Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Alexander Fuller Viguerie Date

Efficient, stable, and reliable solvers for the steady incompressible Navier-Stokes equations in computational hemodynamics.

By

Alexander Fuller Viguerie

Doctor of Philosophy

Mathematics

Alessandro Veneziani, Ph.D.

Advisor

Michele Benzi, Ph.D.

Committee Member

Leo Rebholz, Ph.D.

Committee Member

James Nagy, Ph.D.

Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.

Efficient, stable, and reliable solvers for the steady incompressible Navier-Stokes equations in computational hemodynamics.

By

Alexander Fuller Viguerie
B.A., Emory University, 2013

Advisor: Alessandro Veneziani, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2018

Abstract

Efficient, stable, and reliable solvers for the steady incompressible Navier-Stokes equations in computational hemodynamics.

By Alexander Fuller Viguerie

In recent years, improvements in medical imaging and image-reconstruction algorithms have led to increased interest in the use of Computational Fluid Dynamics (CFD) as a clinical tool in hemodynamics. While such methods have long been employed in the design of medical devices and in basic medical research, many of the techniques commonly employed in these contexts are not ideal in the clinical setting. In particular, in clinical settings typically one is faced with more demanding turnaround times for simulations, less powerful computational resources, and noisy, incomplete, or missing data.

In this thesis, we discuss these challenges and introduce CFD methods which are more practical for direct clinical application. Frequently in these settings, the variable of interest is the temporal average of some time-periodic quantity, such as wall shear-stress, over a cardiac cycle. In these cases, the standard procedure is to perform an unsteady simulation over several cardiac cycles and then to take the time average of the last one. Here, we propose to instead surrogate the unsteady time-averaged solution with the solution of a steady-state problem, allowing us to compute it directly. This approach, if properly applied, can dramatically lower computational cost as we show here; however in many respects the steady problem is arguably more difficult numerically than its unsteady counterpart.

We will address these difficulties and propose effective workarounds. In particular, we aim to develop methods for steady solvers that are *efficient*, *stable*, and *reliable*. Roughly speaking, this work is divided into three parts, with each part focusing on one of these aspects. Concerning efficiency, we extend the inexact algebraic factorization approach popular for the unsteady problem into the steady setting. We will address the issue of stability by taking inspiration from nonlinear filtering techniques used in turbulence modeling to develop stabilization techniques for the steady problem. Finally, we will develop and validate methods for assigning boundary conditions in data-deficient settings while maintaining reliability. Throughout each section, we will provide both theoretical and numerical justification for our methods.

Efficient, stable, and reliable solvers for the steady incompressible Navier-Stokes equations in computational hemodynamics.

By

Alexander Fuller Viguerie
B.A., Emory University, 2013

Advisor: Alessandro Veneziani, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2018

Acknowledgements

So many people have helped me reach this point, I don't even know where to begin. My only hope is that I don't forget anyone!

First I would like to thank my incredible family- my parents Patrick and Susie and my two brothers Shaun and Sam. You have been here for me every step of the way and I have been so lucky to complete graduate school so close to home. Your love and support was invaluable to me while I completed my PhD, and will continue to be during my next stage and beyond. I love you all so much! Oh, and I can't forget to mention the toy poodles Zoe and Cookie (RIP).

My advisor Alessandro Veneziani has been unbelievable throughout this process. His passion for research really rubs off on all of those around him, and has been a continued source of inspiration for me. But while his scientific guidance is unparalleled, what Alessandro cares about most is that we develop and grow as human beings. I feel honored to be his student, but even more honored to be his friend.

My co-advisor Simona Perotto has also been an incredible influence both professionally and personally. I look forward to many more years of collaboration during my postdoc in Italy and beyond.

I have been blessed with an incredible group friends, and you all know who you are. I'd like to specifically mention three, though: Cole Furrh, Ian Kirby, and Laura Zeng. Y'all were my best friends before I started my PhD and are my best friends today. I couldn't ask for more caring, supportive, and fun people to be friends with. I look forward to Ian joining me as a PhD one day very soon. May the memories of 2307 and Edgar Pearson live on forever. Both.

I've learned a lot from the other people in Alessandro's group and the Emory Center for Computing in Mathematics and Medicine (ECM2). I'd like to acknowledge and thank Don Giddens and Habib Samady for giving me the opportunity to be a part of their team and for being such great bosses. Thanks as well in no particular order to Sofia, Rodrigo, Claire, Huijuan, Ashley, Huanhuan, Heesu, and David. Special acknowledgments go to Adrien, with whom I've worked most closely and from whom I've learned so much, and Luca, who is still the first person I ask when I'm stuck on a problem. Alessandro B- ci vediamo da Mario, prima o poi.

It's been a pleasure to share this experience with the other graduate students in the Emory Math and CS department. Again in no order, thanks Massimiliano, Samy, Yunyi (Larry), Clarissa, James, Sarah, Ariel, Anastassia, Robert (both of

you), Christine, Vicman, Jessi, and whoever I may have forgotten (sorry!) A special shout out goes Reed, who has been an especially good friend and without whom graduate school would have been much harder.

Thanks to my other committee members. Michele- I've learned so much from you over the years and your mentorship has meant a lot to me. I doubt anyone will ever come close to taking as many classes with you as I did. Leo- you've been an excellent scientific resource and I really appreciate all of the research opportunities you've given me during the last year.

Miscellaneous thanks go to Alex Moorhead, Ana Susnjara, Dallas Albritton, tutti i Martinittesi senza Alfredo, Steve, Bill W. and his many friends, and the New Connections Group (it's been great serving as your secretary!) I'd also like to thank The Cottage School and all its staff and students, without whom I never would have made it this far.

And lastly, Kirsten. These past few years with you have been amazing. I can't wait to start our next chapter (and beyond!) Whatever challenges the future may hold, I feel safe knowing that I will face them with you by my side. I love you so much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Computational Fluid Dynamics as a clinical tool

In recent years, the use of Computational Fluid Dynamics models (hereafter abbreviated as *CFD*) in cardiology has gained increased acceptance and popularity. Though such models have long been used for basic research and in the design of medical devices [97, 115, 34, 68, 97, 40], their application has extended to the clinical level, with CFD simulations now being used to help inform medical treatment for individual patients. Such models may help physicians diagnose patients, assess different treatment options, and have even been used to develop new surgical treatments [25, 47, 97, 115, 129, 82].

In particular, CFD simulations have been used extensively as a diagnostic tool. In many cases, *in vivo* measurements are costly and highly invasive, and may even be impossible. In these instances, patient-specific CFD simulations provide an attractive noninvasive alternative to *in vivo* tests. Notably, the startup HeartFlow has secured FDA approval by using CFD to non-invasively assess the severity of coronary stenoses requiring only a CT, eliminating the need for unnecessary catheterization [29].

The emergence of CFD as a tool for clinicians has been made possible in large part due to the advances in medical imaging and image processing that have occurred over the last several decades. Reliable algorithms for reconstructing three-dimensional geometries of specific patients based on medical images have been developed, allowing one to simulate blood flow inside the reconstructed domain [4, 92]. Many packages now exist for the creation of such geometries, in particular the freely available softwares Segment [52] and vmtk [5].

In addition to advancements in image reconstruction, the increase in both computational power and in numerical methods for solving the equations of fluid dynamics have made this possible. Perhaps the most important development in this regard was the Finite Element Method (see e.g. [102, 99]), which has been essential in enabling the numerical solution of the governing equations in nontrivial three-dimensional geometries within a strong mathematical framework.

1.2 Challenges in the clinical setting

The use of CFD as a clinical tool presents several challenges when compared to the academic or industrial setting. Unsteady simulations with upwards of millions of degrees of freedom (DOF) and thousands of time-steps are common in industrial and academic work and may require days, perhaps weeks, to complete. While this guarantees accuracy and allows for more precise modeling, this is not acceptable for clinical purposes. The turnaround time on simulations must be roughly comparable, and should certainly not be significantly longer, to *in vivo* tests in order for CFD to present a practical alternative to such tests. This generally means that simulations should be completed in a matter of minutes.

In addition to the restrictions on computational time, in clinical settings computational power may be severely limited. Computational resources may consist

of only a single workstation or laptop. Thus, the large-scale simulations designed for multicore machines, though standard in other contexts, may not be viable for clinical applications. The demand for fast results coupled with limited computing power therefore requires different computational methods than those used in other settings. While the emergence of high-performance cloud computing facilities over the past several years has mitigated this problem somewhat, there remain concerns regarding network reliability and, most importantly, data protection.

The availability of data is another common problem at the clinical level. The proper prescription of boundary conditions is absolutely critical for the development of reliable simulations that closely reflect reality. However, at the clinical level such data is often unavailable, unreliable or incomplete. For example, one may require flow rates for several branches of a lumen vessel, but some or all of these flow rates may be unavailable. Relying solely on literature values in these instances neglects important patient-specific information and may lead to inaccurate results. While methods for data assimilation have been the subject of much investigation, many of these approaches are computationally intensive and not suitable for this application given the restrictions on computational resources.

1.3 Thesis outline

In this thesis we will address the aforementioned difficulties. We aim to develop efficient CFD methods suitable for the clinical setting, that provide fast and reliable results while remaining tractable in settings with limited computational resources. In particular, the methods must be *efficient*, *stable*, and *reliable*. The outline of the thesis is as follows.

In **Chapter 2**, we will introduce the Incompressible Navier-Stokes Equations for a Newtonian Fluid, which will be our model equations for the entirety of this

work. We will discuss the use of the steady Navier-Stokes equations as the basis for our simulations as opposed to the unsteady (time-dependent) equations. We will motivate and justify this by formally regarding the solution of the steady problem as a surrogate for the time-average of the unsteady problem over a given interval. The advantages and disadvantages of the steady problem compared to the unsteady one will be briefly discussed here and is an important recurring theme that we will revisit many times. Once the steady setting has been properly motivated, we will introduce the mathematical theory of the steady problem and traditional approaches for its solution. This will form an important foundation for what follows. Also note that the notation established this chapter will be used for the remainder of this work.

Chapter 3 will focus on *efficient methods*. We will develop efficient methods based algebraic splitting methods for solving the steady Navier-Stokes problem. These form a popular and efficient class of solution methods for the unsteady problem; however their extension to the steady setting is not straightforward as their construction relies on the presence of a time derivative term. We will show how one may extend this approach and present several mathematical results regarding the steady extension. Numerical results in 2D and 3D will be presented and discussed.

Chapter 4 builds directly on the work from the previous chapter. When extending inexact algebraic factorization methods to the steady setting, one introduces several new challenges. In particular, the nonlinear iterations arising from the methods in Chapter 3 are now partially explicit and therefore require stabilization. In some instances, this stabilization delays convergence. We will demonstrate how to retain many the advantages of the methods in Chapter 3 without an explicit term via the use of grad-div stabilization. We will present several important mathematical results and also demonstrate how this method may allow one

to further approximate the Schur complement (to be defined later) and replace it with a diagonal matrix inversion. Numerical results 2D and 3D will again be presented and discussed.

In **Chapter 5** we will focus on the development of *stable methods*. The steady problem necessarily requires the solution of an iterative nonlinear problem. In practice, this iteration may become unstable due to the presence of flow disturbances which may delay the convergence of the nonlinear iteration or prevent it entirely. Traditional stabilization methods may be ineffective in these instances, as they are designed to address instabilities triggered by high convective fields, while many instabilities encountered hemodynamics are caused by other morphological reasons. We will address this by introducing a new class of stabilization schemes inspired by deconvolution filters popular for unsteady turbulence modeling. We will show relevant mathematical results that establish the well-posedness of the procedure and demonstrate its effectiveness with a series of 2D and 3D numerical tests.

Chapter 6 will focus on the development of *reliable methods*, which refers to a simulations' ability to accurately reflect physical reality. In order for reliability to be achieved, it is essential that boundary conditions be appropriately prescribed. In clinical settings, however, we often encounter the problem of *data-deficiency* in which measured data for prescribing boundary conditions is unreliable, incomplete, or missing entirely. In this situation we must develop methods for appropriately determining and assigning boundary conditions with limited information. After a brief discussion of the various ways to surrogating data and different approaches for assigning data, we will develop new techniques for determining boundary conditions that incorporate available patient-specific information, geometrical information, and well-known physical models. We then validate our approach by comparing against a pool of actual patient data.

Concluding remarks and possible future research directions based on the work shown here will be presented in **Chapter 7**.

Chapter 2

The Steady Incompressible Navier-Stokes Problem

This chapter introduces and develops the relevant background material for the steady Navier-Stokes problem. We will begin by motivating this choice of model by demonstrating how the steady problem can be regarded as a surrogate for the time-average of the unsteady problem. We will then discuss several important mathematical preliminaries regarding the well-posedness of the problem and the standard techniques for its numerical solution.

2.1 Interpretation as time-average

Let $\Omega \in \mathbb{R}^d$ be a suitable domain, where $d = 2, 3$ and let $[0, T]$ be a given time interval. We may assume without loss of generality that the initial time is zero. The time-dependent incompressible Navier-Stokes problem for a Newtonian fluid

in primitive variables then reads:

$$\begin{aligned} \rho\frac{\partial \boldsymbol{u}}{\partial t} - \nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = \boldsymbol{f} \quad &\text{in } \Omega\times[0,T] \\ \nabla\cdot\boldsymbol{u} = 0 \quad &\text{in } \Omega\times[0,T] \\ \mathcal{B}\boldsymbol{u} = \boldsymbol{g} \quad &\text{on } \partial\Omega\times[0,T] \\ \boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0 \quad &\text{at } t = 0 \end{aligned} \tag{2.1.1}$$

where ν is the kinematic viscosity and ρ is the fluid density, which are both hereafter assumed to be constant and known. $\boldsymbol{f}$ is a forcing term, $\boldsymbol{u}$ is a d-vector representing the unknown velocity and the scalar function p is the pressure, both generally functions of time and space. $\mathcal{B}$ is an appropriate trace operator defining the different types of boundary conditions of interest.

The top equation in (2.1.1) corresponds to the conservation of momentum, and the second equation corresponds to the conservation of mass. These will hereafter be referred to as the momentum and mass equations accordingly. The mass equation in this case is our *incompressibility* constraint, and ensuring this condition is enforced can be challenging.

From the physical point of view, the term $-\nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$ corresponds to diffusive forces [1] and the term $\rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ to convective forces. We will refer to these as the *diffusive* and *convective* terms accordingly. The behavior of a given flow is determined largely by the relationship between these terms. Flows in which the diffusive term dominates the convective term are regular and smooth, while convection-dominated flows are more irregular and may exhibit physical phenomena such as recirculation.

[1] We have used the full stress tensor formulation of the diffusive term: $-\nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$. This term arises directly from the derivation of the equations (see e.g. [37]) For incompressible flows this is equivalent to $-\nu\Delta\boldsymbol{u}$ and for this reason one often sees the equations formulated in this way. However, the weak forms of the different formulations are not the same. For the purposes of this work the difference is largely irrelevant; however for certain types of problems, such as fluid-structure interaction problems, this may be significant.

As a general rule, (2.1.1) becomes more difficult to solve as the convective term grows larger or the diffusive term grows smaller. The extent to which the convective term dominates the diffusive term is most commonly indicated by the *Reynolds number*, a dimensionless parameter defined by:

$$\mathrm{Re} = \frac{\rho|\boldsymbol{u}|L}{\nu} \tag{2.1.2}$$

Where $|\boldsymbol{u}|$ is a characteristic flow velocity and L is a characteristic length. For all Reynolds numbers reported here we take L to be the inlet diameter and $|\boldsymbol{u}|$ to be the mean inflow velocity unless otherwise stated. Higher values of Re correspond to more convection-dominated flows.

Flows where Re is low-to-moderate (up to around 2000, though this may change depending on the geometry) are called *laminar flows* and are characterized by a smooth, regular flow profile. As Re grows larger, flows transition to the *turbulent* regime, and the profile becomes highly irregular and nonsmooth.

The numerical solution of (2.1.1) necessitates discretization of both the spatial and temporal domains and often requires one to solve a discretized problem for hundreds, if not thousands, of time steps. In general these types of unsteady computations take a long time, perhaps days or even weeks, to complete.

However, for many cardiovascular quantities of clinical interest, one is not interested in the entire temporal solution, but rather a time average. For instance, one may be interested in the time-averaged wall-shear stress in a cerebral aneurysm over a cardiac cycle. In such cases, rather than computing the entire time loop as required by (2.1.1), we would like to instead compute the time-averaged velocity and pressure (defined as $\overline{\boldsymbol{u}} = \frac{1}{T}\int_0^T \boldsymbol{u}\,dt$ and $\overline{p} = \frac{1}{T}\int_0^T p\,dt$ respectively, with $[0, T]$ representing the time interval of a full cardiac cycle) directly.

Let us decompose $\boldsymbol{u}$ and p into a sum of their respective time-averaged and fluctuating parts:

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x},t) &= \overline{\boldsymbol{u}}(\boldsymbol{x}) + \boldsymbol{u}'(\boldsymbol{x},t) \\ p(\boldsymbol{x},t) &= \overline{p}(\boldsymbol{x}) + p'(\boldsymbol{x},t) \end{aligned} \tag{2.1.3}$$

This is known as the *Reynolds decomposition* in time of the velocity and pressure respectively [103]. By construction:

$$\begin{aligned} \overline{\boldsymbol{u}'} &= \frac{1}{T}\int_0^T \boldsymbol{u}'\ dt = \boldsymbol{0} \quad \text{and} \\ \overline{p'} &= \frac{1}{T}\int_0^T p'\ dt = 0 \end{aligned} \tag{2.1.4}$$

Remark 2.1.1 *We recall the following elementary properties of the time-average operator* $\overline{f} = \frac{1}{T}\int_0^T f\,dt$*:*

1. *For c constant in time (though not necessarily space):* $\overline{c} = c$
2. *For f_1, f_2 time-dependent and c_1, c_2 constant in time (though not necessarily space):* $\overline{c_1 f_1 + c_2 f_2} = c_1\overline{f_1} + c_2\overline{f_2}$
3. *For any time-dependent f:* $\overline{\overline{f}} = \overline{f}$
4. *For any time-dependent f, differential operator in space D:* $\overline{Df} = D\overline{f}$

We now write (2.1.1) in terms of (2.1.3) and apply the time-average operator. From (2.1.4) and the properties of the time-average listed above, it is easy to see that

$$\begin{aligned} \overline{-\nu\nabla\cdot(\nabla(\overline{\boldsymbol{u}}+\boldsymbol{u}')+\nabla(\overline{\boldsymbol{u}}+\boldsymbol{u}')^T)} &= -\nu\nabla\cdot(\nabla(\overline{\overline{\boldsymbol{u}}}+\overline{\boldsymbol{u}'})+\nabla(\overline{\overline{\boldsymbol{u}}}+\overline{\boldsymbol{u}'})^T) \\ &= -\nu\nabla\cdot(\nabla\overline{\boldsymbol{u}}+\nabla\overline{\boldsymbol{u}}^T), \end{aligned} \tag{2.1.5}$$

and similarly,

$$\begin{aligned} \overline{\nabla(\overline{p}+p')} &= \overline{p} \\ \overline{\nabla\cdot(\overline{\boldsymbol{u}}+\boldsymbol{u}')} &= \nabla\cdot\overline{\boldsymbol{u}}. \end{aligned} \tag{2.1.6}$$

For the nonlinear term, one has:

$$\begin{aligned} &\overline{\rho((\overline{\boldsymbol{u}}+\boldsymbol{u}')\cdot\nabla)(\overline{\boldsymbol{u}}+\boldsymbol{u}')} \\ &\qquad = \rho\left(\overline{(\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}+(\boldsymbol{u}'\cdot\nabla)\overline{\boldsymbol{u}}+(\overline{\boldsymbol{u}}\cdot\nabla)\boldsymbol{u}'+(\boldsymbol{u}'\cdot\nabla)\boldsymbol{u}'}\right) \\ &\qquad = \rho\left(\overline{(\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}}+\overline{(\boldsymbol{u}'\cdot\nabla)\overline{\boldsymbol{u}}}+\overline{(\overline{\boldsymbol{u}}\cdot\nabla)\boldsymbol{u}'}+\overline{(\boldsymbol{u}'\cdot\nabla)\boldsymbol{u}'}\right). \end{aligned} \tag{2.1.7}$$

Since $\overline{\boldsymbol{u}}$ is constant in time, property (2) of the time-average in Remark 2.1.1 implies:

$$\begin{aligned} &\overline{\rho((\overline{\boldsymbol{u}}+\boldsymbol{u}')\cdot\nabla)(\overline{\boldsymbol{u}}+\boldsymbol{u}')} \\ &\qquad = \rho\left((\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}+(\overline{\boldsymbol{u}'}\cdot\nabla)\overline{\boldsymbol{u}}+(\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}'}+\overline{(\boldsymbol{u}'\cdot\nabla)\boldsymbol{u}'}\right) \\ &\qquad = \rho(\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}+\overline{\rho(\boldsymbol{u}'\cdot\nabla)\boldsymbol{u}'}. \end{aligned} \tag{2.1.8}$$

where the last line follows from (2.1.4). From (2.1.5), (2.1.6), and (2.1.8), the time-average of (2.1.1) is then given by the following system of partial differential equations:

$$\begin{aligned} -\nu\nabla\cdot(\nabla\overline{\boldsymbol{u}}+\nabla\overline{\boldsymbol{u}}^T)+\rho(\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}+\overline{\rho(\boldsymbol{u}'\cdot\nabla)\boldsymbol{u}'}+\nabla\overline{p} &= \overline{\boldsymbol{f}} \quad \text{in } \Omega \\ \nabla\cdot\overline{\boldsymbol{u}} &= 0 \quad \text{in } \Omega \\ \mathcal{B}\overline{\boldsymbol{u}} &= \overline{\boldsymbol{g}} \quad \text{on } \partial\Omega \end{aligned} \tag{2.1.9}$$

These are the *stationary Reynolds-Averaged Navier Stokes Equations*, commonly abbreviated as RANS. This formulation specifically corresponds to the time averaged RANS equations. Though first introduced as a time average, more general

formulations based on ensemble-averaging are also (note these formulations are identical to (2.1.9) for the case of statistically stationary flow) [106, 128, 75, 14]. Unsteady RANS formulations also exist, in which (2.1.1) are not averaged over the whole time-domain, but rather over a series of subintervals. These are sometimes referred to as URANS (*Unsteady Reynolds-Averaged Navier Stokes Equations*) to distinguish them from (2.1.9) [75, 113, 61, 118]. URANS is commonly used in situations in which turbulent dynamics exhibit periodic characteristics (as observed, for instance, with vortex shedding past a bluff body). In this case, one may employ phase-averaging (2.1.1) to resolve the unsteady periodic fluctuations in the flow field [118, 61, 113, 14]. We will hereafter assume a statistically stationary flow regime and restrict our attention to the steady time-averaged formulation (2.1.9) in this work.

The system is not closed due to the presence of the term $\overline{\rho(\boldsymbol{u}' \cdot \nabla)\boldsymbol{u}'}$. This term is known as the *Reynolds stress* or *turbulent stress* term. In order to close the system, this term must be modeled. Many different Reynolds stress models have been proposed and studied. We refer the reader to [103, 83, 128, 93] for more detailed discussion of different models.

For flows with low-to-moderate Reynolds numbers (laminar flows), however, it is reasonable to assume that the contributions of the Reynolds stress term are insignificant and the term may simply be disregarded [93]. For cardiovascular applications, this encapsulates many flows of significant clinical interest, including flows in major arterial systems (notably the coronary and carotid arteries) as well as venous flow. This assumption is generally not valid for flows in the Aorta [40]. As it is commensurate with our application, we will hereafter assume this flow regime and denote $\overline{\boldsymbol{u}}$ and $\overline{p}$ as simply $\boldsymbol{u}$ and p for ease of notation. We note that the numerical methods discussed here will still hold for cases when the Reynolds stress term is non-negligible and modeled explicitly, however we will not consider

these cases.

This now gives us a steady-state system. The steady setting offers several advantages over the unsteady problem for our purposes. Most importantly, we will generally require far fewer iterations to achieve a satisfactory solution. Rather than computing hundreds (or thousands) of time steps, ideally we can compute a reasonable surrogate for the time-average in a far fewer number of nonlinear iterations (generally on the order of tens).

However, the steady problem is not without its difficulties and the development of efficient solution methods remains an open problem [64]. As we will see later, terms originating from time derivatives are numerically beneficial and give one efficient ways for managing the nonlinearities. In the unsteady problem, one may easily decouple the velocity and pressure variables and solve for them separately. This is less straightforward in the steady case. While the steady problem generally requires require fewer iterations, the tradeoff is that now each iteration is much more difficult to solve. Finally, for the steady problem we must rely on some sort of nonlinear iterative process to obtain a solution, introducing all of the potential difficulties associated with such schemes (such as convergence and stability concerns). In contrast, such iterations may be avoided for the unsteady problem without sacrificing stability (for example, by using semi-implicit time discretization, see e.g. [99, 102]).

2.2 Mathematical Background and Preliminaries

The Steady Navier-Stokes Problem for an incompressible Newtonian fluid in primitive variables is given by

$$\begin{aligned} -\nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)+\rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}+\nabla p &= \boldsymbol{f} \quad \text{in } \Omega \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \mathcal{B}\boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \partial\Omega, \end{aligned} \tag{2.2.1}$$

where all the notation is the same as before. (2.2.1) resembles (2.1.1) but with no time derivative. As in (2.1.1), $\mathcal{B}$ is a trace operator defining the boundary conditions. In particular, we consider standard options, namely

$$\text{Dirichlet conditions} : \boldsymbol{u}(\Gamma_D) = \boldsymbol{g}_D,$$

and

$$\text{Neumann conditions} : \left(p\boldsymbol{n} - \nu\nabla\left(\boldsymbol{u}+\boldsymbol{u}^T\right)\cdot\boldsymbol{n}\right)(\Gamma_N) = \boldsymbol{g}_N,$$

where $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$, $\boldsymbol{n}$ is the unit normal vector to the boundary and $\boldsymbol{g}_D$ and $\boldsymbol{g}_N$ are given. In the following, we will assume for simplicity data $\boldsymbol{g} = \boldsymbol{0}$ on both Dirichlet and Neumann boundaries. From the physical point of view, the terms in (2.2.1) have the same interpretation as described in the previous section.

The theory of this problem is addressed e.g. in [45, 116, 70] . The weak formulation is promptly obtained by standard arguments. Let V be the Sobolev space $\left(H^1_{\Gamma_D}\right)^d(\Omega)$ and $Q = L^2(\Omega)$ (in the case $\Gamma_N = \emptyset$, Q is the space $L^2_0(\Omega)$ of the null-average L^2 functions). Denote the dual space of V as V' with its norm

defined in the standard way where $(\cdot,\cdot)$ is the usual scalar product in L^2:

$$\|\boldsymbol{\phi}\|_{-1} := \sup_{\boldsymbol{v}\in V,\, \boldsymbol{v}\neq \mathbf{0}} \frac{|(\boldsymbol{\phi},\boldsymbol{v})|}{\|\boldsymbol{v}\|_V} \tag{2.2.2}$$

Let us define

$$a(\cdot,\cdot) : V\times V \to \mathbb{R}, \text{s.t. } a(\boldsymbol{w},\boldsymbol{v}) = \nu \int_\Omega (\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T) : \nabla \boldsymbol{v}, \tag{2.2.3}$$

$$b(\cdot,\cdot) : V\times Q \to \mathbb{R}, \text{s.t. } b(\boldsymbol{w},q) = -\int_\Omega \nabla\cdot \boldsymbol{w}\, q, \tag{2.2.4}$$

$$c(\cdot,\cdot,\cdot) : V\times V\times V \to \mathbb{R}, \text{s.t. } c(\boldsymbol{w},\boldsymbol{v},\boldsymbol{z}) = \rho \int_\Omega (\boldsymbol{w}\cdot\nabla)\boldsymbol{v}\cdot\boldsymbol{z}. \tag{2.2.5}$$

Then the weak formulation of the steady problem reads: find $\boldsymbol{u}\in V$ and $p\in Q$ s.t. for any $\boldsymbol{v}\in V, q\in Q$

$$a(\boldsymbol{u},\boldsymbol{v}) + c(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{v},p) + b(\boldsymbol{u},q) - (\boldsymbol{f},\boldsymbol{v}) = 0 \tag{2.2.6}$$

The bilinear form $a(\cdot,\cdot)$ is continuous and coercive in $V\times V$ if $\Gamma_D \neq \emptyset$. In particular, this means that there exists a positive constant C_a s.t.[2]

$$a(\boldsymbol{v},\boldsymbol{v}) \geq C_a \|\boldsymbol{v}\|_V^2, \forall \boldsymbol{v}\in V. \tag{2.2.7}$$

Also, the trilinear form is continuous too, so that there exists a constant C_b such that

$$|c(\boldsymbol{w},\boldsymbol{v},\boldsymbol{u})| \leq C_b \|\boldsymbol{w}\|_V \|\boldsymbol{v}\|_V \|\boldsymbol{u}\|_V. \tag{2.2.8}$$

[2]We notice that this inequality is the standard Poincaré inequality when using the Laplacian formulation of the equations. When using the full tensor, as we do here, the same inequality follows from the Korn inequality.

for all $\boldsymbol{w}$, $\boldsymbol{v}$, $\boldsymbol{u}$ in V. Finally, for a homogenous Dirichlet boundary value problem, if we have divergence free vectors, we get the skew-symmetry of $c(\cdot,\cdot,\cdot)$, i.e.

$$c(\boldsymbol{w},\boldsymbol{v},\boldsymbol{u}) = -c(\boldsymbol{w},\boldsymbol{u},\boldsymbol{v}) \Rightarrow c(\boldsymbol{w},\boldsymbol{v},\boldsymbol{v}) = 0, \quad \forall \boldsymbol{v},\boldsymbol{u} \in H_0^1(\Omega). \tag{2.2.9}$$

In order to compute the solution of the problem (2.2.6) we must linearize the non-linear quadratic term via some type of iterative procedure. Two classical methods for linearization are *Picard iterations* and *Newton iterations*.

Provided one has an initial guess $\boldsymbol{u}_0$ (this may be e.g. the solution of the Stokes problem [102]), at an iteration $k+1$ the Picard iteration is given by: find $\boldsymbol{u}^{k+1} \in V$ and $p^{k+1} \in Q$ s.t. for any $\boldsymbol{v} \in V, q \in Q$ [45]:

$$a(\boldsymbol{u}^{k+1},\boldsymbol{v}) + c(\boldsymbol{u}^{k},\boldsymbol{u}^{k+1},\boldsymbol{v}) + b(\boldsymbol{v},p^{k+1}) + b(\boldsymbol{u}^{k+1},q) - (\boldsymbol{f},\boldsymbol{v}) = 0 \tag{2.2.10}$$

While the Newton iteration is given by: find $\boldsymbol{u}^{k+1} \in V$ and $p^{k+1} \in Q$ s.t. for any $\boldsymbol{v} \in V, q \in Q$ [45]:

$$\begin{aligned} a(\boldsymbol{u}^{k+1},\boldsymbol{v}) + c(\boldsymbol{u}^{k},\boldsymbol{u}^{k+1},\boldsymbol{v}) + c(\boldsymbol{u}^{k+1},\boldsymbol{u}^{k},\boldsymbol{v}) + b(\boldsymbol{v},p^{k+1}) + b(\boldsymbol{u}^{k+1},q) \\ -c(\boldsymbol{u}^{k},\boldsymbol{u}^{k},\boldsymbol{v}) - (\boldsymbol{f},\boldsymbol{v}) = 0 \end{aligned} \tag{2.2.11}$$

Under mild assumptions on $\boldsymbol{f}$ and ν, both (2.2.10) and (2.2.11) are known to converge to the true solution $(\boldsymbol{u},p)$, with Picard converging linearly. That is, for a constant $\chi > 1$ at an iteration $k+1$:

$$\|\boldsymbol{u}^{k+1} - \boldsymbol{u}\|_V + \|p^{k+1} - p\|_{L^2} \leq \frac{1}{\chi}\left(\|\boldsymbol{u}^{k} - \boldsymbol{u}\|_V + \|p^{k} - p\|_{L^2}\right) \tag{2.2.12}$$

While Newton converges quadratically (note χ is not necessarily the same as in

2.2.12):

$$\|\boldsymbol{u}^{k+1} - \boldsymbol{u}\|_V + \|p^{k+1} - p\|_{L^2} \leq \frac{1}{\chi}\left(\|\boldsymbol{u}^k - \boldsymbol{u}\|_V^2 + \|p^k - p\|_{L^2}^2\right) \tag{2.2.13}$$

We will provide a proof for the convergence of the discrete version of the Picard iteration in the Chapter 2.3.1. The convergence of Newton's method is more technical and we direct the reader to [66, 45] for the proof of its convergence.

While Newton converges quadratically in theory, it depends more heavily on the initial guess than Picard, and may fail to converge in the absence of a sufficiently close $\boldsymbol{u}_0$. In practice a close enough initial guess may be unavailable, particularly for higher Re. Picard iterations are therefore often preferred, despite their slower convergence, due to their superior reliability. Hybrid schemes are also used, wherein one begins the iteration with several Picard iterations to get sufficiently close to the desired solution before switching to the Newton scheme.

The Oseen Problem

The *Oseen problem* for a vector field $\boldsymbol{b}$, is closely related to the problem (2.2.1) and is given by:

$$\begin{aligned} -\nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + \rho(\boldsymbol{b}\cdot\nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \mathcal{B}\boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \partial\Omega, \end{aligned} \tag{2.2.14}$$

with corresponding weak formulation:

$$a(\boldsymbol{u},\boldsymbol{v}) + c(\boldsymbol{b},\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{v},p) + b(\boldsymbol{u},q) - (\boldsymbol{f},\boldsymbol{v}) = 0. \tag{2.2.15}$$

The Oseen problem strongly resembles (2.2.1), however the convective field is no longer defined by the velocity vector $\boldsymbol{u}$ but by a given vector field $\boldsymbol{b}$. Assuming $\boldsymbol{b}$ is known and does not depend on $\boldsymbol{u}$ or p, the equations (2.2.14-2.2.15) are linear, in contrast to the nonlinear equations (2.2.1-2.2.6).

We can regard the Picard linearization method (2.2.10) for solving (2.2.1) as solving a sequence of linear Oseen problems where at an iteration $k+1$ we have $\boldsymbol{b} = \boldsymbol{u}^k$. This interpretation of the nonlinear problem is important, as we will see that the efficient solution of each Oseen sub-problem is crucical for the efficient solution of (2.2.1). As much of the literature regarding key aspects of incompressible flow problems explored here (including stabilization and linear solution techniques) is formulated in terms of the Oseen problem [8, 53, 28, 79, 81, 16], we believe that this connection between the linear and nonlinear problems is important to emphasize.

The Stokes Problem

The *Stokes problem* is another problem related to (2.2.1) and is given by:

$$\begin{aligned} -\nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)+\nabla p &= \boldsymbol{f} \quad \text{in } \Omega \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \mathcal{B}\boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \partial\Omega, \end{aligned} \tag{2.2.16}$$

with corresponding weak formulation:

$$a(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{v},p) + b(\boldsymbol{u},q) - (\boldsymbol{f},\boldsymbol{v}) = 0. \tag{2.2.17}$$

The Stokes problem resembles (2.2.1) and (2.2.14), but without a convective term. Physically, these equations are an appropriate model for incompressible Newto-

nian flows in which convective forces are negligible (Re $<<$ 1).

In hemodynamics, these equations are not particularly useful . Flows in certain small vessels, such as capillaries, have suitably low Reynolds numbers, however the non-Newtonian nature of flow in these vessels makes these equations an unsuitable model [40]. Nonetheless, the solution of these equations can often serve as a good initial guess for the iterative methods to solve the steady Navier-Stokes problem discussed here. For this reason, as well their general importance in the study of incompressible flow problems, we believe they are they are worth briefly acknowledging.

Different Weak Formulations

Before closing this section, we note that the weak formulations of (2.2.1) are not unique. In particular, there are several different weak formulations of the convective term that are consistent with (2.2.1). The trilinear form (2.2.5) is one such formulation of the convective term. Several other forms exist, and while they are all consistent with the original equations, their discrete solutions will differ in general [21]. One such alternative formulation that we will make use of later in this work is the *skew-symmetric* formulation:

$$\begin{aligned} c^*(\cdot,\cdot,\cdot) : V \times V \times V \to \mathbb{R}, \text{s.t.} \\ c^*(\boldsymbol{w},\boldsymbol{v},\boldsymbol{z}) = \frac{\rho}{2}\left(\int_\Omega (\boldsymbol{w}\cdot\nabla)\boldsymbol{v}\cdot\boldsymbol{z} - \int_\Omega (\boldsymbol{w}\cdot\nabla)\boldsymbol{z}\cdot\boldsymbol{v}\right). \end{aligned} \tag{2.2.18}$$

Additionally we will also use the diffusive term in *Laplacian form* at various points, which corresponds to the following formulation:

$$a^*(\cdot,\cdot) : V \times V \to \mathbb{R}, \text{s.t. } a^*(\boldsymbol{w},\boldsymbol{v}) = \nu \int_\Omega \nabla\boldsymbol{w} : \nabla\boldsymbol{v} \tag{2.2.19}$$

This is the weak formulation obtained when one replaces the diffusive term in (2.2.1) with $-\nu\Delta\boldsymbol{u}$. The two forms are equivalent in the strong formulation as $\nabla\cdot\boldsymbol{u}$, though in the weak formulation one observes different behavior at the boundaries from the contributions of the term $\nabla\boldsymbol{u}^T$ [46].

