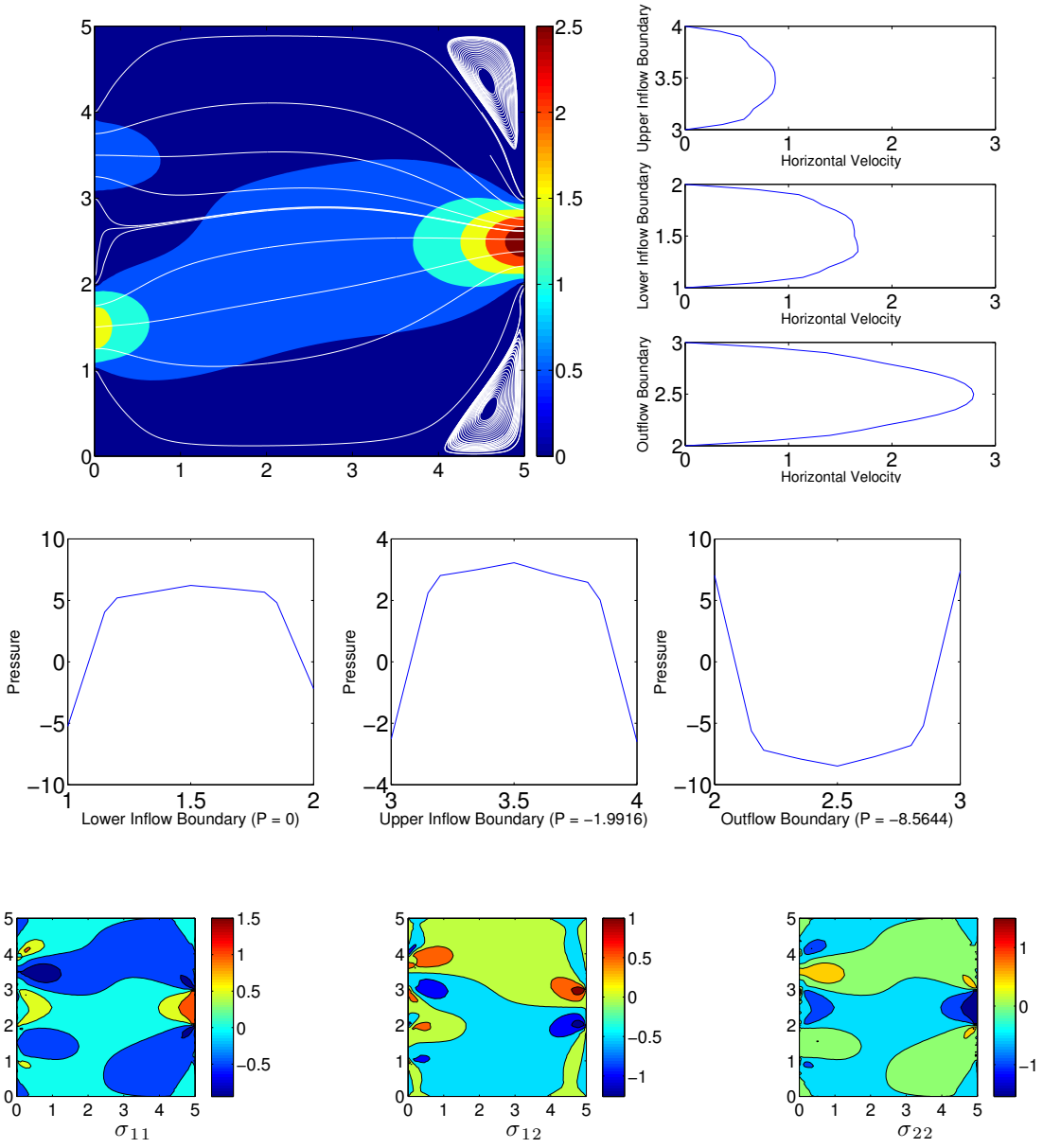


Figure 7.4: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the Gauss-Newton algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$.

Using this shifted solution the mean pressure on S_2 and S_3 (used for the mean pressure matching problem) is $P_2 = -1.992$ and $P_3 = -8.564$. A solution (seen in Figure 7.5) was generated using the steepest descent algorithm using Neumann and Dirichlet controls. The steepest descent algorithm

converged in four iterations on both meshes. We observe from Figure 7.5 that the velocity streamlines generally agree with those found using Dirichlet boundary conditions and flow rate matching, but there are significant differences in the speed contours, inlet/outlet velocity and pressure profiles, as well as in the σ_{12} stress component at the inlets and outlet.

7.6.3 Further flow rate boundary conditions algorithm verification

As further verification of our steepest descent and Gauss-Newton algorithms for flow rate matching, we consider again both algorithms, but now with the constant vector $\mathbf{g} = [5, ..., 5]$ as an initial guess for both controls. Both algorithms converged, with the steepest descent needing 17 iterations while the Gauss-Newton algorithm converged in three iterations. The solutions are displayed in Figures 7.6-7.7, respectively, and we observe they match each other exactly, but are quite different from the solutions found with initial guess $\mathbf{g} = [0.1, ..., 0.1]$. These and previous results indicate that both algorithms converge to a same local minimum and the local minimum found by the algorithms is determined by an initial guess.

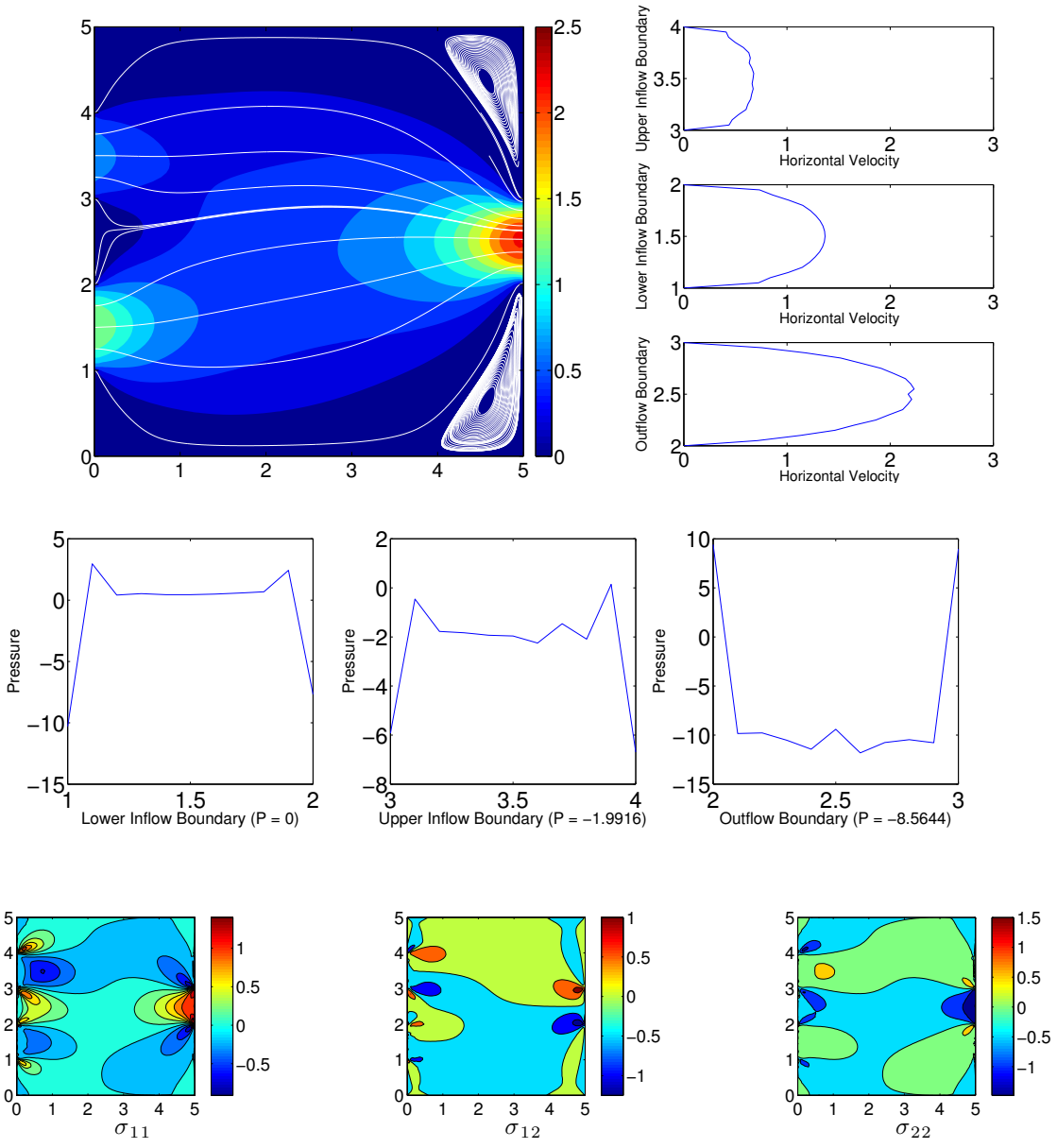


Figure 7.5: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the mean pressure matching problem with initial guess $\mathbf{g} = [0.1, ..., 0.1]$.