2.3 The Discrete Problem

For the numerical solution of (2.2.1) we must discretize the differential operators. In this work we refer to the Finite Element Method (FEM) for the spatial discretization.

With FEM, we postulate the velocity-pressure solution to be piecewise polynomial over a triangulation $\mathcal{T}$ of the domain Ω featuring a representative mesh size h. To guarantee the discrete problem is nonsingular, we assume that the polynomial degrees for the velocity and pressure approximation fulfill the so called the Babushka-Brezzi (or 'inf-sup') condition [102, 45]: There exists a positive constant $\beta > 0$ such that

$$\inf_{q_h\in Q_h}\sup_{\boldsymbol{v}_h\in V_h}\frac{(\nabla\cdot\boldsymbol{v}_h, q_h)}{\|q_h\|\|\nabla\boldsymbol{v}_h\|}\geq\beta>0. \tag{2.3.1}$$

For instance, we may resort to piecewise quadratic velocity and piecewise (continuous) linear pressure fields (Taylor-Hood pair). From now on, we denote by $\boldsymbol{u}_h$ and p_h the discrete velocity and pressure belonging to an inf-sup compatible finite dimensional space pair $V_h\times Q_h\subset V\times Q$. We denote by $N_u(N_p)$ the number of degrees of freedom for the discrete velocity (pressure). The discrete form of the problem (2.2.10) at an iteration $k+1$ is then given by the following *saddle-point*

system:

$$\begin{bmatrix} K + C(\boldsymbol{u}^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{k+1} \\ \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \tag{2.3.2}$$

Where $K + C(\boldsymbol{u}^k)$ is a $N_u \times N_u$ matrix and B is a $N_p \times N_u$ matrix. K is commonly referred to as the *stiffness matrix* and corresponds to the diffusive term in (2.2.1). For $\boldsymbol{\varphi}_i$ being the generic FEM quadratic basis function associated with the $i-$th degree of freedom, $[K]_{ij} = a(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i)$.

Similarly, $C(\boldsymbol{u}^k)$ corresponds to the convective term in (2.2.1) and

$$[C(\boldsymbol{u}^k)]_{ij} = c(\boldsymbol{u}^k, \boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i).$$

Finally, letting ψ_i be a generic basis (piecewise linear) function representing the pressure, B is a $N_p \times N_u$ matrix defined by

$$[B]_{ij} = b(\boldsymbol{\varphi}_j, \psi_i).$$

By standard arguments based on the coercivity of $a(\cdot, \cdot)$, K is readily proved to be symmetric positive-definite (s.p.d.) and $C(\boldsymbol{u}^k)$ to be skew-symmetric for $\Gamma_N = \emptyset$.

The discrete version of (2.2.11) at an iteration $k+1$ is defined similarly and is given by the following saddle-point system:

$$\begin{bmatrix} K + C(\boldsymbol{u}^k) + \hat{C}(\boldsymbol{u}^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{k+1} \\ \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} + C(\boldsymbol{u}^k)\boldsymbol{u}^k \\ \boldsymbol{0} \end{bmatrix}. \tag{2.3.3}$$

with

$$[\hat{C}(\boldsymbol{u})]_{ij} = c(\boldsymbol{\varphi}_j, \boldsymbol{u}, \boldsymbol{\varphi}_i).$$

For the majority of this work, we will focus Picard iterations for simplicity; how-

ever most of what follows also applies for the Newton scheme.

2.3.1 Analysis of the discrete Picard iteration

In this section we analyze the discrete version of (2.2.10). Given that much of the subsequent analysis in this work follows a similar approach, we feel that providing this analysis is instructive. For simplicity, we will assume homogenous Dirichlet boundary conditions; that is, $\boldsymbol{u}_h = \mathbf{0}$ almost everywhere on $\partial\Omega$. The work shown here is based on classical results that can be found in e.g. [66, 116, 70].

Given an LBB compatible velocity-pressure space $V_h \times Q_h \subset V \times Q$, the discrete version of (2.2.6) is obtained in the natural way: find $\boldsymbol{u}_h \in V_h$ and $p_h \in Q_h$ s.t. for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$a^*(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) + b(\boldsymbol{u}_h, q_h) - (\boldsymbol{f}, \boldsymbol{v}_h) = 0. \tag{2.3.4}$$

Note that we use the Laplacian form of a (2.2.19). Since $V_h \times Q_h \subset V \times Q$, the discrete version of the a^* and c forms retain their properties (2.2.7) and (2.2.8), and since we are assuming homogenous Dirichlet boundary conditions, skew-symmetry (2.2.9) holds for c as well. We will assume existence of a solution pair $(\boldsymbol{u}_h, p_h)$ to (2.3.4) and prove conditions for which such a solution is unique.

At an iteration $k+1$ the discrete Picard linearization of (2.3.4) is then given by: find $\boldsymbol{u}_h^{k+1} \in V_h$ and $p_h^{k+1} \in Q_h$ s.t. for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$a^*(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h^{k}, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1}) + b(\boldsymbol{u}_h^{k+1}, q_h) - (\boldsymbol{f}, \boldsymbol{v}_h) = 0. \tag{2.3.5}$$

We derive an estimate before proceeding to our main results. Define the space

$X_h \subset V_h$ of discretely divergence-free functions as:

$$X_h := \{\boldsymbol{v}_h \in V_h,\ b(\boldsymbol{v}_h, q_h) = 0\ \forall q_h \in Q_h\}. \tag{2.3.6}$$

Note that $\boldsymbol{v}_h \in X_h$ does not necessarily imply $\nabla \cdot \boldsymbol{v}_h = 0$ pointwise. This will be addressed in more detail in Chapter 4. Letting $\boldsymbol{v}_h = \boldsymbol{u}_h$ and $q_h = p_h$ in (2.3.4) gives:

$$\nu \|\nabla \boldsymbol{u}_h\|_{L^2}^2 + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{u}_h) + b(\boldsymbol{u}_h, p_h) + b(\boldsymbol{u}_h, p_h) = (\boldsymbol{f}, \boldsymbol{u}_h). \tag{2.3.7}$$

Note $b(\boldsymbol{u}_h, p_h) = 0$ and $c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{u}_h) = 0$ by $\boldsymbol{u}_h \in X_h$ and the skew-symmetry of c respectively, reducing (2.3.7) to:

$$\|\nabla \boldsymbol{u}_h\|_{L^2} = \nu^{-1} \frac{(\boldsymbol{f}, \boldsymbol{u}_h)}{\|\nabla \boldsymbol{u}_h\|_{L^2}} \tag{2.3.8}$$

From the standard Poincare inequality (2.2.7), the norms $\|\nabla \boldsymbol{v}\|_{L^2}$ and $\|\boldsymbol{v}\|_V$ are equivalent [99, 116, 70], hence from (2.2.2):

$$\|\nabla \boldsymbol{u}_h\|_{L^2} \leq \nu^{-1} \|\boldsymbol{f}\|_{-1} \tag{2.3.9}$$

for any solution $\boldsymbol{u}_h$ of (2.3.4). By the inequality (2.2.7) one may also obtain the equivalent condition[3] :

$$\|\boldsymbol{u}_h\|_V \leq \frac{\|\boldsymbol{f}\|_{-1}}{\nu C_a} \tag{2.3.10}$$

We now use this estimate to prove the following uniqueness result:

Lemma 2.3.1 *If $\nu^2 > C_b \|\boldsymbol{f}\|_{-1}$, then the solution pair $(\boldsymbol{u}_h, p_h)$ to the problem (2.3.4) is*

[3]The equivalence of the two conditions (2.3.9) and (2.3.10) implies that all convergence results proven with the Laplacian form (2.2.19) of the diffusive term hold for the full stress tensor formulation (2.2.3).

unique.

Proof. Assume that there exist two solution pairs $(\boldsymbol{u}_h, p_h)$ and $(\widetilde{\boldsymbol{u}}_h, \widetilde{p}_h)$ satisfying (2.3.4). By definition:

$$\begin{aligned} &a^*(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) + b(\boldsymbol{u}_h, q_h) - (\boldsymbol{f}, \boldsymbol{v}_h) \\ &= a^*(\widetilde{\boldsymbol{u}}_h, \boldsymbol{v}_h) + c(\widetilde{\boldsymbol{u}}_h, \widetilde{\boldsymbol{u}}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, \widetilde{p}_h) + b(\widetilde{\boldsymbol{u}}_h, q_h) - (\boldsymbol{f}, \boldsymbol{v}_h) \end{aligned} \tag{2.3.11}$$

Defining $\boldsymbol{e}_u = \boldsymbol{u}_h - \widetilde{\boldsymbol{u}}_h$, adding and subtracting $c(\boldsymbol{u}_h, \widetilde{\boldsymbol{u}}_h, \boldsymbol{v}_h)$ and simplifying gives:

$$\begin{aligned} a^*(\boldsymbol{e}_u, \boldsymbol{v}_h) + b(\boldsymbol{e}_u, q_h) + b(\boldsymbol{v}_h, p_h - \widetilde{p}_h) \\ = \; - c(\boldsymbol{u}_h, \boldsymbol{e}_u, \boldsymbol{v}_h) - c(\boldsymbol{e}_u, \widetilde{\boldsymbol{u}}_h, \boldsymbol{v}_h) \end{aligned} \tag{2.3.12}$$

Letting $(\boldsymbol{v}_h, q_h) = (\boldsymbol{e}_u, p_h - \widetilde{p}_h)$ above gives $b(\boldsymbol{e}_u, p_h - \widetilde{p}_h) = 0$ as $\boldsymbol{e}_u \in X_h$ and $c(\boldsymbol{u}_h, \boldsymbol{e}_u, \boldsymbol{e}_u) = 0$ by skew-symmetry. From (2.2.8) and (2.3.9):

$$\begin{aligned} \nu \|\nabla \boldsymbol{e}_u\|_{L^2}^2 &\leq C_b \|\nabla \boldsymbol{e}_u\|_{L^2}^2 \|\nabla \boldsymbol{u}_h\|_{L^2} \\ &\leq C_b \frac{\|\boldsymbol{f}\|_{-1}}{\nu} \|\nabla \boldsymbol{e}_u\|_{L^2}^2 \\ \left(\nu - \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu}\right) \|\nabla \boldsymbol{e}_u\|_{L^2}^2 &\leq 0 \end{aligned} \tag{2.3.13}$$

If $\nu^2 > C_b \|\boldsymbol{f}\|_{-1}$, then (2.3.13) implies $\boldsymbol{e}_u = \boldsymbol{0}$ and hence $\boldsymbol{u}_h = \widetilde{\boldsymbol{u}}_h$, establishing the uniqueness of the velocity. We note that by (2.2.7) this is equivalent to the condition $\nu^2 C_a^2 > C_b \|\boldsymbol{f}\|_{-1}$. To verify the uniqueness of the pressure, observe that $\boldsymbol{e}_u = \boldsymbol{0}$ implies from (2.3.12):

$$b(\boldsymbol{v}_h, p_h - \widetilde{p}_h) = 0 \tag{2.3.14}$$

for all $\boldsymbol{v}_h$ in V_h. By the inf-sup condition (2.3.1), this gives $p_h = \widetilde{p}_h$, completing the proof.

Define the data-dependent constants:

$$\eta := \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu^2}, \tag{2.3.15}$$

$$\chi := \frac{\nu^2 C_a^2}{C_b \|\boldsymbol{f}\|_{-1}}. \tag{2.3.16}$$

From the lemma, we know that if the *small-data hypothesis*

$$\eta < 1 \text{ or equivalently,} \tag{2.3.17}$$

$$\chi > 1 \tag{2.3.18}$$

holds, then the solution pair $(\boldsymbol{u}_h, p_h)$ of the problem defined by (2.3.4) is unique. We will use both η and χ and their respective (equivalent) small-data hypotheses (2.3.17) and (2.3.18) throughout this work, depending on which is more convenient to use for the current problem. We now proceed to the main result:

Theorem 2.3.1 *Assume (2.3.17) holds and let $(\boldsymbol{u}_h, p_h)$ be the resulting unique solution pair to (2.3.4). Then the sequence given by (2.3.5) converges to $(\boldsymbol{u}_h, p_h)$.*

Proof. We proceed by induction. Denote $\boldsymbol{e}^k = \boldsymbol{u}_h^k - \boldsymbol{u}_h$ as the error in the velocity field between the solution at an iteration k of (2.3.5) and the solution of (2.3.4).

We assume that:

$$\|\nabla \boldsymbol{e}^k\|_{L^2} \leq \eta^k \|\nabla \boldsymbol{e}^0\|_{L^2}, \tag{2.3.19}$$

and seek to show that

$$\|\nabla \boldsymbol{e}^{k+1}\|_{L^2} \leq \eta \|\nabla \boldsymbol{e}^k\|_{L^2} \leq \eta^{k+1} \|\nabla \boldsymbol{e}^0\|_{L^2}. \tag{2.3.20}$$

By the assumption (2.3.17), this will imply $\|\nabla \boldsymbol{e}^{k+1}\|_{L^2} \to 0$ as $k \to \infty$. Subtracting

(2.3.4) from (2.3.5) gives:

$$\begin{aligned} a^*(\boldsymbol{e}^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}, p_h^k - p_h) + b(\boldsymbol{e}^{k+1}, q_h) \\ = -c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) \end{aligned} \tag{2.3.21}$$

Since $\boldsymbol{e}^{k+1} \in X_h$, $b(\boldsymbol{e}^{k+1}, q_h) = 0$. Adding and subtracting $c(\boldsymbol{u}_h^k, \boldsymbol{u}_h, \boldsymbol{v}_h)$ from the right-hand side:

$$a^*(\boldsymbol{e}^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}, p_h^k - p_h) = -c(\boldsymbol{u}_h^k, \boldsymbol{e}^{k+1}, \boldsymbol{v}_h) - c(\boldsymbol{e}^k, \boldsymbol{u}_h, \boldsymbol{v}_h). \tag{2.3.22}$$

Letting $\boldsymbol{v}_h = \boldsymbol{e}^{k+1}$ implies $b(\boldsymbol{e}^{k+1}, p_h^k - p_h) = 0$ by $\boldsymbol{e}^{k+1} \in X_h$ and $c(\boldsymbol{u}_h^k, \boldsymbol{e}^{k+1}, \boldsymbol{e}^{k+1}) = 0$ by skew-symmetry. From (2.2.8):

$$\begin{aligned} \nu \|\nabla \boldsymbol{e}^{k+1}\|_{L^2}^2 &= -c(\boldsymbol{e}^k, \boldsymbol{u}_h, \boldsymbol{e}^{k+1}) \\ &\leq C_b \|\nabla \boldsymbol{e}^k\|_{L^2} \|\nabla \boldsymbol{u}_h\|_{L^2} \|\nabla \boldsymbol{e}^{k+1}\|_{L^2}. \end{aligned} \tag{2.3.23}$$

Owing to (2.3.9) and (2.3.17):

$$\begin{aligned} \|\nabla \boldsymbol{e}^{k+1}\|_{L^2} &\leq C_b \nu^{-1} \|\nabla \boldsymbol{e}^k\|_{L^2} \|\nabla \boldsymbol{u}_h\|_{L^2} \\ &\leq \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu^2} \|\nabla \boldsymbol{e}^k\|_{L^2} \\ &\leq \eta \|\nabla \boldsymbol{e}^k\|_{L^2}, \end{aligned} \tag{2.3.24}$$

which was to be shown.

To complete the proof, all that remains is to verify that the condition (2.3.20) holds for the base case: $\|\nabla \boldsymbol{e}^1\|_{L^2} \leq \eta \|\nabla \boldsymbol{e}^0\|_{L^2}$. Let $\boldsymbol{u}_h^0 = \boldsymbol{0}$. Then $\boldsymbol{e}^0 = \boldsymbol{u}_h$ trivially and $\boldsymbol{u}_h^1$ is given by the solution to the problem: Find $\boldsymbol{u}_h^1$ in V_h and p_h^1 in Q_h s.t. for all $\boldsymbol{v}_h$ in V_h, q_h in Q_h:

$$a^*(\boldsymbol{u}_h^1, \boldsymbol{v}_h) + c(\boldsymbol{0}, \boldsymbol{u}_h^1, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^1) + b(\boldsymbol{u}_h^1, q_h) = (\boldsymbol{f}, \boldsymbol{v}_h) \tag{2.3.25}$$

The term $c(\mathbf{0}, \boldsymbol{u}_h^1, \boldsymbol{v}_h) = 0$ trivially, reducing (2.3.25) to the Stokes problem (2.2.16)-(2.2.17) with the diffusive term in Laplacian form.

Subtracting (2.3.4) from (2.3.25) gives:

$$a^*(\boldsymbol{e}^1, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^1 - p_h) + b(\boldsymbol{e}^1, q_h) = c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) \tag{2.3.26}$$

Letting $(\boldsymbol{v}_h, q_h) = (\boldsymbol{e}^1, p_h^1 - p_h)$ above gives $b(\boldsymbol{e}_1, p_h^1 - p_h) = 0$. Recalling that $\boldsymbol{e}^0 = \boldsymbol{u}_h$ and applying (2.3.9) and (2.2.8):

$$\begin{aligned} \nu \|\nabla \boldsymbol{e}^1\|_{L^2}^2 &= c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{e}^1) \\ &= c(\boldsymbol{u}_h, \boldsymbol{e}^0, \boldsymbol{e}^1) \\ &\leq C_b \|\nabla \boldsymbol{u}_h\|_{L^2} \|\nabla \boldsymbol{e}^0\|_{L^2} \|\nabla \boldsymbol{e}^1\|_{L^2} \\ &\leq \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu} \|\nabla \boldsymbol{e}^0\|_{L^2} \|\nabla \boldsymbol{e}^1\|_{L^2} \end{aligned} \tag{2.3.27}$$

Dividing the left and right hand sides by $\nu \|\nabla \boldsymbol{e}^1\|_{L^2}$, from (2.3.15):

$$\begin{aligned} \|\nabla \boldsymbol{e}^1\|_{L^2} &\leq \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu^2} \|\nabla \boldsymbol{e}^0\|_{L^2} \\ &\leq \eta \|\nabla \boldsymbol{e}^0\|_{L^2} \end{aligned} \tag{2.3.28}$$

verifying that the condition holds for $k = 1$, completing the proof. Note that the above result also justifies the use of the solution to the Stokes problem (2.2.17) as an initial guess for the Picard iteration.

Chapter 3

Algebraic Splitting Methods for the Steady Problem

[1] In this chapter we discuss methods to solve the discrete problem by segregating the velocity and pressure variables and solving for them separately. This is often desirable as solving a full saddle-point system of the form (2.3.2) monolithically can be difficult due to both its large size and its conditioning properties. Much work has been done on developing effective preconditioners for this *discrete Oseen problem*, the class of problem solved at each step of a Picard iteration, (see e.g. [53, 8, 37, 123]), however their application to the discrete nonlinear system still presents challenges as the linearization changes at each iteration and therefore the preconditioner must be updated accordingly.

Although the saddle-point system for the unsteady problem also changes at each time step (unless one uses a fully explicit discretization of the nonlinear term, though this approach suffers from time instabilities in practical applications), the presence of the time derivative introduces a *mass matrix* $\frac{1}{\Delta t}M$. For small Δt, this

[1] This chapter is a modified version of the article: A. Viguerie and A. Veneziani. "Algebraic splitting methods for the steady incompressible Navier-Stokes equations at moderate Reynolds numbers." *Computer Methods in Applied Mechanics and Engineering*, 330:271-291, 2018. [122]

term is generally dominant and enables one to effectively apply preconditioners that do not require updating, such as the Cahouet-Chabard preconditioner [37].

The presence of the mass term also enables one to easily decouple velocity and pressure and solve a series of smaller, simpler problems rather than solving the monolithic system. These schemes may be based on functional analysis arguments [17] or inexact algebraic factorizations [124, 101, 100]. Though such schemes are popular and effective for the unsteady problem, their construction relies explicitly on the presence of the time derivative and as a result the extension of such ideas to the steady problem is not immediate.

Because of the relative ease of solving the linear systems in the unsteady problem (and the difficulty of their solution for the steady problem), a common approach is to regard the steady problem as the asymptotic limit of an unsteady problem and simulate the problem as a transient problem over a long time interval. While this allows one to use the aforementioned sorts of efficient solution schemes, it may require the solution of hundreds or thousands of time steps, and the advantages gained by using a steady-state model to reduce simulation time is quickly lost.

In this section we extend the class of inexact algebraic factorization methods popular for the unsteady problem [124, 101, 100] and develop analogous approaches for the steady problem while maintaining many of the desirable properties of these methods. We will first briefly discuss the unsteady versions of these schemes and then introduce our analogous approaches for the steady problem. We will present several results relevant to our new approach and validate its efficiency with test cases in two and three dimensions.

3.1 Algebraic Splitting Methods

At their core, *algebraic splitting* methods are based on *inexact LU factorizations* of the discrete saddle-point problem (2.3.2). For the sake of notation, let A be the (1,1) block in the saddle point system (often called the *velocity block* for obvious reasons) where the dependence on k is understood. Then the saddle-point system (2.3.2) admits the following block LU-factorization (see e.g. [9, 91]):

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{(k+1)} \\ \boldsymbol{p}^{(k+1)} \end{bmatrix} = \begin{bmatrix} A & 0 \\ B & -BA^{-1}B^T \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{(k+1)} \\ \boldsymbol{p}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \tag{3.1.1}$$

Although this block LU factorization formally yields a segregated computation of velocity and pressure, it is not practical. The system for the pressure requires the solution of the so called (pressure) *Schur complement* $\Sigma = -BA^{-1}B^T$. This matrix changes at each iteration and cannot be computed, as A^{-1} is neither available nor worth computing, for both computational and storage costs induced by the fill-in of the inversion.

A possible approach is to solve the Schur complement by an iterative method so that when the matrix needs to be multiplied by a vector, we instead solve a system in A. We recall that $A = K + C$ (the dependence of C on k is understood here) which in general is neither symmetric nor positive-definite due to the skew-symmetry of C. Therefore, solving systems in both A and Σ is difficult in general due to their lack of favorable numerical properties. This means that the nested iterations obtained in this way (as A is usually solved by an iterative method too) rapidly increase the computational costs and the advantage of segregation is easily lost for problems where the contributions of C are significant.

As previously mentioned, there exist effective workarounds to this problem in the case of the unsteady problem. The idea is that we may approximate the LU

factorization (3.1.1) with a system that is easier to solve while still converging to our desired solution.

We now briefly consider the discrete version of the problem (2.1.1), which is similar to (2.3.2). We discretize with the finite element method in space, and use a finite difference scheme for the time discretization. For simplicity, we use a simple first-order semi-implicit Euler scheme here, so at each time step n we solve the system:

$$\begin{bmatrix} A_t & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{p}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} + \frac{1}{\Delta t} M \boldsymbol{u}^n \\ \mathbf{0} \end{bmatrix} \tag{3.1.2}$$

with $A_t(\boldsymbol{u}^n) = \frac{1}{\Delta t} M + A^n$ where Δt is the time step, $A^n \equiv K + C(\boldsymbol{u}^n)$ is as before and M (called the *mass matrix*) comes from the bilinear form $m(\boldsymbol{u}, \boldsymbol{v}) : V \times V \to \mathbb{R}$:

$$m(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v}. \tag{3.1.3}$$

If I denotes the identity matrix, observe that:

$$\frac{1}{\Delta t} M + A^n = \frac{1}{\Delta t} M (I + \Delta t M^{-1} A^n) \Rightarrow (\frac{1}{\Delta t} M + A^n)^{-1} = \Delta t (I + \Delta t M^{-1} A^n)^{-1} M^{-1} \tag{3.1.4}$$

For Δt sufficiently small, we can ensure that $\rho(\Delta t M^{-1} A_1^n) < 1$ and exploit the well-known Neumann series identity for $(I + \Delta t M^{-1} A_1^n)^{-1}$ to observe that

$$\left(\frac{1}{\Delta t} M + A^n \right)^{-1} = \Big(\sum_{j=0}^{\infty} (-1)^j (\Delta t M^{-1} A^n)^j \Big) \Delta t M^{-1} \approx \Delta t M^{-1} \tag{3.1.5}$$

with a first order truncation of the Neumann expansion. Hence we can regard $\Delta t M^{-1}$ as a first-order approximation for A_t^{-1}. With this approximation in mind we then consider a block factorization of (3.1.2) similar to (3.1.1) defined as:

$$\begin{bmatrix} A_t & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} A_t & 0 \\ B & -BH_1B^T \end{bmatrix} \begin{bmatrix} I & H_2B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{p}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} + \frac{1}{\Delta t} M \boldsymbol{u}^n \\ \mathbf{0} \end{bmatrix} \tag{3.1.6}$$

For $H_1 = H_2 = A_t^{-1}$, we get the exact block factorization of the original system (3.1.2). The *Inexact Factorization methods* seek to replace H_1 and H_2 with approximations of A_t^{-1}, a natural choice being $\Delta t M^{-1}$. Replacing both H_1 and H_2 with $\Delta t M^{-1}$ gives the so called *Algebraic Chorin-Temam* method, as it was pointed out that there is a formal analogy with the original Chorin-Temam projection scheme based on the Ladyzhenskaya decomposition theorem [17, 117, 104, 96, 95, 91]. Using $\Delta t M^{-1}$ for H_1 and A_t^{-1} for H_2 is another viable option leading to the so-called *Yosida* method. These methods are discussed and analyzed in detail in [91, 101, 100]. In analogue with the original unsteady schemes, we will refer to inexact factorizations of the steady problem in which $H_1 = H_2$ as *Algebraic Chorin-Temam-type methods* and where $H_1 \neq H_2$ as *Yosida-type methods*, even if the matrix approximations used no longer correspond formally to the original projection schemes.

The common denominator of this approach is the approximation of the Schur complement having the velocity mass matrix as a critical ingredient, with the parameter Δt modulating the accuracy of the approximation. Variants to the original scheme have been eventually introduced to guarantee higher order of time accuracy and to trigger time adaptivity [43, 108, 44, 123, 124, 125]. It is worth noting that the higher accuracy is not obtained by including more terms in the Neumann expansion, as that would cause some stability and practical problems

[30]. In fact, the approximation of the Schur complement with A_t^{-1} replaced by the inverse of the mass matrix is particularly convenient, as the mass matrix is not only s.p.d., but also diagonal for finite difference, spectral and lumped finite elements. This allows for specific ad hoc solvers based on the sparse QR factorization of Σ [125].

The absence of the mass matrix in the stationary problem prevents an immediate extension of the approach to the steady case. Thus, the goal here is to find a way that accelerates the computation of the steady solution by splitting velocity and pressure with a convenient approximation of the Schur complement. Following the unsteady approach, for the block-factorized system (3.1.1), we approximate the Schur complement with the following arguments.

We first replace the Picard iteration with the following variant. Let $\alpha \in [0,1]$ be a positive parameter . The *modified Picard scheme reads*: given the guess $(\boldsymbol{u}_h{}^{(k)}, p_h{}^{(k)}) \in V_h \times Q_h$, find $(\boldsymbol{u}_h{}^{(k)}, p_h{}^{(k)})$ in the same space pair s.t.

$$\begin{aligned} &a(\boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{v}_h) + \alpha c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{v}_h) + b(p_h{}^{(k+1)}, \boldsymbol{v}_h) \\ &= (\boldsymbol{f}, \boldsymbol{v}_h) - (1-\alpha) c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k)}, \boldsymbol{v}_h) \\ &b(q_h, \boldsymbol{u}_h{}^{(k+1)}) = 0 \end{aligned} \tag{3.1.7}$$

for any $\boldsymbol{v}_h \in V_h$ and $q_h \in Q_h$.

At the algebraic level, for $A_\alpha \equiv K + \alpha C(\boldsymbol{u}^{(k)})$ (we do not emphasize the dependence on k for easiness of notation), the modified Picard iteration reads

$$\begin{bmatrix} A_\alpha & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{k+1} \\ \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} - (1-\alpha) C(\boldsymbol{u}^{(k)}) \boldsymbol{u}^{(k)} \\ \boldsymbol{0} \end{bmatrix} \tag{3.1.8}$$

For $\alpha = 1$ we clearly have the standard Picard scheme, while for $\alpha = 0$ we have a full explicit treatment of the nonlinearity. While the convergence rate is linear

for the original scheme[66], when α approaches 0, we expect the pool of possible initial good guesses for the convergence to shrink. This will be confirmed in the analysis below.

Notice that[2]

$$A_\alpha = K(I + \alpha K^{-1}C) \tag{3.1.9}$$

By a similar argument as seen for the unsteady case (with Δt), if α is chosen such that:

$$\rho(\alpha K^{-1}C) < 1, \tag{3.1.10}$$

then we have:

$$(K + \alpha C)^{-1} = \left(K(I + \alpha K^{-1}C)\right)^{-1} = \left(\sum_{j=0}^{\infty}(-1)^j(\alpha K^{-1}C)^j\right)K^{-1} \tag{3.1.11}$$

justifying the approximation:

$$(K + \alpha C)^{-1} \approx K^{-1} \tag{3.1.12}$$

as a first order truncation of the Neumann expansion. With the LU factorization approximated in this way, we propose the following algorithm for the solution of the single iteration of (3.1.8):

$$\begin{aligned} A_\alpha \boldsymbol{u}^* &= \boldsymbol{f} - (1-\alpha)C\boldsymbol{u}^{(k)} \\ BK^{-1}B^T p^{(k+1)} &= B\boldsymbol{u}^* \\ K\boldsymbol{u}^{(k+1)} &= K\boldsymbol{u}^* - B^T p^{(k+1)} = \boldsymbol{f} - (1-\alpha)C\boldsymbol{u}^{(k)} - \alpha C\boldsymbol{u}^* - B^T p^{(k+1)} \end{aligned} \tag{3.1.13}$$

[2]The dependence of C on the iterate $\boldsymbol{u}^{(k)}$ is understood.

Since K is not dependent on the current iteration, we do not need to reassemble any associated preconditioner for the approximated Schur complement at each step. Additionally, our approximated Schur complement is guaranteed to be s.p.d., allowing for the use of efficient iterative methods.

The sequence of systems (3.1.13) provides the backbone of our method. It corresponds to what we previously called *Algebraic Chorin-Temam* scheme in the unsteady case, as we use the same approximation of A^{-1} in both its occurrences of the factorization. However, a substantial difference must be noted between the steady and unsteady schemes. In the (algebraic) Chorin-Temam method, the final operator recovering the end-of-step velocity is a projection. In the algebraic scheme, this corresponds to the velocity mass matrix. A drawback of this is that there is virtually no control on the tangential boundary conditions for the end-of-step velocity. For this reason, some authors preferred the intermediate velocity to be qualified as the physical field, despite it not being divergence free [50, 49]. On the contrary, in the Yosida approach, for $H_2 = A_1^{-1}$, the end-of-step velocity is solved by a second order operator and the control of the end-of-step velocity boundary conditions is retained.

In the steady case, also following the strategy *á la* Chorin-Temam, the final step is obtained by solving a second order (diffusion) operator, so the enforcement of the boundary conditions to the end-of-step velocity is fulfilled. Because of this, we find that the Yosida-like strategy in the steady case is not justified and we do not pursue it any longer.

If $\mathcal{A}_\alpha$ denotes the matrix of the modified Picard scheme

$$\mathcal{A}_{\alpha,u} \equiv \begin{bmatrix} A_\alpha & B^T \\ B & 0 \end{bmatrix},$$

the matrix resulting from the splitting scheme reads

$$\mathcal{A}_{\alpha,s} \equiv \begin{bmatrix} A_\alpha & 0 \\ B & -BK^{-1}B^T \end{bmatrix} \begin{bmatrix} I & K^{-1}B^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} A_\alpha & A_\alpha K^{-1}B^T \\ B & 0 \end{bmatrix}$$
$$= \mathcal{A}_{\alpha,u} + \begin{bmatrix} 0 & \boxed{\alpha C K^{-1}B^T} \\ 0 & 0 \end{bmatrix}.$$

As the boxed term points out, the splitting error affects only the block (1,2) of the original system. This means that the approximated end-of-step velocity is truly (discretely) divergence free. In addition, the error term scales with α so, as expected, vanishes for $\alpha \to 0$.

To enhance the convergence rate and the robustness of the method we introduce below two variants that eventually we combine in the final formulation of the scheme.

Remark

At the continuous level, the first step of our splitting scheme reads: given $\boldsymbol{u}^{(k)}$

$$\begin{aligned} &-\nabla \cdot \left(\nu \nabla \hat{\boldsymbol{u}}^{(k+1)} + \nabla^T \hat{\boldsymbol{u}}^{(k+1)}\right) + \alpha \left(\boldsymbol{u}^{(k)} \cdot \nabla\right) \hat{\boldsymbol{u}}^{(k+1)} \\ &\quad = \boldsymbol{f} + (\alpha - 1) \left(\boldsymbol{u}^{(k)} \cdot\right) \nabla \boldsymbol{u}^{(k)} \end{aligned} \tag{3.1.14}$$

This formulation sheds light on the role of α and consequently its selection. The Péclet number of this problem reads

$$\mathbb{P} = \frac{\alpha ||\boldsymbol{u}^{(k)}||_\infty h}{2\nu}$$

so we argue that the numerical approximation with standard finite elements is convective stable when

$$\alpha h < \frac{2\nu}{||\boldsymbol{u}^{(k)}||_\infty}. \tag{3.1.15}$$

Since α is selected by the user, this inequality leads to two conclusions:

1. α has a stabilizing effect, so that it can mitigate the need of a fine mesh required by the original Picard scheme;

2. we may use this inequality to develop an adaptive strategy for the automatic selection of this α, that is then assumed to depend on k. We test and validate this approach later in this work.

Remark

A similar approach can be applied to the Newton method. Weighting the linearized term of the Newton iteration by the parameter α and evaluating the residual term weighted by $1-\alpha$ at the previous iteration, the modified Newton scheme reads given the guess $(\boldsymbol{u}_h{}^{(k)}, p_h{}^{(k)}) \in V_h \times Q_h$, find $(\boldsymbol{u}_h{}^{(k)}, p_h{}^{(k)})$ in the same space pair s.t.

$$\begin{aligned} & a(\boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{v}_h) + \alpha \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{u}_h{}^{(k)}, \boldsymbol{v}_h) \right) + b(p_h{}^{(k+1)}, \boldsymbol{v}_h) \\ & \quad = (\boldsymbol{f}, \boldsymbol{v}_h) + (2\alpha - 1) c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k)}, \boldsymbol{v}_h) \\ & \qquad b(q_h, \boldsymbol{u}_h{}^{(k+1)}) = 0 \end{aligned} \tag{3.1.16}$$

for any $\boldsymbol{v}_h \in V_h$ and $q_h \in Q_h$.

This is consistent with the original problem for any value of α (as promptly realized by setting $\boldsymbol{u}_h{}^{(k+1)} = \boldsymbol{u}_h{}^{(k)}$).

As the Newton method requires an initial guess good enough, we expect the choice of α - that affects the pool of good guesses, as we will see later - to be even more delicate with this approach. On the other hand, the Newton method is faster and the actual effect of this improved convergence rate on the splitting algorithm will be investigated elsewhere.

3.1.1 The incremental variant

By mimicking a popular approach for the unsteady problem [108, 55], we consider the following *incremental pressure* form of the problem, letting $\delta \boldsymbol{p}^{k+1} = \boldsymbol{p}^{k+1} - \boldsymbol{p}^{k}$, given $\boldsymbol{u}^{(k)}$ and $p^{(k)}$, solve

$$\begin{bmatrix} K + \alpha C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{k+1} \\ \delta \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} - (1-\alpha) C \boldsymbol{u}^{k} - B^T \boldsymbol{p}^{k} \\ \boldsymbol{0} \end{bmatrix} \tag{3.1.17}$$

The relevance of the incremental approach is promptly realized by noting that the consistency error affects only the block (1,2) of the matrix of the system. Should the method be convergent, the splitting error of the nonincremental scheme will gather on the momentum equation as $\alpha C K^{-1} B^T p$ where p is the convergence pressure. This vanishes for $\alpha \to 0$. In the case of the incremental scheme, the solution $\delta p \to 0$ when approaching convergence. This means that the convergence solution solves the exact problem for all the values of α. The incremental scheme is therefore "strongly consistent", as the convergence solution is exact (up to numerical discretization errors) regardless the value of the parameter.