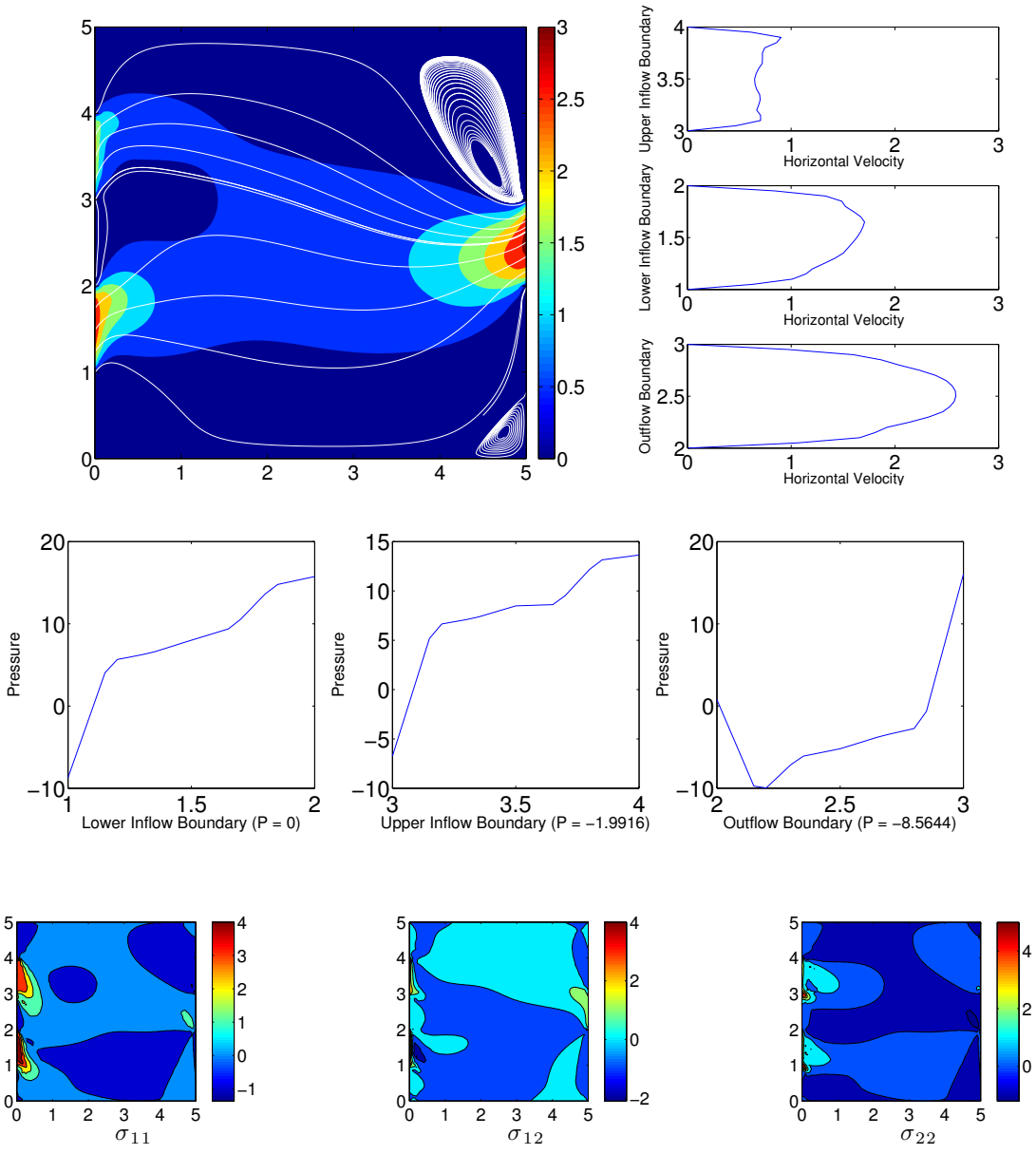


Figure 7.6: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the steepest descent algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [5, ..., 5]$.

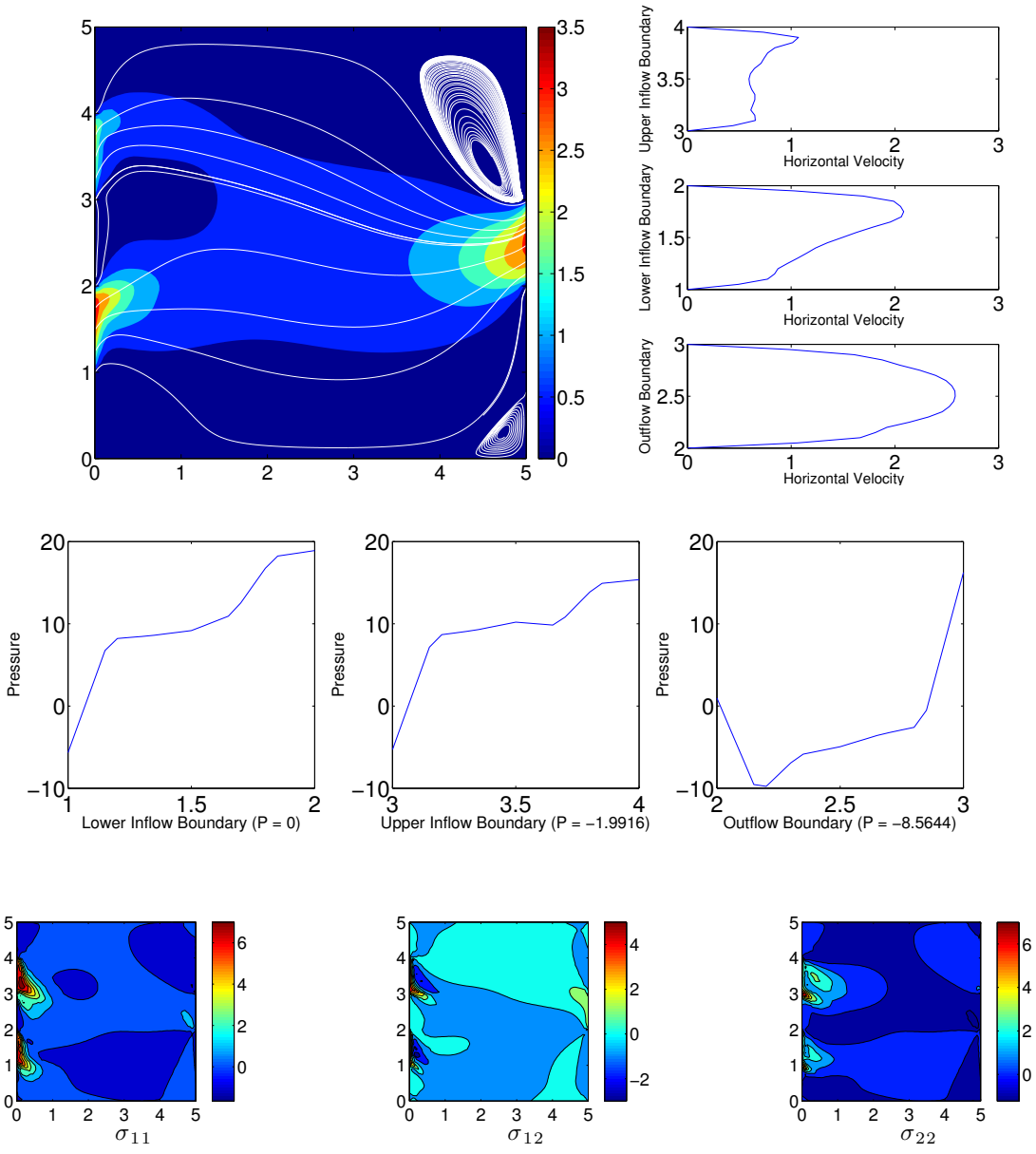


Figure 7.7: Plots of the magnitude of the velocity and streamlines, velocity and pressure profiles on S_1, S_2, and S_3, and stress contours of the solution generated using the Gauss-Newton algorithm for the flow rate matching problem with initial guess $\mathbf{g} = [5, ..., 5]$.

Chapter 8

Conclusions

This work began in Chapter 3 where we studied a finite element method for the time-dependent NSE based on the recently developed VVH formulation. Through a rigorous stability analysis we have shown that the velocity is unconditionally stable, as is the vorticity if we penalize the discrete solution to be 'close', in some sense, to the curl of the discrete velocity. Numerical experiments show mixed results: for an idealized problem, optimal convergence rates are recovered for the velocity and near-optimal convergence rates are recovered for the vorticity. However, on channel flow problems over a step, it is clear that improved vorticity boundary conditions need to be developed in order for this method to be competitive when higher order elements are used. In Chapter 4 we addressed this problem by proposing a new, simple vorticity boundary condition. This boundary condition, intended for portions of the domain where no-slip velocity boundary conditions are to be enforced, was derived directly from the vorticity equation, and consisted of boundary integrals of the pressure and forcing function. A 3d finite element method was then presented to solve the steady NSE and vorticity equations implementing our new vorticity boundary conditions. On a simple 3d problem, we verified that optimal convergence rates are achieved for the velocity and vorticity solutions, and we are currently working on performing several 3d benchmark problems to further test the method.

Chapter 5 derived a new, reduced order MDM, based off of the original MDM proposed in [22]. The RMDM allowed us to derive a C^0 finite element method that is unconditionally stable. Additionally, the method consists of a linearized backward Euler timestepping scheme which decouples the filter solve, making the method efficient. We then proved analytically that optimal

convergence rates are achieved (with respect to the model solution). The chapter ended by implementing our method on two benchmark 2d flow problems. In both examples, more accurate solutions are recovered from the RMDM when we use two different spatial scales for the filters, verifying the effectiveness of our multiscale model.

Our study of the defective boundary problem for non-Newtonian flows began in Chapter 6, where we considered a numerical method for generalized-Newtonian flows governed by the Cross model with flow rate boundary conditions. The problem was formulated as a constrained optimal control problem for which we proved solutions must exist. We then proved the existence of Lagrange multipliers, and used the Lagrange multiplier method to derive an optimality system. Finally, a steepest descent method was proposed that decouples the optimality system. We ended the chapter by studying a complex 2d flow problem for which we were able to accurately and efficiently produce solutions, verifying the robustness of our numerical method.

Chapter 7 investigated two numerical methods for viscoelastic flows with flow rate or mean pressure boundary conditions. As in Chapter 6, the defective boundary problem was transformed into an optimal control problem. The first method to solve this system stems from that presented in Chapter 6, where an optimality system is derived using the Lagrange multiplier method, and a steepest descent method is used to decouple and solve the optimality system. The second method reconsidered the optimality system from a nonlinear least squares (NLS) viewpoint. We solved the NLS problem by solving a related linear least squares problem (which was done using a conjugate gradient method). Finally, a numerical experiment was considered to compare and contrast the two methods, showing that the former is more robust, but the latter is more efficient.