3.1.2 Relaxation

The parameter α is introduced to modulate the weight of the convective component and to make the Schur complement approximation acceptable. While for $\alpha = 0$ the splitting error is null, the original Picard convergent (and robust) iteration is obtained for $\alpha = 1$. The proper selection of this parameter is therefore critical for the overall performance of the method. To add some flexibility, we introduce a second parameter for the purpose of enhancing the convergence. Denoting by $\hat{\boldsymbol{u}}^{k+1}$ the velocity field obtained by the basic unrelaxed (either incre-

mental or not) scheme, the relaxed scheme computes

$$\boldsymbol{u}^{k+1} = \gamma \hat{\boldsymbol{u}}^{k+1} + (1-\gamma)\boldsymbol{u}^{k}$$

with $\gamma \in (0, 1]$. The actual bound on the relaxation parameter γ that guarantees convergence will be discussed in the next section.

The final splitting incremental relaxed scheme then reads as in Algorithm 1.

Algorithm 1 Incremental relaxed segregated scheme for the steady Navier-Stokes equations.

1: **while** Convergence criterion not met: **do**
2: solve $A_\alpha \hat{\boldsymbol{u}}^{k+1} = \boldsymbol{f} + (\alpha - 1)C\boldsymbol{u}^k - B^T \boldsymbol{p}^k$
3: solve $BK^{-1}B^T \delta \boldsymbol{p}^{k+1} = B\hat{\boldsymbol{u}}^{k+1}$
4: solve $K\hat{\boldsymbol{u}}^{k+1} = K\hat{\boldsymbol{u}}^{k+1} - B^T \delta \boldsymbol{p}^{k+1}$
5: update the pressure: $\boldsymbol{p}^{k+1} = \delta \boldsymbol{p}^{k+1} + \boldsymbol{p}^k$
6: relax the velocity: $\boldsymbol{u}^{k+1} = \gamma \hat{\boldsymbol{u}}^{k+1} + (1-\gamma)\boldsymbol{u}^k$
7: check tolerance
8: **end while**

Remark

The expected bottleneck in the algorithm is represented by the approximated Schur complement $BK^{-1}B^T$. For K to be the pure diffusion operator (i.e. for $\nabla \cdot (\nabla + \nabla^T)$ reduced to the Laplace operator), we may notice that this matrix is spectrally equivalent to a pressure mass matrix [102]. In principle, we could replace this matrix with a lumped pressure mass matrix so as to minimize the cost of the solution of the system. Nevertheless, in our numerical experiments (not shown) we found that this choice is overall detrimental for the performance of the entire splitting scheme. While we will further investigate this option and its smart implementation, we notice however that the pressure mass matrix is a natural and excellent preconditioner for the approximate Schur complement. It will be used in the entire section on Numerical Results.

3.1.3 The Yosida-like variant

In principle, following [101], we can think of a variant to the splitting scheme where the third step is not approximated:

Algorithm 2 Steady Yosida Method.

1: **while** Convergence criterion not met: **do**
2: solve $A_\alpha \hat{\hat{\boldsymbol{u}}}^{k+1} = \boldsymbol{f} + (\alpha - 1)C\boldsymbol{u}^k - B^T\boldsymbol{p}^k$
3: solve $BK^{-1}B^T\delta\boldsymbol{p}^{k+1} = B\hat{\hat{\boldsymbol{u}}}^{k+1}$
4: solve $A_\alpha \hat{\boldsymbol{u}}^{k+1} = \boldsymbol{f} + (\alpha - 1)C\boldsymbol{u}^k - B^T\boldsymbol{p}^k - B^T\delta\boldsymbol{p}^{k+1}$
5: update the pressure: $\boldsymbol{p}^{k+1} = \delta\boldsymbol{p}^{k+1} + \boldsymbol{p}^k$
6: relax the velocity: $\boldsymbol{u}^{k+1} = \gamma\hat{\boldsymbol{u}}^{k+1} + (1-\gamma)\boldsymbol{u}^k$
7: check tolerance
8: **end while**

In this case, the splitting error is gathered in the mass conservation equation as opposed to the momentum equation as in our proposed scheme. In the unsteady splitting method, this leads to the Yosida scheme. As pointed out previously, the Yosida approach has the advantage of enforcing the entire boundary conditions to the end-of-step velocity. In the present case, our method also enforces the boundary conditions, and is therefore more justified. In particular, the third step of our method requires the solution of a s.p.d. system, which can be solved more efficiently than the corresponding system in Algorithm 2. For this reason we do not analyze in detail this variant.

3.1.4 Algebraic Splitting-Inspired Preconditioner

As mentioned previously, the use of algebraic splitting schemes is widespread for unsteady problems. In addition to their direct application, such schemes have been used to develop block preconditioners for the monolithic unsteady Navier-Stokes problem [123, 43, 33]. Similarly, we may extend the ideas of the previous section to develop block preconditioners for the modified α-Picard scheme (3.1.7):

$$P^{-1} = \left(\begin{bmatrix} \tilde{A}_\alpha & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & \tilde{K}^{-1} B^T \\ 0 & I \end{bmatrix} \right)^{-1} \tag{3.1.18}$$

Alternatively, we may instead define a block-upper triangular preconditioner:

$$P^{-1} = \begin{bmatrix} \tilde{A}_\alpha & B^T \\ 0 & S \end{bmatrix}^{-1} \tag{3.1.19}$$

where S is an approximation of $-BA_\alpha^{-1}B^T$ and $\tilde{A}$, $\tilde{K}$ are approximations of A_α and K, though we may also use the original matrices. Such block-triangular or block-LU preconditioners are common for this problem [37, 123, 53]. This may be an effective approach for a few reasons. It is well-known that for high Reynolds numbers, the approximation of the Schur complement becomes more difficult. The pressure mass-matrix M_p, simple to construct and apply, provides a good approximation for the Stokes problem and for low Reynolds numbers, but loses effectiveness as the Reynolds number increases. Common approximations used in these situations include the least-squares commuter and pressure-convection diffusion (PCD) preconditioners. However, while these preconditioners are effective, they require reassembly at each iteration and occasional difficulties can arise from the need to assign boundary conditions for the second-order operators used to construct the matrices [37]. In contrast, for sufficiently small α in (3.1.7), the pressure mass-matrix again becomes a reasonable approximation of the Schur complement. We could also in principle approximate S by solving a system in $BK^{-1}B^T$ (as we do in the splitting scheme), however this is unlikely to improve efficiency.

Another possible advantage of this approach is that, as previously shown, the

modified Picard scheme provides convective stabilization without requiring the introduction of new terms. Further, for large scale problems with higher Reynolds numbers, solving the A block, either exactly or approximately, in a block triangular (or block diagonal) preconditioner such as those shown in [37] becomes quite challenging. The modified velocity block $A_\alpha = K + \alpha C$ is comparatively dominated by the SPD diffusive matrix K, making its preconditioning and solution much easier. For instance, for small enough α one would expect Laplacian preconditioners to be effective. In fact, for small enough α one may also justify the use of $\tilde{A}_\alpha = \tilde{K}$. This preconditioner has low assembly costs and is easy to implement; however the explicit term in the underlying scheme still necessitates the use of under-relaxation.

In general we do not expect this approach to be as efficient as the splitting scheme, as the pressure mass matrix is an optimal preconditioner for the approximate Schur complement, ensuring a low number of outer iterations independently of h. Nonetheless, the above scheme provides the advantage of reduced splitting error and being easily adaptable to existing codes which use preconditioners of this type. Further, in applications where solving the velocity block and inner Schur Complement block exactly, as required in the splitting schemes, is not possible, one may implement the above preconditioners with easier-to-solve approximations of A_α (or K). Compared to similar block-triangular preconditioners, it is easy to implement and has a lower assembly cost, with the Schur complement approximation not requiring any updating. Further investigation of preconditioners of this type could be the subject of future work.

3.2 Analysis of the modified Picard scheme

In this section, we provide a complete convergence analysis of the modified Picard scheme. The proof follows a similar approach to the one shown in Section 2.3.1. As in that section, we assume for simplicity that the boundary conditions are of (homogeneous) Dirichlet type over the entire boundary $\partial\Omega$, so that $V = H_0^1(\Omega)$ and $Q = L^2 \setminus \mathbb{R}$. As a consequence, as mentioned earlier the trilinear form is skew-symmetric, i.e.

$$c(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{u}) = 0$$

for any $\boldsymbol{v} \in V$ and divergence free velocity $\boldsymbol{u} \in V$.

Recall the bound (2.3.10):

$$\|\boldsymbol{u}_h\|_V \leq \frac{\|\boldsymbol{f}\|_{-1}}{\nu C_a} \tag{3.2.1}$$

where $\|\cdot\|_{-1}$ denotes the norm of the dual space of $H_0^1(\Omega)$. We assume that the small-data hypothesis (2.3.18) holds:

$$\chi = \frac{\nu^2 C_a^2}{C_b \|\boldsymbol{f}\|_{-1}} > 1. \tag{3.2.2}$$

which guarantees a unique solution to the problem from Lemma 2.3.1. As we are concerned with moderate Reynolds numbers, we assume additionally that:

$$\chi > 2, \tag{3.2.3}$$

which is helpful for our analysis. This assumption is reasonable, as a small-to-moderate Reynolds number implies that ν is relatively large in comparison to the data $\|\boldsymbol{f}\|_{-1}$.

Theorem 3.2.1 *Let $\boldsymbol{u}_h^{(0)} \times p_h^{(0)}$ be an initial guess for the modified Picard scheme, be-*

longing to $V_h \times Q_h$. Under the assumptions (3.2.2), (3.2.3), *for $\alpha \in [0,1]$, and $k > 0$ the sequence $\boldsymbol{u}_h^{(k)} \times p_h^{(k)}$ computed by the modified Picard scheme converges to the (unique) solution of the steady Navier-Stokes problem if the initial guess is close enough to the exact solution. For $\alpha \to 1$, the assumption* (3.2.3) *can be relaxed to* (3.2.2).

Proof

Let[3] $\boldsymbol{e}^{(k)} \equiv \boldsymbol{u}_h{}^{(k)} - \boldsymbol{u}_h$ and $\varepsilon^{(k)} \equiv p_h{}^{(k)} - p_h$ be the errors associated with the modified Picard scheme at iteration k.

By direct subtraction of the (weak) Navier-Stokes problem from the modified Picard iteration we obtain the error equation, for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$\begin{aligned} & a(\boldsymbol{e}^{(k+1)}, \boldsymbol{v}_h) + b(\varepsilon^{(k+1)}, \boldsymbol{v}_h) - b(q_h, \boldsymbol{e}^{(k+1)}) \\ & \quad = -\alpha \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{v}_h) - c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) \right) \\ & \quad = (1-\alpha) \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k)}, \boldsymbol{v}_h) - c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) \right). \end{aligned} \tag{3.2.4}$$

In particular, we select $\boldsymbol{v}_h = \boldsymbol{e}^{(k+1)}$ and $q_h = \varepsilon^{(k+1)}$ so that the the left hand side reduces to $a(\boldsymbol{e}^{(k+1)}, \boldsymbol{e}^{(k+1)})$. Let us focus on the two terms within parentheses on the right hand side.

$$\begin{aligned} & a \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k+1)}, \boldsymbol{e}^{(k+1)}) - c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)}) \right) \\ & \quad = \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{e}^{(k+1)}, \boldsymbol{e}^{(k+1)}) + c(\boldsymbol{e}^{(k)}, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)}) \right) \\ & \quad = c(\boldsymbol{e}^{(k)}, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)}). \end{aligned}$$

[3] We do not add the subscript h to the error symbols. Even if appropriate, we feel it makes the notation much heavier.

Also,

$$\begin{aligned} & a\left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{u}_h{}^{(k)}, \boldsymbol{e}^{(k+1)}) - c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)})\right) \\ &\quad = \left(c(\boldsymbol{u}_h{}^{(k)}, \boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k+1)}) + c(\boldsymbol{e}^{(k)}, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)})\right) \\ &\quad = \left(c(\boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k+1)}) + c(\boldsymbol{u}_h, \boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k+1)}) + c(\boldsymbol{e}^{(k)}, \boldsymbol{u}_h, \boldsymbol{e}^{(k+1)})\right) \end{aligned}$$

By collecting all these results, for the coercivity of the bilinear form $a(\cdot,\cdot)$ we obtain

$$\begin{aligned} \nu C_a \|\boldsymbol{e}^{(k+1)}\|_V^2 \leq C_b \Bigg(& \|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{(k)}\|_V \|\boldsymbol{e}^{(k+1)}\|_V \\ & + (1-\alpha)\left(\|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{(k)}\|_V \|\boldsymbol{e}^{(k+1)}\|_V + \|\boldsymbol{e}^{(k)}\|_V^2 \|\boldsymbol{e}^{(k+1)}\|_V\right)\Bigg) \end{aligned} \tag{3.2.5}$$

After trivial manipulation, and recalling (3.2.1) we find

$$\|\boldsymbol{e}^{(k+1)}\|_V \leq (2-\alpha)\frac{C_b\|\boldsymbol{f}\|_{-1}}{\nu^2 C_a^2}\|\boldsymbol{e}^{(k)}\|_V + \frac{(1-\alpha)C_b}{\nu C_a}\|\boldsymbol{e}^{(k)}\|_V^2. \tag{3.2.6}$$

if we set $\rho \equiv \|\boldsymbol{e}^{(k)}\|_V$ and $\sigma \equiv \|\boldsymbol{e}^{(k+1)}\|_V$, we have the inequality

$$\omega\rho^2 + \zeta\rho > \sigma$$

with

$$\omega = \frac{(1-\alpha)C_b}{\nu C_a}, \quad \zeta = (2-\alpha)\frac{C_b\|\boldsymbol{f}\|_{-1}}{\nu^2 C_a^2} = \frac{2-\alpha}{\chi}.$$

Now, if $\rho > \omega\rho^2 + \zeta\rho$ we have that $\sigma = \beta\rho$ with $0 < \beta < 1$ proving that the modified Picard method is convergent with dumping factor β. For $\alpha = 1$ this is trivially true, as expected by the original Picard scheme, since we find $\omega = 0, \zeta = \chi^{-1} < 1$.

In general, since $\alpha \in [0,1]$, the inequality $\rho > \omega\rho^2 + \zeta\rho$ leads to

$$\rho < \frac{1-\zeta}{\omega} = \frac{\nu C_a}{(1-\alpha)C_b}\frac{\chi - 2 + \alpha}{\chi}.$$

Notice that for $\alpha \in [0,1]$ under the assumption (3.2.3), the right hand side is positive and the bound can be fulfilled.

If we assume that the velocity initial guess is such that $\|\boldsymbol{u}_h{}^{(0)}\|_V < \frac{\nu C_a}{(1-\alpha)C_b}\frac{\chi-2}{\chi}$, then by induction the entire sequence is convergent. Notice, however, that for $\alpha \to 1$ this restriction relaxes, and (3.2.2) suffices for the right hand side to be positive as the original Picard method is recovered.

We finally notice that the assumption (3.2.3) is sufficient for the Theorem. It can be replaced by an assumption on α close enough to 1, precisely $\alpha > 2 - \chi$ (for $\chi < 2$).

Remark

The previous analysis suggests a possible practical approach to obtain a good initial guess. In practice, we can initiate the iterative process with $\alpha = 1$ (original Picard) and eventually switch to $\alpha < 1$ so as to apply the splitting scheme. This approach is robust, even if it requires one to solve the problem with a mesh fine enough to stand the physical Reynolds number. In a previous Remark, we suggested using α to mitigate the mesh requirements for a given (moderate) Reynolds number. A smart dynamic fine tuning of α can clearly attain the purpose of both robustness and efficiency. This dynamic strategy will be investigated elsewhere. For the problems presented herafter, we did not find any problem for the initial guess, so we opted for "small" constant values of α to guarantee computational efficiency.

3.3 Analysis of the splitting method

A complete convergence analysis of the splitting scheme is still missing, and requires a deep analysis of the error equations governing the behavior of the difference between the solution of the modified Picard method and the segregated solution. A similar analysis for the unsteady splitting Yosida scheme was carried out in [100]. Here, we limit ourselves to the consistency analysis, showing that the splitting error locally (i.e. at each modified Picard iteration) vanishes with α. Then, we discuss the empirical selection of the relaxation parameter related to boundness of the generic guess $\boldsymbol{u}_{h,s}^{(k)}$. Here s denotes the solution of the split scheme as opposed to u that we use for the unsplit modified Picard iterate.

We resort to algebraic arguments. To this aim, we will extensively use the following formula, for two generic invertible matrices R and T,

$$R^{-1} - T^{-1} = -R^{-1}(R - T)T^{-1}. \tag{3.3.1}$$

3.3.1 Consistency

As mentioned earlier, the splitting scheme is an inexact block LU factorization of the original saddle point system. Notice that the (1,2) block of the splitting matrix reads

$$A_\alpha K^{-1} B^T = B^T + \alpha C K^{-1} B^T.$$

The consistency of the splitting error block with the unsplit method for $\alpha \to 0$ follows promptly as K, C and B are independent of α.

For the sake of notation, we denote

$$S \equiv BK^{-1}B^T \qquad \text{and} \quad \Sigma_\alpha \equiv BA_\alpha^{-1}B^T.$$

By direct inspection we find that:

$$\mathcal{A}_{\alpha,u}^{-1} = \begin{bmatrix} (I - A_\alpha^{-1} B^T \Sigma_\alpha^{-1} B) A_\alpha^{-1} & A_\alpha^{-1} B^T \Sigma_\alpha^{-1} \\ \\ \Sigma_\alpha^{-1} B A_\alpha^{-1} & -\Sigma_\alpha^{-1} \end{bmatrix}$$

and:

$$\mathcal{A}_{\alpha,s}^{-1} = \begin{bmatrix} (I - K^{-1} B^T S^{-1} B) A_\alpha^{-1} & K^{-1} B^T S^{-1} \\ \\ S^{-1} B A_\alpha^{-1} & -S^{-1} \end{bmatrix}.$$

To analyze the error introduced at each iteration, let us assume to perform one iteration of the unsplit as well as of the split method starting from the same guess $\boldsymbol{u}_h^{(k)}, p_h^{(k)}$. Denoting

$$\boldsymbol{f}_\alpha \equiv \boldsymbol{f} - (1-\alpha) C \boldsymbol{u}_h^{(k)},$$

we have then the two systems

$$\mathcal{A}_{\alpha,u} \begin{bmatrix} \boldsymbol{u}_{h,u}{}^{(k+1)} \\ \\ p_{h,u}{}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_\alpha \\ \\ 0 \end{bmatrix}, \qquad \mathcal{A}_{\alpha,s} \begin{bmatrix} \boldsymbol{u}_{h,s}{}^{(k+1)} \\ \\ p_{h,s}{}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_\alpha \\ \\ 0 \end{bmatrix}.$$

Let $\boldsymbol{e}_s{}^{(k+1)} \equiv \boldsymbol{u}_{h,u}{}^{(k+1)} - \boldsymbol{u}_{h,s}{}^{(k+1)}$ and $\varepsilon_s{}^{(k+1)} \equiv p_{h,u}{}^{(k+1)} - p_{h,s}{}^{(k+1)}$ denote the splitting error for velocity and pressure respectively. By direct computation, the local splitting error reads

$$\begin{bmatrix} \boldsymbol{e}_s{}^{(k+1)} \\ \\ \varepsilon_s{}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \left(K^{-1} B^T S^{-1} B A_\alpha^{-1} - A_\alpha^{-1} B^T \Sigma_\alpha^{-1} B A_\alpha^{-1}\right) \boldsymbol{f}_\alpha \\ \\ \left(S^{-1} - \Sigma_\alpha^{-1}\right) B A_\alpha^{-1} \boldsymbol{f}_\alpha \end{bmatrix}. \tag{3.3.2}$$

From (3.3.1), we obtain

$$\begin{aligned} K^{-1} - A_\alpha^{-1} &= \alpha K^{-1} C A_\alpha^{-1}, \\ S^{-1} - \Sigma_\alpha^{-1} &= -S^{-1}\left(S - \Sigma_\alpha\right)\Sigma_\alpha^{-1} \\ &= -S^{-1} B \left(K^{-1} - A_\alpha^{-1}\right) B^T \Sigma_\alpha^{-1} \\ &= -\alpha S^{-1} B K^{-1} C B^T \Sigma_\alpha^{-1} \end{aligned}$$

Also, set

$$Z_\alpha = K^{-1} C A_\alpha^{-1} B^T \Sigma_\alpha^{-1} B A_\alpha^{-1}$$

and since K is s.p.d. we obtain

$$\begin{aligned} & K^{-1} B^T S^{-1} B A_\alpha^{-1} - A_\alpha^{-1} B^T \Sigma_\alpha^{-1} B A_\alpha^{-1} \\ &\quad = K^{-1} B^T \left(S^{-1} - \Sigma_\alpha^{-1}\right) B A_\alpha^{-1} + \left(K^{-1} - A_\alpha^{-1}\right) B^T \Sigma_\alpha^{-1} B A_\alpha^{-1} \\ &\quad = \alpha \left(I - K^{-1} B^T S^{-1} B\right) Z_\alpha \\ &\quad = \alpha K^{-1/2} \left(I - K^{-1/2} B^T S^{-1} B K^{-1/2}\right) K^{1/2} Z_\alpha. \end{aligned}$$

After introducing the QR factorization of the matrix $K^{-1/2} B^T$ by standard arguments the previous matrix reads

$$\alpha K^{-1/2} Q^T \begin{bmatrix} 0 & 0 \\ & \\ 0 & I_{N_p} \end{bmatrix} Q K^{1/2} Z_\alpha.$$

For the assumption on the Reynolds number, we postulate that the matrix A_α is invertible for any value of $\alpha \in [0,1]$ and its spectrum depends continuously on

α and it is bounded accordingly. The same holds for Σ_α and Z_α. Consequently,

$$\begin{bmatrix} \boldsymbol{e}_s{}^{(k+1)} \\ \\ \varepsilon_s{}^{(k+1)} \end{bmatrix} = \alpha \begin{bmatrix} WZ_\alpha \boldsymbol{f}_\alpha \\ \\ -S^{-1}BK^{-1}CB^T\Sigma_\alpha^{-1}BA_\alpha^{-1}\boldsymbol{f}_\alpha \end{bmatrix} \tag{3.3.3}$$

where

$$W \equiv I - K^{-1}B^TS^{-1}B = K^{-1/2}Q^T \begin{bmatrix} 0 & 0 \\ & \\ 0 & I_{N_p} \end{bmatrix} QK^{1/2}$$

has N_u null eigenvalues and N_p eigenvalues = 1. This points out that the splitting error reads $\alpha \boldsymbol{w}_\alpha$ where the vector $\boldsymbol{w}_\alpha$ has a norm bounded with $\alpha \in [0,1]$. This proves the local consistency of the splitting error at each with $\alpha \to 0$.

3.3.2 Boundedness and empirical selection of the relaxation parameter

As proven in the analysis of the modified Picard scheme, the introduction of an explicit component in the iteration can in general cause some stability issues which can be mitigated through the introduction of the relaxation step with parameter γ. While, in practice, the selection of an appropriate parameter can require some trial and error, we nonetheless can demonstrate some heuristic guidelines for its selection. Here we consider the splitting method applied to a simplified problem with no forcing term, as for any stability analysis, so that $\boldsymbol{f}_\alpha$ reduces to $(\alpha - 1)C\boldsymbol{u}^k$ (we keep working on the non incremental scheme).

By direct inspection, the unrelaxed velocity, hereafter denoted by $\hat{\boldsymbol{u}}^{(k+1)}$ computed by the splitting scheme reads

$$\hat{\boldsymbol{u}}^{(k+1)} = (A_\alpha^{-1} - K^{-1}B^T S^{-1} B A_\alpha^{-1})\boldsymbol{f}_\alpha = (\alpha - 1)WA_\alpha^{-1}C\boldsymbol{u}^{(k)}.$$

Notice that from (3.3.1), we have

$$A_\alpha^{-1} = K^{-1} - \alpha K^{-1} C A_\alpha^{-1}$$

so that

$$\begin{aligned}(\alpha - 1)WA_\alpha^{-1}C &= (\alpha - 1)WK^{-1}C\left(I - \alpha A_\alpha^{-1}C\right) \\ &= (\alpha - 1)WK^{-1}C\left(I - \alpha K^{-1}C\right) + \mathcal{O}(\alpha^2)\end{aligned}$$

Hereafter, we approximate the matrix up to terms of order α,

$$(\alpha - 1)WA_\alpha^{-1}C \approx (\alpha - 1)WK^{-1}C.$$

Fix $\epsilon > 0$. Since $\rho(W) = 1$ as noted in the previous Section, there exists a matrix norm $\|\cdot\|$ such that $\|W\| < (1 + \epsilon)$.

Now, for the relaxed velocity, we have

$$\boldsymbol{u}^{(k+1)} = \gamma\hat{\boldsymbol{u}}^{(k+1)} + (1 - \gamma)\boldsymbol{u}^{(k)} = \left[(\alpha - 1)\gamma WK^{-1}C + (1 - \gamma)\right]\boldsymbol{u}^{(k)}$$

so that for the norm introduced before and assuming that $\alpha\|K^{-1}C\| < 1$ (justified by (3.1.10)):

$$\|\boldsymbol{u}^{(k+1)}\| \leq \left(\frac{(\alpha - 1)}{\alpha}\gamma(1 + \epsilon) + (1 - \gamma)\right)\|\boldsymbol{u}^{(k)}\|.$$

Now,

$$\left|\frac{(\alpha - 1)}{\alpha}\gamma(1 + \epsilon) + (1 - \gamma)\right| \leq 1$$

leads to:

$$\gamma \leq 2\alpha + \mathcal{O}(\epsilon)$$

Therefore, we empirically follow the bound:

$$\gamma < 2\alpha \tag{3.3.4}$$

in Sect. 3.4, as this resulted to be effective.

3.4 Numerical results

3.4.1 Analytical Solution Test

We devise a test problem with a known analytic solution to verify the convergence of the method. We consider problem (2.2.1) defined on $\Omega = [0,1]^2$ with $\nu = .01$. We define our forcing term as:

$$\boldsymbol{f} = -2\nu \begin{bmatrix} \sin(x)\cos(y) + \frac{1}{2}\sin(2x) + \cos(x) \\ \cos(x)\sin(y) + \frac{1}{2}\sin(2y) + \cos(y) \end{bmatrix}$$

with the corresponding solution

$$\boldsymbol{u}_{ex} = [-\sin(x)\cos(y), \cos(x)\sin(y)]', \quad \boldsymbol{p}_{ex} = \sin(x) + \sin(y)$$

We simulated the problem with both a coarse (800 elements) and a fine (5000 elements) mesh for several different values of α and γ. We note that as the solution is inviscid, changing the viscosity does not affect the behavior of the solver and this test is not well-suited for assessing our method's robustness with respect to ν. Thus the purpose of this test is to verify the consistency of our method with an expected solution and to see how the convergence depends on the α, γ, and the

mesh size. As convergence criterion we compared the L^2 norm of velocity between each iteration for tolerance levels of 1e-3, 1e-4, and 1e-5. These computations were performed in FreeFem++ on a 2013 MacBook Pro. We solved the linear systems with UMFPACK, except for the Schur Complement step which was solved with PCG (using the simple Stokes pressure mass preconditioner, see e.g. [119]) with a stopping tolerance of 1e-6. The results are reported in Tab. 3.1 and 3.2.

Coarse Mesh, 800 Elements					
Parameters	Tolerance	$\|\boldsymbol{u}-\boldsymbol{u}_{ex}\|_{H1}$	$\|\boldsymbol{u}-\boldsymbol{u}_{ex}\|_{L2}$	$\|\boldsymbol{p}-\boldsymbol{p}_{ex}\|_{L2}$	Num. Iter.
$\alpha = .175, \gamma = .3$	1e-3	.0313	.00197	.00185	19
$\alpha = .175, \gamma = .3$	1e-4	.00325	.000161	.000345	25
$\alpha = .175, \gamma = .3$	1e-5	.00072	1.86e-5	.00022	32
$\alpha = .2, \gamma = .35$	1e-3	.02061	.00129	.00128	17
$\alpha = .2, \gamma = .35$	1e-4	.00210	7.75e-5	.000253	23
$\alpha = .2, \gamma = .35$	1e-5	.000671	1.57e-5	.000206	32
$\alpha = .25, \gamma = .4$	1e-3	.01259	.00179	.00207	18
$\alpha = .25, \gamma = .4$	1e-4	.00158	.00021	.000371	31
$\alpha = .25, \gamma = .4$	1e-5	.000674	1.748e-5	.000212	47

Table 3.1: Performance of the segregated scheme on the Analytical Solution Test. Coarse Mesh.

Fine Mesh, 5000 Elements					
Parameters	Tolerance	$\|\boldsymbol{u}-\boldsymbol{u}_{ex}\|_{H1}$	$\|\boldsymbol{u}-\boldsymbol{u}_{ex}\|_{L2}$	$\|\boldsymbol{p}-\boldsymbol{p}_{ex}\|_{L2}$	Num. Iter.
$\alpha = .175, \gamma = .3$	1e-3	.0313	.00197	.00169	18
$\alpha = .175, \gamma = .3$	1e-4	.00320	.000163	.000174	25
$\alpha = .175, \gamma = .3$	1e-5	.00033	1.61e-5	4.46e-5	32
$\alpha = .2, \gamma = .35$	1e-3	02055	.00127	.00114	17
$\alpha = .2, \gamma = .35$	1e-4	00202	7.28e-5	.000109	23
$\alpha = .2, \gamma = .35$	1e-5	000153	8.16e-6	3.32e-5	32
$\alpha = .25, \gamma = .4$	1e-3	.01251	.00179	.00204	18
$\alpha = .25, \gamma = .4$	1e-4	.00144	.000211	.000264	31
$\alpha = .25, \gamma = .4$	1e-5	.000171	1.40e-5	4.04e-5	47

Table 3.2: Performance of the segregated scheme on the Analytical Solution Test. Fine mesh.

These results confirm the convergence to the desired solution for appropriate values of α and γ and that convergence rate is independent of mesh size. Conver-

gence was most rapid for $\alpha = .2$ and $\gamma = .35$. We also observe that, as expected, refining the mesh results in better convergence to the exact solution, particularly for the pressure. This confirms that the splitting error does not dominate the discretization error after refining the mesh, despite little change in the convergence rate.

3.4.2 Laminar Flow Past a Cylinder

Problem Setting

We now test the method on the classic benchmark problem of flow past a cylinder in both the 2D and 3D settings. We model our problem setting as the one given in [110].

We prescribe at the inflow the Dirichlet boundary condition:

$$\begin{aligned} \boldsymbol{u}(0,y) &= \frac{1}{.41^2}(1.2y(0.41-y),0), \quad 0 \le y \le .41 \ \text{ in 2D} \\ \boldsymbol{u}(0,y,z) &= \frac{1}{.41^4}(6.56yz(.41-y)(.41-z),0,0), \quad 0 \le y,z \le .41 \ \text{ in 3D} \end{aligned} \tag{3.4.1}$$

We prescribe no-slip boundary conditions along the cylinder and the top and bottom of the domain and traction-free boundary conditions at the outflow. The benchmark parameters of interest are the lift and drag coefficients at the cylinder c_l and c_d respectively and the difference in pressure between the front and back of the cylinder defined as:

$$\begin{aligned} \Delta p &= p(.15,.2) - p(.25,.2) \ \text{ in 2D} \\ \Delta p &= p(.45,.2,.205) - p(.55,.2,.205) \ \text{ in 3D} \end{aligned} \tag{3.4.2}$$

The drag and lift coefficients c_d and c_l are defined classically in two dimensions as:

$$c_d = \frac{2}{U^2 L} \int_{\Gamma_{cyl}} \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \boldsymbol{n}_y - p\boldsymbol{n}_x \, \partial\Gamma_{cyl}, \; c_l = -\frac{2}{U^2 L} \int_{\Gamma_{cyl}} \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \boldsymbol{n}_y + p\boldsymbol{n}_y \, \partial\Gamma_{cyl} \tag{3.4.3}$$

and in three dimensions as:

$$c_d = \frac{2}{U^2 LH} \int_{\Gamma_{cyl}} \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \boldsymbol{n}_y - p\boldsymbol{n}_x \, \partial\Gamma_{cyl}, \; c_l = -\frac{2}{U^2 LH} \int_{\Gamma_{cyl}} \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \boldsymbol{n}_y + p\boldsymbol{n}_y \, \partial\Gamma_{cyl} \tag{3.4.4}$$

where $\boldsymbol{n}$ is the unit normal vector and U, L and H are the characteristic velocity, length and height respectively. Here the kinematic viscosity ν is set to .001. As shown in [7], the computation of c_d and c_l based on this formulation may yield inaccurate numerical results. Accordingly, we employ a technique used in [65] to rewrite (3.4.3) as a volume integral. Define $\boldsymbol{v}_d$ as $(\boldsymbol{v}_d)|_{\Gamma_{cyl}} = (1,0)^T$ and vanishing at all other boundaries. Following the argument given in [65] we see that for two dimensions we obtain the alternate formulation

$$c_d = -500 \int_\Omega \nu \nabla \boldsymbol{u} : \nabla \boldsymbol{v}_d + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v}_d - p(\nabla \cdot \boldsymbol{v}_d) \, \partial\Omega \tag{3.4.5}$$

For three dimensions, we define $\boldsymbol{v}_d$ as $(\boldsymbol{v}_d)|_{\Gamma_{cyl}} = (1,0,0)^T$, vanishing at all other boundaries and obtain:

$$c_d = -\frac{500}{.41} \int_\Omega \nu \nabla \boldsymbol{u} : \nabla \boldsymbol{v}_d + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v}_d - p(\nabla \cdot \boldsymbol{v}_d) \, \partial\Omega \tag{3.4.6}$$

Similarly, defining $\boldsymbol{v}_l$ as $(\boldsymbol{v}_l)|_{\Gamma_{cyl}} = (0,1)^T$ and vanishing at all other boundaries for two dimensions we find:

$$c_l = -500 \int_\Omega \nu \nabla \boldsymbol{u} : \nabla \boldsymbol{v}_l + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v}_l - p(\nabla \cdot \boldsymbol{v}_l) \, \partial\Omega \tag{3.4.7}$$

For three dimensions, let $\boldsymbol{v}_l$ as $(\boldsymbol{v}_l)|_{\Gamma_{cyl}} = (0,1,0)^T$ and vanishing at all other

boundaries:

$$c_l = -\frac{500}{.41}\int_\Omega \nu\nabla\boldsymbol{u} : \nabla\boldsymbol{v}_l + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}_l - p(\nabla\cdot\boldsymbol{v}_l)\,\partial\Omega \tag{3.4.8}$$

For our finite element discretization, we use standard Taylor-Hood P2/P1 for velocity and pressure. We test convergence by comparing the $L2$ norm of the current iteration with the previous iteration and stop iterating at a tolerance level of 1e-3. For accuracy, we compare our values to benchmark values found in the literature. For 2D we use the values reported in [86]:

$$c_d = 5.57953523384, \quad c_l = 0.010618948146, \quad \Delta p = 0.11752016697 \tag{3.4.9}$$

and for the 3D case we use the values found in [63]:

$$c_d = 6.1853267, \quad c_l = .0094012217, \quad \Delta p = 0.17077855 \tag{3.4.10}$$

2D Results

For this problem our experimental design was as follows: we considered two mesh levels: coarse and fine. We use our inequality (3.1.15) with $\|\boldsymbol{u}\|_\infty = .3$ (the maximum inflow velocity) to determine α for each mesh level. Referring to (3.3.4), we set $\gamma = 1.9\alpha$. The values of these parameters for each mesh are reported in the tables below.

For each mesh level, we compare the performance our method in terms of both accuracy and time with standard Picard iterations. To solve the linear problem at each iteration in Standard Picard, we will test both direct and iterative approaches. We will use UMFPACK for the direct solves and GMRES preconditioned with block-triangular approximate Schur complement preconditioners of the kind found in [37] for the iterative solves (again using UMFPACK for the

2D Flow Past Cylinder, Coarse Mesh (831 triangles,4037 DOF)						
Method	c_d/c_d Rel.Err.	c_l/c_l Rel.Err.	$\Delta p/\Delta p$ Rel.Err.	Num. Iter.	Time/Iter.	Total time
$\alpha = .13, \gamma = .25$	5.560/.0034	.0315/1.97	.1166/.0073	16	.21s	3.36s
UMFPACK	5.558/.0037	.0109/.0302	.11668/.0057	6	.242s	1.45s
GMRES (PCD)	5.589/.0036	.0109/.029	.1168/.0057	6	.56s	3.36s

2D Flow Past Cylinder, Fine Mesh (1815 triangles, 8615 DOF)						
Method	c_d/c_d Rel.Err.	c_l/c_l Rel.Err.	$\Delta p/\Delta p$ Rel.Err.	Num. Iter.	Time/Iter.	Total time
$\alpha = .19, \gamma = .36$	5.619/.0071	.0108/.0195	.1167/.00709	13	.44s	5.72s
UMFPACK	5.568/.0020	.10167/.0047	.1162/.0104	6	.5s	3s
GMRES (PCD)	5.568/.0020	.0106/.0051	.1162/.0104	6	1.28s	7.68s

Table 3.3: Numerical Results on the 2D Schaefer/Turek Flow Past a Cylinder Test Case.

inner solves, tolerance of 1e-6). In particular, we will approximate the Schur complement with the Pressure-Convection Diffusion (PCD) preconditioner (using UMFPACK for the inner solves). We note that the PCD preconditioner requires updating at each step.

These computations were performed in FreeFem++ on a 2017 MacBook Pro. For our splitting method, we solved the linear systems with UMFPACK, except for the Schur Complement step which was solved with PCG preconditioned with the scaled Stokes lumped pressure mass preconditioner (see e.g. [119], [37]) with a stopping criterion of 1e-6 for the outer solve and UMFPACK for the inner solves. To check the convergence, we computed the difference in L^2 of the current and previous velocity at each step and terminated the iteration at for a value below 1e-3. We report our results in Tab. 3.3.

Our method performs well. On the coarse mesh, it failed to compute the lift coefficient accurately; however for both mesh levels all other parameters were computed with good accuracy. In terms of cost, we observe that the number of iterations is larger when using our method, which is consistent with our expectation. However, the cost of each iteration is lower. Our scheme has the lowest average iteration time across all of the methods, outperforming even the direct solver on a small-scale 2D problem. When considering the savings at each iteration, we see that our method outperforms preconditioned GMRES. For problems

of this type, however, the savings per iteration do not appear significant enough to outperform a Picard scheme with a direct solver.

3D Test Case

In three dimensions we tested the algorithm locally on a moderate mesh with 36095 elements, giving 174941 total degrees of freedom. This is comparable to a mesh level used for this problem in [63], where the relevant parameters were computed up to three accurate leading digits. We therefore expect to be able to attain acceptable results for this mesh level. We also computed two fine-mesh solutions on a cluster with four Intel Xeon E5-4627 CPUs with 40 cores and 1 TB of memory. The mesh levels for the cluster computations were 75223 tetrahedra and 333677 total degrees of freedom and 152055 tetrahedra and 675460 degrees of freedom.

Based on our mesh configuration and the problem parameters, we estimate the Peclet number to be around .22 and set α and γ in accordance with (3.1.15) and (3.3.4). Our experimental design is similar to the one in the previous subsection, comparing our scheme with the same set of standard Picard solution methods (UMFPACK and GMRES with the PCD preconditioner) for the MacBook simulation. On the cluster we compared our method only to GMRES with PCD, as the problem was too large for UMFPACK. We report the results in Tab. 3.4.

Our approach performs excellently in this case, significantly outperforming the direct solver and performing comparably to PCD-preconditioned GMRES in the coarse case and outperforming it for the fine mesh cases. Although we require twice as many iterations (as in the 2D case), the savings gained in the assembly and solution process make up for the higher iteration count. The savings were especially pronounced on the cluster. The methods performed similarly in terms of accuracy.

3D Flow Past Cylinder, 36095 tetrahedra (174941 DOF, MacBook)						
Method	c_d/c_d Rel.Err.	c_l/c_l Rel.Err.	$\Delta p/\Delta p$ Rel.Err.	Num. Iter.	Time/Iter.	Total time
$\alpha = .175$, $\gamma = .325$	6.152/.0053	.0092/.0229	.1694/.0082	14	68.2s	954.8s
UMFPACK	6.149/.0059	.0099/.0581	.1693/.0086	7	178.8s	1251.6s
GMRES (PCD)	6.149/.0059	.0099/.0582	.1693/.0086	7	147.8s	1034.6s
3D Flow Past Cylinder, 75223 tetrahedra (333677 DOF, 40-Core Cluster)						
Method	c_d/c_d Rel.Err.	c_l/c_l Rel.Err.	$\Delta p/\Delta p$ Rel.Err.	Num. Iter.	Time/Iter.	Total time
$\alpha = .175$, $\gamma = .325$	6.141/.0072	.0097/.0337	.1704/.0022	15	269.5s	4042.5s
GMRES (PCD)	6.134/.0082	.0099/.0575	.1702/.0031	7	782.4s	5476.8s
3D Flow Past Cylinder, 152055 tetrahedra (675460 DOF, 40-Core Cluster)						
Method	c_d/c_d Rel.Err.	c_l/c_l Rel.Err.	$\Delta p/\Delta p$ Rel.Err.	Num. Iter.	Time/Iter.	Total time
$\alpha = .175$, $\gamma = .325$	6.135/.0081	.0093/.0127	.1677/.0180	15	633.52	9502.9s
GMRES (PCD)	6.123/.0100	.0095/.0175	.1666/.0247	7	2757.6s	19303.4s

Table 3.4: Numerical Results on the 3D Schaefer/Turek Flow Past a Cylinder Test Case.

A finer mesh did not necessarily guarantee more accurate computations of our parameters of interest. As this behavior was observed for both our method and for the comparison, it is most likely a result of the specific mesh configurations and not the solution techniques. The literature shows that the computation of these values varies widely across different problem configurations, and our values were within the ranges reported in [110] for each mesh level.

Note that the timings reported are taken from two different machines (a MacBook for the coarse simulation and a cluster for the fine simulations). Referring to the solutions on the cluster, our method appears to scale well with the problem size, with the advantage becoming more pronounced as problem size increases (26% speedup for 75223k tetrahedra and 51% for 152055k tetrahedra). For the coarse mesh run on a MacBook, our method outperformed the other methods; however the advantage was less pronounced (8% speedup compared to PCD-GMRES, 23.7% compared to the direct solver).

This is unsurprising, as the pressure mass matrix is optimal with respect to mesh size for our approximate Schur complement system [37] and hence the iterative solver used to compute its solution does not require more iterations as the problem size increases. Additionally, the preconditioner does not require

reassembly at each nonlinear iteration, which is a major bottleneck with PCD-preconditioned GMRES for large-scale problems.

In addition to the savings in computational time, there are other practical reasons one may prefer our approach. Importantly, it is very straightforward to implement and can be easily adapted to existing Stokes solvers which segregate velocity and pressure. Additionally, as we split the nonlinear term, the velocity block A_α in step 1 is dominated by the diffusive term, making it relatively easy to precondition. We expect this advantage to be significant for problems too large for A_α to be solved directly; for problems of this scale the preconditioning of the A block when using a PCD-type preconditioner can be difficult.

3.4.3 Heywood-Rannacher-Turek Bifurcation: Dependence on Re and Automatic Parameter Selection

In this section we solve the problem on a bifurcated domain shown in Fig. 3.1 (inspired by a similar geometry in [56]) for $\nu = .01$, $\nu = .0067$, and $\nu = .005$. The purpose of this test is twofold: first, we want to see how changes in ν affect the method's performance. Second, we want to test the selection of α and γ by an adaptive approach based on the inequalities (3.1.15) and (3.3.4).

At each iteration k, we chooose α^k and γ^k adaptively by computing the local Peclet number for each element K:

$$\mathbb{P}_K = \frac{\|\boldsymbol{u}^k\|_K h_K}{2\nu} \tag{3.4.11}$$

The theory states that we should enforce $\mathbb{P}_K < 1$; however in practice we find that enforcing $\mathbb{P}_K < 2$ provides faster convergence without sacrificing stability.

Therefore, we set:

$$\alpha^k = .75 \min_{K \in \mathcal{T}} \frac{4\nu}{\|\boldsymbol{u}^k\|_K h_K} \tag{3.4.12}$$

Using (3.3.4), we then set $\gamma^k = 1.9\,\alpha^k$.

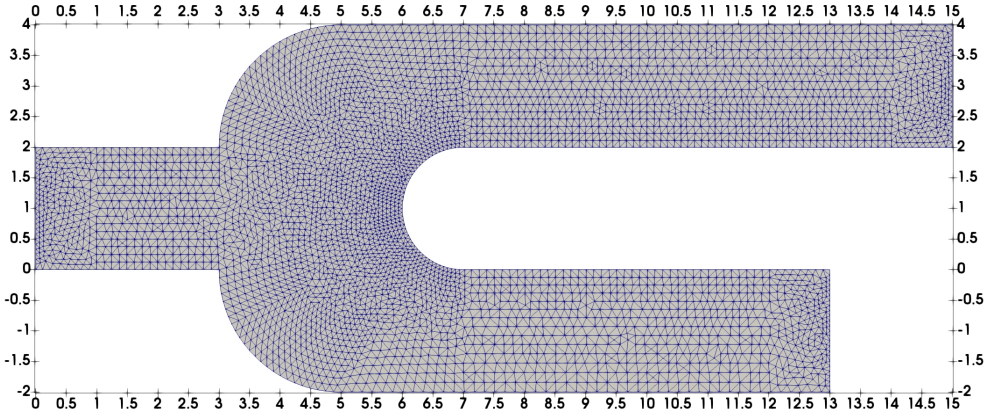

Figure 3.1: Domain for the Heywood-Rannacher-Turek Bifurcation test case.

Along the sides we prescribed no-slip boundary conditions. At the inlet we prescribed a standard parabolic profile with maximum velocity 1.0 and at the outlets homogeneous Neumann conditions. We stop the iteration when the difference in velocity L^2 norm compared to the previous iteration is below 1e-3. The results are reported in Tab 3.5. Note that the splitting errors correspond to the difference in norm between the solution computed with our method and a monolithic solution computed with a direct solver. The results are reported in Tab. 3.5.

As expected, the number of iterations required for convergence increases with the Reynolds number, as we must correspondingly choose smaller α and γ. The required number of iterations appears to increase linearly with decreases in ν, as we expect. The adaptive selection of α seems quite robust, not significantly im-

Bifurcation Test					
h	ν	Iterations	Time/Iter	Velocity Error (L^2)	Pressure Error (L^2)
.1	.01	24	.857s	.0041	.0162
.075	.01	23	1.71s	.0036	.0166
.05	.01	26	2.68s	.0030	.0163
.1	.0067	42	.877s	.0065	.0142
.075	.0067	39	1.72s	.0055	.0109
.05	.0067	41	2.69s	.0054	.0108
.1	.005	69	.904s	.0034	.0083
.075	.005	70	1.82s	.0036	.0094
.05	.005	68	2.84s	.0042	.0079

Table 3.5: Results for the Heywood-Rannacher-Turek 2D bifurcation.

pacting the required number of iterations or the solution accuracy. This suggests that larger values of α may be used as the mesh is refined.

We also notice that the time per iteration across each test appears to depend on the mesh size h, but is largely unaffected by the Reynolds number. This is unsurprising as we solve the A_α-block system with a direct solver and the approximate Schur complement does not contain any convection-dependent terms. We expect that if one uses iterative solvers for A_α and the inner solve of the approximate Schur complement (which does depend on ν) this will no longer be true; however referring to (3.1.15) we note that our splitting approach will serve to partially mitigate the impact of ν on the velocity solves.

This test demonstrates that as Re increases, the convergence of our method slows and we require more iterations. This the expected behavior for all nonlinear iterative schemes for this problem. This test also confirms that an adaptive approach based on (3.1.15) and (3.3.4) for automatic parameter selection is a possible strategy that eliminates the need for user input. The adaptive scheme presented here is a simple one and by no means the only option. We feel that more investigation of similar schemes may lead to further improvements in performance and are a possible area for a follow-up to this work.

3.4.4 3D Brain Aneurysm

Our last test is designed to demonstrate our method's applicability for large-scale, nontrivial problems of clinical interest in hemodynamics. We test our method on case 32 from the publicly available ANEURISK database [3]. We will then compare our solution with the time-average of the unsteady benchmark computation.

We run our simulation with a fine mesh with 354980 tetrahedra and 1621338 total degrees of freedom on a cluster with four Intel Xeon E5-4627 CPUs with 40 cores and 1 TB of memory (the same cluster used in Section 5.2.3). We use the same problem configuration as the benchmark computation, with $\nu = .04$ g/cm-s (based on [23]). For the inflow, we prescribe a parabolic Poiseulle profile with a flow rate of .4005 ml/s (determined based on [24]). We prescribe no-slip boundary conditions along the walls and homogenous Neumann boundary conditions at the outflows. We terminate the nonlinear iterations when the difference in L^2 norm of velocity between consecutive iterations falls below 1e-3. For the purposes of comparing solution times, we also compute the solution with a monolithic approach in which the linear system at each step is solved with GMRES preconditioned with the Pressure Convection-Diffusion preconditioner. We report the timings in Tab. 3.6.

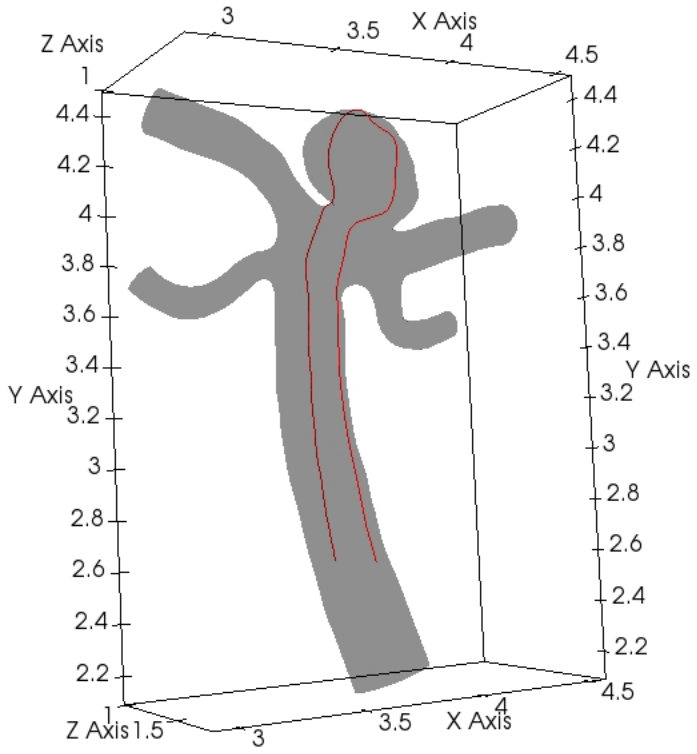

Figure 3.2: We compare the WSS along the highlighted curve.

We compare the wall shear stress on the center of the vessel along the pictured curve in Fig. 3.2 of our simulation with the benchmark computation. The curve is formed by the intersecting surface with a plane with origin $(3.74, 3.29, 1.39)$ normal to $(1, 0, 0)$ beginning at the point $(3.74, 3.29, 1.30)$ and ending at the point $(3.74, 3.29, 1.04)$. The curve was intentionally chosen to be far from the inlet to minimize the impact of the Dirichlet boundary condition on the flow. Note that our computation is steady-state while the benchmark computation is time-averaged; since the Reynolds number is low (approximately 40) we may assume the impact of the unsteady Reynolds stresses is low and therefore the steady-state solution provides a reasonable approximation of the time-average. We plot the results in Fig. 3.3 and Fig. 3.4.

BSA Aneurysm, 354980 tetrahedra (1621338 DOF, 40-Core Cluster)		
Method	Num. Iterations	Avg time/Iteration
$\alpha = .25, \gamma = .45$	13	2521.3s
GMRES (PCD)	7	7754.0s

Table 3.6: Results for the 3D BSA Aneurysm test case.

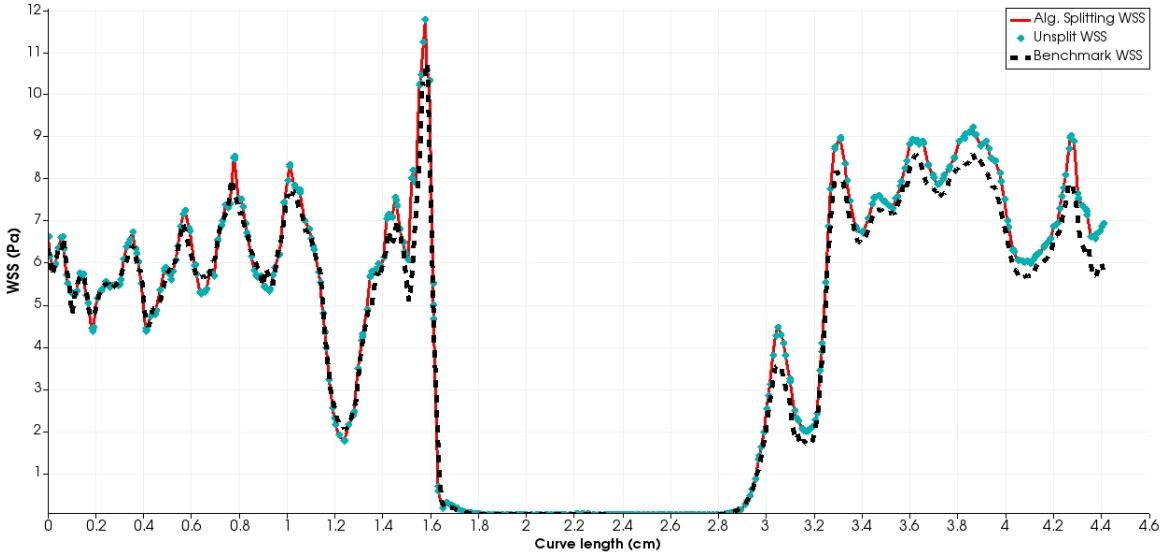

Figure 3.3: Comparison of computed wall-shear stress with the benchmark computation.

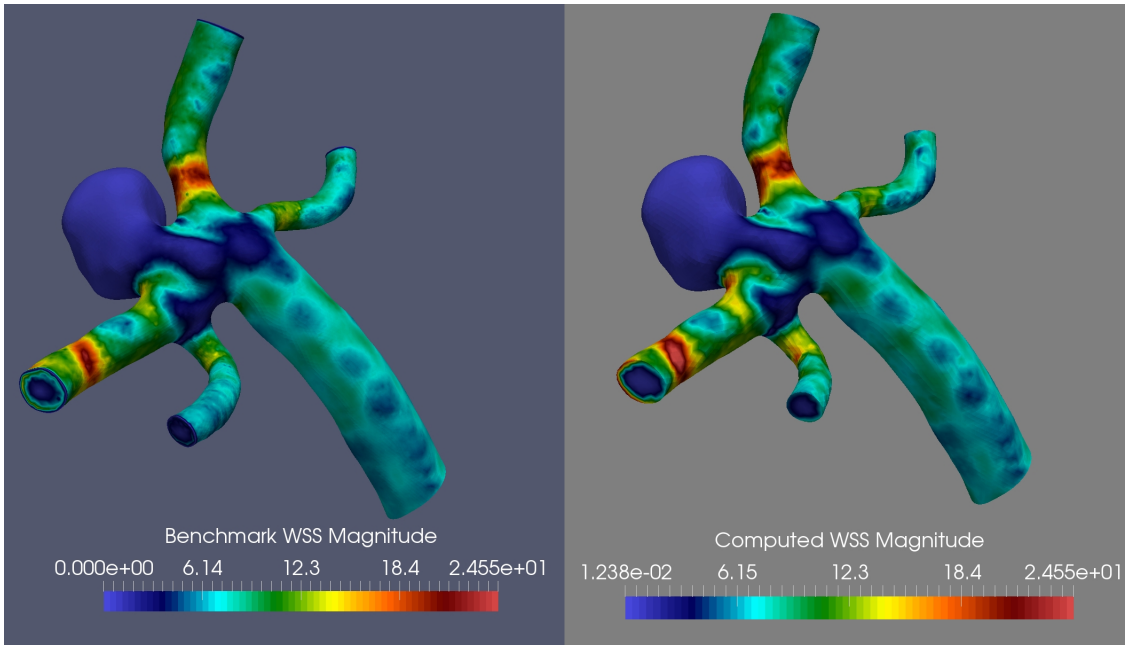

Figure 3.4: Comparison of benchmark computation (left) and the solution computed with our method (right).

We observe several things: first, we are in excellent agreement with the benchmark computation. We again see our method results in computational savings when compared to monolithic approach using GMRES preconditioned with PCD, while retaining the same level of accuracy. Though we again require nearly twice as many iterations (13 compared to 7) each iteration is less than half the total cost, resulting in roughly 40% savings in computational time. In terms of acccuracy, the GMRES solution and the solution computed with our method appear identical. This demonstrates the method's possible utility on problems of clinical or research interest and its competitiveness with standard steady-state solution techniques.

Chapter 4

Splitting Methods Using Grad-Div Stabilization

[1] The approaches of the previous chapter represent an effective extension of the traditional algebraic splitting schemes to the steady setting. However, there are some limitations with this approach. Most crucially, their performance requires the selection of parameters α and γ. While criterion for the automatic selection of these parameters exists, for higher Reynolds numbers α may be small, leading to small γ, delaying convergence. We therefore would like to develop a scheme that retains the good aspects of the previously discussed schemes (in particular, a segregated velocity/pressure approach with a stationary Schur complement that is easy to precondition and solve) while mitigating its drawbacks (selection of user parameters, necessary under-relaxation possibly slowing convergence). We are able to develop such schemes using *grad-div stabilization* (see e.g. [53, 88]).

[1] This chapter is a modified version of the following works: L. Rebholz, A. Viguerie, and M. Xiao. "Efficient nonlinear iteration schemes based on algebraic splitting for the incompressible Navier-Stokes equations," in revision; A. Viguerie and M. Xiao. "Effective Chorin-Temam Algebraic Splitting Schemes for the Steady Navier-Stokes Equations," in preparation.

4.1 Grad-div Stabilization for Steady Navier-Stokes

We remind the reader that the incompressibility condition in the Incompressible Navier-Stokes equations corresponds to conservation of mass and is given by:

$$\nabla \cdot \boldsymbol{u} = 0.$$

However, in the weak formulation at the discrete level the incompressibility condition reads:

$$\int_\Omega (\nabla \cdot \boldsymbol{u}_h)\ q_h = 0$$

for all q_h in Q_h. This enforces mass conservation over the domain, but does not necessarily imply $\nabla \cdot \boldsymbol{u}_h = 0$ pointwise, depending on the choice of finite element pair $V_h \times Q_h$. For example, Taylor-Hood finite element pairs $\mathbb{P}^k/\mathbb{P}^{k-1}$ are not pointwise divergence-free in general [22]. Define the following the bilinear form:

$$g_{div}(\cdot,\cdot) : V \times V \to \mathbb{R}, \text{s.t. } g_{div}(\boldsymbol{w},\boldsymbol{v}) = \xi \int_\Omega (\nabla \cdot \boldsymbol{w})(\nabla \cdot \boldsymbol{v}). \tag{4.1.1}$$

This is known as the *grad-div stabilization* term. Adding g_{div} to the weak Navier-Stokes equations corresponds to adding the term $-\xi\nabla(\nabla \cdot \boldsymbol{u})$, zero by incompressibility, to the (2.2.1). It is therefore consistent with our original equations.

For suitable $\xi > \nu$, the term g_{div} penalizes $\nabla \cdot \boldsymbol{u}_h \neq 0$, forcing $\|\nabla \cdot \boldsymbol{u}_h\|$ to be small and improving the accuracy and stability of the solution [88, 51]. Grad-div stabilization is an especially effective way to improve solution accuracy in situations when additional mesh refinement is not feasible (as may be common in clinical hemodynamics). Although the theory requires that $\xi > \nu$, it is well-known that $\xi \sim \mathcal{O}(1)$ gives good performance for many applications [105, 51, 62].

At an iteration $k+1$, the grad-div stabilized Picard iteration is given by: find

$\boldsymbol{u}_h^{k+1} \in V_h$ and $p_h^{k+1} \in Q_h$ s.t. for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$a^*(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c^*(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1}) + g_{div}(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) = (\boldsymbol{f}, \boldsymbol{v}_h) \quad (4.1.2)$$

$$b(\boldsymbol{u}_h^{k+1}, q_h) = 0 \quad (4.1.3)$$

By Theorem 2.3.1, (4.1.2-4.1.3) converges to a unique solution $(\boldsymbol{u}_h, p_h)$[2] obeying the bound:

$$\|\nabla \boldsymbol{u}_h\|_{L^2} \leq \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu} \quad (4.1.4)$$

provided that the small-data hypothesis (2.3.17) holds:

$$\eta := \frac{C_b \|\boldsymbol{f}\|_{-1}}{\nu^2} < 1 \quad (4.1.5)$$

For the remainder of this section, we assume a discrete functional space $V_h \times Q_h$ and denote $(\boldsymbol{u}_h, p_h)$ as simply $(\boldsymbol{u}, p)$ without ambiguity. As before, the algebraic problem associated with (4.1.2-4.1.3) has the form:

$$\begin{bmatrix} A + D & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix} \quad (4.1.6)$$

where D is the matrix arising from the grad-div stabilization term and $A = K + C$ as before, with K corresponding to the diffusive term and C to the convective term[3].

We define the *Steady Incremental Grad-Div Picard-Algebraic Chorin-Temam Iteration* (hereafter abbreviated as GISACT) as follows:

[2]The addition of the grad-div term does not affect the result of Theorem 2.3.1 (in fact it strengthens it), as the same argument shown there establishes that the sequence given by $\xi \|\nabla \cdot \boldsymbol{e}^{k+1}\|_{L^2}^2 + \|\nabla \boldsymbol{e}^{k+1}\|_{L^2}^2$ decays to zero as $k \to \infty$.

[3]The dependence of A and C on the iterate $\boldsymbol{u}^k$ is understood.

Algorithm 4.1.1 *Given initial guess $(\boldsymbol{u}^0, p^0)$, at an iteration $k+1$ the Steady Incremental Grad-Div Picard-Algebraic Chorin-Temam Iteration is given by:*

1. *Find $\boldsymbol{z}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$:*

$$a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{z}^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^k)$$

2. *Find $(\boldsymbol{w}^{k+1}, \delta_p^{k+1}) \in V_h \times Q_h$ satisfying for all $(\boldsymbol{v}, q) \in V_h \times Q_h$:*

$$\begin{aligned} a^*(\boldsymbol{w}^{k+1}, \boldsymbol{v}) - b(\delta_p^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{w}^{k+1}, \boldsymbol{v}) &= 0, \\ b(\boldsymbol{w}^{k+1}, q) &= -b(\boldsymbol{z}^{k+1}, q) \end{aligned}$$

3. *Find $\boldsymbol{u}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$,*

$$a^*(\boldsymbol{u}^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{u}^{k+1}, \boldsymbol{v}) = b(\boldsymbol{v}, \delta_p^{k+1}) + a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v})$$

4. *Set $p^{k+1} = \delta_p^{k+1} + p^k$.*

Note that we use the Laplacian formulation for the diffusive term 2.2.19 and the skew-symmetric formulation of the convective term 2.2.18. This will be helpful in the analysis to follow.

While defining step 2 as above is crucial for the purposes of our analysis, observe that at discrete level step 2 is given by:

$$\begin{bmatrix} K+D & 0 \\ B & -B(K+D)^{-1}B^T \end{bmatrix} \begin{bmatrix} I & (K+D)^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{k+1} \\ \delta \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -B\boldsymbol{z}^{k+1} \end{bmatrix} \tag{4.1.7}$$

Where D is the matrix arising from the grad-div stabilization. The block-LU solution is then obtained by:

(i) Solve $(K+D)\widetilde{\boldsymbol{w}}^{k+1} = 0$;

(ii) Solve $B\left(K+D\right)^{-1}B^T\,\delta\boldsymbol{p}^{k+1} = -B\boldsymbol{z}^{k+1} - B\widetilde{\boldsymbol{w}}^{k+1}$;

(iii) Correction step: $\boldsymbol{w}^{k+1} = \widetilde{\boldsymbol{w}}^{k+1} - (K+D)^{-1}\delta\boldsymbol{p}^{k+1}$

However, by the full rank of $K+D$, we have $\widetilde{\boldsymbol{w}}^{k+1} = 0$, reducing (ii) to $B\left(K+D\right)^{-1}B^T\,\delta\boldsymbol{p}^{k+1} = -B\boldsymbol{z}^{k+1}$. Referring again to Alg. 4.1.1, we do not utilize $\boldsymbol{w}^{k+1}$ in the subsequent steps and its value does not affect that of $\delta\boldsymbol{p}^{k+1}$, allowing us to disregard (iii).

The discrete form of Algorithm 4.1.1 is therefore equivalent to solving the following sequence of problems:

$$\left(K + C(\boldsymbol{u}^k) + D\right)\boldsymbol{z}^{k+1} = \boldsymbol{f} + B^T\boldsymbol{p}^k \tag{4.1.8}$$

$$B\left(K+D\right)^{-1}B^T\,\delta\boldsymbol{p}^{k+1} = -B\boldsymbol{z}^{k+1} \tag{4.1.9}$$

$$\left(K+D\right)\boldsymbol{u}^{k+1} = \left(K+D\right)\boldsymbol{z}^{k+1} + B^T\delta\boldsymbol{p}^{k+1} \tag{4.1.10}$$

$$\boldsymbol{p}^{k+1} = \delta\boldsymbol{p}^{k+1} + \boldsymbol{p}^k \tag{4.1.11}$$

This can also be interpreted as solving the following inexact-LU factorization of (4.1.6) in pressure-incremental form:

$$\begin{bmatrix} A+D & 0 \\ B & -B(K+D)^{-1}B^T \end{bmatrix}\begin{bmatrix} I & (K+D)^{-1}B^T \\ 0 & I \end{bmatrix}\begin{bmatrix} \boldsymbol{u}^{k+1} \\ \delta\boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} - B^T\boldsymbol{p}^k \\ 0 \end{bmatrix}$$

Referring to (3.1.6), as $H_1 = H_2$ above, we see that this indeed a Chorin-Temam type splitting, justifying its name. This scheme retains many of the favorable properties of the schemes shown in Chapter 3. In particular, our approximate Schur complement is SPD and easy to precondition (an optimal preconditioner is given by $(\nu+\xi)M_p^{-1}$ [8]) and as it does not change at each iteration, there is no need to update the preconditioner [88]. However, unlike the schemes in

the previous section, this approach does not require under-relaxation. This is advantageous as under-relaxation is known to slow down convergence.

However, one should note that the presence of the matrix D now makes the velocity systems more difficult to solve compared to the analogous systems in the earlier schemes. Several preconditioners have been used in the literature, in particular algebraic multigrid and ILU approaches [53, 26] as well as block triangular approximations [13]. However, the best way to precondition and solve $K + C + D$ remains an open question [88]. This is in contrast to the schemes from the previous section, in which the solution of the velocity was comparatively easy.

Remark 1. A closely related *Steady Incremental Grad-Div Picard-Yosida Iteration* (IPY) also exists and is defined as follows:

Algorithm 4.1.2 *Given initial guess $(\boldsymbol{u}^0, p^0)$, at an iteration $k+1$ the Steady Incremental Grad-Div Picard-Yosida Iteration is given by:*

1. *Find $\boldsymbol{z}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$:*

$$a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{z}^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^k)$$

2. *Find $(\boldsymbol{w}^{k+1}, \delta_p^{k+1}) \in V_h \times Q_h$ satisfying for all $(\boldsymbol{v}, q) \in V_h \times Q_h$:*

$$\begin{aligned} a^*(\boldsymbol{w}^{k+1}, \boldsymbol{v}) - b(\delta_p^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{w}^{k+1}, \boldsymbol{v}) &= 0, \\ b(\boldsymbol{w}^{k+1}, q) &= -b(\boldsymbol{z}^{k+1}, q) \end{aligned}$$

3. *Set $p^{k+1} = \delta_p^{k+1} + p^k$.*

4. *Find $\boldsymbol{u}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$,*

$$a^*(\boldsymbol{u}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{u}^{k+1}, \boldsymbol{v}) + g_{div}(\boldsymbol{u}^{k+1}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^{k+1})$$

This has the corresponding algebraic form:

$$\begin{bmatrix} A+D & 0 \\ B & -B(K+D)^{-1}B^T \end{bmatrix} \begin{bmatrix} I & (A+D)^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{k+1} \\ \delta \boldsymbol{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} - B^T \boldsymbol{p}^k \\ 0 \end{bmatrix}$$

Referring again to (3.1.6), as $H_1 \neq H_2$ we verify that this is a Yosida-type splitting by our naming convention. Much of what follows in the analysis and numerical tests sections holds also for this scheme. Due to the similarity between the methods, we focus on the Algebraic Chorin-Temam variant in this thesis in the interest of brevity and avoiding redundancy. We will, however, compare against this alternate approach during our numerical tests.

Remark 2. One may also use this approach to define Newton-type iterations for both Alg. 4.1.1 and 4.1.2. One obtains these algorithms by replacing step 1 in both 4.1.1 and 4.1.2 with:

Find $\boldsymbol{z}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$:

$$\begin{aligned} a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{z}^{k+1}, \boldsymbol{u}^k, \boldsymbol{v}) + g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v}) \\ = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^k) + c^*(\boldsymbol{u}^k, \boldsymbol{u}^k, \boldsymbol{v}) \end{aligned}$$

And for Alg. 4.1.2, replacing step 3 with:

Find $\boldsymbol{u}^{k+1} \in V_h$ satisfying for all $\boldsymbol{v} \in V_h$,

$$\begin{aligned} a^*(\boldsymbol{u}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{u}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^{k+1}, \boldsymbol{u}^k, \boldsymbol{v}) + g_{div}(\boldsymbol{u}^{k+1}, \boldsymbol{v}) \\ = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^{k+1}) + c^*(\boldsymbol{u}^k, \boldsymbol{u}^k, \boldsymbol{v}) \end{aligned}$$

We will include an analysis of the Newton version of Algorithm 4.1.1 and note that the analysis of the Newton version of Algorithm 4.1.2, is similar. A brief numerical test verifying the faster convergence of these methods is included at

the end of this section. We recall (2.3.9):

4.1.1 Analysis

Theorem 4.1.1 *Assume* $\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}_0)\|_{L^2}^2+\xi^{-1}\|p-p_0\|_{L^2}^2 \leq \|\nabla\boldsymbol{u}\|_{L^2}^2$ *and* $\xi \geq \nu$, *where* $(\boldsymbol{u},p)$ *is the solution of* (4.1.2)-(4.1.3) *and* $(\boldsymbol{u}_0,p_0)\in V_h\times Q_h$ *the initial guess. Let* $(\boldsymbol{u}^{k+1},p^{k+1})\in V_h\times Q_h$ *be the solution of Algorithm 4.1.1 at step* $k+1$, *then it converges linearly to* $(\boldsymbol{u},p)$, *provided* [4]

$$\eta < \min\{\nu\left(8\beta^{-2}(2\nu+2\nu^{1/2}+3)+2(1+\nu^{1/2})^2\right)^{-1},(16\beta^{-2}+3)^{-1},1\}$$

Proof. Denote $\boldsymbol{e}_u^{k+1}:=\boldsymbol{u}-\boldsymbol{u}^{k+1}$ and $\boldsymbol{e}_z^{k+1}:=\boldsymbol{u}-\boldsymbol{z}^{k+1}$. As all norms in V_h and Q_h in this analysis are $L^2(\Omega)$, we denote $\|\cdot\|_{L^2}$ as simply $\|\cdot\|$ hereafter. Our proof will assume:

$$\nu\|\nabla\boldsymbol{e}_u^k\|^2+\xi^{-1}\|p-p^k\|^2\leq\|\nabla\boldsymbol{u}\|^2, \tag{4.1.12}$$

and we seek to show that:

$$\nu\|\nabla\boldsymbol{e}_u^{k+1}\|+\xi^{-1}\|p-p^{k+1}\|\leq\eta\left(\nu\|\nabla\boldsymbol{e}_u^k\|+\xi^{-1}\|p-p^k\|\right) \tag{4.1.13}$$

which will imply the condition at the next iteration.

Subtracting step 1 of Algorithm 4.1.1 from the unique steady solution equation (4.1.2), we obtain for all $\boldsymbol{v}\in V_h$,

$$\xi(\nabla\cdot\boldsymbol{e}_z^{k+1},\nabla\cdot\boldsymbol{v})+\nu(\nabla\boldsymbol{e}_z^{k+1},\nabla\boldsymbol{v})=(p-p^k,\nabla\cdot\boldsymbol{v})-c^*(\boldsymbol{e}_u^k,\boldsymbol{u},\boldsymbol{v})-c^*(\boldsymbol{u}^k,\boldsymbol{e}_z^{k+1},\boldsymbol{v}).$$

[4] Note β is the inf-sup constant (2.3.1).

Choosing $\boldsymbol{v} = \boldsymbol{e}_z^{k+1}$ vanishes the last nonlinear term, and provides the bound

$$\xi\|\nabla\cdot\boldsymbol{e}_z^{k+1}\|^2 + \nu\|\nabla\boldsymbol{e}_z^{k+1}\|^2 \leq \xi^{-1}\|p-p^k\|^2 + \nu\eta^2\|\nabla\boldsymbol{e}_u^k\|^2, \tag{4.1.14}$$

thanks to the definition of η, the bound (4.1.4) on the true solution $\boldsymbol{u}$, Young's inequality and (2.2.8).