Appendices

Appendix A deal.II code for 3d vorticity equation

```
1  /*************************************************/
2  /* Author - Keith Galvin                         */
3  /* Last updated - 6/4/13                         */
4  /* Model - 3d steady NSE and vorticity equation  */
5  /* Velocity boundary conditions: Dirichlet       */
6  /* Vorticity boundary conditions: New Natural vort bc */
7  /* Nonlinear Method for steady NSE: Newton's Method */
8  /* Solver - UMFPACK Sparse Direct Solver         */
9  /*************************************************/
10
11 // Header Files
12 #include <deal.II/base/timer.h>
13 #include <deal.II/base/thread_management.h>
14 #include <deal.II/base/multithread_info.h>
15 #include <deal.II/grid/tria.h>
16 #include <deal.II/grid/grid_generator.h>
17 #include <deal.II/grid/tria_accessor.h>
18 #include <deal.II/grid/tria_iterator.h>
19 #include <deal.II/grid/grid_tools.h>
20 #include <deal.II/grid/grid_refinement.h>
21 #include <deal.II/grid/grid_out.h>
22 #include <deal.II/dofs/dof_handler.h>
23 #include <deal.II/dofs/dof_tools.h>
24 #include <deal.II/dofs/dof_accessor.h>
25 #include <deal.II/dofs/dof_renumbering.h>
26 #include <deal.II/fe/fe_q.h>
27 #include <deal.II/fe/fe_values.h>
28 #include <deal.II/fe/fe_system.h>
29 #include <deal.II/base/quadrature_lib.h>
```

```
30 #include <deal.II/base/function.h>
31 #include <deal.II/base/logstream.h>
32 #include <deal.II/base/utilities.h>
33 #include <deal.II/base/tensor_function.h>
34 #include <deal.II/numerics/vectors.h>
35 #include <deal.II/numerics/matrices.h>
36 #include <deal.II/numerics/data_out.h>
37 #include <deal.II/numerics/error_estimator.h>
38 #include <deal.II/lac/full_matrix.h>
39 #include <deal.II/lac/solver_cg.h>
40 #include <deal.II/lac/precondition.h>
41 #include <deal.II/lac/constraint_matrix.h>
42 #include <deal.II/lac/sparse_direct.h>
43 #include <deal.II/lac/matrix_lib.h>
44 #include <fstream>
45 #include <iostream>
46 #include <sstream>
47
48 using namespace dealii;
49
50 const long double pi = 3.14159265358979323846264 3;
51
52 /*************************************/
53 /*                                   */
54 /* Exact solution class implementations */
55 /*                                   */
56 /*************************************/
57 /***********************************/
58 /*  Exact solution class - velocity  */
59 /***********************************/
```

```
60 template <int dim>
61 class ExactSolution : public Function<dim>
62 {
63 public:
64   ExactSolution () : Function<dim> (dim+1) {}
65
66   virtual double value (const Point<dim> &p, const unsigned int
       component) const;
67   virtual void vector_value (const Point<dim> &p, Vector<double> &
       values) const;
68   virtual Tensor<1,dim> gradient (const Point<dim> &p, const unsigned
       int component) const;
69   virtual void vector_gradient (const Point<dim> &p, std::vector<Tensor
       <1,dim> > &gradients) const;
70 };
71
72 template <int dim>
73 double ExactSolution<dim>::value (const Point<dim> &p, const unsigned
     int component) const
74 {
75   if (component == 0)
76     return std::sin(2*pi*p[1]);
77   else if (component == 1)
78     return std::cos(2*pi*p[2]);
79   else if (component == 2)
80     return std::exp(p[0]);
81   else if (component == 3)
82     return std::sin(2*pi*p[0]) + cos(2*pi*p[1]) + sin(2*pi*p[2]);
83
84   return 0;
```

```
85 }
86
87 template <int dim>
88 void ExactSolution<dim>::vector_value (const Point<dim> &p, Vector<
       double> &values) const
89 {
90   for (unsigned int c=0; c<this->n_components; c++)
91     values(c) = ExactSolution<dim>::value (p,c);
92 }
93
94 template <int dim>
95 Tensor<1,dim> ExactSolution<dim>::gradient (const Point<dim>   &p, const
       unsigned int component) const
96 {
97   Tensor<1,dim> return_value;
98
99   if (component == 0)
100    {
101      return_value[0] = 0;
102      return_value[1] = 2*pi*std::cos(2*pi*p[1]);
103      return_value[2] = 0;
104    }
105  else if (component == 1)
106    {
107      return_value[0] = 0;
108      return_value[1] = 0;
109      return_value[2] = -2*pi*std::sin(2*pi*p[2]);
110    }
111  else if (component == 2)
112    {
```

```
113         return_value[0] = std::exp(p[0]);
114         return_value[1] = 0;
115         return_value[2] = 0;
116       }
117     else
118       {
119         return_value[0] = 2*pi*std::cos(2*pi*p[0]);
120         return_value[1] = -2*pi*std::sin(2*pi*p[1]);
121         return_value[2] = 2*pi*std::cos(2*pi*p[2]);
122       }
123
124     return return_value;
125 }
126
127 template <int dim>
128 void ExactSolution<dim>::vector_gradient (const Point<dim> &p, std::
        vector<Tensor<1,dim> > &gradients) const
129 {
130     for (unsigned int c=0; c<this->n_components; c++)
131       gradients[c] = ExactSolution<dim>::gradient (p,c);
132 }
133
134 /************************************/
135 /*  Exact solution class - vorticity  */
136 /************************************/
137 template <int dim>
138 class ExactSolutionVorticity : public Function<dim>
139 {
140 public:
141     ExactSolutionVorticity () : Function<dim> (dim) {}
```

```
142
143   virtual double value (const Point<dim> &p, const unsigned int
          component) const;
144   virtual void vector_value (const Point<dim> &p, Vector<double>  &
          values) const;
145   virtual Tensor<1,dim> gradient (const Point<dim> &p, const unsigned
          int component) const;
146   virtual void vector_gradient (const Point<dim> &p, std::vector<Tensor
          <1,dim> > &gradients) const;
147 };
148
149 template <int dim>
150 double ExactSolutionVorticity<dim>::value (const Point<dim> &p, const
      unsigned int component) const
151 {
152   if (component == 0)
153     return 2*pi*std::sin(2*pi*p[2]);
154   else if (component == 1)
155     return -std::exp(p[0]);
156   else if (component == 2)
157     return -2*pi*std::cos(2*pi*p[1]);
158
159   return 0;
160 }
161
162 template<int dim>
163 void ExactSolutionVorticity<dim>::vector_value (const Point<dim> &p,
      Vector<double> &values) const
164 {
165   for (unsigned int c=0; c<this->n_components; c++)
```

```
166       values(c) = ExactSolutionVorticity<dim>::value (p,c);
167   }
168
169   template <int dim>
170   Tensor<1,dim> ExactSolutionVorticity<dim>::gradient (const Point<dim>
          &p, const unsigned int component) const
171   {
172     Tensor<1,dim> return_value;
173
174     if (component == 0)
175       {
176         return_value[0] = 0;
177         return_value[1] = 0;
178         return_value[2] = 4*pi*pi*std::cos(2*pi*p[2]);
179       }
180     else if (component == 1)
181       {
182         return_value[0] = -std::exp(p[0]);
183         return_value[1] = 0;
184         return_value[2] = 0;
185       }
186     else if (component == 2)
187       {
188         return_value[0] = 0;
189         return_value[1] = 4*pi*pi*std::sin(2*pi*p[1]);
190         return_value[2] = 0;
191       }
192
193     return return_value;
194   }
```

```
195
196 template <int dim>
197 void ExactSolutionVorticity<dim>::vector_gradient (const Point<dim> &p,
        std::vector<Tensor<1,dim> > &gradients) const
198 {
199   for (unsigned int c=0; c<this->n_components; c++)
200     gradients[c] = ExactSolutionVorticity<dim>::gradient (p,c);
201 }
202
203 /*****************************************/
204 /*  Exact solution class (tensor) - velocity */
205 /*****************************************/
206 template <int dim>
207 class ExactSolutionTensor : public TensorFunction<1,dim>
208 {
209 public:
210   ExactSolutionTensor() : TensorFunction<1,dim>() {}
211   virtual Tensor<1,dim> value (const Point<dim> &p) const;
212   virtual void value_list (const std::vector<Point<dim> > &points, std::
        vector<Tensor<1,dim> > &values) const;
213   virtual Tensor<2,dim> gradient (const Point<dim> &p) const;
214   virtual void gradient_list (const std::vector<Point<dim> > &points,
        std::vector<Tensor<2,dim> > &values) const;
215 };
216
217 template <int dim>
218 Tensor<1,dim> ExactSolutionTensor<dim>::value (const Point<dim> &p)
      const
219 {
220   ExactSolution<dim> utrue;
```

```
221     Tensor<1,dim> ten;
222
223     for (unsigned int d=0; d<dim; d++)
224       ten[d] = utrue.value(p,d);
225
226     return ten;
227   }
228
229   template <int dim>
230   void ExactSolutionTensor<dim>::value_list (const std::vector<Point<dim>
        > &points, std::vector<Tensor<1,dim> > &values) const
231   {
232     for (unsigned int p=0; p<points.size(); p++)
233       {
234         values[p].clear ();
235         values[p] = ExactSolutionTensor<dim>::value(points[p]);
236       }
237   }
238
239   template <int dim>
240   Tensor<2,dim> ExactSolutionTensor<dim>::gradient (const Point<dim> &p)
        const
241   {
242     ExactSolution<dim> utrue;
243     Tensor<2,dim> ten;
244
245     for (unsigned int d=0; d<dim; d++)
246       for (unsigned int c=0; c<dim; c++)
247         ten[d][c] = utrue.gradient(p,d)[c];
248
```

```
249   return ten;
250 }
251
252 template <int dim>
253 void ExactSolutionTensor<dim>::gradient_list (const std::vector<Point<
      dim> > &points, std::vector<Tensor<2,dim> > &values) const
254 {
255   for (unsigned int p=0; p<points.size(); p++)
256     {
257       values[p].clear ();
258       values[p] = ExactSolutionTensor<dim>::gradient(points[p]);
259     }
260 }
261
262 /*****************************************/
263 /*  Exact solution class (tensor) - vorticity */
264 /*****************************************/
265 template <int dim>
266 class ExactSolutionVorticityTensor : public TensorFunction<1,dim>
267 {
268 public:
269   ExactSolutionVorticityTensor() : TensorFunction<1,dim>() {}
270   virtual Tensor<1,dim> value (const Point<dim> &p) const;
271   virtual void value_list (const std::vector<Point<dim> > &points, std::
        vector<Tensor<1,dim> > &values) const;
272 };
273
274 template <int dim>
275 Tensor<1,dim> ExactSolutionVorticityTensor<dim>::value (const Point<dim>
      &p) const
```

```
276 {
277   ExactSolutionVorticity<dim> curl_utrue;
278   Tensor<1,dim> ten;
279
280   for (unsigned int d=0; d<dim; d++)
281     ten[d] = curl_utrue.value(p,d);
282
283   return ten;
284 }
285
286 template <int dim>
287 void ExactSolutionVorticityTensor<dim>::value_list (const std::vector<
      Point<dim> > &points, std::vector<Tensor<1,dim> > &values) const
288 {
289   for (unsigned int p=0; p<points.size(); p++)
290     {
291       values[p].clear ();
292       values[p] = ExactSolutionVorticityTensor<dim>::value(points[p]);
293     }
294 }
295
296
297 /*****************************/