Next, we bound $\|p-p^{k+1}\|$. Begin by adding Steps 1 and 2, which gives for all $\boldsymbol{v}\in V_h$,

$$\begin{aligned}\xi(\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}),\nabla\cdot\boldsymbol{v}) + \nu(\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}),\nabla\boldsymbol{v})\\ = (p_k,\nabla\cdot\boldsymbol{v}) - c^*(\boldsymbol{u}^k,\boldsymbol{z}^{k+1},\boldsymbol{v}) + (\boldsymbol{f},\boldsymbol{v}),\end{aligned} \tag{4.1.15}$$

and we note that $(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1})\in X_h$. Subtracting the unique steady solution equation (4.1.2) from this, we obtain the error equation for all $\boldsymbol{v}\in V_h$:

$$\begin{aligned}\xi(\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u}),\nabla\cdot\boldsymbol{v}) + \nu(\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u}),\nabla\boldsymbol{v})\\ = (p_k-p,\nabla\cdot\boldsymbol{v}) - c^*(\boldsymbol{e}_u^k,\boldsymbol{e}_z^{k+1},\boldsymbol{v}) - c^*(\boldsymbol{u},\boldsymbol{e}_z^{k+1},\boldsymbol{v}) - c^*(\boldsymbol{e}_u^k,\boldsymbol{u},\boldsymbol{v})\end{aligned} \tag{4.1.16}$$

Choosing $\boldsymbol{v} = (\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\in X_h$ gives $(p^k-p,\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})) = 0$, yielding the bound:

$$\begin{aligned}\xi\|\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2 + \nu\|\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2\\ \leq 2C_b^2\nu^{-1}\|\nabla\boldsymbol{e}_u^k\|^2\|\nabla\boldsymbol{e}_z^{k+1}\|^2 + \nu\eta^2\|\nabla\boldsymbol{e}_z^{k+1}\|^2 + \nu\eta^2\|\nabla\boldsymbol{e}_u^k\|^2,\end{aligned} \tag{4.1.17}$$

thanks to (2.2.8), Young's inequality, (4.1.4). Using the assumption that $\nu\|\nabla\boldsymbol{e}_u^k\|^2 \leq \|\nabla\boldsymbol{u}\|^2$, this reduces to

$$\begin{aligned}\xi\|\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2 + \nu\|\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2\\ \leq (\nu+2)\eta^2\|\nabla\boldsymbol{e}_z^{k+1}\|^2 + \nu\eta^2\|\nabla\boldsymbol{e}_u^k\|^2.\end{aligned} \tag{4.1.18}$$

We now use this bound to bound the pressure error, after applying inf-sup to (4.1.16) to find

$$\begin{aligned}\beta\|p-p^{k+1}\| &\leq \xi\|\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|+\nu\|\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\| \\ &\quad + C_b\|\nabla\boldsymbol{u}\|\|\nabla\boldsymbol{e}_z^{k+1}\| + C_b\|\nabla\boldsymbol{u}\|\|\nabla\boldsymbol{e}_u^{k}\| + C_b\|\nabla\boldsymbol{e}_u^{k}\|\|\nabla\boldsymbol{e}_z^{k+1}\| \\ &\leq \xi\|\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|+\nu\|\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\| \\ &\quad + \eta(\nu+\nu^{1/2})\|\nabla\boldsymbol{e}_z^{k+1}\| + \eta\nu\|\nabla\boldsymbol{e}_u^{k}\|.\end{aligned} \tag{4.1.19}$$

Squaring both sides, using that $\xi \geq \nu$, and reducing yields

$$\begin{aligned}\beta^2\|p-p^{k+1}\|^2 &\leq \\ &4\xi\left(\xi\|\nabla\cdot(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2+\nu\|\nabla(\boldsymbol{w}^{k+1}+\boldsymbol{z}^{k+1}-\boldsymbol{u})\|^2\right) \\ &+4\xi\left((\nu^{1/2}+1)^2\eta^2\|\nabla\boldsymbol{e}_z^{k+1}\|^2+\eta^2\nu\|\nabla\boldsymbol{e}_u^{k}\|^2\right).\end{aligned} \tag{4.1.20}$$

Using the bound (4.1.18) and multiplying both sides by ξ^{-1} reduces this estimate to

$$\begin{aligned}\xi^{-1}\|p-p^{k+1}\|^2 &\leq \\ &4\beta^{-2}\left((2\nu+2\nu^{1/2}+3)\eta^2\|\nabla\boldsymbol{e}_z^{k+1}\|^2+2\nu\eta^2\|\nabla\boldsymbol{e}_u^{k}\|^2\right).\end{aligned} \tag{4.1.21}$$

Next, we use (4.1.21) and (4.1.14) to bound $\|\nabla\boldsymbol{e}_u^{k+1}\|$. Adding Step 1 and Step 3, and then subtracting it from (4.1.2) obtains

$$\begin{aligned}&\xi(\nabla\cdot\boldsymbol{e}_u^{k+1},\nabla\cdot\boldsymbol{v})+\nu(\nabla\boldsymbol{e}_u^{k+1},\nabla\boldsymbol{v}) \\ &\quad=(p-p_k,\nabla\cdot\boldsymbol{v})-c^*(\boldsymbol{e}_u^{k},\boldsymbol{u},\boldsymbol{v})-c(u,\boldsymbol{e}_z^{k+1},\boldsymbol{v})-c^*(\boldsymbol{e}_u^{k},\boldsymbol{e}_z^{k+1},\boldsymbol{v}).\end{aligned} \tag{4.1.22}$$

Letting $\boldsymbol{v} = \boldsymbol{e}_u^{k+1}$ and applying Cauchy-Schwarz inequality, (2.2.8) and as-

sumption $\|\nabla \boldsymbol{e}_u^k\|^2 \leq \nu^{-1}\|\nabla u\|^2$ produces

$$\begin{aligned}
&\xi\|\nabla \cdot \boldsymbol{e}_u^{k+1}\|^2 + \nu\|\nabla \boldsymbol{e}_u^{k+1}\|^2 \\
&\quad \leq \|p - p^{k+1}\|\|\nabla \cdot \boldsymbol{e}_u^{k+1}\| + \left(\nu\eta\|\nabla \boldsymbol{e}_u^k\| + (\nu + \nu^{1/2})\eta\|\nabla \boldsymbol{e}_z^{k+1}\|\right)\|\nabla \boldsymbol{e}_u^{k+1}\|
\end{aligned} \tag{4.1.23}$$

Applying Young's inequality yields

$$\begin{aligned}
&\xi\|\nabla \cdot \boldsymbol{e}_u^{k+1}\|^2 + \nu\|\nabla \boldsymbol{e}_u^{k+1}\|^2 \\
&\qquad \leq \xi^{-1}\|p - p^{k+1}\|^2 + 2\nu\eta^2\|\nabla \boldsymbol{e}_u^k\|^2 + 2(1 + \nu^{1/2})^2\eta^2\|\nabla \boldsymbol{e}_z^{k+1}\|^2
\end{aligned} \tag{4.1.24}$$

Adding it to (4.1.21) and combining it with (4.1.14) and (4.1.21), we obtain

$$\begin{aligned}
&\xi^{-1}\|p - p^{k+1}\|^2 + \nu\|\nabla \boldsymbol{e}_u^{k+1}\|^2 \\
&\quad \leq \left(2(1 + \nu^{1/2})^2 + 8\beta^{-2}(2\nu + 2\nu^{1/2} + 3)\right)\eta^2\nu^{-1}\xi^{-1}\|p - p^k\|^2 \\
&\quad + \left((2(1 + \nu^{1/2})^2 + 8\beta^{-2}(2\nu + 2\nu^{1/2} + 3))\eta^2\nu^{-1} + 2 + 16\beta^{-2}\right)\eta^2\nu\|\nabla \boldsymbol{e}_u^k\|^2
\end{aligned} \tag{4.1.25}$$

Using the small data condition $\eta < \nu\left(8\beta^{-2}(2\nu + 2\nu^{1/2} + 3) + 2(1 + \nu^{1/2})^2\right)^{-1}$, $\eta < (16\beta^{-2} + 3)^{-1}$, we find that:

$$\begin{aligned}
&\xi^{-1}\|p - p^{k+1}\|^2 + \nu\|\nabla \boldsymbol{e}_u^{k+1}\|^2 \\
&\qquad \leq \eta\xi^{-1}\|p - p^k\|^2 + \left(\eta + 16\beta^{-2} + 2\right)\eta^2\nu\|\nabla \boldsymbol{e}_u^k\|^2 \\
&\qquad \leq \eta\left(\xi^{-1}\|p - p^k\|^2 + \nu\|\nabla \boldsymbol{e}_u^k\|^2\right)
\end{aligned} \tag{4.1.26}$$

We have thus proven that $\xi^{-1}\|p - p^{k+1}\|^2 + \nu\|\nabla \boldsymbol{e}_u^{k+1}\|^2$ is a contractive sequence in k, and thus converges. Since the solution of the (finite dimensional) problem (4.1.2)-(4.1.3) is unique and bounded by the data, we have that the limit of the incremental Picard-Yosida iteration converges to the solution of (4.1.2)-(4.1.3).

4.2 Numerical Results

In this section, we present two numerical tests: a 2D flow in a bifurcated domain and 3D BSA aneurysm. Here we want to show two things: that our incremental method converges linearly to the solution of discrete steady Navier-Stokes system (4.1.2), and that the approximation of the Schur complement does not severely affect the convergence rate.

4.2.1 Case 1: 2D Bifurcation Flow.

We first test our proposed Algorithm 4.1.1 by solving the 2D steady Navier-Stokes problem in the same bifurcated domain used in section 3.4.3 (seen in Fig. 3.1) . For the discretization, we employ Taylor-Hood $\mathbb{P}^2/\mathbb{P}^1$ elements on a fine mesh of 8988 elements ($h = .05$), leading to 41,589 total DOF. We prescribe a parabolic inflow profile with peak velocity 2.0 and assign traction-free boundary conditions at both outflows. We solve for $\nu = .0133$ and $\nu = .0067$ for $\xi = 1$ and $\xi = 2$ using the software FreeFEM on a 2017 MacBook Pro.

We compare the solutions computed by Algorithm 4.1.2 (IPY) and Algorithm 4.1.1 (GISACT) to the reference solution from the standard Picard iterations up to a very high level of accuracy (1e-12). To ensure the best possible accuracy, we solved the full saddle-point system with UMFPACK at each iteration. For IPY and GISACT, we solve both velocity systems with UMFPACK (Step k.1 and Step k.2), and the Schur Complement with CG preconditioned by the lumped pressure mass-matrix for outer solve and UMFPACK for inner solve.

For the purposes of comparison, we also computed the solution for the same problem configuration with a standard Picard scheme (4.1.2-4.1.3). At each iteration we solved the full saddle-point system with GMRES preconditioned with the

Figure 4.1: Bifurcation test streamlines; $\nu = .0133$ (top), $\nu = .0067$ (bottom).

following block-triangular preconditioner found in [53]:

$$P^{-1} = \begin{bmatrix} A_\xi & B^T \\ 0 & -(\nu + \xi)^{-1} M_p \end{bmatrix}^{-1} \tag{4.2.1}$$

where M_p is the lumped pressure mass matrix and A_ξ the velocity block with grad-div stabilization. We will solve the exact velocity block using UMFPACK. In the available literature using this preconditioner, typically the exact velocity block (rather than an approximation) is solved either directly or with an iterative method [53, 26]. We set the outer solve tolerance to 1e-6. These computations were performed with FreeFem++.

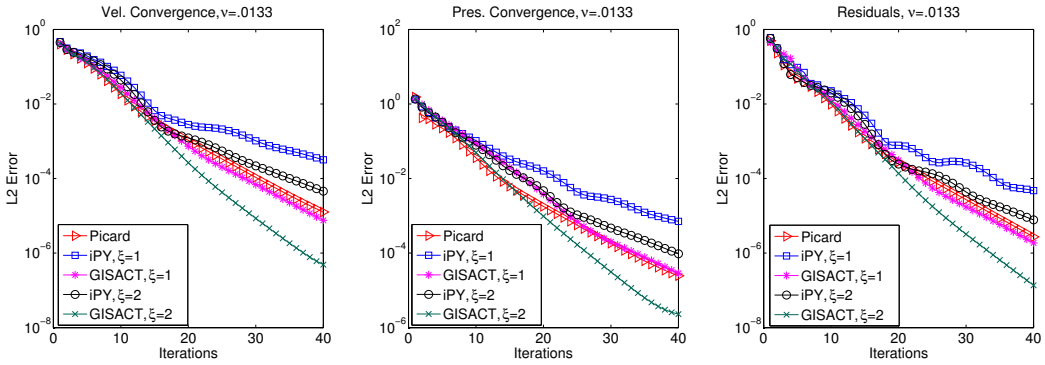

Figure 4.2: Comparison of convergence for test case 1, $\nu = .0133$.

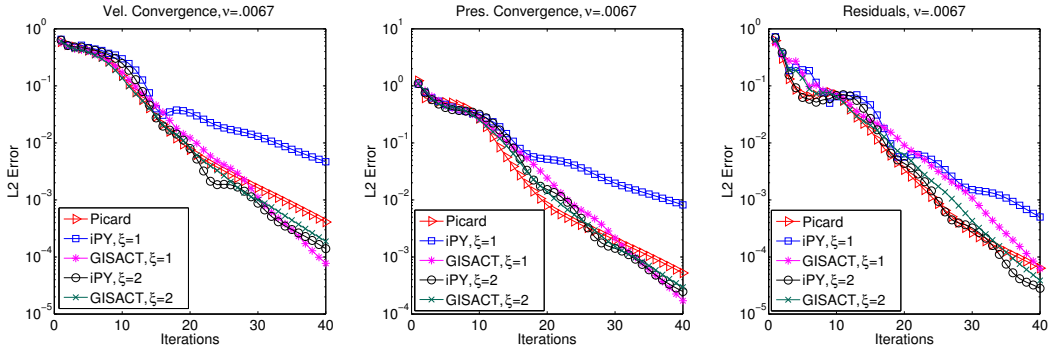

Figure 4.3: Comparison of convergence for test case 1, $\nu = .0067$.

For $\nu = .0133$ (fig. 4.2), we observe a similar rate of convergence for standard Picard and GISACT. IPY converges more slowly, with $\xi = 1$ being non-competitive and $\xi = 2$ converging faster but still more slowly than standard Picard or GISACT. For $\nu = .0067$ (fig. 4.3) , we see that IPY with $\xi = 1$ is again not competitive, converging very slowly. However, GISACT with both values of ξ and IPY with $\xi = 2$ are both comparable or slightly superior to standard Picard.

We note that in those figures above, the convergence of velocity although has the trend of linear convergence, but there are some oscillations. We are not sure as to their exact cause and this requires further investigation.

In Table 4.1 we provide information regarding the numerical cost of each non-linear iteration. Here, outer Krylov iterations refers to the number of outer GMRES iterations for Standard Picard and the number of outer PCG iterations for

the approximate Schur complement solve for IPY and GISACT. The average solve time is the time required to solve the systems at each nonlinear iteration, and does not factor in the assembly costs (similar across all algorithms and therefore excluded to more clearly show the differences between methods). For all three methods, the number of outer iterations does appear to change significantly with ν. Standard Picard shows a mild but observable dependence on ξ for its required outer iterations, but both IPY and GISACT appear less sensitive in this regard. Overall, the splitting schemes require fewer outer iterations per solve, resulting in clear computational savings at each step.

2D Bifurcation: $\nu = .0133$ (Re=200)						
	GISACT		IPY		Standard Picard	
ξ	Outer Krylov iter.	Avg solve time	Outer Krylov iter.	Avg solve time	Outer Krylov iter.	Avg solve time
1.0	11	.600s	11	.594s	30	1.346s
2.0	10	.549s	10	.552s	25	1.134s
2D Bifurcation: $\nu = .00667$ (Re=400)						
	GISACT		IPY		Standard Picard	
ξ	Outer Krylov iter.	Avg solve time	Outer Krylov iter.	Avg solve time	Outer Krylov iter.	Avg solve time
1.0	11	.603s	11	.595s	30	1.409s
2.0	11	.598s	11	.593s	25	1.146s

Table 4.1: Iteration statistics for 2D Bifurcation test.

Let us now briefly consider the underlying operations at a given iteration for both the Standard Picard and our grad-div splitting schemes. Using the preconditioner 4.2.1, each outer GMRES iteration for Standard Picard requires a diagonal matrix inversion in Q_h, a matrix-vector multiply, and the solution of a system in V_h. For both of the grad-div splitting methods, at each iteration we solve the Schur complement system with PCG ($P^{-1} = (\nu + \xi)M_p^{-1}$), requiring at each outer PCG iteration a solve in V_h, two matrix-vector multiplies, and a diagonal mass matrix inversion in Q_h. We must also solve two more systems in V_h for the intermediate and corrected velocity fields.

In both cases, the number of V_h solves is the dominant driver of cost and is determined directly by the number of outer Krylov (GMRES for standard Picard, PCG for splitting) iterations k, with standard Picard requiring k solves and the

splitting methods requiring $k+2$. As we used the same direct sparse linear solver for all inner solves in V_h, by this reasoning we expect:

$$t^* = \frac{\text{Avg. solve time}}{k^*}, \quad k^* = \begin{cases} \text{Outer Krylov iterations} & \text{for Standard Picard} \\ \text{Outer Krylov iterations + 2} & \text{for GISACT/IPY} \end{cases}$$

Method	ν	ξ	Avg solve time	k^*	t^*
IPY	.0133	1.0	.594s	13	.0457s
GISACT	.0133	1.0	.600s	13	.0462s
Std. Picard	.0133	1.0	1.356s	30	.0452s
IPY	.0133	2.0	.552s	12	.0450s
GISACT	.0133	2.0	.549s	12	.0458s
Std. Picard	.0133	2.0	1.134	22	.0454s
IPY	.0067	1.0	.595s	13	.0458s
GISACT	.0067	1.0	.603s	13	.0464s
Std. Picard	.0067	1.0	1.389s	30	.0463s
IPY	.0067	2.0	.593s	13	.0456s
GISACT	.0067	2.0	.598s	13	.0460s
Std. Picard	.0067	2.0	1.146s	25	.0458s

Table 4.2: Relationship between average solve time and outer Krylov iterations across all Bifurcation test cases.

to be more or less constant across all cases. Referring to Table 4.2 we confirm this is the case, with the overall variation over t^* being about one-thousandth of a second, a relative difference of only 3 percent between the maximum and minimum t^*.

We note briefly that the insensitivity of solve *times* to ν observed here is a consequence of using direct solvers for the inner blocks. If one solves the inner blocks iteratively, we expect the solve times to change with ν; however we still expect the outer iteration count to remain ν-independent. We will explore this issue in the following section.

Overall, from this test we conclude that the convergence of GISACT is roughly

the same, or slightly faster, than Standard Picard for both values of ξ. IPY is somewhat slower than both Picard and GISACT for $\xi = 1$, but performs comparably for $\xi = 2$. Changing ξ does appear to affect the convergence of GISACT somewhat, but on the whole it appears less sensitive to ξ than IPY. In terms of cost per iteration, we find both IPY and GISACT to be cheaper than Standard Picard, resulting in around 50% savings in solve time. As shown by Table 4.2, these temporal savings are explained by the differences in outer iteration count.

4.2.2 Case 2: 3D Brain Aneurysm.

We next test our method on case 32 from the publicly available ANEURISK database [3], which is the same geometry used in Chapter 3. We use $\mathbb{P}^2/\mathbb{P}^1$ Taylor-Hood elements on a coarser mesh than used previously, with 36,693 total tetrahedra leading to 224,180 total DOF and ran the simulations with the software FreeFem++ 2017 MacBook Pro. We chose to run the simulations locally, as opposed to a cluster, as we wanted to test our algorithm's suitability for small-scale local computations, often encountered in clinical settings. We solve the velocity blocks with the sparse direct solver UMFPACK. For the Schur complement system we use PCG preconditioned with the lumped pressure Mass matrix with an outer stopping tolerance of 1e-6, again using UMFPACK for the inner solve. We end our computation when the difference in L^2 norm of velocity between consecutive iterations falls below 1e-3. We test both grad-div splitting schemes (4.1.1) and (4.1.2). For the standard Picard comparison, we solved the problem using the same comparison solver configuration as in section 4.2.2 (GMRES preconditioned with 4.2.1).

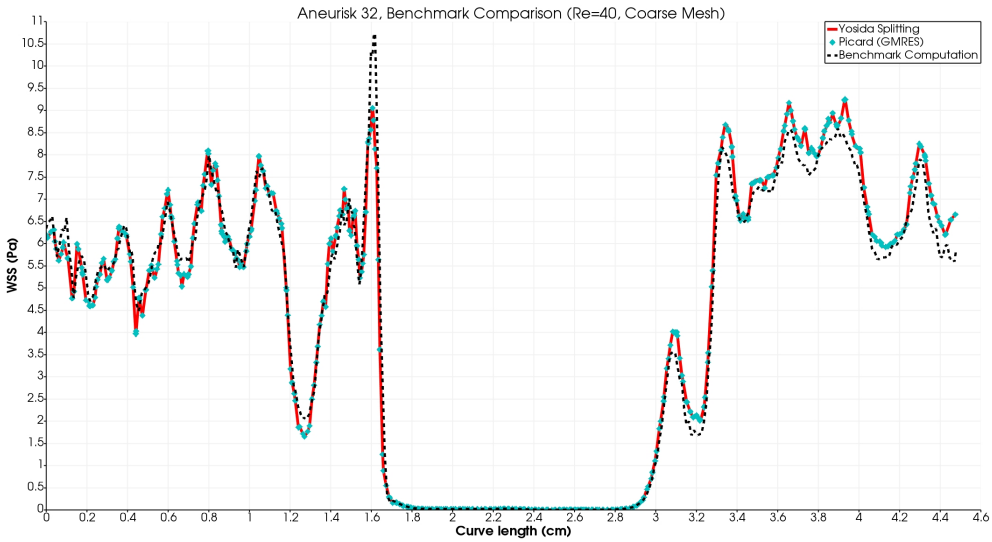

Figure 4.4: Results for BSA Aneurysm Test Case

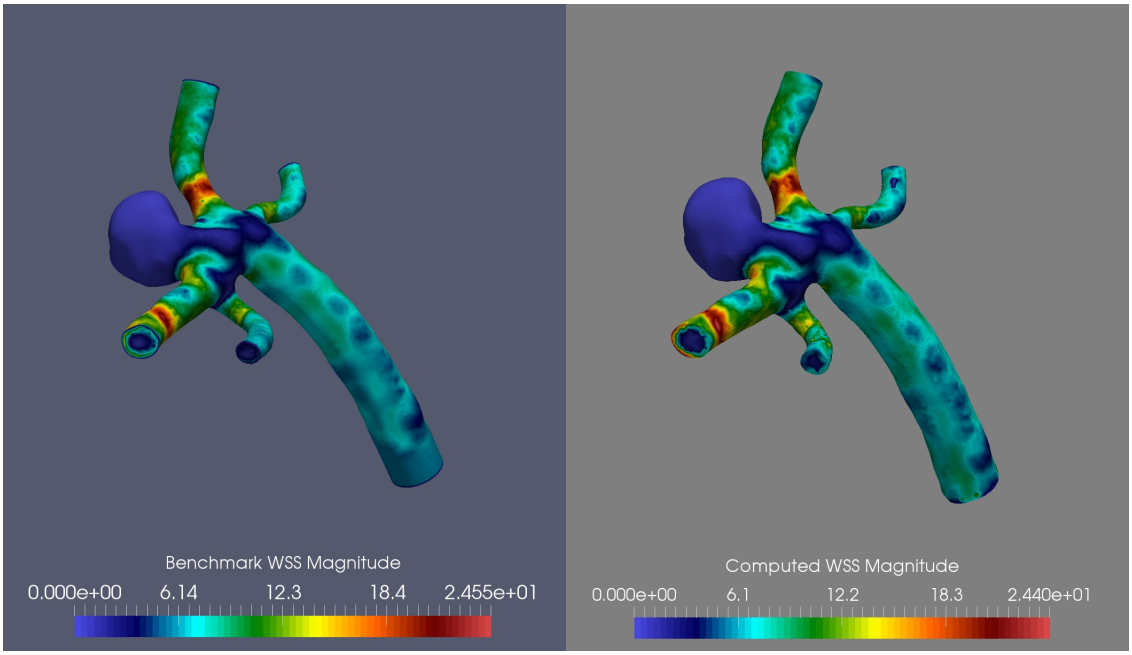

Figure 4.5: Left: Benchmark solution, Right: computed solution (using IPY)

We observe good agreement with the benchmark values, though due to the coarse mesh the agreement is not quite as strong as the results shown in Section 3.4.4. We note again that as our computation is a steady computation and the benchmark computation was a time-averaged unsteady computation, we do not

3D BSA Aneurysm, 224,180 DOF: $Q = .4005$ ml/s, $\nu = .04$ g/cm-s (Re=40)					
Method	Nonlin. Iter.	Outer Krylov Iter.	Avg solve time	Avg. total time	Total time
IPY	12	32	34.5s	67.0s	804.0s
GISACT	8	32	33.8s	66.1s	528.8s
Std. Picard	7	48	47.9s	89.2s	713.6s

Table 4.3: Avg. time/solve only includes the system solve time; avg. total time includes assembly costs. Outer Krylov iterations denotes avg. num. of outer GMRES iterations for Standard Picard and outer PCG iterations for the Schur complement step for splitting methods per nonlinear iteration.

expect perfect agreement. The GISACT splitting converged in 8 iterations at 66.1 seconds per iteration (528.8 total seconds), while IPY reached convergence in 12 iterations at 67.0 seconds per iteration (804.0s seconds). Standard Picard required 7 iterations and 89.2 seconds per iteration (713.6 seconds). GISACT is clearly superior to the standard Picard scheme for this test, while the slower convergence of IPY negates its advantage in cost per iteration, leading to worse overall performance than standard Picard.

Dependence on ν and Q; Numerical cost.

Using the same Aneurisk 32 geometry, we ran further tests at Re=100 and Re=160 to assess the robustness of the methods with respect to different flow configurations. We recall that the Reynolds number (2.1.2) can be increased by either decreasing ν or increasing the inflow rate Q. Although the different configurations of ν and Q may have identical Re, the problems are *not* the same, neither physically nor numerically, and we should not expect the same behavior from numerical schemes. From the perspective of hemodynamics, both situations are relevant and correspond to distinct physical phenomena. For example, when modeling the effect of a blood-thinning medication, one may change ν and not Q, while the opposite may be more appropriate for modeling a change in cardiac out-

put[5]. This distinction is important for the accurate computation of dimensional parameters. Therefore, in this test we assess performance for both situations: one in which we keep our original flow rate of .4005 ml/s and lower the viscosity, and one in which we keep the original viscosity of .04 g/cm-s and increase the flow rate. We again solve all inner blocks with sparse direct solvers. We report the results in Table 4.4.

3D BSA Aneurysm, 224,180 DOF: $Q = 1.005$ ml/s, $\nu = .04$ g/cm-s (Re=100)					
Method	Nonlin. Iter.	Outer Krylov Iter.	Avg solve time	Avg. total time	Total time
IPY	19	31	32.2s	63.8s	1212.2s
GISACT	15	31	33.1s	65.7s	985.5s
Std. Picard	11	66	70.1s	113.2s	1245.2s
3D BSA Aneurysm, 224,180 DOF: $Q = .4005$ ml/s, $\nu = .016$ g/cm-s (Re=100)					
Method	Nonlin. Iter.	Outer Krylov Iter.	Avg solve time	Avg. total time	Total time
IPY	15	28	31.4s	63.7s	955.5s
GISACT	13	28	31.1s	63.5s	825.5s
Std. Picard	11	47	47.1s	87.8s	965.8s
3D BSA Aneurysm, 224,180 DOF: $Q = 1.6005$ ml/s, $\nu = .04$ g/cm-s (Re=160)					
Method	Nonlin. Iter.	Outer Krylov Iter.	Avg solve time	Avg. total time	Total time
IPY	30	31	32.5s	63.9s	1917.0s
GISACT	27	31	32.7s	64.8s	1749.6s
Std. Picard	28	90	95.3s	138.6s	3880.8s
3D BSA Aneurysm, 224,180 DOF: $Q = .4005$ ml/s, $\nu = .01$ g/cm-s (Re=160)					
Method	Nonlin. Iter.	Outer Krylov Iter.	Avg solve time	Avg. total time	Total time
IPY	28	26	28.6s	60.7s	1699.6s
GISACT	26	26	28.5s	59.9s	1557.4s
Std. Picard	28	45	47.7s	89.8s	2514.4s

Table 4.4: Avg. time/solve only includes the system solve time; avg. total time includes assembly costs. Outer Krylov iterations denotes avg. num. of outer GMRES iterations for GMRES and outer PCG iterations for the Schur complement step for splitting methods per nonlinear iteration.

For all methods, the number of required nonlinear iterations increases as both ν decreases and Q increases. The required number of nonlinear iterations is identical for standard Picard for cases with the same Reynolds number, however for the splitting schemes we observe minor differences, with the high ν/high Q cases converging slightly more slowly. This appears to affect the IPY scheme more than

[5]While both situations are relevant and important, the scenario in which we keep ν constant and vary Q is encountered much more frequently.

GISACT.

We now turn our attention to the cost per each nonlinear iteration. Referring to the results in Table 4.4, we see that GMRES preconditioned with 4.2.1 (used for standard Picard), is robust with respect to ν, confirming the findings in [53], but its performance worsens significantly with increases in Q. In comparison, the algebraic splitting schemes are robust with respect to both parameters, showing no sensitivity at all to Q and a mild *improvement* in performance as ν decreases. Over these tests, the GISACT splitting method demonstrates clear superiority over the comparison in both performance and versatility. The IPY scheme is also clearly superior over the comparison in terms of cost per iteration, however this advantage is offset somewhat by its slower convergence, leading to more modest advantages and worse overall performance for the Re=40 case.

Note that we do not have any benchmark data for Re=100 and Re=160; however by comparing to a reference solution computed with monolithic methods (to ensure the best possible convergence and accuracy) to a high degree of precision (stopping tolerance of 10^{-8}) as in the previous section, we again confirm that our schemes converge to the desired solution at a rate comparable to that of standard Picard. We show plots for the case with $Q = 1.6005$, $\nu = .04$ below in Figure 4.6. We observe the same pattern as in the previous test, with GISACT converging at about the same rate as standard Picard and IPY slightly more slowly.

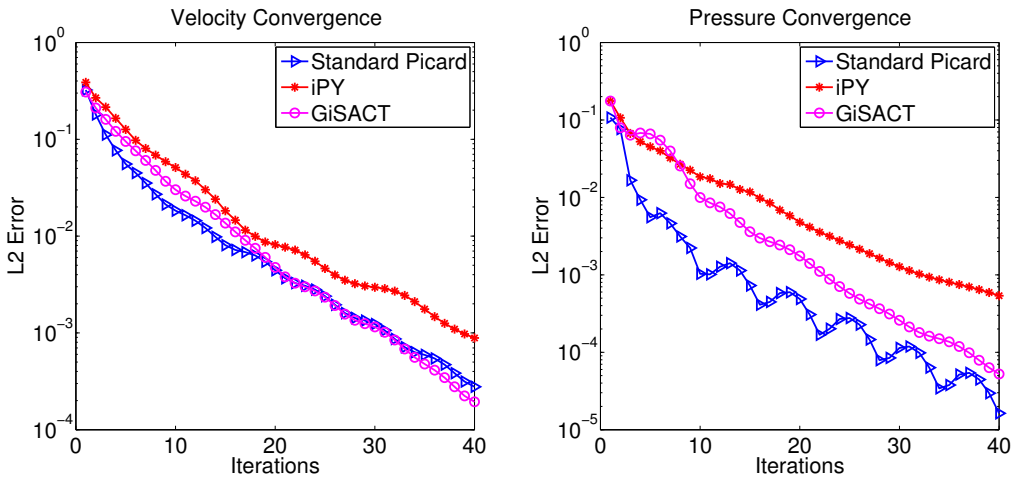

Figure 4.6: Aneurisk 32, $Q = 1.6005$ ml/s, $\nu = .04$ (Re=160): Convergence in L^2 norm for velocity (left), pressure (right).

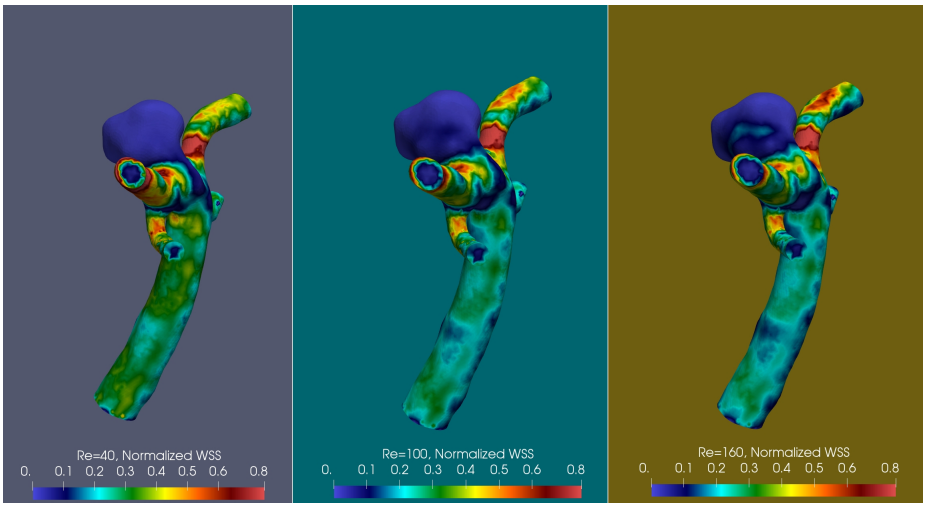

Figure 4.7: Aneurisk 32 normalized WSS for Re=40 (left), Re=100 (center), Re=160 (right) ($\nu = .04$ g-cm/s for each, Q varying)

3D Brain Aneurysm: Iterative Inner Solves

In many applications, problems may become too large to use direct methods, even for the inner solves. In these instances we must also use iterative methods to solve the A_ξ and $K_\xi = K + \xi D$ blocks. If solved to a sufficiently high tolerance level, we do not expect iteratively solved inner blocks to have an significant im-

pact on the number of outer Krylov iterations. However, note that the majority of inner solves for the splitting schemes are for the system $K_{\tilde{\xi}}$, while for GMRES preconditioned with 4.2.1, one solves $A_{\tilde{\xi}}$. The lack of a convective term makes $K_{\tilde{\xi}}$ considerably easier to precondition and solve iteratively than $A_{\tilde{\xi}}$, especially for higher Reynolds numbers. Thus, for the splitting schemes the majority of the cost is passed to an easier system. We therefore expect IPY and GISACT to maintain or improve their advantages in cost per iteration over our implementation of standard Picard when iterative inner solves are used.

We verify this by running the case Re=40 and both cases of Re=100 solving inner blocks with iterative methods. We will solve $A_{\tilde{\xi}}$ and $K_{\tilde{\xi}}$ using GMRES preconditioned with the block-triangular preconditioner proposed in [13]. For the $A_{\tilde{\xi}}$ solves in (4.2.1), we set the stopping tolerance to 1e-3. This tolerance was chosen as our tests showed that higher tolerances lead to an excessive amount of outer solves and more restrictive tolerances lead to more inner solves without a correspondingly significant decrease in total outer solves. We note that replacing $A_{\tilde{\xi}}$ in 4.2.1 with the block triangular approximation was not effective. For the $A_{\tilde{\xi}}$ and inner SC ($K_{\tilde{\xi}}$) solves in the splitting schemes, we set the stopping tolerance to 1e-6. We again use $\tilde{\xi} = 1$. The results are reported in Table 4.5. Here 'inner Krylov iterations' refers to the average number of GMRES iterations required to solve the inner blocks.

The key takeaway here is to observe that, although we are now solving the inner blocks iteratively, both the outer Krylov iteration counts and the number of nonlinear iterations are unaffected for GISACT and IPY. Recall that the number of inner solves are somewhat misleading, as for GMRES we solve $A_{\tilde{\xi}}$ with a stopping tolerance of 1e-3 and for IPY and GISACT we solve $K_{\tilde{\xi}}$ to a tolerance of 1e-6; $K_{\tilde{\xi}}$ is in fact easier to solve. We again find significant savings for IPY and GISACT, with the required time between 28% and 44% of the time for standard Picard in

3D BSA Aneurysm, 224,180 DOF: $Q = .4005$ ml/s, $\nu = .04$ g/cm-s (Re=40). Iterative inner blocks				
Method	Nonlinear Iter.	Outer Krylov Iter.	Inner Krylov Iter.	Avg solve time
IPY	12	32	68 (115 for $A_{\tilde{\xi}}$ solves)	741.7s
GISACT	8	32	68 (115 for $A_{\tilde{\xi}}$ solve)	760.6s
GMRES	7	99	57	1659.6s
3D BSA Aneurysm, 224,180 DOF: $Q = .4005$ ml/s, $\nu = .016$ g/cm-s (Re=100). Iterative inner blocks				
Method	Nonlinear Iter.	Outer Krylov Iter.	Inner Krylov Iter.	Avg solve time
IPY	15	28	86 (196 for $A_{\tilde{\xi}}$ solves)	921.3
GISACT	13	28	91 (194 for $A_{\tilde{\xi}}$ solve)	955.4
GMRES	11	93	95	2995.3s
3D BSA Aneurysm, 224,180 DOF: $Q = 1.005$ ml/s, $\nu = .04$ g/cm-s (Re=100). Iterative inner blocks				
Method	Nonlinear Iter.	Outer Krylov Iter.	Inner Krylov Iter.	Avg solve time
IPY	19	32	67 (137 for $A_{\tilde{\xi}}$ solves)	758.3s
GISACT	15	32	67 (135 for $A_{\tilde{\xi}}$ solve)	744.7s
GMRES	11	133	71	2706.1s

Table 4.5: Results for Aneursym test with inner blocks solved iteratively.

each case. The savings in this case are even more dramatic than those observed when solving the inner blocks directly.

The iterative solution of both $A_{\tilde{\xi}}$ and $K_{\tilde{\xi}}$ appears to depend on ν, with lower ν making these systems more difficult to solve. However, this affects standard Picard just as severely as it does GISACT and IPY. On the other hand, increases in Q appear to make the solution of $A_{\tilde{\xi}}$ more costly, but do not affect $K_{\tilde{\xi}}$, further evidence of the splitting schemes' superior robustness with respect to changes in this parameter.

Overall, this test provides compelling evidence that the IPY and GISACT schemes may provide significant savings over standard monolithic solving methods on nontrivial three-dimensional problems. This advantage is particularly pronounced for cases in which the inner blocks are solved iteratively or when the problem is convection-dominated with a relatively high ν. We do again observe slightly slower nonlinear convergence for IPY compared to standard Picard; however the savings in cost per iteration generally offset this. GISACT, on the other hand, does not appear to converge more slowly than standard Picard, and given its savings at each iteration, clearly outperforms standard Picard for all cases on this test.

4.3 Newton-type Scheme

As mentioned before, it is also possible to define a Newton-type version of Algorithms 4.1.1 and 4.1.2. We now present and study a higher order of version of Algorithm 4.1.1, which is defined as follows. Note that the corresponding analysis for the higher-order version of 4.1.2 is similar.

Algorithm 4.3.1 *The higher order algebraic Chorin-Temam iteration for the steady Navier-Stokes is given by:*

Step 1: Guess $\boldsymbol{u}_0 \in V_h$, $p_0 \in Q_h$.
Step k consists of the following 4 steps:

k.1 Find $\boldsymbol{z}^{k+1} \in V_h$ *satisfying for all* $\boldsymbol{v} \in V_h$,

$$\begin{aligned} &g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{u}^k, \boldsymbol{z}^{k+1}, \boldsymbol{v}) + c^*(\boldsymbol{z}^{k+1}, \boldsymbol{u}^k, \boldsymbol{v}) + a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}) \\ &\quad = (\boldsymbol{f}, \boldsymbol{v}) + b(\boldsymbol{v}, p^k) + c^*(\boldsymbol{u}^k, \boldsymbol{u}^k, \boldsymbol{v}). \end{aligned}$$

k.2 Find $(\boldsymbol{w}^{k+1}, \delta_p^{k+1}) \in (V_h, Q_h)$ *satisfying for all* $(\boldsymbol{v}, q) \in (V_h, Q_h)$,

$$\begin{aligned} g_{div}(\boldsymbol{w}^{k+1}, \boldsymbol{v}) - b(\boldsymbol{v}, \delta_p^{k+1}) + a^*(\boldsymbol{w}^{k+1}, \boldsymbol{v}) &= 0, \\ b(\boldsymbol{w}^{k+1}, q) &= -b(\boldsymbol{z}^{k+1}, q). \end{aligned}$$

k.3 Find $\boldsymbol{u}^{k+1} \in V_h$ *satisfying for all* $\boldsymbol{v} \in V_h$,

$$g_{div}(\boldsymbol{u}^{k+1}, \boldsymbol{v}) + a^*(\boldsymbol{u}^{k+1}, \boldsymbol{v}) = b(\boldsymbol{v}, \delta_p^{k+1}) + g_{div}(\boldsymbol{z}^{k+1}, \boldsymbol{v}) + a^*(\boldsymbol{z}^{k+1}, \boldsymbol{v}),$$

k.4 Set $p^{k+1} = p^k + \delta_p^{k+1}$.

4.3.1 Analysis

In this section, we prove the convergence of Algorithm (4.3.1) to the unique solution of (4.1.2)-(4.1.3). We assume the following small data condition:

$$\widetilde{\eta} = (1+\epsilon)C_b\nu^{-2}\|f\|_{-1} < 1 \tag{4.3.1}$$

for some $\epsilon > 0$. Note this is more restrictive than the standard small data condition (2.3.15) used previously.

Theorem 4.3.1 *Let $\epsilon > 0$ and*

$$\widetilde{\eta} < \min\left\{1,\ (9+10\beta^{-2})^{-1},\ \nu^{-1}(1+\epsilon)^2\|f\|_{-1}^2\left(2(1+12\beta^{-2})+(8+80\beta^{-2})^{-1}\right)^{-1}\right\}.$$

Denote by $(\boldsymbol{u}, p)$ the solution of system (4.1.2)-(4.1.3)*, $(\boldsymbol{u}_0, p_0) \in (V_h, Q_h)$ the initial guess of Algorithm 4.3.1, and $(\boldsymbol{u}^k, p^k)$ the step k solution. Then if $\xi \geq \nu$ and $\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}_0)\|^2 + \xi^{-1}\|p-p_0\|^2 \leq \|\nabla\boldsymbol{u}\|^2$, the sequence $(\boldsymbol{u}^{k+1}, p^{k+1})$ converges to $(\boldsymbol{u}, p)$.*

Remark

Even though Newton's method converges quadratically, we would not expect Algorithm 4.3.1 to converge quadratically, since approximations are being made. From the proof, in particular (4.3.14), observe that if the pressure terms are small, then quadratic convergence of the velocity is recovered.

Proof. We begin the proof by giving one assumption that the sequence $\{\boldsymbol{u}-\boldsymbol{u}^k\}$ is bounded by $\min\{\nu^{-1/2}, \epsilon\|\nabla\boldsymbol{u}\|\}$ for all $k \in \mathbb{N}$, where ϵ is the same constant used in the definition of $\widetilde{\eta}$. Hence for all k, we have

$$\|\nabla\boldsymbol{u}^k\| \leq \|\nabla(\boldsymbol{u}-\boldsymbol{u}^k)\| + \|\nabla\boldsymbol{u}\| \leq (1+\epsilon)\nu^{-1}\|f\|_{-1}. \tag{4.3.2}$$

by using the triangle inequality and the upper bound (4.1.4) of true solution $\boldsymbol{u}$.

Also, we assume that

$$\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}^k)\|^2+\xi^{-1}\|p-p^k\|^2\leq\|\nabla\boldsymbol{u}\|^2, \tag{4.3.3}$$

and by proving that the sequence defined by $\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}^k)\|^2+\xi^{-1}\|p-p^k\|^2$ is decreasing, this will imply the condition at the next iteration.

Denote $\boldsymbol{e}_u^{k+1}:=\boldsymbol{u}-\boldsymbol{u}^{k+1}$ and $\boldsymbol{e}_z^{k+1}:=\boldsymbol{u}-\boldsymbol{z}^{k+1}$. Subtracting Step k.1 of algorithm 4.3.1 from the unique steady solution equation (4.1.2), we obtain for all $\boldsymbol{v}\in V_h$,

$$\begin{aligned}&\xi(\nabla\cdot\boldsymbol{e}_z^{k+1},\nabla\cdot\boldsymbol{v})+\nu(\nabla\boldsymbol{e}_z^{k+1},\nabla\boldsymbol{v})\\&\qquad=(p-p^k,\nabla\cdot\boldsymbol{v})-c^*(\boldsymbol{e}_u^k,\boldsymbol{e}_u^k,\boldsymbol{v})-c^*(\boldsymbol{e}_z^{k+1},\boldsymbol{u}^k,\boldsymbol{v})-c^*(\boldsymbol{u}^k,\boldsymbol{e}_z^{k+1},\boldsymbol{v}).\end{aligned}$$

Choosing $\boldsymbol{v}=\boldsymbol{e}_z^{k+1}$ vanishes the last term, giving:

$$\xi\|\nabla\cdot\boldsymbol{e}_z^{k+1}\|^2+\nu(1-\widetilde{\eta})\|\nabla\boldsymbol{e}_z^{k+1}\|^2\leq\xi^{-1}\|p-p^k\|^2+\frac{C_b^2}{\nu(1-\widetilde{\eta})}\|\nabla\boldsymbol{e}_u^k\|^4, \tag{4.3.4}$$

thanks to the Young's inequality, (4.3.2), the definition of $\widetilde{\eta}$.

Next, we give a bound of $\|p-p^{k+1}\|$. Begin by adding step k.1 and step k.2, and subtracting it from the unique steady solution equation (4.1.2), we obtain the error equation for all $\boldsymbol{v}\in V_h$,

$$\begin{aligned}&\xi(\nabla\cdot(\boldsymbol{z}^{k+1}+\boldsymbol{w}^{k+1}-\boldsymbol{u}),\nabla\cdot\boldsymbol{v})+\nu(\nabla(\boldsymbol{z}^{k+1}+\boldsymbol{w}^{k+1}-\boldsymbol{u}),\nabla\boldsymbol{w})\\&\qquad=(p^{k+1}-p,\nabla\cdot\boldsymbol{v})+c^*(\boldsymbol{e}_u^k,\boldsymbol{e}_u^k,\boldsymbol{v})+c^*(\boldsymbol{u}^k,\boldsymbol{e}_z^{k+1},\boldsymbol{v})+c^*(\boldsymbol{e}_z^{k+1},\boldsymbol{u}^k,\boldsymbol{v}).\end{aligned} \tag{4.3.5}$$

Letting $\boldsymbol{v} = \boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u} \in X_h$ gives $(p^{k+1} - p, \nabla \cdot (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})) = 0$, yielding the bound:

$$\begin{aligned} \xi \|\nabla \cdot (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\|^2 + \nu \|\nabla (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\|^2 \\ \leq 8\nu \widetilde{\eta}^2 \|\nabla \boldsymbol{e}_z^{k+1}\|^2 + 2C_b^2 \nu^{-1} \|\nabla \boldsymbol{e}_u^k\|^4. \end{aligned} \tag{4.3.6}$$

thanks to Young's inequality, (4.3.2), and the definition of $\widetilde{\eta}$.

Applying the inf-sup condition to (4.4.15) one finds:

$$\begin{aligned} \beta \|p^{k+1} - p\| \leq \xi \|\nabla \cdot (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\| + \nu \|\nabla (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\| \\ + C_b \|\nabla \boldsymbol{e}_u^k\|^2 + 2\nu \widetilde{\eta} \|\nabla \boldsymbol{e}_z^{k+1}\|. \end{aligned} \tag{4.3.7}$$

Squaring both sides, using that $\xi \geq \nu$, and reducing yields:

$$\begin{aligned} &\beta^2 \|p^{k+1} - p\|^2 \\ &\quad \leq 4\xi \Big(\xi \|\nabla \cdot (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\|^2 + \nu \|\nabla (\boldsymbol{z}^{k+1} + \boldsymbol{w}^{k+1} - \boldsymbol{u})\|^2 \\ &\qquad + \nu^{-1} C_b^2 \|\nabla \boldsymbol{e}_u^k\|^4 + 2\nu \widetilde{\eta}^2 \|\nabla \boldsymbol{e}_z^{k+1}\|^2 \Big). \end{aligned} \tag{4.3.8}$$

Using the bound (4.4.16) reduces this estimate to:

$$\|p - p^{k+1}\|^2 \leq 4\xi \beta^{-2} \left(10\nu \widetilde{\eta}^2 \|\nabla \boldsymbol{e}_z^{k+1}\|^2 + 3\nu^{-1} C_b^2 \|\nabla \boldsymbol{e}_u^k\|^4 \right). \tag{4.3.9}$$

Combining (4.4.13) and (4.3.9) and multliplying both sides by ξ^{-1} produces:

$$\begin{aligned} &\xi^{-1} \|p - p^{k+1}\|^2 \\ &\quad \leq 4\beta^{-2} \left(\frac{10\widetilde{\eta}^2}{1 - \widetilde{\eta}} \xi^{-1} \|p - p^k\|^2 + \nu^{-1} M^2 \left(3 + \frac{10\widetilde{\eta}^2}{(1 - \widetilde{\eta})^2} \right) \|\nabla \boldsymbol{e}_u^k\|^4 \right). \end{aligned} \tag{4.3.10}$$

Next, we use (4.3.10) and (4.4.13) to bound $\|e_u^{k+1}\|$. Adding step k.3 and step k.1 and then subtracting from (4.1.2), we have:

$$\begin{aligned}&\xi(\nabla\cdot\boldsymbol{e}_u^{k+1},\nabla\cdot\boldsymbol{v})+\nu(\nabla\boldsymbol{e}_u^{k+1},\nabla\boldsymbol{v})\\&\quad=(p-p^{k+1},\nabla\cdot\boldsymbol{v})-c^*(\boldsymbol{e}_u^k,\boldsymbol{e}_u^k,\boldsymbol{v})-c^*(\boldsymbol{e}_z^{k+1},\boldsymbol{u}^k,\boldsymbol{v})-c^*(\boldsymbol{u}^k,\boldsymbol{e}_z^k,\boldsymbol{v}).\end{aligned}\tag{4.3.11}$$

Choosing $\boldsymbol{v}=\boldsymbol{e}_k^u$ yields:

$$\begin{aligned}&\xi\|\nabla\cdot\boldsymbol{e}_u^{k+1}\|^2+\nu\|\nabla\boldsymbol{e}_u^{k+1}\|^2\\&\quad\le\xi^{-1}\|p-p^{k+1}\|^2+2\nu^{-1}C_b^2\|\nabla\boldsymbol{e}_u^k\|^4+8\widetilde{\eta}^2\nu\|\nabla\boldsymbol{e}_z^{k+1}\|^2,\end{aligned}\tag{4.3.12}$$

thanks to Young's inequality, (4.3.2) and the definition of $\widetilde{\eta}$.

Adding this bound with (4.3.10) gives:

$$\begin{aligned}&\xi^{-1}\|p-p^{k+1}\|^2+\nu\|\nabla\boldsymbol{e}_u^{k+1}\|^2\\&\quad\le\frac{8\widetilde{\eta}^2}{1-\widetilde{\eta}}(10\beta^{-2}+1)\xi^{-1}\|p-p^k\|^2\\&\qquad+\left(2\nu^{-2}C_b^2(12\beta^{-2}+1)+\frac{8\widetilde{\eta}^2C_b^2}{\nu^2(1-\widetilde{\eta})^2}(10\beta^{-2}+1)\right)\nu\|\nabla\boldsymbol{e}_u^k\|^4.\end{aligned}\tag{4.3.13}$$

Using the assumptions $\widetilde{\eta}<(9+10\beta^{-2})^{-1}$ and
$\tilde{\alpha}<\nu^{-1}(1+\epsilon)^2\|f\|_{-1}^2\left(2(1+12\beta^{-2})+(8+80\beta^{-2})^{-1}\right)^{-1}$,

$$\begin{aligned}&\xi^{-1}\|p-p^{k+1}\|^2+\nu\|\nabla\boldsymbol{e}_u^{k+1}\|^2\\&\le\widetilde{\eta}\xi^{-1}\|p-p^k\|^2+\left(2(12\beta^{-2}+1)+\frac{\widetilde{\eta}}{1-\widetilde{\eta}}\right)\frac{C_b^2}{\nu^3}\nu^2\|\nabla\boldsymbol{e}_u^k\|^4\\&\le\widetilde{\eta}\xi^{-1}\|p-p^{k-1}\|^2+\left(2(12\beta^{-2}+1)+(8+80\beta^{-2})^{-1}\right)\frac{\widetilde{\eta}^2\nu}{\|f\|_{-1}^2(1+\epsilon)^2}\nu^2\|\nabla\boldsymbol{e}_u^k\|^4\\&\le\widetilde{\eta}\left(\xi^{-1}\|p-p^k\|^2+\nu^2\|\nabla\boldsymbol{e}_u^k\|^4\right).\end{aligned}\tag{4.3.14}$$

By (4.3.3), $\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}^k)\|^2 \leq 1$. We have therefore proved that $\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}^k)\|^2 + \xi^{-1}\|p-p^k\|^2$ is a contractive sequence in k, and thus converges. Since the solution of the problem (4.1.2)-(4.1.3) is unique and bounded by the data, we have that the limit of Algorithm 4.3.1 converges to the solution of (4.1.2)-(4.1.3).

4.3.2 Numerical Test: 2D Bifurcation

In this section we verify the faster convergence of Newton's method by repeating the same 2D bifurcation test (case $\nu = .0133$) from Section 4.2.1 using the Newton versions of the Yosida (iNY) and ACT (GISACTN) type splitting schemes. We use the same solver setup as in that test, however we will change the stopping tolerance for step 2 of iNY/GISACTN and the outer GMRES tolerance for standard Newton from 1e-6 to 1e-12. This is to better visualize and monitor convergence at high levels of precision. We run the test over the values $\xi = 1, 5, 10$. We plot the results below in figure 4.8[6].

As the above figure shows, for $\xi = 1.0$ the Newton versions of our splitting schemes do not perform well, either converging at the same rate as standard Picard (iNY) or converging non-monotonically and more slowly (GISACTN). However, for the higher values of ξ, we begin to see the convergence speedup. The convergence is still linear (unlike Usual Newton), but with a much improved convergence rate compared to the corresponding Picard versions. This is not surprising since the scheme's linear error term (the pressure term) in the analysis is scaled by ξ^{-1}, so as ξ increases this term's effect is reduced. This test suggests that for problems with large pressure, or bad initial guesses at pressure, the Newton version may not perform much better than the Picard version unless ξ is taken relatively large. However, if the problems associated with large ξ are not a

[6]The reference solution was computed on the same mesh with standard Picard techniques until the difference in L^2 norms between consecutive velocity iterations fell below 1e-12. To ensure maximum accuracy for the reference solution, we used only sparse direct solvers.

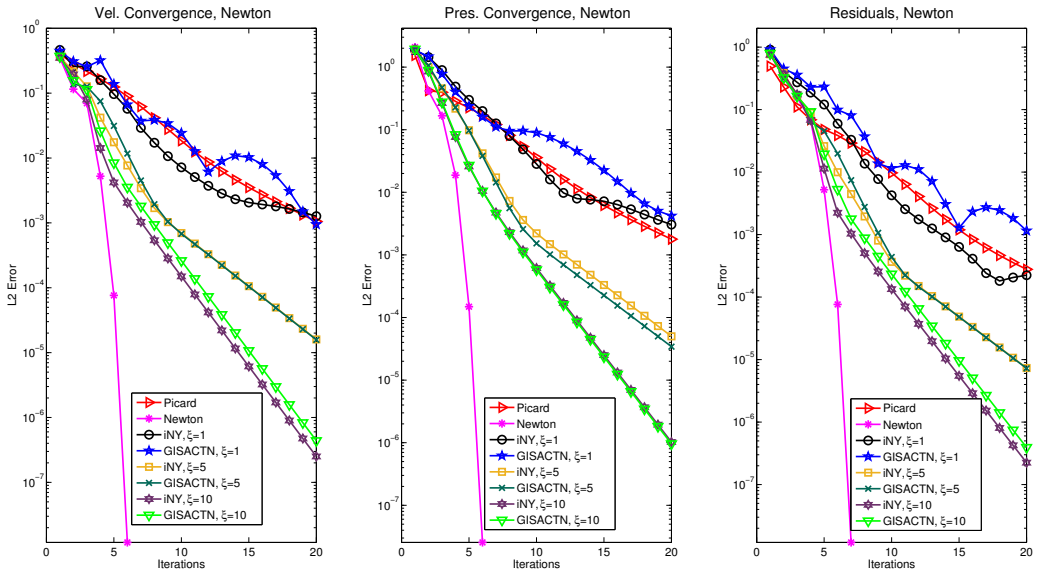

Figure 4.8: Convergence for Newton-type splitting schemes

concern (as may be the case when using direct solvers, for instance), then using the Newton version of these schemes is a possible way to accelerate convergence. We note also that step 2 is not different for the Newton and Picard versions of the scheme and hence the difference in numerical cost is only apparent in steps 1 (and step 3 for Yosida-type schemes), which comprise a comparatively minor portion of the overall cost.

Remark. In terms of cost per iteration, the behavior was similar to the Picard case shown earlier and hence we omit this information for brevity and to avoid redundancy.

4.4 Approximation of the Schur Complement

We can improve the numerical efficiency of the scheme by approximating the Schur Complement. For clarity in this section we will write K as νK and D as ξD to emphasize the matrices' dependence on these quantities. As proven in

[53], at the discrete level the matrix D has the following form where M_p is the pressure-mass matrix:

$$\xi D = \xi B^T M_p^{-1} B + \xi\, h\, R \tag{4.4.1}$$

So in the limit as h tends to zero only the $\xi B^T M_p^{-1} B$ term remains. We will make use of the following lemma:

Lemma 4.4.1 *$B(\nu K + \xi D)^{-1} B^T$ can be expressed as $B(\nu K + \xi B^T M_p B)^{-1} B^T + \mathcal{O}(h/\xi)$.*

Proof. Begin by recalling the following formula for two invertible matrices X and Y (see e.g. [54]):

$$(X+Y)^{-1} = X^{-1} - X^{-1}Y(I + X^{-1}Y)^{-1}X^{-1} \tag{4.4.2}$$

By (4.4.1):

$$(\nu K + \xi D)^{-1} = (\nu K + \xi B^T M_p^{-1} B + \xi\, hR)^{-1} \tag{4.4.3}$$

Applying formula (4.4.2) with $X = \nu K + \xi B^T M_p^{-1} B$ and $Y = \xi\, hR$:

$$\begin{aligned}(\nu K + \xi D)^{-1} &= (\nu K + \xi B^T M_p^{-1} B)^{-1} \\ &-h\xi(\nu K + \xi B^T M_p^{-1} B)^{-1} R \left(I + h\xi(\nu K + \xi B^T M_p^{-1} B)^{-1} R\right)^{-1} (\nu K + \xi B^T M_p^{-1} B)^{-1}\end{aligned} \tag{4.4.4}$$

Call the second term in (4.4.4) H. We will analyze the growth order of H with respect to the parameters ξ, ν and h. Clearly:

$$(\nu K + \xi B M_p^{-1} B)^{-1} \sim \mathcal{O}\left((\nu + \xi)^{-1}\right) \tag{4.4.5}$$

From (4.4.5) we then have:

$$\begin{aligned} H &\sim \mathcal{O}\left(h\xi(\nu+\xi)^{-1}\left(1+h\xi(\nu+\xi)^{-1}\right)^{-1}(\nu+\xi)^{-1}\right) \\ &\sim \mathcal{O}\left(\frac{h\xi}{(\nu+\xi)^2}\left(1+\frac{h\xi}{\nu+\xi}\right)^{-1}\right) \\ &\sim \mathcal{O}\left(\frac{h\xi}{(\nu+\xi)^2}\frac{\nu+\xi}{\nu+\xi+h\xi}\right) \\ &\sim \mathcal{O}\left(\frac{h\xi}{(\nu+\xi)(\nu+\xi+h\xi)}\right) \end{aligned} \tag{4.4.6}$$

We recall now that $0 < \nu < \xi$ by hypothesis and therefore we assume:

$$\mathcal{O}\left((\nu+\xi)^{-1}\right) \sim \mathcal{O}(\xi^{-1}) \tag{4.4.7}$$

And so the last line of (4.4.6) becomes:

$$\begin{aligned} H &\sim \mathcal{O}\left(\frac{h\xi}{\xi^2+h\xi^2}\right) \\ &\sim \mathcal{O}\left(\frac{h}{(1+h)\xi}\right) \\ &\sim \mathcal{O}(h/\xi) \end{aligned} \tag{4.4.8}$$

Left and right multiply (4.4.4) by B and B^T respectively to obtain:

$$B(\nu K + \xi B^T M_p^{-1} B)^{-1} B^T + BHB^T \tag{4.4.9}$$

which together with 4.4.8 gives:

$$B(\nu K + \xi B^T M_p^{-1} B)^{-1} B^T + \mathcal{O}(h/\xi) \tag{4.4.10}$$

completing the proof. This suggests that we must have $\xi >> h$ for $B(\nu K + \xi B^T M_p^{-1} B)^{-1} B^T$ to be an effective approximation of $B(\nu K + \xi D)^{-1} B^T$; for $\xi \sim$

$\mathcal{O}(h)$, this approximation becomes less reliable as the other term is no longer small. However, throughout this work we take $\xi \sim \mathcal{O}(1)$, in which case this is not an issue.

We will also use this identity (see [8]):

$$\left(B(\nu K + \xi B^T M_p^{-1} B)^{-1} B^T\right)^{-1} = \nu \left(BK^{-1}B^T\right)^{-1} + \xi M_p^{-1} \tag{4.4.11}$$

It is a well-known fact ([37, 8]) that the matrix $BK^{-1}B^T$ is spectrally equivalent to M_p. Based on this fact and the preceding two lemmas, we then propose to use the following approximation for the Schur complement:

$$\left(B(\nu K + \xi D)^{-1} B^T\right)^{-1} \approx (\nu + \xi) M_p^{-1} \tag{4.4.12}$$

where M_p is lumped so it is diagonal. This approximation replaces the inversion of the Schur complement system with a diagonal matrix inversion, which can be easily applied directly even for very large systems. Before we continue, let's first take a look at the new system by converting it back to finite element setup. Using the ACT algorithm with this approximation is equivalent to: find $\boldsymbol{z}^k \in V_h$ such that for any $\boldsymbol{v} \in V_h$:

$$a^*(\boldsymbol{z}^k, \boldsymbol{v}) + g_{div}(\boldsymbol{v}^k, \boldsymbol{v}) + c^*(\boldsymbol{u}^{k-1}, \boldsymbol{z}^k, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) + b(p^{k-1}, \boldsymbol{v}), \tag{4.4.13}$$

then find $(\boldsymbol{u}^k, p^k) \in V_h \times Q_h$ satisfying for all $(\boldsymbol{v}, q) \in V_h \times Q_h$:

$$(\nu + \xi)^{-1}(\delta_p^k, q) = -b(\boldsymbol{z}^k, q), \tag{4.4.14}$$

$$a^*(\boldsymbol{u}^k, \boldsymbol{v}) + g_{div}(\boldsymbol{u}^k, \boldsymbol{v}) = a(\boldsymbol{z}^k, \boldsymbol{v}) + g_{div}(\boldsymbol{z}^k, \boldsymbol{v}) + b(\delta_p^k, \boldsymbol{v}), \tag{4.4.15}$$

$$p^k = \delta_p^k + p^{k-1}. \tag{4.4.16}$$

Equations (4.4.13) - (4.4.16) are essentially the penalty projection methods shown in [77], suitably adapted for steady problems. Both have the same intermediate velocity solved by (4.4.13). As the projection step in [77] is heavily dependent on timestep, in our work we must adjust it as (4.4.14)-(4.4.15) so that u^k is divergence-free. Finally, we recover the pressure by (4.4.16).

Using this approach, computational costs are reduced. Our numerical tests confirm that this approach is efficient, though it does have some inherent limitations. Numerical tests and a rigorous error analysis will follow.

Remark. As mentioned also in the previous chapter, algebraic splitting schemes are readily adapted into preconditioners for the monolithic problem. The Schur Complement approximation (4.4.12) suggests that the following preconditioner may be effective:

$$P^{-1} = \left(\begin{bmatrix} \tilde{A}_{\xi} & 0 \\ B & -(\nu + \xi)^{-1} M_p \end{bmatrix} \begin{bmatrix} I & \tilde{K}^{-1} B^T \\ 0 & I \end{bmatrix} \right)^{-1} \tag{4.4.17}$$

This preconditioner is conceptually similar to the ones proposed by Heister and Rapin [53] (used in previous simulations) and Benzi and Olshanskii [8], as it uses the same approximation of the Schur complement. Indeed, this preconditioner could be regarded as a block-LU version of the aforementioned block triangular preconditioners. These types of block-LU preconditioners are known to be effective for the unsteady problem [123]. Further analysis and testing of this preconditioner is a possible subject of future work.

4.4.1 Analysis

In this section we rigorously analyze the approximation error incurred by the use of (4.4.12). Let $\tilde{S} := B(\nu K + \xi B^T M_p^{-1} B)^{-1} B^T$, $S := BK^{-1}B^T$, and recall that $B(K + \xi D)^{-1} B^T = \tilde{S} + \mathcal{O}(h/\xi)$.

We will proceed as follows: we will first show that $\|I - (\nu + \xi) M_p^{-1} \tilde{S}\|_N$ is bounded continuously by ξ in an appropriate norm $\|\cdot\|_N$. That result combined with (4.4.1) will imply that the approximation error is controlled by the mesh level h and the user controlled parameter ξ.

Recall that S and M_p are spectrally equivalent (see e.g. [8],[37]) implying that

$$0 < d_{min} < \sigma(S^{-1} M_p) < d_{max} < \infty \tag{4.4.18}$$

Where d_{min} and d_{max} are independent of the mesh size h. Fix $\epsilon > 0$. As the spectral radius of $\rho(S^{-1}M_p) < d_{max}$ there exists a matrix norm $\|\cdot\|_N$ such that:

$$\|S^{-1} M_p\|_N < \rho\left(S^{-1} M_p\right) + \epsilon < d_{max} + \epsilon < 2\, d_{max} \tag{4.4.19}$$

Theorem 4.4.1 *$\|I - (\nu + \xi) M_p^{-1} \tilde{S}\|_N$ is bounded by ν/ξ independently of h.*

Proof. Observe that:

$$I - (\nu + \xi) M_p^{-1} \tilde{S} = I - \left(\tilde{S}^{-1} \frac{1}{\nu + \xi} M_p\right)^{-1} \tag{4.4.20}$$

Applying (4.4.11):

$$I - (\nu + \xi)M_p^{-1}\tilde{S} = I - \left(\frac{\nu}{\nu+\xi}S^{-1}M_p + \frac{\xi}{\nu+\xi}I\right)^{-1} \tag{4.4.21}$$

$$= I - \left(\frac{\xi}{\nu+\xi}(I + \frac{\nu(\nu+\xi)}{(\nu+\xi)\xi}S^{-1}M_p)\right)^{-1} \tag{4.4.22}$$

$$= I - \frac{\nu+\xi}{\xi}\left(I + \frac{\nu}{\xi}S^{-1}M_p\right)^{-1} \tag{4.4.23}$$

Let ξ be such that:

$$\|\frac{\nu}{\xi}S^{-1}M_p\|_N \leq \frac{1}{2} \tag{4.4.24}$$

with $\|\cdot\|_N$ defined as (4.4.19). Note that the choice of $1/2$ here is simply for convenience and one could use any value less than one without loss of generality. By (4.4.24) the Neumann series converges:

$$\left(I + \frac{\nu}{\xi}S^{-1}M_p\right)^{-1} = \sum_{j=0}^{\infty}\left(-\frac{\nu}{\xi}S^{-1}M_p\right)^j \tag{4.4.25}$$

And therefore:

$$I - (\nu + \xi)M_p^{-1}\tilde{S} = I - \frac{\nu+\xi}{\xi}\sum_{j=0}^{\infty}\left(-\frac{\nu}{\xi}S^{-1}M_p\right)^j \tag{4.4.26}$$

$$= -\frac{\nu}{\xi}I - \sum_{j=1}^{\infty}\left(-\frac{\nu}{\xi}S^{-1}M_p\right)^j \tag{4.4.27}$$

$$= -\frac{\nu}{\xi}I - \frac{\nu}{\xi}S^{-1}M_p\sum_{j=0}^{\infty}(-1)^{j+1}\left(\frac{\nu}{\xi}S^{-1}M_p\right)^j \tag{4.4.28}$$

$$\tag{4.4.29}$$

We then take norms using the norm defined in (4.4.19):

$$\|I-(\nu+\xi)M_p^{-1}\tilde{S}\|_N = \|-\frac{\nu}{\xi}I-\frac{\nu}{\xi}S^{-1}M_p\sum_{j=0}^{\infty}(-1)^{j+1}\left(\frac{\nu}{\xi}S^{-1}M_p\right)^j\|_N \quad (4.4.30)$$

$$\leq \frac{\nu}{\xi}+\frac{\nu}{\xi}\|S^{-1}M_p\|_N\sum_{j=0}^{\infty}\|\frac{\nu}{\xi}S^{-1}M_p\|_N^j \quad (4.4.31)$$

$$\leq \frac{\nu}{\xi}(1+4\,d_{max}) \quad (4.4.32)$$

Noting that d_{max} is independent of h completes the proof.

This then immediately implies our main result:

Theorem 4.4.2 *The splitting error $\|I-(\nu+\xi)M_p^{-1}B(\nu K+\xi D)^{-1}B^T\|$ incurred by the approximation (4.4.12) is bounded by ν/ξ and h/ξ and tends to zero as $\xi\to\infty$.*

Proof. Applying the previous theorem and Lemma 4.4.1 gives:

$$\begin{aligned}&\|I-(\nu+\xi)M_p^{-1}B(\nu K+\xi D)^{-1}B^T\|\\ &\quad\leq \|I-(\nu+\xi)M_p^{-1}\tilde{S}\|+\|(\nu+\xi)M_p^{-1}\mathcal{O}(h/\xi)\| \quad (4.4.33)\\ &\quad\leq \frac{\nu}{\xi}C_1+\frac{h}{\xi}C_2\end{aligned}$$

Theorem 4.4.3 *The local splitting error at an iteration incurred by replacing $(B(\nu K+\xi D)^{-1}B^T)^{-1}$ with $(\nu+\xi)M_p^{-1}$ at an iteration k in the discrete version of (4.1.1) is bounded by ν/ξ and h.*

Proof. By direct inspection we may find that one step of the discrete problem given by (4.1.1) is equivalent to solving the following system in the block matrix A_F (letting $\tilde{K}=(\nu K+\xi D)$ and $\tilde{S}_F=B\tilde{K}^{-1}B^T$ for the sake of notation):

$$A_F=\begin{pmatrix}A & A\tilde{K}^{-1}B^T\\ B & 0\end{pmatrix}\begin{pmatrix}\boldsymbol{u}^k\\ \boldsymbol{\delta}_p^k\end{pmatrix}=\begin{pmatrix}f+B^Tp^{k-1}\\ 0\end{pmatrix} \quad (4.4.34)$$

Letting $\tilde{M}_p = (\nu + \xi)M_p$, the approximate inverse of A_F using (4.4.12) is given by:

$$A_{apx}^{-1} = \begin{pmatrix} (I - \tilde{K}^{-1}B^T\tilde{M}_p^{-1}B)A^{-1} & \tilde{K}^{-1}B^T\tilde{M}_p^{-1} \\ \tilde{M}_p^{-1}BA^{-1} & -\tilde{M}_p^{-1} \end{pmatrix} \tag{4.4.35}$$

Let $\boldsymbol{v}_F$ be the solution vector computed by solving the system (4.4.34) and $\boldsymbol{v}_{apx}$ the approximated solution computed by applying (4.4.35) to the same right hand side $\boldsymbol{b}$ given in (4.4.34). Then we define the splitting error vector $\boldsymbol{e}_s = [e_u, e_p]^T$ as:

$$\boldsymbol{e}_s = \boldsymbol{v}_F - \boldsymbol{v}_{apx} = \boldsymbol{v}_F - A_{apx}^{-1}\boldsymbol{b} = \boldsymbol{v}_F - A_{apx}^{-1}(A_F\boldsymbol{v}_F) = (I - A_{apx}^{-1}A_F)\boldsymbol{v}_F \tag{4.4.36}$$

Expanding this expression:

$$(I - A_{apx}^{-1}A_F)\boldsymbol{v}_F = \begin{pmatrix} 0 & \tilde{K}^{-1}B^T(I - \tilde{M}_p^{-1}\tilde{S}_F) \\ 0 & I - \tilde{M}_p^{-1}\tilde{S}_F \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_F \\ \boldsymbol{\delta}_{p,F} \end{pmatrix} \tag{4.4.37}$$

Note that neither the momentum nor mass equation is satisfied, unlike the full solution which conserves mass. By our bounds from part 1,

$$\|\boldsymbol{e}_s\| \leq \left(\frac{\nu}{\xi}C_1 + \frac{h}{\xi}C_2\right)\|\boldsymbol{\delta}_{p,F}\| \tag{4.4.38}$$

With a bound on the local splitting error, we can now prove a bound for the global splitting error.

Theorem 4.4.4 *Let $\boldsymbol{v}_{apx} = [\boldsymbol{u}_{apx}, p_{apx}]$ be the solution obtained from Picard iteration given by approximating the Schur complement solve with (4.4.12) and $\boldsymbol{v}_F = [\boldsymbol{u}_F, p_F]$ be the solution obtained from 4.1.1. Then:*

$$\|\boldsymbol{v}_{apx} - \boldsymbol{v}_{ex}\| \leq C_1\frac{\nu}{\xi} + C_2\frac{h}{\xi}$$

for suitable ξ and h.