298 /*                           */
299 /* RHS class implementations */
300 /*                           */
301 /*****************************/
302 /*********************************/
303 /*  RHS function class - velocity */
304 /*********************************/
```

```
305 template <int dim>
306 class RightHandSide : public Function<dim>
307 {
308 public:
309   RightHandSide(const double Reynolds) : Function<dim> (dim+1), Reynolds
        (Reynolds) {};
310   virtual double value(const Point<dim> &p, const unsigned int component
        ) const;
311   virtual void vector_value(const Point<dim> &p, Vector<double> &values)
         const;
312   virtual void vector_value_list(const std::vector<Point<dim> > &points,
         std::vector<Vector<double> > &value_list) const;
313 private:
314   const double Reynolds;
315 };
316
317 template <int dim>
318 double RightHandSide<dim>::value (const Point<dim> &p, const unsigned
     int component) const
319 {
320   const ExactSolutionTensor<dim> uten;
321   const ExactSolution<dim> u;
322
323   if (component == 0)
324     return 2*pi*std::cos(2*pi*p[1])*std::cos(2*pi*p[2]) + (1.0/Reynolds)
          *4*pi*pi*std::sin(2*pi*p[1]) + 2*pi*std::cos(2*pi*p[0]);
325   else if (component == 1)
326     return -2*pi*std::sin(2*pi*p[2])*std::exp(p[0]) + (1.0/Reynolds)*4*
          pi*pi*std::cos(2*pi*p[2]) - 2*pi*std::sin(2*pi*p[1]);
327   else if (component == 2)
```

```
328       return std::sin(2*pi*p[1])*std::exp(p[0]) - (1.0/Reynolds)*std::exp(
              p[0]) + 2*pi*std::cos(2*pi*p[2]);
329     else if (component == 3)
330       return 0;
331
332     return 0;
333   }
334
335   template <int dim>
336   void RightHandSide<dim>::vector_value(const Point<dim> &p, Vector<double
        > &values) const
337   {
338     for (unsigned int c=0; c<this->n_components; c++)
339       values(c) = RightHandSide<dim>::value (p,c);
340   }
341
342   template <int dim>
343   void RightHandSide<dim>::vector_value_list(const std::vector<Point<dim>
        > &points, std::vector<Vector<double> > &value_list) const
344   {
345     const unsigned int n_points = points.size();
346
347     for (unsigned int p=0; p<n_points; p++)
348       RightHandSide<dim>::vector_value(points[p], value_list[p]);
349   }
350
351   /*********************************/
352   /*  RHS function class - vorticity */
353   /*********************************/
354   template <int dim>
```

```
355 class RightHandSideVorticity : public Function<dim>
356 {
357 public:
358   RightHandSideVorticity(const double Reynolds) : Function<dim> (dim),
        Reynolds(Reynolds) {};
359   virtual double value(const Point<dim> &p, const unsigned int component
        ) const;
360   virtual void vector_value(const Point<dim> &p, Vector<double> &values)
         const;
361   virtual void vector_value_list(const std::vector<Point<dim> > &points,
         std::vector<Vector<double> > &value_list) const;
362
363 private:
364   const double Reynolds;
365 };
366
367 template <int dim>
368 double RightHandSideVorticity<dim>::value (const Point<dim> &p, const
     unsigned int component) const
369 {
370   if (component == 0)
371     return 2*pi*std::cos(2*pi*p[1])*std::exp(p[0]) + 4*pi*pi*std::cos(2*
         pi*p[2])*std::exp(p[0]) + (1.0/Reynolds)*8*pi*pi*pi*std::sin(2*
         pi*p[2]);
372   else if (component == 1)
373     return -4*pi*pi*std::sin(2*pi*p[2])*std::cos(2*pi*p[1]) - std::sin
         (2*pi*p[1])*std::exp(p[0]) + (1.0/Reynolds)*std::exp(p[0]);
374   else if (component == 2)
375     return -2*pi*std::sin(2*pi*p[2])*std::exp(p[0]) + 4*pi*pi*std::cos
         (2*pi*p[2])*std::sin(2*pi*p[1]) - (1.0/Reynolds)*8*pi*pi*pi*std
```

```
          ::cos(2*pi*p[1]);
376
377   return 0;
378 }
379
380 template <int dim>
381 void RightHandSideVorticity<dim>::vector_value(const Point<dim> &p,
      Vector<double> &values) const
382 {
383   for (unsigned int c=0; c<this->n_components; c++)
384     values(c) = RightHandSideVorticity<dim>::value (p,c);
385 }
386
387 template <int dim>
388 void RightHandSideVorticity<dim>::vector_value_list(const std::vector<
      Point<dim> > &points, std::vector<Vector<double> > &value_list)
      const
389 {
390   const unsigned int n_points = points.size();
391
392   for (unsigned int p=0; p<n_points; p++)
393     RightHandSideVorticity<dim>::vector_value(points[p], value_list[p]);
394 }
395
396 /*****************************************/
397 /*  RHS function class (tensor) - velocity */
398 /*****************************************/
399 template <int dim>
400 class RightHandSideTensor : public TensorFunction<1,dim>
401 {
```

```
402 public:
403   RightHandSideTensor(const double Reynolds) : TensorFunction<1,dim>(),
        Reynolds(Reynolds) {}
404   virtual Tensor<1,dim> value (const Point<dim> &p) const;
405   virtual void value_list (const std::vector<Point<dim> > &points, std::
        vector<Tensor<1,dim> > &values) const;
406
407 private:
408   const double Reynolds;
409 };
410
411 template <int dim>
412 Tensor<1,dim> RightHandSideTensor<dim>::value (const Point<dim> &p)
      const
413 {
414   const RightHandSide<dim> rhs(Reynolds);
415   Tensor<1,dim> ten;
416
417   for (unsigned int d=0; d<dim; d++)
418     ten[d] = rhs.value(p,d);
419
420   return ten;
421 }
422
423 template <int dim>
424 void RightHandSideTensor<dim>::value_list (const std::vector<Point<dim>
     > &points, std::vector<Tensor<1,dim> > &values) const
425 {
426   for (unsigned int p=0; p<points.size(); p++)
427     {
```

```
428         values[p].clear ();
429         values[p] = RightHandSideTensor<dim>::value(points[p]);
430       }
431 }
432
433 /*****************************************/
434 /*  RHS function class (tensor) - vorticity*/
435 /*****************************************/
436 template <int dim>
437 class RightHandSideVorticityTensor : public TensorFunction<1,dim>
438 {
439 public:
440   RightHandSideVorticityTensor(const double Reynolds) : TensorFunction
          <1,dim>(), Reynolds(Reynolds) {}
441   virtual Tensor<1,dim> value (const Point<dim> &p) const;
442   virtual void value_list (const std::vector<Point<dim> > &points, std::
          vector<Tensor<1,dim> > &values) const;
443
444 private:
445   const double Reynolds;
446 };
447
448 template <int dim>
449 Tensor<1,dim> RightHandSideVorticityTensor<dim>::value (const Point<dim>
       &p) const
450 {
451   const RightHandSideVorticity<dim> rhs(Reynolds);
452   Tensor<1,dim> ten;
453
454   for (unsigned int d=0; d<dim; d++)
```

```
455       ten[d] = rhs.value(p,d);
456
457     return ten;
458   }
459
460   template <int dim>
461   void RightHandSideVorticityTensor<dim>::value_list (const std::vector<
        Point<dim> > &points, std::vector<Tensor<1,dim> > &values) const
462   {
463     for (unsigned int p=0; p<points.size(); p++)
464       {
465         values[p].clear ();
466         values[p] = RightHandSideVorticityTensor<dim>::value(points[p]);
467       }
468   }
469
470
471   /*************************************************/
472   /*                                               */
473   /* Dirichlet boundary function class implementations */
474   /*                                               */
475   /*************************************************/
476   /***************************************/
477   /*  Boundary function class - velocity  */
478   /***************************************/
479   template <int dim>
480   class BoundaryValues: public Function<dim>
481   {
482   public:
483     BoundaryValues() : Function<dim> (dim+1) {};
```

```
484     virtual double value (const Point<dim> &p, const unsigned int
            component) const;
485     virtual void vector_value (const Point<dim> &p, Vector<double> &value)
             const;
486   };
487
488   template <int dim>
489   double BoundaryValues<dim>::value (const Point<dim> &p, const unsigned
        int component) const
490   {
491     const ExactSolution<dim> utrue;
492     return utrue.value(p, component);
493
494     return 0;
495   }
496
497   template <int dim>
498   void BoundaryValues<dim>::vector_value (const Point<dim> &p, Vector<
        double> &values) const
499   {
500     for (unsigned int c=0; c < this->n_components; c++)
501       values(c) = BoundaryValues<dim>::value(p,c);
502   }
503
504   /*************************************/
505   /*  Boundary function class - vorticity */
506   /*************************************/
507   template <int dim>
508   class BoundaryValuesVorticity: public Function<dim>
509   {
```

```
510 public:
511   BoundaryValuesVorticity() : Function<dim> (dim) {};
512   virtual double value (const Point<dim> &p, const unsigned int
        component) const;
513   virtual void vector_value (const Point<dim> &p, Vector<double> &value)
         const;
514 };
515
516 template <int dim>
517 double BoundaryValuesVorticity<dim>::value (const Point<dim> &p, const
      unsigned int component) const
518 {
519   const ExactSolutionVorticity<dim> wtrue;
520   return wtrue.value(p,component);
521
522
523   return 0;
524 }
525
526 template <int dim>
527 void BoundaryValuesVorticity<dim>::vector_value (const Point<dim> &p,
      Vector<double> &values) const
528 {
529   for (unsigned int c=0; c < this->n_components; c++)
530     values(c) = BoundaryValuesVorticity<dim>::value(p,c);
531 }
532
533 /*********************************************/
534 /* Boundary function class (tensor) - velocity  */
535 /*********************************************/
```

```
536 template <int dim>
537 class BoundaryValuesTensor : public TensorFunction<1,dim>
538 {
539 public:
540   BoundaryValuesTensor() : TensorFunction<1,dim>() {}
541   virtual Tensor<1,dim> value (const Point<dim> &p) const;
542   virtual void value_list (const std::vector<Point<dim> > &points, std::
        vector<Tensor<1,dim> > &values) const;
543 };
544
545 template <int dim>
546 Tensor<1,dim> BoundaryValuesTensor<dim>::value (const Point<dim> &p)
      const
547 {
548   const BoundaryValues<dim> g;
549   Tensor<1,dim> ten;
550
551   for (unsigned int d=0; d<dim; d++)
552     ten[d] = g.value(p,d);
553
554   return ten;
555 }
556
557 template <int dim>
558 void BoundaryValuesTensor<dim>::value_list (const std::vector<Point<dim>
      > &points, std::vector<Tensor<1,dim> > &values) const
559 {
560   for (unsigned int p=0; p<points.size(); p++)
561     {
562       values[p].clear ();
```

```
563         values[p] = BoundaryValuesTensor<dim>::value(points[p]);
564       }
565   }
566
567
568   /*********************************************************/
569   /*                                                       */
570   /* Steady NSE / vorticity equation class implementations */
571   /*                                                       */
572   /*********************************************************/
573   template <int dim>
574   class SNSE
575   {
576   public:
577     SNSE ();
578     ~SNSE ();
579     void run ();
580
581   private:
582     void make_grid ();
583     void setup_system();
584     void assemble_velocity_system ();
585     void assemble_velocity_system_interval (const typename DoFHandler<dim
          >::active_cell_iterator &begin, const typename DoFHandler<dim>::
          active_cell_iterator &end);
586     void assemble_vorticity_system ();
587     void solve_vel ();
588     void solve_vort ();
589     double compute_relative_norm ();
590     void compute_error ();
```

```
591     void output_results ();
592
593     std::string outpath;
594     Threads::Mutex assembler_lock;
595
596     const unsigned int num_mesh_div;
597     const unsigned int degree_vel;
598     const unsigned int degree_vort;
599     const double Re;
600
601     Triangulation<dim> triangulation;
602
603     FESystem<dim> fe_vel;
604     ConstraintMatrix constraints_vel;
605     DoFHandler<dim> dof_handler_vel;
606     SparsityPattern sparsity_pattern_vel;
607     SparseMatrix<double> system_matrix_vel;
608     Vector<double> system_rhs_vel;
609     Vector<double> prev_solution_vel;
610
611     FESystem<dim> fe_vort;
612     ConstraintMatrix constraints_vort;
613     DoFHandler<dim> dof_handler_vort;
614     SparsityPattern sparsity_pattern_vort;
615     SparseMatrix<double> system_matrix_vort;
616     Vector<double> system_rhs_vort;
617 };
618
619
620 /*******************************/
```

```
621 /* Class constructor/destructor */
622 /*******************************/
623 template <int dim>
624 SNSE<dim>::SNSE ()
625   :
626   num_mesh_div(4),
627   degree_vel(2),
628   degree_vort(2),
629   Re(1.0),
630   fe_vel (FE_Q<dim>(degree_vel), dim, FE_Q<dim>(degree_vel-1),1),
631   dof_handler_vel(triangulation),
632   fe_vort (FE_Q<dim>(degree_vort), dim),
633   dof_handler_vort(triangulation)
634 {
635   outpath = "VorticityEq_";
636 }
637
638 template <int dim>
639 SNSE<dim>::~SNSE ()
640 {
641   dof_handler_vel.clear();
642   dof_handler_vort.clear();
643 }
644
645
646 /********************/
647 /*  Mesh Generation  */
648 /********************/
649 template <int dim>
650 void SNSE<dim>::make_grid ()
```

```
651 {
652   std::cout << "Reynolds number is: " << Re << std::endl;
653   std::cout << "Constructing the mesh..." << std::endl;
654
655   GridGenerator::subdivided_hyper_cube(triangulation, num_mesh_div, 0,
          1);
656
657   std::string filename = outpath+"Mesh.eps";
658   std::ofstream output (filename.c_str());
659   GridOut grid_out;
660   grid_out.write_eps (triangulation, output);
661
662   for (typename Triangulation<dim>::active_cell_iterator cell =
          triangulation.begin_active(); cell != triangulation.end(); ++cell)
663     for (unsigned int f=0; f<GeometryInfo<dim>::faces_per_cell; ++f)
664       {
665         if (cell->face(f)->center()[dim-2] == 0 || cell->face(f)->center
              ()[dim-2] == 1)
666           cell->face(f)->set_all_boundary_indicators(1);
667         if (cell->face(f)->center()[dim-1] == 0 || cell->face(f)->center
              ()[dim-1] == 1)
668           cell->face(f)->set_all_boundary_indicators(2);
669       }
670 }
671
672
673 /*****************/
674 /*  System Setup  */
675 /*****************/
676 template <int dim>
```

```
677 void SNSE<dim>::setup_system ()
678 {
679   std::cout << "Initializing..." << std::endl;
680
681   // Initialize velocity/pressure
682   dof_handler_vel.distribute_dofs (fe_vel);
683
684   constraints_vel.clear ();
685   {
686     std::vector<bool> component_mask_vel (dim+1, true);
687     component_mask_vel[dim] = false;
688     VectorTools::interpolate_boundary_values (dof_handler_vel, 0,
          BoundaryValues<dim>(), constraints_vel, component_mask_vel);
689     VectorTools::interpolate_boundary_values (dof_handler_vel, 1,
          BoundaryValues<dim>(), constraints_vel, component_mask_vel);
690     VectorTools::interpolate_boundary_values (dof_handler_vel, 2,
          BoundaryValues<dim>(), constraints_vel, component_mask_vel);
691
692     std::vector<bool> component_mask_pres (dim+1, false);
693     component_mask_pres[dim] = true;
694     std::vector<bool> boundary_dofs (dof_handler_vel.n_dofs(), false);
695     DoFTools::extract_boundary_dofs (dof_handler_vel,
          component_mask_pres, boundary_dofs);
696
697     const unsigned int first_boundary_dof = std::distance (boundary_dofs
          .begin(), std::find (boundary_dofs.begin(), boundary_dofs.end(),
           true));
698
699     constraints_vel.add_line (first_boundary_dof);
700     for (unsigned int i=first_boundary_dof+1; i<dof_handler_vel.n_dofs()
```

```
        ; ++i)
701       if (boundary_dofs[i] == true)
702           constraints_vel.add_entry (first_boundary_dof, i, -1);
703   }
704   constraints_vel.close ();
705
706   CompressedSparsityPattern csp_vel (dof_handler_vel.n_dofs());
707   DoFTools::make_sparsity_pattern (dof_handler_vel, csp_vel,
          constraints_vel, false);
708   sparsity_pattern_vel.copy_from (csp_vel);
709   system_matrix_vel.reinit (sparsity_pattern_vel);
710   system_rhs_vel.reinit (dof_handler_vel.n_dofs());
711   prev_solution_vel.reinit (dof_handler_vel.n_dofs());
712
713   std::cout << "Number of velocity/pressure degrees of freedom: " <<
          dof_handler_vel.n_dofs() << std::endl;
714
715   // Initialize vorticity
716   dof_handler_vort.distribute_dofs (fe_vort);
717
718   constraints_vort.clear();
719   {
720     std::vector<bool> component_mask_x (dim,false);
721     component_mask_x[dim-3] = true;
722     VectorTools::interpolate_boundary_values (dof_handler_vort, 0,
            BoundaryValuesVorticity<dim>(), constraints_vort,
            component_mask_x);
723     std::vector<bool> component_mask_y (dim,false);
724     component_mask_y[dim-2] = true;
725     VectorTools::interpolate_boundary_values (dof_handler_vort, 1,
```

```
            BoundaryValuesVorticity<dim>(), constraints_vort,
            component_mask_y);
726       std::vector<bool> component_mask_z (dim,false);
727       component_mask_z[dim-1] = true;
728       VectorTools::interpolate_boundary_values (dof_handler_vort, 2,
            BoundaryValuesVorticity<dim>(), constraints_vort,
            component_mask_z);
729     }
730     constraints_vort.close();
731
732     CompressedSparsityPattern csp_vort (dof_handler_vort.n_dofs());
733     DoFTools::make_sparsity_pattern (dof_handler_vort, csp_vort,
          constraints_vort, false);
734     sparsity_pattern_vort.copy_from (csp_vort);
735     system_matrix_vort.reinit (sparsity_pattern_vort);
736     system_rhs_vort.reinit (dof_handler_vort.n_dofs());
737
738     std::cout << "Number of vorticity degrees of freedom: " <<
          dof_handler_vort.n_dofs() << std::endl;
739 }
740
741
742 /*****************************/
743 /*  Velocity System Assembly  */
744 /*****************************/
745 template <int dim>
746 void SNSE<dim>::assemble_velocity_system ()
747 {
748     Timer AssemblyTimer;
749     AssemblyTimer.start ();
```

```
750
751     const unsigned int n_threads = multithread_info.n_default_threads;
752     std::cout << "Assembling the velocity system matrix and rhs vector ("
            << n_threads << " threads assembling in parallel)..." << std::endl
            ;
753
754     system_matrix_vel = 0;
755     system_rhs_vel = 0;
756
757     Threads::ThreadGroup<> threads;
758
759     typedef typename DoFHandler<dim>::active_cell_iterator
            active_cell_iterator;
760     std::vector<std::pair<active_cell_iterator,active_cell_iterator> >
            thread_ranges = Threads::split_range<active_cell_iterator> (
            dof_handler_vel.begin_active (), dof_handler_vel.end (), n_threads
            );
761
762     for (unsigned int thread=0; thread<n_threads; ++thread)
763       threads += Threads::new_thread (&SNSE<dim>::
              assemble_velocity_system_interval, *this, thread_ranges[thread].
              first, thread_ranges[thread].second);
764
765     threads.join_all ();
766
767     AssemblyTimer.stop ();
768     std::cout << "  Elapsed wall time: " << std::floor(AssemblyTimer.
            wall_time() + 0.5) << " seconds" << std::endl;
769 }
770
```

```
771
772 /*************************************/
773 /*  Velocity System Assembly Interval */
774 /*************************************/
775 template <int dim>
776 void SNSE<dim>::assemble_velocity_system_interval (const typename
        DoFHandler<dim>::active_cell_iterator &begin, const typename
        DoFHandler<dim>::active_cell_iterator &end)
777 {
778   system_matrix_vel = 0;
779   system_rhs_vel = 0;
780
781   QGauss<dim>  quadrature_formula(degree_vel + 1);
782   const unsigned int n_q_points = quadrature_formula.size();
783
784   FEValues<dim> fe_values (fe_vel, quadrature_formula, update_values |
          update_gradients | update_quadrature_points | update_JxW_values);
785   const unsigned int dofs_per_cell = fe_vel.dofs_per_cell;
786   std::vector<unsigned int> local_dof_indices (dofs_per_cell);
787
788   const RightHandSide<dim> right_hand_side(Re);
789
790   FullMatrix<double> cell_matrix (dofs_per_cell, dofs_per_cell);
791   Vector<double> cell_rhs (dofs_per_cell);
792
793   std::vector<Vector<double> > rhs_values(n_q_points, Vector<double>(dim
          +1));
794   std::vector<Tensor<1,dim> > prev_vel_values(n_q_points);
795   std::vector<Tensor<2,dim> > prev_vel_grads(n_q_points);
796
```

```
797   std::vector<Tensor<2,dim> > grad_phi_u (dofs_per_cell);
798   std::vector<double> div_phi_u (dofs_per_cell);
799   std::vector<Tensor<1,dim> > phi_u (dofs_per_cell);
800   std::vector<double> phi_p (dofs_per_cell);
801
802   const FEValuesExtractors::Vector velocities(0);
803   const FEValuesExtractors::Scalar pressure (dim);
804
805   typename DoFHandler<dim>::active_cell_iterator cell;
806
807   for (cell = begin; cell!=end; cell++)
808     {
809       fe_values.reinit(cell);
810
811       cell_matrix = 0;
812       cell_rhs = 0;
813
814       right_hand_side.vector_value_list (fe_values.get_quadrature_points
              (), rhs_values);
815       fe_values[velocities].get_function_values(prev_solution_vel,
              prev_vel_values);
816       fe_values[velocities].get_function_gradients(prev_solution_vel,
              prev_vel_grads);