Proof. We will proceed by induction. Starting with the same initial guess $\boldsymbol{v}^0 = (\boldsymbol{u}_0, p_0)$, the local splitting error derived in the previous theorem implies:

$$\|\boldsymbol{e}_s^1\| < \left(\frac{\nu}{\tilde{\xi}} C_1 + \frac{h}{\tilde{\xi}} C_2\right) \|\boldsymbol{\delta}_{p,F}^1\|$$

Define $\epsilon := \frac{\nu}{\tilde{\xi}} C_1 + \frac{h}{\tilde{\xi}} C_2$. The above bound implies the existence of a vector $\boldsymbol{\eta}$ such that $\boldsymbol{v}_{apx}^1 := \boldsymbol{v}_F^1 + \tilde{\epsilon}\,\boldsymbol{\eta}$. Similarly, there exists a matrix Δ such that $A_{apx}^2 := A_F^2 + \tilde{\epsilon}\,\Delta$. Note that the $\tilde{\epsilon}$ are both $\mathcal{O}(\epsilon)$ but possibly distinct.

Now assume that $\|\boldsymbol{e}_s^{k-1}\| \sim \mathcal{O}(\epsilon)$ and that $\boldsymbol{v}_{F,k-1}$ is bounded. Then we again have $\boldsymbol{v}_{apx}^{k-1} := \boldsymbol{v}_F^{k-1} + \tilde{\epsilon}\,\boldsymbol{\eta}$ for some $\boldsymbol{\eta}$ and $A_{apx}^k := A_F^k + \tilde{\epsilon}\,\Delta$ for some Δ. We seek to show that $\|\boldsymbol{e}_s^k\| \sim \mathcal{O}(\epsilon)$ (note that the base case holds for iteration $\|\boldsymbol{e}_s^1\|$). This establishes that $\boldsymbol{v}_{apx}^k$ is always in an ϵ-neighborhood of $\boldsymbol{v}_F^k$ for each k, implying that as $k \to \infty$, $\boldsymbol{v}_{apx}$ converges to a solution within an ϵ neighborhood of $\boldsymbol{v}_F$.

$$A_F^k\,\boldsymbol{v}_F^k = \boldsymbol{r}_F^{k-1} \tag{4.4.39}$$

$$(A_F^k + \tilde{\epsilon}\,\Delta)\boldsymbol{v}_{apx}^k = \boldsymbol{r}_F^{k-1} + \tilde{\epsilon}\,\tilde{\boldsymbol{\eta}} \tag{4.4.40}$$

As the right-hand side depends on the solution at the last iteration. Then:

$$\|\boldsymbol{v}_F^k - \boldsymbol{v}_{apx}^k\| = \left\| \left((A_F^k)^{-1} - (A_F^k + \tilde{\epsilon}\,\Delta)^{-1}\right) \boldsymbol{r}_F^{k-1} - (A_F^k + \tilde{\epsilon}\,\Delta)^{-1}\tilde{\epsilon}\,\tilde{\boldsymbol{\eta}} \right\| \tag{4.4.41}$$

$$\leq \left\| (A_F^k)^{-1} - (A_F^k + \tilde{\epsilon}\,\Delta)^{-1} \right\| \left\| \boldsymbol{r}_F^{k-1} \right\| + \tilde{\epsilon} \left\| (A_F^k + \tilde{\epsilon}\,\Delta)^{-1} \right\| \|\boldsymbol{\eta}\| \tag{4.4.42}$$

$$\leq \tilde{\epsilon} \left(\|\boldsymbol{r}_F^{k-1}\| + \|(A_F^k + \tilde{\epsilon}\,\Delta)^{-1}\| \|\boldsymbol{\eta}\| \right) \tag{4.4.43}$$

Where the last line follows from the continuity of matrix inverses. This completes the proof.

Note that strictly speaking Theorem 4.4.4 does not imply the convergence of $(\boldsymbol{u}_{apx}, p_{apx})$ to $(\boldsymbol{u}_F, p_F)$, but only convergence to within a certain neighborhood. The (true) convergence of $(\boldsymbol{u}_F, p_F)$ to the solution $(\boldsymbol{u}, p)$ of (4.1.2-4.1.3) then implies that $(\boldsymbol{u}_{apx}, p_{apx})$ converges to within a certain neighborhood of $(\boldsymbol{u}, p)$ as well. This neighborhood can be made arbitrarily small for large ξ; however for any fixed ξ this only establishes convergence up to a fixed level of accuracy.

This theorem shows the practical limitations of this scheme. As stated previously, large ξ can make the problem difficult to solve. In practice, with $\xi \sim \mathcal{O}(1)$, the the bound predicted by Theorem 4.4.4 may be small due to low values of ν and h; however for problems with higher viscosity or relatively coarse meshes, it is possible that $\xi \sim \mathcal{O}(1)$ is not sufficiently large to make the splitting error suitably small, and taking ξ extremely large is not a viable option in general. While the substantial numerical savings offered by this approach make it potentially worthwile, we acknowledge it has significant limitations and may not be applicable for some problems. Nonetheless, in our tests we still found that it performed quite well. Although we did observe some evidence of limiting behavior likely arising from the global splitting error, the effect was small.

4.4.2 Numerical Test: Bifurcation Flow

We repeat the same 2-dimensional bifurcation test previously used to compare the solutions computed by Algorithm 4.1.1 (GISACT), and Algorithm 4.1.1 using the approximated Schur complement (4.4.12) (Apx. GISACT) to the reference solution from the standard Picard iterations up to a very high level of accuracy (1e-12). For the reference case, we solved the full saddle-point system with UMFPACK at each iteration. For ACT, we solve both velocity systems with UMFPACK (Step k.1 and Step k.2), and the Schur Complement with CG preconditioned by the lumped pressure mass-matrix for outer solve and UMFPACK for inner solve.

For Apx. GISACT, we use UMFPACK for the velocity solves (Step k.1 and Step k.2), and solve the Schur complement system by multiplying a diagonal matrix. For our comparison Picard method, we use the same solver configuration as used in section 4.2.1. We will test the method for $\xi = 1, 2, 3$.

We display the L^2 norm of velocity and pressure convergence, as well as the nonlinear residuals, in Figures 4.9 and 4.10. Note here that $\xi = 2$ for the plotted Picard and GISACT comparisons.

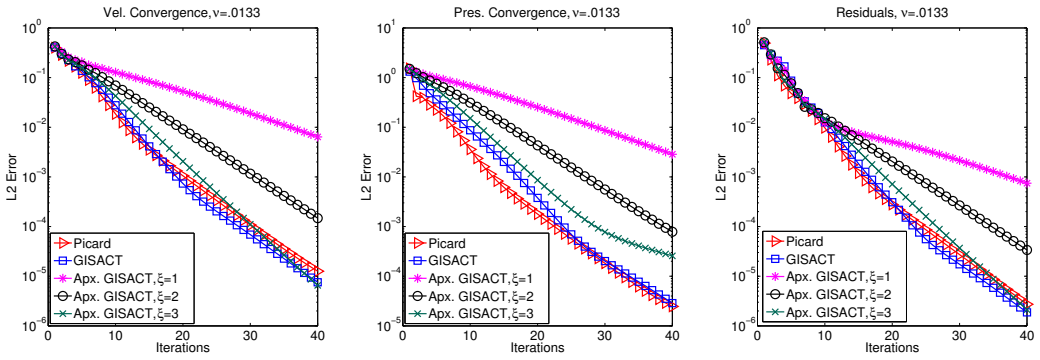

Figure 4.9: Bifurcation test case, $\nu = .0133$.

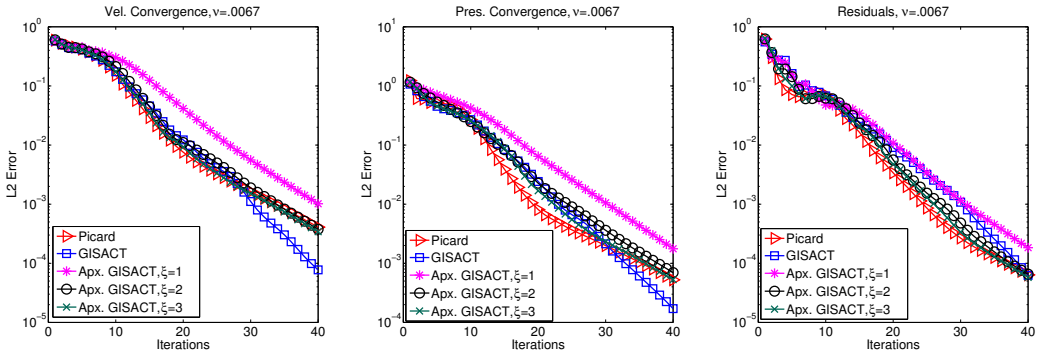

Figure 4.10: Bifurcation test case, $\nu = .0067$.

For $\nu = .0133$ (fig. 4.9), the approximated Schur Complement solution converges more slowly than GISACT with $\xi = 1$, as might be expected, but we clearly observe monotonic convergence to the desired solution. For $\xi = 2$ the convergence is still slower than full GISACT, but the gap narrows considerably,

while for $\xi = 3$ the convergence is the same as standard Picard and full GISACT in the velocity space. This is expected based on our bound (4.4.33), which suggests that large ξ implies a better approximation. In the pressure space for $\xi = 3$, we see the convergence start comparably to GISACT and Picard, but then flatten out. Recall from Theorem 4.4.4 that the approximate Schur complement scheme converges only to within a certain neighborhood, the size of which depends on ξ; we suspect this flattening is caused by the solution reaching the threshold of convergence. Unsurprisingly, the average solve time across all cases was .091s, a small fraction of the times shown in Table 4.1.

For $\nu = .0067$ (fig. 4.10), we notice that Apx. GISACT is still slower than GISACT for $\xi = 1$, however the difference is much less pronounced than for the case $\nu = .0133$. For $\xi = 2$ and $\xi = 3$, we see nearly identical convergence behavior for GISACT and Apx. GISACT. This is consistent with our expectation based on Theorem 4.4.4, as the quality of the approximation depends on ν/ξ and therefore as we decrease ν or increase ξ the approximated scheme should perform more similarly to GISACT.

As expected, the convergence of the approximate GISACT depends both on the grad-div parameter ξ and the viscosity parameter ν. For cases with higher ν, it appears one must use a relatively high value of ξ in order for the approximate scheme to converge at a similar rate; however as ν decreases the schemes behave more similarly, to where one must use a higher value of ξ anyway in order for GISACT to converge rapidly (as seen in section 4.2.1). In these instances, it appears the approximate scheme offers a similar convergence rate at a fraction of the cost. Although we only expect the approximate scheme to converge up to a certain level of accuracy, in this test this we found $\xi \sim \mathcal{O}(1)$ provided a small enough threshold for us to obtain accurate solutions.

4.4.3 Numerical Test: Newton-type Version

We now compare the convergence of the Newton and Picard formulations of our scheme and the effect of the approximate Schur complement. We run the same bifurcation test with $\nu = .0133$ as in section 4.3.1. We again compare the standard GISACTN scheme with the approximate GISACTN scheme for $\xi = 1, 5, 10$. We plot the results below (fig. 4.11) :

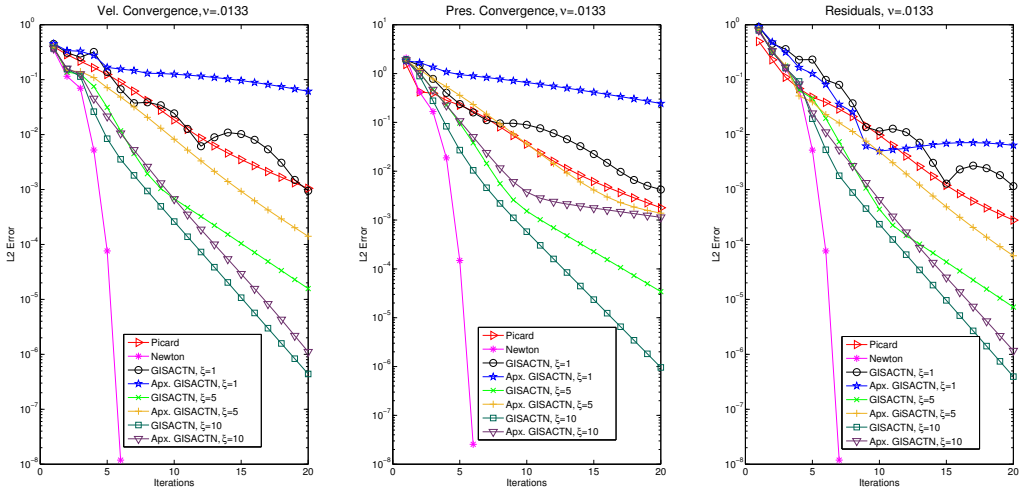

Figure 4.11: Bifurcation test, GISACTN and approximate GISACTN.

The first thing we note is that the approximate GISACTN algorithm does not converge as quickly as standard GISACTN for any value of ξ. For $\xi = 1$, approximate GISACTN does not seem to converge at all. We clearly see in the pressure plot that the approximate version GISACTN with $\xi = 10$ reaches its neighborhood of convergence around ten iterations, after which it does not converge further. This is the expected behavior based on Theorem 4.4.4. Interestingly, in this test this does not seem to happen in the velocity space.

4.5 Comparison of Algebraic Splitting Methods

Before closing the discussion of efficient solvers based on algebraic splitting, we would like to clarify a couple of important points regarding the schemes shown in Chapter 3 and Chapter 4. Both types of methods attempt to solve the same problem and, from a practical standpoint, are very similar in terms of implementation. However, the methods presented have somewhat different advantages and disadvantages.

The methods in Chapter 3 are based on the introduction of a semi-explicit term modulated by a parameter α. Generally, due to the explicit term, one must also introduce an underrelaxation parameter. Because of this, these schemes have the disadvantage of slower convergence compared to Chapter 4. However, the lack of grad-div stabilization and the introduction of α also provide some positive aspects. In particular, the weighted velocity block A_α is easier to solve and, compared to the grad-div schemes in chapter 4, the inner block K of the approximate Schur complement is also easier to manage.

In contrast, the methods in Chapter 4 do not require under-relaxation and do not suffer from delayed convergence. However, while grad-div stabilization has positive effects on accuracy and convergence, it does make the inner solves more difficult, with the difficulty increasing as ξ increases. While higher ξ is beneficial for the solution of the nonlinear problem, its effect on the conditioning of the inner blocks is negative.

If using direct solvers for the inner solves, the superior convergence properties and general positive impact of grad-div stabilization on solution accuracy make the methods in Chapter 4 preferable over those in Chapter 3. If the inner solves are a serious challenge, then one may prefer the methods from Chapter 3.

Importantly, although presented separately for ease of explanation, the methods shown in Chapter 3 and Chapter 4 are in no way mutually exclusive. It is

certainly possible for one to develop hybrid schemes employing techniques from both chapters. Such schemes may in fact allow one to retain advantages from both methods. One may, for instance, use a very small ξ ($\xi \sim \mathcal{O}(\nu)$) in order to improve the convergence of the outer solve of the approximate Schur complement in (3.1.13) without significantly increasing the difficulty of its inner solve. The theory of 4.1.1 or 4.1.2 does not allow for ξ this small, but when used in conjunction with an α-splitting scheme the approximate Schur complement is justified by the Neumann series argument. In general, these hybrid-type schemes will allow one to pick larger values of α and γ while using smaller values of ξ, allowing for potentially beneficial trade-offs. This is worthy of further investigation and may be an interesting area for additional research.

Chapter 5

Deconvolution-based Stabilization of the Steady Problem

[1]A major difficulty encountered when solving problems in CFD is the issue of stability. In practice, instabilities generating spurious numerical oscillations may be triggered by the presence of significant dynamics at small scales unresolved by the space discretization. In the case of the steady problem, the presence of these numerical oscillations can disrupt the steady solution. In the numerical approximation, this may delay or even prevent the convergence of the nonlinear iterative solver. While the theory of the problem suggests that one may refine the mesh in order to resolve these scales, in practice the amount of refinement may greatly increase the number of unknowns, causing the problem to become numerically intractable.

Small-scale dynamics are often triggered by high convective fields. In this case, a number of stabilization techniques exist, in particular Streamline Diffusion, GLS and SUPG-type methods (see e.g. [20, 59, 42, 58, 102, 99, 107] for

[1]This chapter is a modified version of the following work: A. Viguerie and A. Veneziani. "Deconvolution-based stabilization of the incompressible Navier-Stokes equations," in preparation.

details). The classical Streamline Diffusion (or Streamline Upwind) method is a multidimensional generalization of the well known Upwind scheme, where a numerical viscosity - vanishing generally with the space-discretization step - is added along the direction of the convective field. GLS and SUPG belong to the family of *strongly consistent methods*, where the additional viscosity is the result of a more sophisticated procedure that guarantees the consistency with the exact solution also for the discrete problem. The additional stabilization term is generally weighted by the convective term, so these techniques are very effective in high-convection regions. Strongly consistent schemes can be regarded an approximation of a general variational multiscale residual-based formulation of the fluid problem used in modeling the turbulence within the framework of Large Eddy Simulation models [6].

Within the context of hemodynamics, however, small scale dynamics can also be triggered by other phenomena besides high convective fields. Certain geometrical features can also induce this behavior, for example recirculation in regions induced by bends, obstructions, or rough boundaries. These small-scale behaviors may be physically meaningful or an entirely spurious numerical artifact. For example, stenoses caused by atherosclerotic plaques may cause recirculation that is also present in the physical problem. However, we may also observe these sorts of small-scale dynamics due to roughness in boundaries caused by the reconstruction of noisy data, in which case the disturbed flow is non-physical. In both these cases, even for relatively low convective fields, we may experience instabilities for which traditional methods are not always the optimal approach.

Consider for instance the 2D example of flow in a pipe with a sharp 90-degree curve. Here the velocity streamlines are pictured, colored according to the velocity magnitude as illustrated in Fig. 5.1 (inflow bottom-left, outflow top-right). The Reynolds number in this example is not exceedingly large (667), and the solution

still falls within the steady regime. Nonetheless, the presence of the recirculation region along the inner wall past the curve causes stability issues, making the simulation of this test case quite challenging, requiring both a very fine mesh and a large number of iterations to converge to a steady solution. In this case, the regions causing the instability are exactly the recirculation areas, which are not a high-convective region but rather a low-convection region. Therefore, stabilizing the high-convective regions as is done with SUPG or Streamline Diffusion will not effectively resolve the instabilities.

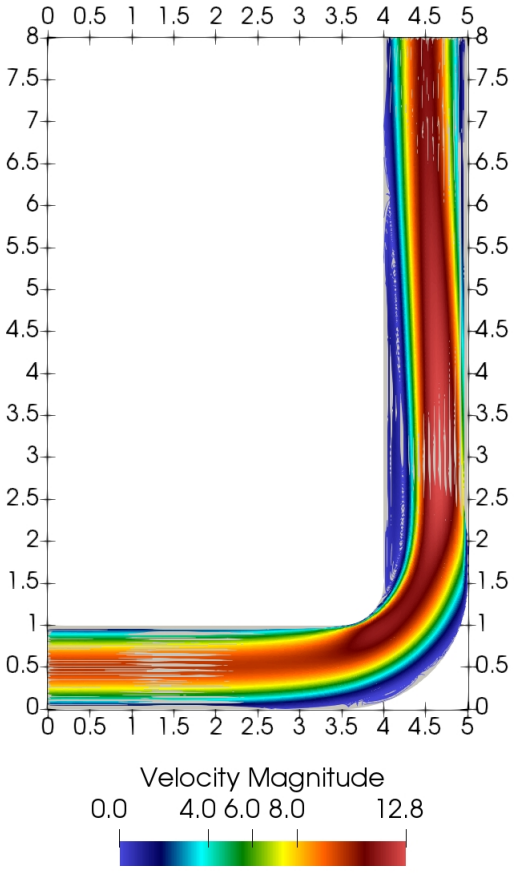

Figure 5.1: Example, illustrating instabilities arising from recirculation.

In this chapter, we present a modified version of the aforementioned stabilization techniques, where the additional numerical viscosity is not designed on the

strength and the direction of the convective field. We take inspiration from techniques developed for *Large Eddy Simulation* approaches to turbulence modeling in order present a different choice of the stabilization path. More specifically, the definition of the numerical viscosity is based on the deconvolution filter used for the indicator function in the LES modeling investigated in [72, 10]. This indicator function can be effective in detecting unresolved small-scale dynamics that need stabilization, even when they are not induced by high convective fields. While our stabilization method is inspired by approaches used in turbulence modeling, we note that the methods discussed here should not be regarded as turbulence models.

We will begin this chapter by reviewing the construction and basic properties of standard stabilization methods. We will then introduce and analyze our LES-inspired approach, first in a consistent framework analogous to the Streamline-Diffusion method, and then in a strongly consistent formulation. We will then demonstrate the efficacy of our approach in comparison with traditional stabilizations over a series of two and three dimensional test problems. In accordance with the rest of this work, our focus here is in on the Steady Navier-Stokes problem, but we also note that our findings in this section apply also to the unsteady problem.

5.1 Numerical stabilization

As the dynamics of fluids involve the interaction between large and small scales, the mesh for numerical discretization should be fine enough to resolve all scales of motion. The consequent computational cost can be prohibitively high. As an alternative to using a fine mesh, one may also employ numerical methods ("stabilizations") to obtain reasonable solutions on coarse meshes. Techniques

to stabilize (2.2.6) generally involve the addition of a new term to the velocity equation on each element K.

To pursue convergence, these terms must be *consistent* with the original problem; that is, the original problem is the limit of the discrete stabilized problem as $h \to 0$ where h is the representative measure of the mesh size. There are several well-known methods that follow this general procedure, and we will focus broadly on two classes of these methods: *Streamline-diffusion methods* and *strongly consistent methods*. The former are generally easier to implement, while the latter have the advantage of strong consistency with the original problem, leading to better convergence rates in general [20].

5.1.1 Classical Streamline-Diffusion Methods

For the purposes of generality, we will consider these methods as applied to the Oseen problem (2.2.14-2.2.15) for a given vector field $\boldsymbol{b}_h$. This class of methods are derived from approaches originally developed for advection-diffusion-reaction problems and involve the introduction of an additional term to the momentum equation of the form $-W_k h_K \left[(\boldsymbol{b}_h \cdot \nabla \boldsymbol{u}_h)\, \boldsymbol{b}_h\right]$. This term adds additional viscosity to the problem along $\boldsymbol{b}_h$. Here W_k is a user-defined scalar function and h_K is the function representing the size of the Kth mesh element. A popular example is

$$W_k \equiv \delta_K / \|\boldsymbol{b}_h\|_K, \tag{5.1.1}$$

where δ_K is in turn a user-defined scalar function (often but not necessarily constant) and $\|\boldsymbol{b}_h\|_K$ is the norm of $\boldsymbol{b}$ on the current element K. Another possible choice, recommended in [119] is:

$$W_k \equiv \frac{\delta_K}{\|\boldsymbol{b}_h\|_{L^2(\Omega)}} \frac{2h_K \|\boldsymbol{b}_h\|_{L^2(K)}}{1 + h_K \|\boldsymbol{b}_h\|_{L^2(K)}} \leq \frac{2\delta_k}{\|\boldsymbol{b}_h\|_{L^2(\Omega)}} \tag{5.1.2}$$

Ideally, this additional diffusivity helps one gain by adding numerical viscosity along the convective field, avoiding oscillations by minimizing crosswind numerical diffusion [58, 102, 119]. In the weak form, for a given W_k and mesh size h_K it adds the form $\tau(\cdot,\cdot,\cdot;W_k,h_K) : V_h \times V_h \times V_h \to \mathbb{R}$, s.t. [119, 102]

$$\tau(\boldsymbol{b}_h,\boldsymbol{v}_h,\boldsymbol{z}_h;W_k,h_K) \equiv \sum_{K\in\mathcal{T}} W_k h_K \int_K ((\boldsymbol{b}_h\cdot\nabla)\boldsymbol{v}_h)\cdot((\boldsymbol{b}_h\cdot\nabla)\boldsymbol{z}_h)\,. \tag{5.1.3}$$

Note that for a given reticulation $\mathcal{T}$ of a domain Ω, for a given $\boldsymbol{b}_h$, $W_k > 0$ and h_K, we have

$$\tau(\boldsymbol{b}_h,\boldsymbol{v}_h,\boldsymbol{v}_h;W_k,h_K) = \sum_{K\in\mathcal{T}} W_k h_K \int_K \|(\boldsymbol{b}_h\cdot\nabla)\boldsymbol{v}_h\|_{L^2}^2 \geq 0. \tag{5.1.4}$$

By standard arguments of functional analysis we also have

$$|\tau(\boldsymbol{b}_h,\boldsymbol{v}_h,\boldsymbol{u}_h;W_k,h_K)| \leq C\|\boldsymbol{b}_h\|_V^2\|\boldsymbol{v}_h\|_V\|\boldsymbol{z}_h\|_V. \tag{5.1.5}$$

The weak formulation of the stabilized problem reads: given $\boldsymbol{b}_h$ regular enough, find $\boldsymbol{u}_h \in V$ and $p_h \in Q$ s.t. for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$\begin{aligned} a(\boldsymbol{u}_h,\boldsymbol{v}_h) + c(\boldsymbol{b}_h,\boldsymbol{u}_h,\boldsymbol{v}_h) + b(\boldsymbol{v}_h,p_h) + \\ \tau(\boldsymbol{b}_h,\boldsymbol{u}_h,\boldsymbol{v}_h;W_k,h_K) + b(\boldsymbol{u}_h,q_h) - (\boldsymbol{f},\boldsymbol{v}_h) = 0. \end{aligned} \tag{5.1.6}$$

The well-posedness of this problem follows from the well-posedness of (2.2.6) under standard assumptions [45, 102, 20, 99, 58, 119] and (5.1.4), (5.1.5). Referring to (5.1.3), we note for each element K, the form τ depends on the element diameter h_K. As the mesh is refined $h_{MAX} \equiv \max_K(h_K) \to 0$, it follows that $\tau \to 0$ everywhere in $\mathcal{T}$. This establishes the consistency of these stabilization methods, as the numerical problem converges to the original one when the mesh size vanishes.

An iteration $k+1$ of the Picard iteration (2.2.10) for problem (2.2.1-2.2.6) one

sets $\boldsymbol{b} = \boldsymbol{u}_h^k$, the velocity field from the previous iteration to give the stabilized problem: Find $\boldsymbol{u}_h^{k+1} \in V$ and $p_h^{k+1} \in Q$ s.t. for any $\boldsymbol{v}_h \in V_h, q_h \in Q_h$:

$$\begin{aligned} a(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1}) + \\ \tau(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h; W_k, h_K) + b(\boldsymbol{u}_h^{k+1}, q_h) - (\boldsymbol{f}, \boldsymbol{v}_h) = 0 \end{aligned} \tag{5.1.7}$$

Intuitively, this method works by adding artificial viscosity only in the direction of the solution streamlines. It can be regarded as a generalization of the *upwinding methods* common in one-dimensional problems. For this reason it is also referred to as the *streamlined upwinding method* in the literature [20, 58].

Remark. Note that the functional form τ can be conveniently regarded as a quadrilinear form in general:

$$\tau(\boldsymbol{w}_h, \boldsymbol{v}_h, \boldsymbol{y}_h, \boldsymbol{z}_h; W_k, h_K) \equiv \sum_{K \in \mathcal{T}} W_k h_K \int_K ((\boldsymbol{w}_h \cdot \nabla)\boldsymbol{v}_h) \cdot ((\boldsymbol{y}_h \cdot \nabla)\boldsymbol{z}_h) \tag{5.1.8}$$

5.1.2 Strongly Consistent Methods

Another important class of stabilization techniques are the *strongly consistent methods* [59, 42, 99]. These techniques are consistent with the original problem for any size of the mesh discretization, unlike the methods discussed in the previous section (which are only consistent for $h_{MAX} \to 0$). Methods of this type work by the addition of the strong element-wise formulation of a residual term of $\mathcal{L}_h(\boldsymbol{u}_h, p_h; \boldsymbol{b}_h, \boldsymbol{f}, \boldsymbol{v}_h, q_h)$ to the problem (2.2.6), i.e. a stabilizing term such that

$$\mathcal{L}_h(\boldsymbol{u}, p; \boldsymbol{b}_h, \boldsymbol{f}, \boldsymbol{v}_h, q_h) = 0 \;\; \forall \, (\boldsymbol{v}_h, q_h) \;\in V_h \times Q_h \tag{5.1.9}$$

where $(\boldsymbol{u},\ p)$ is the exact solution to (2.2.6). Hence, the true solution satisfies the stabilized problem exactly and not just asymptotically, as in the previous case. Consequently, these methods are generally more accurate than the classi-

cal streamline-diffusion type methods described in the previous section [102]. In fact, convergence rate - in an appropriate norm - depends on the degree of the finite elements used, which is generally not true for classical Streamline Diffusion, as a consequence of a standard application of the Strang Lemma [99]. Classical Streamline Diffusion is often over-diffusive; while this can reduce the instabilities so to accelerate the convergence to a steady solution, it adversely affects the accuracy. Strongly consistent methods alleviate this problem [20, 58]. This comes at a cost, however, and these methods are often more difficult to formulate and implement, and in our experience may negatively affect the conditioning of the associated linear systems. Two common methods of this type are the *Streamlined Upwind Galerkin Method* (SUPG) and the *Galerkin Least-Squares Method* (GLS). For the sake of brevity, we limit here only to SUPG.

For the steady Navier-Stokes problem, we let $\mathcal{L}_h$ be defined as (setting $\boldsymbol{b}_h = \boldsymbol{u}_h$):

$$\begin{aligned} &\mathcal{L}_h(\boldsymbol{u}_h, p_h; \boldsymbol{u}_h, \boldsymbol{f}, \boldsymbol{v}_h, q_h) = \\ &\textstyle\sum_{K\in\mathcal{T}_h} \delta_K \left(-\nu\Delta\boldsymbol{u}_h + (\boldsymbol{u}_h\cdot\nabla)\,\boldsymbol{u}_h + \nabla p_h - \boldsymbol{f},\ \ (\boldsymbol{u}_h\cdot\nabla)\,\boldsymbol{v}_h + \nabla q_h\right)_{L^2(K)}. \end{aligned} \tag{5.1.10}$$

Here, δ_K is an element-wise constant parameter defined by the user and is generally dependent on the mesh size. The well-posedness of these and other similar formulations is shown in [107]. The strong consistency is easily observed by noting that if the above bilinear forms are evaluated with the exact solution $(\boldsymbol{u},\ p)$ then in both cases the above inner products will be zero for each K. The unsteady case is defined analogously. In addition to their general convective stabilizing properties, methods of this type enjoy the additional property of stabilizing non-LBB inf-sup compatible element pairs for velocity and pressure (for example, $\mathbb{P}^1/\mathbb{P}^1$) [107].

5.2 LES-inspired stabilization

Our novel approach is inspired by the *generalized Leray models* [76] for Large Eddy simulation explored in [72, 10]. These methods work by the construction of a *filtered velocity field* $\overline{\boldsymbol{u}}$ which identifies regions where the solution requires stabilization. In the context of turbulence modeling, the idea of this filtering step is the identification of relevant small scales, unresolved by the space discretization. In this way, by construction, $\overline{\boldsymbol{u}}$ is small in regions where $\boldsymbol{u}$ is "smooth" and large where $\boldsymbol{u}$ is "not smooth", indicating possible instabilities. The filtered velocity field determines where the fluid velocity can be simulated directly and where small scales should be explicitly modeled.

We borrow from this idea to develop our stabilization technique. We focus particularly on the steady problem here, even though the idea of 'large eddy simulation' is inherently unsteady. The application of such techniques to steady problems represents, in fact, a novel approach.

Given an appropriate filter function $F\boldsymbol{u}$ and an available velocity field $\boldsymbol{u}^k$ we can apply the following algorithm at an iteration $k+1$ to promptly obtain a modified stabilization scheme:

1. Compute $\overline{\boldsymbol{u}}^{k+\frac{1}{2}} = F\boldsymbol{u}^k$;

2. Solve (5.1.6) with a stabilization term guided by the filtered field $\overline{\boldsymbol{u}}^{k+\frac{1}{2}}$;

3. Check Convergence: end if criterion met; else update and loop.

A major difference between the approach discussed in this work and the existing methods for LES is that we use the filtered field to construct an alternative stabilization, not to activate turbulence modeling. While in the existing literature, it is used in conjunction with a differential filter to obtain a regularized convective field for the current time step or as part of an evolution/relaxation scheme [10]

we use the indicator function directly in a streamline diffusion-style manner as in (5.1.6).

To motivate our deconvolution-based approach, we return to the same two-dimensional example shown in the introduction. Observe the plot of $\overline{\boldsymbol{u}}$ for the same problem, obtained by applying a nonlinear deconvolution-based filter to $\boldsymbol{u}$ function similar to those described in [15, 10], as we detail hereafter - see Fig. 5.2, right panel.

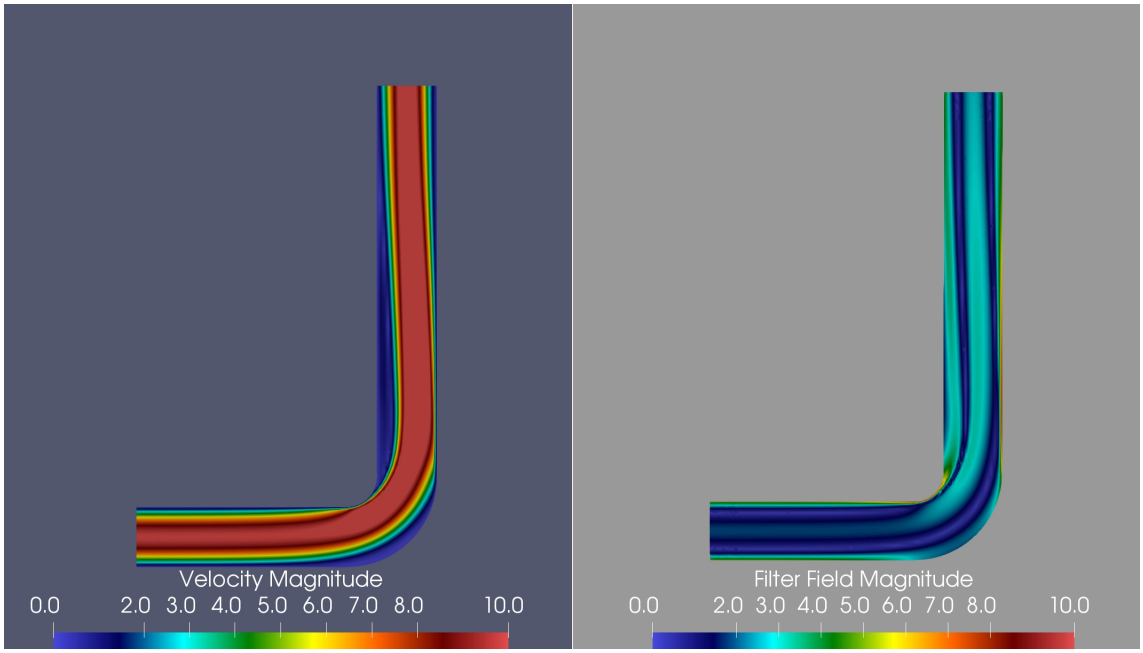

Figure 5.2: A simplified example of small-scale dynamics induced by the geometry of a domain with a relatively low Reynolds number. Left: velocity field $\boldsymbol{u}$ in a bending pipe. Right: filtered field $\overline{\boldsymbol{u}} = F\boldsymbol{u}$ with F specified in the text.

The figure illustrates why stabilizing along the vector field $\overline{\boldsymbol{u}}$ is preferable to the traditional streamline-diffusion scheme; the recirculation region is no longer neglected. Thus, when we add numerical viscosity along $\overline{\boldsymbol{u}}$, we are now adding it to the small-scale and then potentially unstable area, which does not happen when using traditional streamline-based techniques.

Nonlinear filtering techniques are widely used in Large Eddy simulation, and

many filters have been investigated in the literature. These filters may be physics-based, such as those discussed in [72, 15] or may arise from mathematical arguments, as in [12, 10]. In this work we focus our attention on *deconvolution-based nonlinear filters*, which belong to this latter group [73]. This class of indicator functions has an elegant mathematical formulation and demonstrates good performance properties; however generally the use of these indicator functions entails some extra numerical cost when compared to other types of indicator functions. Let V be a Hilbert space and $L : V \to V$ be a linear, invertible, self-adjoint, compact operator. By the spectral theorem (see e.g. [109]), we have:

$$L\boldsymbol{u} = \sum_{i=0}^{\infty} \lambda_i \langle \boldsymbol{u}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i, \quad L^{-1}\boldsymbol{u} = \sum_{i=0}^{\infty} \frac{1}{\lambda_i} \langle \boldsymbol{u}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i \tag{5.2.1}$$

where $\{\boldsymbol{e}_i\}_{i=0}^{\infty}$ are eigenfunctions of F and form an orthonormal basis for V. Since F is compact, it follows that F^{-1} is unbounded. Let D be a bounded, finite-dimensional approximation of L^{-1}, defined as:

$$D = \sum_{i=0}^{N} \frac{1}{\lambda_i} \langle \boldsymbol{u}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i \tag{5.2.2}$$

for some integer N. In regions where $\boldsymbol{u}$ is smooth, we expect that $\langle \boldsymbol{u}, \boldsymbol{e}_i \rangle$ is only significant for small values of i. We may then define the indicator function in the following way:

$$F\boldsymbol{u} = \boldsymbol{u} - D(L\boldsymbol{u})$$

We expect $F\boldsymbol{u}$ to be 'small' in the regions where $\boldsymbol{u}$ is smooth and 'large' where $\boldsymbol{u}$ is not smooth (ie, where $\boldsymbol{u}$ requires regularization).