817
818       for (unsigned int q=0; q<n_q_points; q++)
819         {
820           for (unsigned int k=0; k<dofs_per_cell; k++)
821             {
822               grad_phi_u[k] = fe_values[velocities].gradient (k,q);
823               div_phi_u[k] = fe_values[velocities].divergence (k,q);
```

```
824                 phi_u[k] = fe_values[velocities].value (k,q);
825                 phi_p[k] = fe_values[pressure].value (k,q);
826               }
827
828             for (unsigned int i=0; i<dofs_per_cell; i++)
829               {
830                 for (unsigned int j=0; j<dofs_per_cell; j++)
831                   cell_matrix(i,j) += (grad_phi_u[j]*prev_vel_values[q]*
                        phi_u[i]
832                                        + prev_vel_grads[q]*phi_u[j]*phi_u[
                                             i]
833                                        + (1.0/Re)*double_contract(
                                             grad_phi_u[j], grad_phi_u[i])
834                                        - phi_p[j]*div_phi_u[i]
835                                        + div_phi_u[j]*phi_p[i]
836                                       )*fe_values.JxW(q);
837
838                 const unsigned int component_i = fe_vel.
                        system_to_component_index(i).first;
839                 cell_rhs(i) += (fe_values.shape_value(i,q)*rhs_values[q](
                        component_i) + prev_vel_grads[q]*prev_vel_values[q]*
                        phi_u[i])*fe_values.JxW(q);
840               }
841           }
842
843       cell->get_dof_indices (local_dof_indices);
844
845       assembler_lock.acquire ();
846       constraints_vel.distribute_local_to_global (cell_matrix, cell_rhs,
              local_dof_indices, system_matrix_vel, system_rhs_vel);
```

```
847         assembler_lock.release ();
848       }
849 }
850
851
852 /*****************************/
853 /*  Vorticity System Assembly  */
854 /*****************************/
855 template <int dim>
856 void SNSE<dim>::assemble_vorticity_system ()
857 {
858   std::cout << std::endl << "Assembling the vorticity system matrix and
          RHS vector... " << std::endl;
859
860   Timer AssemblyTimer;
861   AssemblyTimer.start();
862
863   QGauss<dim>  quadrature_formula(degree_vort + 1);
864   QGauss<dim-1> face_quadrature_formula(degree_vort + 1);
865   const unsigned int n_q_points = quadrature_formula.size();
866   const unsigned int n_face_q_points = face_quadrature_formula.size();
867
868   FEValues<dim> fe_values_vel (fe_vel, quadrature_formula, update_values
          | update_gradients | update_quadrature_points | update_JxW_values
         );
869   FEValues<dim> fe_values_vort (fe_vort, quadrature_formula,
         update_values | update_gradients | update_quadrature_points |
         update_JxW_values);
870   FEFaceValues<dim> fe_face_values_vel (fe_vel, face_quadrature_formula,
          update_values | update_gradients | update_normal_vectors |
```

```
        update_quadrature_points | update_JxW_values);
871     FEFaceValues<dim> fe_face_values_vort (fe_vort,
        face_quadrature_formula, update_values | update_gradients |
        update_normal_vectors | update_quadrature_points |
        update_JxW_values);
872
873     const unsigned int dofs_per_cell = fe_vort.dofs_per_cell;
874     std::vector<unsigned int> local_dof_indices (dofs_per_cell);
875
876     const RightHandSideVorticity<dim> right_hand_side_vort(Re);
877     const RightHandSideTensor<dim> right_hand_side_tensor(Re);
878     const BoundaryValuesTensor<dim> g;
879
880     FullMatrix<double> cell_matrix (dofs_per_cell, dofs_per_cell);
881     Vector<double> cell_rhs (dofs_per_cell);
882
883     std::vector<Vector<double> > rhs_values(n_q_points, Vector<double>(dim
        ));
884     std::vector<Tensor<1,dim> > velocity_values(n_q_points);
885     std::vector<Tensor<2,dim> > velocity_grads(n_q_points);
886     std::vector<Tensor<1,dim> > rhs_face_values(n_face_q_points);
887     std::vector<Tensor<1,dim> > rhs_face_fxn(n_face_q_points);
888     std::vector<Tensor<1,dim> > g_values(n_face_q_points);
889     std::vector<Tensor<1,dim> > velocity_face_values(n_face_q_points);
890     std::vector<double> pressure_face_values(n_face_q_points);
891
892     std::vector<Tensor<2,dim> > grad_phi_w (dofs_per_cell);
893     std::vector<double> div_phi_w (dofs_per_cell);
894     std::vector<Tensor<1,dim> > phi_w (dofs_per_cell);
895     std::vector<Tensor<1,dim> > curl_phi_w (dofs_per_cell);
```

```
896     std::vector<Tensor<1,dim> > face_phi_w (dofs_per_cell);
897     std::vector<Tensor<1,dim> > face_curl_phi_w (dofs_per_cell);
898
899     const FEValuesExtractors::Vector velocities(0);
900     const FEValuesExtractors::Scalar pressure(dim);
901     const FEValuesExtractors::Vector vorticity(0);
902
903     typename DoFHandler<dim>::active_cell_iterator cell = dof_handler_vort
            .begin_active(), endc = dof_handler_vort.end();
904     typename DoFHandler<dim>::active_cell_iterator vel_cell =
            dof_handler_vel.begin_active();
905
906     for (; cell!=endc; cell++, vel_cell++)
907       {
908         cell_matrix = 0;
909         cell_rhs = 0;
910
911         fe_values_vel.reinit (vel_cell);
912         fe_values_vort.reinit (cell);
913
914         right_hand_side_vort.vector_value_list (fe_values_vort.
                get_quadrature_points(), rhs_values);
915         fe_values_vel[velocities].get_function_values (system_rhs_vel,
                velocity_values);
916         fe_values_vel[velocities].get_function_gradients (system_rhs_vel,
                velocity_grads);
917
918         for (unsigned int q=0; q<n_q_points; q++)
919           {
920             for (unsigned int k=0; k<dofs_per_cell; k++)
```

```
921             {
922               grad_phi_w[k] = fe_values_vort[vorticity].gradient (k,q);
923               curl_phi_w[k] = fe_values_vort[vorticity].curl (k,q);
924               div_phi_w[k] = fe_values_vort[vorticity].divergence (k,q);
925               phi_w[k] = fe_values_vort[vorticity].value (k,q);
926             }
927
928           for (unsigned int i=0; i<dofs_per_cell; i++)
929             {
930               for (unsigned int j=0; j<dofs_per_cell; j++)
931                 cell_matrix(i,j) += (grad_phi_w[j]*velocity_values[q]*
                        phi_w[i]
932                                      - velocity_grads[q]*phi_w[j]*phi_w[
                                          i]
933                                      + (1.0/Re)*curl_phi_w[j]*curl_phi_w
                                          [i]
934                                      + (1.0/Re)*div_phi_w[j]*div_phi_w[i
                                          ]
935                                     )*fe_values_vort.JxW(q);
936
937               const unsigned int component_i = fe_vort.
                      system_to_component_index(i).first;
938               cell_rhs(i) += (fe_values_vort.shape_value(i,q)*rhs_values
                      [q](component_i))*fe_values_vort.JxW(q);
939             }
940         }
941
942
943       // Boundary Integral Contribution
944       for (unsigned int face=0; face < GeometryInfo<dim>::faces_per_cell
```

```
      ; face++)
945       if (cell->at_boundary(face))
946         {
947           fe_face_values_vel.reinit (vel_cell, face);
948           fe_face_values_vort.reinit (cell, face);
949
950           right_hand_side_tensor.value_list (fe_face_values_vort.
                  get_quadrature_points(), rhs_face_values);
951           g.value_list(fe_face_values_vort.get_quadrature_points(),
                  g_values);
952
953           fe_face_values_vel[velocities].get_function_values (
                  system_rhs_vel, velocity_face_values);
954           fe_face_values_vel[pressure].get_function_values (
                  system_rhs_vel, pressure_face_values);
955
956           for (unsigned int q=0; q<n_face_q_points; q++)
957             {
958               double BPressure = 0.5*velocity_face_values[q]*
                      velocity_face_values[q] + pressure_face_values[q];
959               Tensor<1,dim> fxn;
960               cross_product(fxn, rhs_face_values[q],
                      fe_face_values_vort.normal_vector(q));
961
962               for (unsigned int k=0; k<dofs_per_cell; k++)
963                 {
964                   face_phi_w[k] = fe_face_values_vort[vorticity].value
                          (k,q);
965                   face_curl_phi_w[k] = fe_face_values_vort[vorticity].
                          curl(k,q);
```

```
966                     }
967                   for (unsigned int i=0; i<dofs_per_cell; i++)
968                     {
969                       for (unsigned int j=0; j<dofs_per_cell; j++)
970                         {
971                           Tensor<1,dim> wxg;
972                           cross_product(wxg, face_phi_w[j], g_values[q]);
973                           Tensor<1,dim> wxgxn;
974                           cross_product(wxgxn, wxg, fe_face_values_vort.
                                normal_vector(q));
975
976                           cell_matrix(i,j) += wxgxn*face_phi_w[i]*
                                fe_face_values_vort.JxW(q);
977                         }
978
979                       cell_rhs(i) += (fxn*face_phi_w[i]
980                                       - BPressure*(face_curl_phi_w[i]*
                                           fe_face_values_vort.
                                           normal_vector(q))
981                                       )*fe_face_values_vort.JxW(q);
982                     }
983                 }
984             }
985
986         cell->get_dof_indices (local_dof_indices);
987         constraints_vort.distribute_local_to_global (cell_matrix, cell_rhs
              , local_dof_indices, system_matrix_vort, system_rhs_vort);
988       }
989
990     AssemblyTimer.stop();
```

```
991     std::cout << "  Elapsed wall time: " << std::floor(AssemblyTimer.
            wall_time() + 0.5) << " seconds" << std::endl;
992   }
993
994
995   /*****************************/
996   /*  Velocity System Solver  */
997   /*****************************/
998   template <int dim>
999   void SNSE<dim>::solve_vel ()
1000  {
1001    std::cout << "Solving the velocity system... " << std::endl;
1002
1003    Timer SolveTimer;
1004    SolveTimer.start();
1005
1006    SparseDirectUMFPACK  dsolver;
1007    dsolver.solve(system_matrix_vel, system_rhs_vel);
1008
1009    constraints_vel.distribute(system_rhs_vel);
1010
1011    SolveTimer.stop();
1012    std::cout << "  Elapsed wall time: " << std::floor(SolveTimer.
            wall_time() + 0.5) << " seconds" << std::endl;
1013  }
1014
1015
1016  /*****************************/
1017  /*  Vorticity System Solver  */
1018  /*****************************/
```

```
1019 template <int dim>
1020 void SNSE<dim>::solve_vort ()
1021 {
1022   std::cout << "Solving the vorticity system... " << std::endl;
1023
1024   Timer SolveTimer;
1025   SolveTimer.start();
1026
1027   SparseDirectUMFPACK  dsolver;
1028   dsolver.solve(system_matrix_vort, system_rhs_vort);
1029
1030   constraints_vort.distribute(system_rhs_vort);
1031
1032   SolveTimer.stop();
1033   std::cout << "  Elapsed wall time: " << std::floor(SolveTimer.
          wall_time() + 0.5) << " seconds" << std::endl;
1034 }
1035
1036
1037 /*****************************/
1038 /* Compute Relative Difference */
1039 /*****************************/
1040 template <int dim>
1041 double SNSE<dim>::compute_relative_norm ()
1042 {
1043   Vector<double> relative_diff(system_rhs_vel);
1044   relative_diff -= prev_solution_vel;
1045
1046   Vector<double> norm_per_cell (triangulation.n_active_cells());
1047   const ComponentSelectFunction<dim> velocity_mask(std::make_pair(0, dim
```

```
        ), dim+1);
1048    VectorTools::integrate_difference (dof_handler_vel, relative_diff,
            ZeroFunction<dim>(4), norm_per_cell, QGauss<dim>(3), VectorTools::
            L2_norm, &velocity_mask);
1049    double norm = norm_per_cell.l2_norm();
1050
1051    std::cout <<"L2 norm of u_n - u_{n-1}: " << norm << std::endl;
1052    return norm;
1053  }
1054
1055
1056  /*******************/
1057  /*  Compute Error  */
1058  /*******************/
1059  template <int dim>
1060  void
1061  SNSE<dim>::compute_error ()
1062  {
1063    std::cout << std::endl << "Computing error..." << std::endl;
1064
1065    const ComponentSelectFunction<dim> velocity_mask(std::make_pair(0, dim
            ), dim+1);
1066
1067    ExactSolution<dim> exact_solution;
1068    Vector<double> cellwise_errors (triangulation.n_active_cells());
1069
1070    VectorTools::integrate_difference (dof_handler_vel, system_rhs_vel,
            exact_solution, cellwise_errors, QGauss<dim>(3), VectorTools::
            L2_norm, &velocity_mask);
1071    const double u_L2_error = cellwise_errors.l2_norm();
```

```
1072    std::cout << "L2 velocity error: " << u_L2_error << std::endl;
1073
1074    VectorTools::integrate_difference (dof_handler_vel, system_rhs_vel,
            exact_solution, cellwise_errors, QGauss<dim>(3), VectorTools::
            H1_norm, &velocity_mask);
1075    const double u_H1_error = cellwise_errors.l2_norm();
1076    std::cout << "H1 velocity error: " << u_H1_error << std::endl;
1077
1078    ExactSolutionVorticity<dim> exact_solution_vort;
1079
1080    VectorTools::integrate_difference (dof_handler_vort, system_rhs_vort,
            exact_solution_vort, cellwise_errors, QGauss<dim>(3), VectorTools
            ::L2_norm);
1081    const double w_L2_error = cellwise_errors.l2_norm();
1082    std::cout << "L2 vorticity error: " << w_L2_error << std::endl;
1083
1084    VectorTools::integrate_difference (dof_handler_vort, system_rhs_vort,
            exact_solution_vort, cellwise_errors, QGauss<dim>(3), VectorTools
            ::H1_norm);
1085    const double w_H1_error = cellwise_errors.l2_norm();
1086    std::cout << "H1 vorticity error: " << w_H1_error << std::endl;
1087 }
1088
1089
1090 /********************/
1091 /*  Output Results  */
1092 /********************/
1093 template <int dim>
1094 void SNSE<dim>::output_results ()
1095 {
```

```
1096    {
1097      std::string filename = outpath+"Components.vtk";
1098      std::ofstream output (filename.c_str());
1099      DataOut<dim> data_out;
1100      data_out.attach_dof_handler (dof_handler_vel);
1101
1102      std::vector<std::string> solution_names;
1103      solution_names.push_back ("x_vel");
1104      solution_names.push_back ("y_vel");
1105      solution_names.push_back ("z_vel");
1106      solution_names.push_back ("p");
1107
1108      data_out.add_data_vector (system_rhs_vel, solution_names);
1109      data_out.build_patches ();
1110      data_out.write_vtk (output);
1111    }
1112
1113    {
1114      std::vector<std::string> solution_names (dim,"velocity");
1115      solution_names.push_back ("pressure");
1116
1117      std::vector<DataComponentInterpretation::DataComponentInterpretation
            > data_component_interpretation (dim,
            DataComponentInterpretation::component_is_part_of_vector);
1118      data_component_interpretation.push_back (DataComponentInterpretation
            ::component_is_scalar);
1119
1120      DataOut<dim> data_out;
1121      data_out.attach_dof_handler (dof_handler_vel);
1122      data_out.add_data_vector (system_rhs_vel, solution_names,DataOut<dim
```

```
        >::type_dof_data, data_component_interpretation);
1123    data_out.build_patches ();
1124
1125    std::string filename = outpath+"Solution.vtk";
1126    std::ofstream output (filename.c_str());
1127    data_out.write_vtk (output);
1128  }
1129
1130  {
1131    std::vector<std::string> solution_names (dim,"vorticity");
1132
1133    std::vector<DataComponentInterpretation::DataComponentInterpretation
        > data_component_interpretation (dim,
        DataComponentInterpretation::component_is_part_of_vector);
1134
1135    DataOut<dim> data_out;
1136    data_out.attach_dof_handler (dof_handler_vort);
1137    data_out.add_data_vector (system_rhs_vort, solution_names,DataOut<
        dim>::type_dof_data,data_component_interpretation);
1138    data_out.build_patches ();
1139
1140    std::string filename = outpath+"Solution_Vorticity.vtk";
1141    std::ofstream output (filename.c_str());
1142    data_out.write_vtk (output);
1143  }
1144
1145  {
1146    // Output points (velocity)
1147    double n=50;
1148    std::vector<double> xvec(n);
```

```
1149      std::vector<double> yvec(n);
1150      std::vector<double> zvec(n);
1151
1152      for (double ix=0; ix<n; ix++)
1153        {
1154          xvec[ix] = ix/(n-1);
1155          yvec[ix] = ix/(n-1);
1156          zvec[ix] = ix/(n-1);
1157        }
1158
1159      double u1val, u2val, u3val, pval;
1160      Vector<double> solution_val(dim+1);
1161
1162      std::string filename = outpath+"Values";
1163      std::ofstream output (filename.c_str());
1164      output << xvec.size() << " " << yvec.size() << " " << zvec.size() <<
              " " << xvec.size() << " " << yvec.size() << " " << zvec.size()
             << " " << xvec.size() << std::endl;
1165
1166      for (unsigned int ix = 0; ix < xvec.size(); ix++)
1167        {
1168          std::cout << "ix = " << ix << std::endl;
1169
1170          for (unsigned int iy = 0; iy < yvec.size(); iy++)
1171            for (unsigned int iz = 0; iz < zvec.size(); iz++)
1172              {
1173                VectorTools::point_value(dof_handler_vel, system_rhs_vel,
                        Point<dim> (xvec[ix], yvec[iy], zvec[iz]),
                        solution_val);
1174
```

```
1175                u1val = solution_val[0];
1176                u2val = solution_val[1];
1177                u3val = solution_val[2];
1178                pval = solution_val[3];
1179
1180                output << xvec[ix] << " " << yvec[iy] << " " << zvec[iz]
                       << " " << u1val << " " << u2val << " " << u3val << " "
                        << pval << std::endl;
1181              }
1182        }
1183    }
1184
1185    // {
1186    //   // Output points (vorticity)
1187    //   double n=50;
1188    //   std::vector<double> xvec(n);
1189    //   std::vector<double> yvec(n);
1190    //   std::vector<double> zvec(n);
1191
1192    //   for (double ix=0; ix<n; ix++)
1193    //     {
1194    //     xvec[ix] = ix/(n-1);
1195    //     yvec[ix] = ix/(n-1);
1196    //     zvec[ix] = ix/(n-1);
1197    //      }
1198
1199    //   double w1val, w2val, w3val;
1200    //   Vector<double> solution_val(dim);
1201
1202    //   std::string filename = outpath+"Vorticity_Values";
```

```
1203    //    std::ofstream output (filename.c_str());
1204    //    output << xvec.size() << " " << yvec.size() << " " << zvec.size()
            << " " << xvec.size() << " " << yvec.size() << " " << zvec.size()
            << " " << std::endl;
1205
1206    //    for (unsigned int ix = 0; ix < xvec.size(); ix++)
1207    //      {
1208    //      std::cout << "ix = " << ix << std::endl;
1209
1210    //      for (unsigned int iy = 0; iy < yvec.size(); iy++)
1211    //        for (unsigned int iz = 0; iz < zvec.size(); iz++)
1212    //          {
1213    //            VectorTools::point_value(dof_handler_vort, system_rhs_vort
            , Point<dim> (xvec[ix], yvec[iy], zvec[iz]), solution_val);
1214
1215    //            w1val = solution_val[0];
1216    //            w2val = solution_val[1];
1217    //            w3val = solution_val[2];
1218
1219    //            output << xvec[ix] << " " << yvec[iy] << " " << zvec[iz]
            << " " << w1val << " " << w2val << " " << w3val << " " << std::
            endl;
1220    //          }
1221    //      }
1222    // }
1223  }
1224
1225
1226  /*********/
1227  /*  Run  */
```

```
1228 /*********/
1229
1230 template <int dim>
1231 void SNSE<dim>::run ()
1232 {
1233   Timer ProgramTimer;
1234   ProgramTimer.start();
1235
1236   make_grid();
1237   setup_system();
1238
1239   double NLerr = 1;
1240   int NLiter = 0;
1241
1242   while ((NLerr>1e-6) && (NLiter < 10))
1243   {
1244     std::cout << std::endl << "*** Newton iteration " << NLiter+1 << "
          ***" << std::endl;
1245     prev_solution_vel = system_rhs_vel;
1246
1247     assemble_velocity_system ();
1248     solve_vel();
1249
1250     if (NLiter > 0)
1251       NLerr = compute_relative_norm();
1252
1253     NLiter++;
1254   }
1255
1256   if (NLerr < 1e-6)
```

```
1257      std::cout << std::endl << "*** Newton method converged in " <<
              NLiter << " iterations ***" << std::endl;
1258
1259    assemble_vorticity_system ();
1260    solve_vort();
1261
1262    compute_error ();
1263    //output_results ();
1264
1265    ProgramTimer.stop();
1266    std::cout << std::endl << "Total wall time: " << std::floor(
            ProgramTimer.wall_time() + 0.5) << " seconds" << std::endl;
1267  }
1268
1269
1270  /*****************/
1271  /* Main Function */
1272  /*****************/
1273  int main ()
1274  {
1275    using namespace dealii;
1276    deallog.depth_console (0);
1277    {
1278      SNSE<3> flow_problem_3d;
1279      flow_problem_3d.run ();
1280    }
1281
1282    return 0;
1283  }
```

Bibliography

[1] F. Abrahan, M. Behr, and M. Heinkenschloss. Shape optimization in unsteady blood flow: A numerical study of non-newtonain effects. *Comput. Methods Biomech. Biomed. Engng.*, 8:201–212, 2005.

[2] N. A. Adams and S. Stolz. On the Approximate Deconvolution procedure for LES. *Phys. Fluids*, 2:1699–1701, 1999.

[3] N. A. Adams and S. Stolz. Deconvolution methods for subgrid-scale approximation in large eddy simulation. *Modern Simulation Strategies for Turbulent Flow*, 2001.

[4] J. C. André and M. Lesieur. Influence of helicity on high Reynolds number isotropic turbulence. *Journal of Fluid Mechanics*, 81:187–207, 1977.

[5] B.F. Armaly, F. Durst, J.C.F. Pereira, and B. Schönung. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics*, 127:473–496, 1983.

[6] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1–24/27, 2007.

[7] W. Bangerth, T. Heister, G. Kanschat, et al. `deal.II` *Differential Equations Analysis Library, Technical Reference*. `http://www.dealii.org`.

[8] J. Baranger and K. Najib. Analyse numerique des ecoulements quasi-Newtoniens dont la viscosite obeit a la loi puissance ou la loi de carreau. *Numerical Mathematics*, 58:35–49, 1990.

[9] L.C. Berselli and D. Cordoba. On the regularity of the solutions to the 3d Navier-Stokes equations: a remark on the role of the helicity. *Comptes Rendus Mathematique*, 347(11-12):613–618, 2009.

[10] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1991.

[11] J. Burkardt and M. Gunzburger. Sensitivity discrepancy for geometric parameters. *CFD for Design and Optimization, ASME, New York*, pages 9–15, 1995.

[12] Q. Chen, S. Chen, and G. Eyink. The joint cascade of energy and helicity in three dimensional turbulence. *Physics of Fluids*, 15(2):361–374, 2003.

[13] Q. Chen, S. Chen, G. Eyink, and D. Holm. Intermittency in the joint cascade of energy and helicity. *Physical Review Letters*, 90: 214503, 2003.

[14] S.-S. Chow and G. F. Carey. Numerical approximation of generalized Newtonian fluids using Powell-Sabin-Heindl elements: I. theoretical estimates. *International Journal for Numerical Methods in Fluids*, 41(10):1085–1118, 2003.

[15] B. Cousins, L. Rebholz, and N. Wilson. Enforcing energy, helicity and strong mass conservation in FE computations for incompressible Navier-Stokes simulations. *Applied Mathematics and Computation*, 281:1208–1221, 2011.

[16] M. Cross. Rheology of non-Newtonian fluids: A new flow equation for pseudo plastic systems. *Journal of Colloid Science*, 20(5):417–437, 1965.

[17] C. Davies and P. W. Carpenter. A novel velocity-vorticity formulation of the Navier-Stokes equations with applications to boundary layer disturbance evolution. *Journal of Computational Physics*, 172:119–165, 2001.

[18] S. De, K. Nagendra, and K.N. Lakshmisha. Simulation of laminar flow in a three-dimensional lid-driven cavity by the Lattice-Boltzmann method. *International Journal of Numerical Methods for Heat and Fluid Flow*, 19(6):790–815, 2009.

[19] P. Ditlevsen and P. Giuliani. Dissipation in helical turbulence. *Physics of Fluids*, 13, 2001.

[20] P. Ditlevson and P. Guiliani. Cascades in helical turbulence. *Physical Review E*, 63, 2001.

[21] Q. Du, M. Gunzburger, and L.S. Hou. Analysis and finite element approximation of optimal control problems for a Ladyzhenskaya model for stationary, incompressible viscous flows. *Journal of Computational and Applied Mathematics*, 61:323–343, 1995.

[22] A. Dunca. A two-level multiscale deconvolution method for the large eddy simulation of turbulent flows. *Mathematical Models and Methods in Applied Sciences*, 22(6):1–30, 2012.

[23] V.J. Ervin and N. Heuer. Approximation of time-dependent, viscoelastic fluid flow: Crank-Nicolson, finite element approximation. *Numer. Methods Partial Differential Eq.*, 20:248–283, 2003.

[24] V.J. Ervin and H.K. Lee. Numerical approximation of a quasi-newtonian stokes flow problem with defective boundary conditions. *SIAM J. Numer. Anal.*, 45:2120–2140, 2007.

[25] C. Foias, L. Hoang, and B. Nicolaenko. On the helicity in 3D-periodic Navier-Stokes equations I: The non-statistical case. *Proc. London Math. Soc.*, 94:53–90, 2007.

[26] L. Formaggia, J. F. Gerbeau, F. Nobile, and A. Quarteroni. Numerical treatment of defective boundary conditions for the Navier-Stokes equations. *SIAM J. Numer. Anal.*, 40:376–401, 2002.

[27] L. Formaggia, A. Veneziani, and C. Vergara. A new approach to numerical solution of defective boundary value problems in incompressible fluid dynamics. *SIAM J. Numer. Anal.*, 46:2769–2794, 2008.

[28] G.P. Galdi. *An introduction to the Mathematical Theory of the Navier-Stokes Equations, Volume I.* Springer, Berlin, 1994.

[29] K. Galvin, H.K. Lee, and L. Rebholz. Approximation of viscoelastic flows with defective boundary conditions. *Journal of Non-Newtonian Fluid Mechanics*, 169-170:104–113, 2012.

[30] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations : Theory and algorithms.* Springer-Verlag, 1986.

[31] G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University, Baltimore, 1989.

[32] P. Gresho and R. Sani. *Incompressible Flow and the Finite Element Method*, volume 2. Wiley, 1998.

[33] C.W. Groetsch. *Generalized Inverses of Linear Operators.* Marcel Dekker, New York, 1977.

[34] S. Großand and A. Reusken. *Numerical Methods for Two-phase Incompressible Flows*. Springer, Berlin, 2011.

[35] M. Gunzburger. *Perspectives on Flow Control and Optimization*. SIAM, 2003.

[36] M. Gunzburger, L. Hou, and T. Svobodny. Heating and cooling control of temperature distributions along boundaries of flow domains. *J. Math. Syst. Estimat. Control*, 3:147–172, 1993.

[37] M. Gunzburger and H.-C. Lee. Analysis, approximation, and computation of a coupled solid/fluid temperature control problem. *Comput. Methods Appl. Mech. Engrg.*, 118(1-2):133–152, 1994.

[38] M. Gunzburger and H. Wood. Adjoint and sensitivity-based methods for optimization of gas centrifuges. *in:Proceedings of 7th Work. Separation Phenomena in Liquids and Gases; Moscow Engineering Physics Institute*, 2000.

[39] F. Hecht, O. Pironneau, and K. Ohtsuka. Software freefem++. http://www.freefem.org, 2005.

[40] J. Heywood and R. Rannacher. Finite element approximation of the nonstationary Navier-Stokes problem. Part IV: Error analysis for the second order time discretization. *SIAM J. Numer. Anal.*, 2:353–384, 1990.

[41] J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *Int. J. Numer. Methods Fluids*, 22:325–352, 1996.

[42] M. Hribersek, L. Skerget, and Z. Zunic. 3d driven cavity flow by mixed boundary and finite element method. In *European Conference on Computational Fluid Dynamics, ECCOMAS CFD, 2006*.

[43] V. John and A. Liakos. Time dependent flow across a step: the slip with friction boundary condition. *International Journal of Numerical Methods in Fluids*, 50:713 – 731, 2006.

[44] J. Kim, H. Le, and P. Moin. Direct numerical simulation of turbulent flow over a backward-facing step. *Journal of Fluid Mechanics*, 330:349–374, 1997.

[45] K. Kunisch and X. Marduel. Optimal control of non-isothermal viscoelastic fluid flow. *Journal of Non-Newtonian Fluid Mechanics*, 80:261–301, 2000.

[46] W. Layton. *An introduction to the numerical analysis of viscous incompressible flows*. SIAM, 2008.

[47] W. Layton and L. Rebholz. *Approximate Deconvolution Models of Turbulence: Analysis, Phenomenology and Numerical Analysis*. Springer, 2012.

[48] H.K. Lee. Optimal control for quasi-Newtonian flows with defective boundary conditions. *Comput. Methods. Appl. Mech. Engrg.*, 200:2498–2506, 2011.

[49] H.K. Lee and H.C. Lee. Analysis and finite element approximation of an optimal control problem for the oseen viscoelastic fluid flow. *J. Math. Anal. Appl.*, 336:1090–1160, 2007.

[50] H.K. Lee, M.A. Olshanskii, and L.G. Rebholz. On error analysis for the 3D Navier-Stokes equations in Velocity-Vorticity-Helicity form. *SIAM Journal on Numerical Analysis*, 49(2):711–732, 2011.

[51] C. Manica, M. Neda, M.A. Olshanskii, L. Rebholz, and N. Wilson. On an efficient finite element method for Navier-Stokes-omega with strong mass conservation. *Computational Methods in Applied Mathematics*, 11(1):3–22, 2011.

[52] H. L. Meitz and H. F. Fasel. A compact-difference scheme for the Navier-Stokes equations in vorticity-velocity formulation. *Journal of Computational Physics*, 157:371–403, 2000.

[53] H. Moffatt and A. Tsoniber. Helicity in laminar and turbulent flow. *Annual Review of Fluid Mechanics*, 24:281–312, 1992.

[54] M.A. Olshanskii. A fluid solver based on vorticity-helical density equations with application to a natural convection in a cubic cavity. *Int. J. Num. Meth. Fluids*, 2011.

[55] M.A. Olshanskii and L. Rebholz. Velocity-Vorticity-Helicity formulation and a solver for the Navier-Stokes equations. *Journal of Computational Physics*, 229:4291–4303, 2010.

[56] M.A. Olshanskii and L. Rebholz. Application of barycenter refined meshes in linear elasticity and incompressible fluid mechanics. *ETNA: Electronic Transactions in Numerical Analysis*, 38:258–274, 2011.

[57] J. Qin. *On the convergence of some low order mixed finite elements for incompressible fluids.* PhD thesis, Pennsylvania State University, 1994.

[58] L. Rebholz. Efficient, unconditionally stable, and optimally accurate FE algorithms for approximate deconvolution models of fluid flow, submitted.

[59] V. Ruas. A new formulation of the three-dimensional velocity-vorticity system in viscous incompressible flow. *Math. Mech.*, 79:29–36, 1999.

[60] L.R. Scott and M. Vogelius. Conforming finite element methods for incompressible and nearly incompressible continua. volume Lectures in Applied Mathematics, 22-2 of *Large-scale computations in fluid mechanics, Part 2*, pages 221–244. Amer. Math. Soc., 1985.

[61] J. Trujillo and G. E. Karniadakis. A penalty method for the vorticity-velocity formulation. *Journal of Computational Physics*, 149:32–58, 1999.

[62] K.L. Wong and A.J. Baker. A 3d incompressible Navier-Stokes velocity-vorticity weak form finite element algorithm. *International Journal for Numerical Methods in Fluids*, 38:99–123, 2002.

[63] X.H. Wu, J.Z. Wu, and J.M. Wu. Effective vorticity-velocity formulations for the three dimensional incompressible viscous flows. *Journal of Computational Physics*, 122:68–82, 1995.

[64] E. Zeidler. *Nonlinear Functional Analysis and its Applications*, volume III. Springer, New York, 1985.

[65] S. Zhang. A new family of stable mixed finite elements for the 3d Stokes equations. *Mathematics of Computation*, 74:543–554, 2005.

[66] S. Zhang. Divergence-free finite elements on tetrahedral grids for $k \geq 6$. *Math. Comp.*, 80:669–695, 2011.

[67] S. Zhang. Quadratic divergence-free finite elements on Powell-Sabin tetrahedral grids. *Calcolo*, 48:211–244, 2011.