A popular choice for D is the *Van Cittert deconvolution operator*, given by:

$$D_N = \sum_{n=0}^{N} (I - L)^n \tag{5.2.3}$$

N is typically small, and in this work we will use $N = 0$, so that

$$F_{D_0} = \boldsymbol{u} - L(\boldsymbol{u}) \tag{5.2.4}$$

A possible choice for L is the linear *Helmholtz filter operator* L_H defined as:

$$L_H = \left(I - \mu^2 \Delta\right)^{-1} \tag{5.2.5}$$

where $\mu > 0$ is the *filtering radius* and Δ is the Laplacian: $\Delta = \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2}$ where $d = 2, 3$ is the number of spatial dimensions. From now on, we use this definition of $F \equiv (I - L_H)\boldsymbol{u}$.

We must also address the prescription of boundary conditions associated with the nonlinear filter L_H. As the scope of filtering is not the same of LES modeling, we significantly modify the original choice. Setting $L(\boldsymbol{u}) = \boldsymbol{u}$ everywhere on the boundary does not yield optimal results. In fact, the auxiliary field obtained by the filtering is intended to identify regions that may be in proximity of the boundary. As the essential no-slip condition on $\boldsymbol{u}$ limits the variability of $L(\boldsymbol{u})$ near the walls, this is not ideal for our purposes. The recirculation regions generally occur near walls and therefore we want to allow $L(\boldsymbol{u})$ to be possibly large in these regions. We opted for homogeneous Neumann conditions here, as they give the best results. To take advantage from the Poincaré inequality, which is helpful for the analysis, we then let $L(\boldsymbol{u}) = \boldsymbol{u}$ on the outflows.

We then have a sort of rule-of-thumb for the selection of the boundary conditions for the filter operator, in which we set natural conditions where the physical filed is subject to Dirichlet conditions and vice versa. This is a purely numerical choice, and we do not ascribe to it any physical meaning. We stress that $\overline{\boldsymbol{u}}$ has significance only as an auxiliary numerical tool. As the name "filtered velocity"

can be misleading and may imply a physical interpretation, we will instead use the term "*filtered field*".

Remark 5.2.1 *Some properties of the filter operator are immediately deduced once we establish the link between the filtering operator F and the Yosida regularization of the Laplace operator $-\Delta$ associated with the boundary conditions specified above. Following [18] and Chap. 7 of [19], L_H is the so-called resolvent of $-\Delta$ and $\mu^{-2}F$ its Yosida regularization. Notice that the domain of the resolvent is $L^2(\Omega)$ and that the norm of the resolvent as linear and continuous operator on L^2 is ≤ 1.*
From the general theory, the following properties follow for any $\mu \neq 0$:

(a_1) $\mu^{-2}F = -\Delta(L_H)$ *for any field in* $L^2(\Omega)$*;*

(a_2) $\mu^{-2}F = -L_H(\Delta)$ *for any field in* $H^1(\Omega)$*;*

(b) $\|\mu^{-2}F\boldsymbol{v}\|_{L^2} \leq \| - \Delta\boldsymbol{v}\|_{L^2}$ *for any velocity field* $\boldsymbol{v}$ *in* H^1*;*

(c) $\lim_{\mu\to 0} L_H\boldsymbol{v} = \boldsymbol{v}$ *for any field in* L^2*;*

(d) $\lim_{\mu\to 0} \|F\boldsymbol{v} - (-\mu^2\Delta)\|_{L^2} = 0$ *for any velocity field* $\boldsymbol{v}$ *in* H^1*;*

(e) $(F\boldsymbol{v}, \boldsymbol{v}) \geq 0$ *for any field* $\boldsymbol{v}$ *in* L^2*;*

(f) $\|F\boldsymbol{v}\|_{L^2} \leq \|\boldsymbol{v}\|_{L^2}$ *for any field* $\boldsymbol{v}$ *in* L^2*.*

These properties emphasize that the filtered field pursues a non-negative (dissipative) action (property (e)), with a regularizing action milder that the one of the Laplace operator ((b)).

Theorem 5.2.2 *For a divergence-free vector field $\boldsymbol{u}$, the filtered field $\overline{\boldsymbol{u}} = F\boldsymbol{u}$ is also divergence free.*

Proof. The proof follows from property (a_2) listed above and the commutativity of the Laplace and divergence operators. In fact, we have

$$L_H^{-1} F\boldsymbol{u} = -\mu^2(\Delta)\boldsymbol{u}$$

so that if $\nabla \cdot \boldsymbol{u} = 0$,

$$\nabla \cdot \left(L_H^{-1} F\boldsymbol{u} \right) = 0.$$

Recalling that $L_H^{-1} = I - \mu^2\Delta$, then

$$\nabla \cdot \left(L_H^{-1} F\boldsymbol{u} \right) = L_H^{-1} \left(\nabla \cdot F\boldsymbol{u} \right) = 0$$

from which the theorem follows as the Helmholtz operator is invertible.

Remark. In general, at the discrete level, the divergence of the velocity $\boldsymbol{u}_h$ is only weakly free, i.e.

$$\int_\Omega \nabla \cdot \boldsymbol{u}_h \, q_h = 0 \quad \forall q \in Q_h.$$

Unfortunately, the finite element pair of spaces V_h, Q_h does not guarantee that a weakly divergence-free field is also strongly divergence-free. In general, the divergence of the filtered field will be nonzero. However, we notice that $\forall q \in Q_h$

$$\int_\Omega \nabla \cdot F\boldsymbol{u}_h \, q_h = \int_\Omega \nabla \cdot \boldsymbol{u}_h \, q_h - \mu^2 \int_\Omega \nabla \cdot L_H \boldsymbol{u}_h \, q_h = -\mu^2 \int_\Omega \nabla \cdot L_H \boldsymbol{u}_h \, q_h.$$

So, the weak divergence scales with μ^2.

Theorem 5.2.3 *For a given velocity field* $\boldsymbol{u}$ *in* $H^2(\Omega) \cap H_0^1(\Omega)$, $\|F\boldsymbol{u}\|_V \leq \|\boldsymbol{u}\|_V$.

Proof. From the definition of F,

$$F\boldsymbol{u} = \boldsymbol{u} - \left(I - \mu^2\Delta \right)^{-1} \boldsymbol{u} \tag{5.2.6}$$

Let $\boldsymbol{w}$ in $H^2(\Omega)$ be defined such that:

$$\boldsymbol{w} = \left(I - \mu^2 \Delta\right)^{-1} \boldsymbol{u} \tag{5.2.7}$$

Trivially:

$$\begin{aligned} \left(I - \mu^2 \Delta\right) \boldsymbol{w} &= \boldsymbol{u} \\ \Delta \boldsymbol{w} &= \frac{1}{\mu^2}(\boldsymbol{w} - \boldsymbol{u}) \\ &= -\frac{1}{\mu^2} F\boldsymbol{u} \end{aligned} \tag{5.2.8}$$

From (5.2.6), (5.2.7), and (5.2.8) we have have that $F\boldsymbol{u}$ solves the following variational problem: Find $F\boldsymbol{u}$ in $H^2(\Omega)$ such that for all $\boldsymbol{v}$ in $H^2(\Omega)$,

$$-\int_\Omega F\boldsymbol{u} \cdot \boldsymbol{v} = -\int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega \boldsymbol{w} \cdot \boldsymbol{v} \tag{5.2.9}$$

Let $\boldsymbol{v} = \Delta F\boldsymbol{u}$. Integration by parts[2] gives:

$$\begin{aligned} \|\nabla F\boldsymbol{u}\|_{L^2}^2 &= \int_\Omega \nabla \boldsymbol{u} : \nabla F\boldsymbol{u} - \int_\Omega \nabla \boldsymbol{w} : \nabla F\boldsymbol{u} \\ &= \int_\Omega \nabla \boldsymbol{u} : \nabla F\boldsymbol{u} + \int_\Omega \Delta \boldsymbol{w} \cdot F\boldsymbol{u} \\ &= \int_\Omega \nabla \boldsymbol{u} : \nabla F\boldsymbol{u} + \int_\Omega (-\frac{1}{\mu^2} F\boldsymbol{u}) \cdot F\boldsymbol{u} \\ &= \int_\Omega \nabla \boldsymbol{u} : \nabla F\boldsymbol{u} - \frac{1}{\mu^2} \|F\boldsymbol{u}\|_{L^2}^2 \end{aligned} \tag{5.2.10}$$

Discarding the strictly positive second term on the right-hand side above and

[2]Recall that Δ is associated with homogenous Neumann boundary conditions on $\Gamma_{in} \cup \Gamma_{wall}$ and the Dirichlet condition $L\boldsymbol{u} = \boldsymbol{u}$ on Γ_{out}, equivalent to a homogenous condition Dirichlet here.

applying Cauchy-Schwarz then gives:

$$\|\nabla F\boldsymbol{u}\|_{L^2}^2 \leq \int_\Omega \nabla \boldsymbol{u} : \nabla F\boldsymbol{u} \tag{5.2.11}$$

$$\leq \|\nabla \boldsymbol{u}\|_{L^2} \|\nabla F\boldsymbol{u}\|_{L^2} \tag{5.2.12}$$

$$\|\nabla F\boldsymbol{u}\|_{L^2} \leq \|\nabla \boldsymbol{u}\|_{L^2} \tag{5.2.13}$$

Which, by equivalence of the norms $\|\nabla \boldsymbol{v}\|_{L^2}$ and $\|\boldsymbol{v}\|_V$, was to be shown.

5.2.1 Filtered-Streamline Diffusion

The filtered field $\bar{\boldsymbol{u}}$ is used in our modified classical streamline diffusion scheme instead of the velocity field $\boldsymbol{u}$. In an iterative Picard scheme, this leads to the following algorithm, a particularization of the one introduced at the beginning of the present section.

Given the velocity $\boldsymbol{u}_h^k$:

1. Compute $\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} = F\boldsymbol{u}_h^k$;

2. Solve the stabilized problem: find $\boldsymbol{u}_h^{k+1}$ in V_h, p_h^{k+1} in Q_h such that for all $\boldsymbol{v}_h$ in V_h, q_h in Q_h:

$$\begin{aligned} &a(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1}) \\ &\quad + b(\boldsymbol{u}_h^{k+1}, q_h) + \tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h; W_K, h_K) - (\boldsymbol{f}, \boldsymbol{v}_h) = 0 \end{aligned} \tag{5.2.14}$$

3. Check Convergence: end if criterion met; else update and loop.

Note that, unlike in (5.1.6) and (5.1.7), the convective field $\boldsymbol{u}_h^k$ in the trilinear form c is not the same as the convective field $\bar{\boldsymbol{u}}^{k+\frac{1}{2}}$ in the stabilization term τ. However, as property (5.1.4) still holds trivially and (5.1.5) follows from property (f) in Remark (5.2.1), the iteration (5.2.14) is still well posed for given $\boldsymbol{u}_h^k$ and $\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}$.

Because of this distinction, it is somewhat ambiguous whether we should define W_K based on the norm of the physical velocity $\|\boldsymbol{u}_h\|$ or the filter field $\|\bar{\boldsymbol{u}}_h\|$; we found both approaches effective in our numerical experiments but somewhat better performance when using the physical velocity.

This method requires the solution of the Helmholtz system at each iteration. This represents an additional cost; however we note that the Helmholtz operator does not change throughout during the iterative scheme and therefore the associated matrix need only be assembled once. A solution of this problem is therefore an arguably small fraction of the cost of full Picard iteration on the saddle-point system; in some tests we found it to be less than one percent of the computational time. Accordingly, if the stabilized method allows for convergence in fewer iterations, we do not expect the cost of the filter problem to offset these savings.

We also notice that one may approximate the Helmholtz system using only matrix-vector products by truncating the Neumann series at a finite number of terms. This is justified for μ small enough to make the Neumann expansion convergent:

$$\overline{\boldsymbol{u}}_h^{k+1/2} = \left(I - \mu^2\Delta\right)^{-1} \boldsymbol{u}_h^k \approx \tag{5.2.15}$$

$$\sum_{j=0}^{N} \left(-\mu^2\Delta_h\right)^j \boldsymbol{u}^k = \boldsymbol{u}^k - \mu^2\Delta_h\boldsymbol{u}^k + \ldots \tag{5.2.16}$$

where Δ_h is a matrix representing a discretization of the Laplacian with the boundary conditions specified as above. How this approximation affects computational time and accuracy is a possible subject for future investigation.

Convergence Proof

Here we present a convergence proof for the iteration (5.2.14). This proof is inspired heavily by the convergence proof of the standard Picard method pre-

sented in Chapter 2.3.1. As in the other convergence analyses shown in this work, we assume homogenous Dirichlet boundary conditions. As we will be performing arithmetic operations with τ, we will express it as a quadrilinear form as in (5.1.8) in order to make these manipulations more clear. For ease of notation, we write $\tau(\boldsymbol{w}_h, \boldsymbol{v}_h, \boldsymbol{y}_h, \boldsymbol{z}_h; W_K, h_K)$ as simply $\tau(\boldsymbol{w}_h, \boldsymbol{v}_h, \boldsymbol{y}_h, \boldsymbol{z}_h)$, with the dependence on the parameter W_K and mesh size h_K understood. We will assume that the bilinear form a and trilinear form c shown here are in their Laplacian and skew-symmetric forms, defined previously as a^* (2.2.19) and c^* (2.2.18) respectively.

We define the *consistent Deconvolution-stabilized Navier-Stokes problem* as: find $\boldsymbol{u}_h$ in V_h, p_h in Q_h such that for all $\boldsymbol{v}_h$ in V_h, q_h in Q_h:

$$\begin{aligned} &a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) \\ &\quad + b(\boldsymbol{u}_h, q_h) + \tau(\bar{\boldsymbol{u}}_h, \boldsymbol{u}_h, \boldsymbol{v}_h; W_K, h_K) - (\boldsymbol{f}, \boldsymbol{v}_h) = 0 \end{aligned} \tag{5.2.17}$$

The velocity solution $\boldsymbol{u}_h$ of (5.2.17) satisfies the bound (2.3.10) derived for (2.3.4) in Section 2.3.1:

$$\|\boldsymbol{u}_h\|_V \leq \frac{\|\boldsymbol{f}\|_{-1}}{\nu C_a}, \tag{5.2.18}$$

This is easily seen by applying the same arguments (2.3.7)-(2.3.8) to (5.2.17) and noting that the additional term on the left-hand side $\sum_{K\in\mathcal{T}} W_K h_K \|(\bar{\boldsymbol{u}} \cdot \nabla)\boldsymbol{u}\|^2_{L^2(K)}$ is strictly positive. We also recall the definition of χ and assume the corresponding small data hypothesis (2.3.18) is filled:

$$\chi \equiv \frac{\nu^2 C_a^2}{C_b \|\boldsymbol{f}\|_{-1}} > 1. \tag{5.2.19}$$

Theorem 5.2.1 *Let us assume that there exists a unique solution pair $(\boldsymbol{u}_h, p_h)$ pair to (5.2.17), resulting from regularity assumptions and the small data hypothesis (2.3.18).*

Assume data sufficiently regular such that $\boldsymbol{u}_h \in H^2(\Omega)$. *Then the sequence given by (5.1.7) converges to* $(\boldsymbol{u}_h, p_h)$ *provided the initial guess* $(\boldsymbol{u}_h^0, p_h^0)$ *is sufficiently close and* δ_K *is chosen such that:*

$$\delta_K < \frac{C_b(\chi - 1)}{2\, h_{MAX}} \tag{5.2.20}$$

is satisfied, where h_{MAX} *is the size of the largest element in* $\mathcal{T}$.

Proof.

We proceed by induction. Denote $\boldsymbol{e}^k := \boldsymbol{u}_h^k - \boldsymbol{u}_h$ and $\bar{\boldsymbol{e}}^k := \bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} - \bar{\boldsymbol{u}}_h$. We assume at an iteration k for some $\widetilde{\chi} > 1$,

$$\|\boldsymbol{e}^k\|_V \leq \frac{1}{\widetilde{\chi}^k}\|\boldsymbol{e}^0\|_V \tag{5.2.21}$$

and we seek to show that:

$$\|\boldsymbol{e}^{k+1}\|_V \leq \frac{1}{\widetilde{\chi}}\|\boldsymbol{e}^k\|_V \leq \frac{1}{\widetilde{\chi}^{k+1}}\|\boldsymbol{e}^0\|_V \tag{5.2.22}$$

Based on the fact that $\|\overline{\boldsymbol{u}}_h\|_{L^2} \leq \|\boldsymbol{u}_h\|_{L^2}$ (see Remark 5.2.1), we assume additionally that for an initial guess close enough:

$$\|\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}\|_{L^2} \leq \|\boldsymbol{u}_h\|_{L^2} \tag{5.2.23}$$

Begin by subtracting (5.2.17) from (5.2.14) at the iteration $k+1$ to obtain:

$$\begin{aligned} &a(\boldsymbol{e}^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1} - p_h) + b(\boldsymbol{e}^{k+1}, q_h) + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) - c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) \\ &\quad + \tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{u}_h^{k+1}, \bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{v}_h) - \tau(\bar{\boldsymbol{u}}_h, \boldsymbol{u}_h, \bar{\boldsymbol{u}}_h, \boldsymbol{v}_h) = 0 \end{aligned} \tag{5.2.24}$$

Noting that $\boldsymbol{e}^{k+1}$ is in the space X_h (2.3.6) implies $b(\boldsymbol{e}^{k+1}, q_h) = 0$. After some rearrangement and adding and subtracting $c(\boldsymbol{u}_h^k, \boldsymbol{u}_h, \boldsymbol{v}_h)$, $\tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{u}_h, \bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{v}_h)$, and

$\tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{u}_h, \bar{\boldsymbol{u}}_h, \boldsymbol{v}_h)$:

$$\begin{aligned} a(\boldsymbol{e}^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1} - p_h) + \tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{e}^{k+1}, \bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{v}_h) = \\ -c(\boldsymbol{e}^k, \boldsymbol{u}_h, \boldsymbol{v}_h) - c(\boldsymbol{u}_h^k, \boldsymbol{e}^{k+1}, \boldsymbol{v}_h) - \tau(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{u}_h, \bar{\boldsymbol{e}}^k, \boldsymbol{v}_h) - \tau(\bar{\boldsymbol{e}}^k, \boldsymbol{u}_h, \bar{\boldsymbol{u}}_h, \boldsymbol{v}_h) \end{aligned} \tag{5.2.25}$$

We now let $\boldsymbol{v}_h = \boldsymbol{e}^{k+1}$. This gives $b(\boldsymbol{e}^{k+1}, p_h^{k+1} - p_h) = 0$ by $\boldsymbol{e}^{k+1} \in X_h$ and $c(\boldsymbol{u}_h^k, \boldsymbol{e}^{k+1}, \boldsymbol{e}^{k+1}) = 0$ by skew-symmetry, yielding:

$$\begin{aligned} &\nu C_a \|\boldsymbol{e}^{k+1}\|_V^2 + W_K h_{MIN} \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2 \\ &\quad \leq C_b \|\boldsymbol{e}^k\|_V \|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{k+1}\|_V + \sum_{K \in \mathcal{T}} W_K h_K \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{u}_h\|_{L^2(K)} \|(\bar{\boldsymbol{e}}^k \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2(K)} \\ &\quad + \sum_{K \in \mathcal{T}} W_K h_K \|(\bar{\boldsymbol{e}}^k \cdot \nabla)\boldsymbol{u}_h\|_{L^2(K)} \|(\bar{\boldsymbol{u}}_h \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2(K)} \end{aligned} \tag{5.2.26}$$

Applying the Cauchy-Schwarz inequality on the summation terms above gives:

$$\begin{aligned} &\nu C_a \|\boldsymbol{e}^{k+1}\|_V^2 + W_K h_{MIN} \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2 \\ &\leq C_b \|\boldsymbol{e}^k\|_V \|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{k+1}\|_V + W_K h_{MAX} \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{u}_h\|_{L^2} \|(\bar{\boldsymbol{e}}^k \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2} \\ &\quad + W_K h_{MAX} \|(\bar{\boldsymbol{e}}^k \cdot \nabla)\boldsymbol{u}_h\|_{L^2} \|(\bar{\boldsymbol{u}}_h \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2} \\ &\leq C_b \|\boldsymbol{e}^k\|_V \|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{k+1}\|_V + W_K h_{MAX} \|\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}\|_{L^2} \|\nabla \boldsymbol{u}_h\|_{L^2} \|\bar{\boldsymbol{e}}^k\|_{L^2} \|\nabla \boldsymbol{e}^{k+1}\|_{L^2} \\ &\quad + W_K h_{MAX} \|\bar{\boldsymbol{e}}^k\|_{L^2} \|\nabla \boldsymbol{u}_h\|_{L^2} \|\bar{\boldsymbol{u}}_h\|_{L^2} \|\nabla \boldsymbol{e}^{k+1}\|_{L^2} \end{aligned} \tag{5.2.27}$$

where the last line follows from the basic inequality: $\|(\boldsymbol{w} \cdot \nabla)\boldsymbol{v}\|_{L^2} \leq \|\boldsymbol{w}\|_{L^2} \|\nabla \boldsymbol{v}\|_{L^2}$. Let W_K be such that:

$$W_K := \delta_K / \|\boldsymbol{u}_h\|_{L^2} \tag{5.2.28}$$

Then from (5.2.28), (5.2.23), $\|\bar{\boldsymbol{u}}_h\|_{L^2} < \|\boldsymbol{u}_h\|_{L^2}$ and Theorem 5.2.3, (5.2.29) reduces

to:

$$\begin{aligned}
&\nu C_a \|\boldsymbol{e}^{k+1}\|_V^2 + W_K h_{MIN} \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2 \\
&\quad \leq (C_b + 2\delta_K h_{MAX}) \|\boldsymbol{e}^k\|_V \|\boldsymbol{u}_h\|_V \|\boldsymbol{e}^{k+1}\|_V \\
&\quad \leq (C_b + 2\delta_K h_{MAX}) \frac{\|\boldsymbol{f}\|_{-1}}{\nu C_a} \|\boldsymbol{e}^k\|_V \|\boldsymbol{e}^{k+1}\|_V
\end{aligned} \tag{5.2.29}$$

where the last line follows from (5.2.18). Then from (5.2.19):

$$\|\boldsymbol{e}^{k+1}\|_V + W_K h_{MIN} \frac{\|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2}{\nu C_a \|\boldsymbol{e}^{k+1}\|_V} \leq \left(\frac{1}{\chi} + \frac{2\delta_K h_{MAX} \|\boldsymbol{f}\|_{-1}}{\nu^2 C_a^2} \right) \|\boldsymbol{e}^k\|_V \tag{5.2.30}$$

Which implies the result if:

$$\widetilde{\chi} \equiv \frac{\nu^2 C_a^2}{(C_b + 2\delta_K h_{MAX}) \|\boldsymbol{f}\|_{-1}} > 1 \tag{5.2.31}$$

This holds provided[3]:

$$\delta_K < \frac{C_b(\chi - 1)}{2\, h_{MAX}} \tag{5.2.32}$$

Note the right hand side of (5.2.32) is guaranteed to be positive by (5.2.19).

While (5.2.30) immediately implies (5.2.22), looking at its left hand side shows that it in fact establishes a stronger result that explains how this method stabilizes. By the established bounds, we may assume that for large enough k:

$$\|\boldsymbol{e}^{k+1}\|_V \leq \frac{W_K h_{MIN}}{\nu C_a}, \text{ implying: } 1 \leq \frac{W_K h_{MIN}}{\nu C_a \|\boldsymbol{e}^{k+1}\|_V} \tag{5.2.33}$$

[3]Verification that the base case holds for $\boldsymbol{u}_h^0 = \boldsymbol{0}$ (as done in the proof of Theorem 2.3.1) is straightforward and omitted for brevity.

By which one obtains:

$$\|\boldsymbol{e}^{k+1}\|_V + \|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2 \leq \frac{1}{\widetilde{\chi}}\|\boldsymbol{e}^k\|_V \tag{5.2.34}$$

The contraction constant $\widetilde{\chi}^{-1}$ is similar to the constant χ^{-1} in the standard (non-stabilized) Picard iteration [66], but is now bounding two positive terms dependent on $\boldsymbol{e}^{k+1}$: the error in the H^1 norm $\|\boldsymbol{e}^{k+1}\|_V$ as well as the error along the filtered field $\|(\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla)\boldsymbol{e}^{k+1}\|_{L^2}^2$. Based on this, we expect the stabilization term to accelerate convergence by providing stronger control on the error in the regions identified by $\bar{\boldsymbol{u}}_h^{k+\frac{1}{2}}$. This is confirmed by our numerical experiments.

5.2.2 A Strongly Consistent Method

Applying this approach to strongly consistent methods is not immediate. Classical strongly consistent methods rely on the residual:

$$-\nu\Delta\boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\,\boldsymbol{u} + \nabla p - \boldsymbol{f} \tag{5.2.35}$$

Implicit in this definition is the reliance on the streamline vector field $\boldsymbol{u}$, which serves both as the stabilization path and to guarantee strong consistency. Simply replacing $\boldsymbol{u}$ with $\overline{\boldsymbol{u}}$ in the convective term, as we did for the Filtered-Streamline Diffusion method, yields:

$$-\nu\Delta\boldsymbol{u} + (\overline{\boldsymbol{u}} \cdot \nabla)\,\boldsymbol{u} + \nabla p - \boldsymbol{f} \tag{5.2.36}$$

which is in general nonzero when evaluated at the exact solution and therefore not strongly consistent. A different approach is required. Expanding (5.2.36) with

the generic $\bar{\boldsymbol{u}} = \boldsymbol{u} - D(L\boldsymbol{u})$ gives

$$-\nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} - (D(L(\boldsymbol{u}))\cdot\nabla)\,\boldsymbol{u} + \nabla p - \boldsymbol{f} \tag{5.2.37}$$

We therefore need a term to offset the contributions of $(D(L(\boldsymbol{u}))\cdot\nabla)\,\boldsymbol{u}$ in order for (5.2.37) to be consistent with (5.2.35). This leads to the following SUPG-type approach for stabilizing along $\overline{\boldsymbol{u}}$:

$$\begin{aligned}\mathcal{L}_h(\boldsymbol{u}_h, p_h; \overline{\boldsymbol{u}}_h, \boldsymbol{f}\boldsymbol{v}_h, q_h) = \\ \sum_{K\in\mathcal{T}_h} h_K W_K \left(-\nu\Delta\boldsymbol{u}_h + (\boldsymbol{u}_h\cdot\nabla)\,\boldsymbol{u}_h + \nabla p_h - \boldsymbol{f},\ \ (\overline{\boldsymbol{u}}_h\cdot\nabla)\,\boldsymbol{v}_h + \nabla q_h\right)_{L^2(K)}.\end{aligned} \tag{5.2.38}$$

Numerical experiments suggest however that this is unstable, and we will demonstrate in the following section that indeed we cannot guarantee coercivity, and hence well-posedness, for this problem. Let $\widehat{\widetilde{\boldsymbol{u}}} = D(L(\boldsymbol{u}))$ for the sake of notation and note that $\widehat{\widetilde{\boldsymbol{u}}} = \boldsymbol{u} - \overline{\boldsymbol{u}}$. We then take advantage of our iterative framework and at a given iteration $k+1$ we introduce the following two bilinear forms:

$$\begin{aligned}\mathcal{L}_{FIL}(\boldsymbol{u}_h^{k+1}, p_h^{k+1}; \overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{f}, \boldsymbol{v}_h, q_h) = \\ \sum_{K\in\mathcal{T}_h} h_K W_K \left(-\nu\Delta\boldsymbol{u}_h^{k+1} + (\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla)\boldsymbol{u}_h^{k+1} + \nabla p_h^{k+1} - \boldsymbol{f},\ \ (\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla)\boldsymbol{v}_h + \nabla q_h\right)_{L^2(K)}, \\ \mathcal{L}_{EXP}(\boldsymbol{u}_h^{k}, \widehat{\widetilde{\boldsymbol{u}}}_h^{k+\frac{1}{2}}, \overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{v}_h, q_h) = \\ \sum_{K\in\mathcal{T}_h} h_K W_K \left(\ \ (\widehat{\widetilde{\boldsymbol{u}}}_h^{k+\frac{1}{2}}\cdot\nabla)\boldsymbol{u}_h^{k},\ \ (\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla)\boldsymbol{v}_h + \nabla q_h\right)_{L^2(K)}.\end{aligned} \tag{5.2.39}$$

The term $\mathcal{L}_{EXP}$ is entirely explicit. For a convergent iterative scheme $\boldsymbol{u}_h^{k+1} \approx \boldsymbol{u}_h^{k}$, and $\mathcal{L}_{EXP}$ negates the contributions of $(\widehat{\widetilde{\boldsymbol{u}}}_h^{k+1}\cdot\nabla)\boldsymbol{u}_h^{k+1}$ without affecting the coercivity of $\mathcal{L}_{FIL}$. Then at each step we solve the following problem: Find $(\boldsymbol{u}_h^{k+1}, p_h^{k+1})$

in $V_h \times Q_h$ such that

$$a(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h^{k+1}) + b(\boldsymbol{u}_h^{k+1}, q_h) + \tag{5.2.40}$$
$$\mathcal{L}_{FIL}(\boldsymbol{u}_h^{k+1}, p_h^{k+1}; \overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{f}; \boldsymbol{v}_h, q_h) = -\mathcal{L}_{EXP}(\boldsymbol{u}_h^k, \widehat{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{v}_h, q_h) + (\boldsymbol{f}, \boldsymbol{v}_h)$$

for all $(\boldsymbol{v}_h, q_h)$ in $V_h \times Q_h$. The strong consistency is easily seen by noting when evaluating (5.2.40) for an exact solution $\boldsymbol{u}$ we have $\boldsymbol{u}_h^k = \boldsymbol{u}_h^{k+1} = \boldsymbol{u}$. Note that this scheme is inherently steady and to properly apply it to the unsteady case requires its use as a nonlinear solver at a given time step.

We defined the parameter W_K as:

$$W_K := \frac{\delta_K}{2\|\boldsymbol{u}_h\|_K} \min\left(1, \frac{h\,\|\boldsymbol{u}_h\|_K \,\mathrm{Re}}{2\|\boldsymbol{u}_h\|_{L^2}}\right)$$

where δ_K is a small parameter. This definition was chosen based on empirical investigation. Although we have broken it into two parts (explicit and implicit), our term is still constructed with the residual of the physical velocity. Therefore, we believe it is more consistent to construct W_K in terms of $\boldsymbol{u}_h$ rather than $\bar{\boldsymbol{u}}_h$, as shown above. The proper selection of W_K for the existing methods is a nontrivial issue. We expect that the performance of this method can be properly tuned and improved through informed selection of this parameter and this is worth exploring in future work.

Remark. Like the related SUPG and GLS methods, this scheme also stabilizes the element spaces, allowing one to use non-LBB stable finite element pairs (such as $\mathbb{P}^1/\mathbb{P}^1$) for velocity and pressure. This can be seen by expanding the expression

for $\mathcal{L}_{FIL}$:

$$\begin{aligned}\mathcal{L}_{FIL}(\boldsymbol{u}_h^{k+1}, p_h^{k+1}; \overline{\boldsymbol{u}}^{k+\frac{1}{2}}, \boldsymbol{f}, \boldsymbol{v}_h, q_h) = \\ \sum_{K\in\mathcal{T}_h} h_K W_K \left(-\nu\Delta \boldsymbol{u}_h^{k+1} + \left(\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla\right)\boldsymbol{u}_h^{k+1} - \boldsymbol{f},\ \left(\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla\right)\boldsymbol{v}_h + \nabla q_h\right)_{L^2(K)} \\ + \sum_{K\in\mathcal{T}_h} h_K W_K \left(\nabla p_h,\ \left(\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}\cdot\nabla\right)\boldsymbol{v}_h\right)_{L^2(K)} \\ + \sum_{K\in\mathcal{T}_h} h_K W_K \left(\nabla p_h,\ \nabla q_h\right)_{L^2(K)}.\end{aligned} \tag{5.2.41}$$

The discretization of the term $\sum_{K\in\mathcal{T}_h} h_K W_K \left(\nabla p_h,\ \nabla q_h\right)_{L^2(K)}$ adds a nonsingular negative discrete Laplacian matrix to the (2,2) block of (2.3.2), ensuring the nonsingularity of the system regardless of element choice [59].

Theorem 5.2.4 *At each iteration, the stabilized problem (5.2.40) is well-posed.*

Proof. We will utilize the following estimate for Oseen problems with divergence-free vector fields [107]. Assume that $\boldsymbol{u}$ is strongly divergence-free. Then by Theorem 5.2.2 $\overline{\boldsymbol{u}}_h$ is divergence-free as well.

Let

$$X := [\sum_{K\in\mathcal{T}_h} \|(\overline{\boldsymbol{u}}_h\cdot\nabla)\boldsymbol{u}_h + \nabla p_h\|_K^2]^{1/2}$$

for the sake of brevity:

$$\left|\sum_{K\in\mathcal{T}_h} h_K W_K(-\nu\Delta\boldsymbol{u}_h,\ (\overline{\boldsymbol{u}}_h\cdot\nabla)\boldsymbol{u}_h + \nabla p_h)_{L^2(K)}\right| \leq \frac{1}{2}\left(\nu\|\nabla\boldsymbol{u}_h\|_{L^2}^2 + X^2\right) \tag{5.2.42}$$

We first show the coercivity of the bilinear form:

$$\begin{aligned} A((\boldsymbol{u}_h^{k+1}, p_h^{k+1}), (\boldsymbol{v}_h, q_h)) &= a(\boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{v}_h) \\ +b(\boldsymbol{v}_h, p_h^{k+1}) - b(\boldsymbol{u}_h^{k+1}, q_h) &+ \mathcal{L}_{FIL}(\boldsymbol{u}_h^{k+1}, p_h^{k+1}; \overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}, \boldsymbol{f}; \boldsymbol{v}_h, q_h) \end{aligned} \tag{5.2.43}$$

Set $(\boldsymbol{v}_h, q_h) = (\boldsymbol{u}_h^{k+1}, p_h^{k+1})$. Then

$$\begin{aligned} A((\boldsymbol{u}_h^{k+1}, p_h^{k+1}), (\boldsymbol{u}_h^{k+1}, p_h^{k+1})) &= \nu \|\nabla \boldsymbol{u}_h\|_{L^2}^2 + c(\boldsymbol{u}_h^k, \boldsymbol{u}_h^{k+1}, \boldsymbol{u}_h^{k+1}) \\ +X^2 + \sum_{K \in \mathcal{T}_h} h_K W_K (-\nu \Delta \boldsymbol{u}_h^{k+1}, &(\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla) \boldsymbol{u}_h^{k+1} + \nabla p_h^{k+1})_{L^2(K)}. \end{aligned} \tag{5.2.44}$$

The second term on the right-hand side is zero by the skew-symmetry of c. Then by (5.2.42) we can group terms together and find:

$$\begin{aligned} A((\boldsymbol{u}_h^{k+1}, p_h^{k+1}), (\boldsymbol{u}_h^{k+1}, p_h^{k+1})) &> \frac{1}{2}(\nu \|\nabla \boldsymbol{u}_h^{k+1}\|_{L^2}^2 + X^2) \\ &\geq \frac{\nu}{2} \|\nabla \boldsymbol{u}_h^{k+1}\|_{L^2}^2 \\ &\geq C \|\boldsymbol{u}_h^{k+1}\|_V^2. \end{aligned} \tag{5.2.45}$$

The continuity of A and the right-hand side are obvious assuming that $\boldsymbol{f}$, $\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}}$, and $\widehat{\overline{\boldsymbol{u}}}_h^{k+\frac{1}{2}}$ are bounded at each iteration k. The well-posedness follows. Note that a proof of well-posedness at each iteration does not imply the convergence of the iterative scheme to the desired solution; a rigorous proof of this (similar to that provided for the consistent scheme in the previous section) is still missing. **Remark:** It was mentioned earlier that stability is not guaranteed for the formulation (5.2.38). One can see this by observing that after testing against $(\boldsymbol{u}_h^{k+1}, p_h^{k+1})$, one obtains an expression similar to (5.2.44), but with the following term instead of X^2:

$$\sum_{K \in \mathcal{T}_h} h_K W_K ((\boldsymbol{u}_h^k \cdot \nabla) \boldsymbol{u}_h^{k+1} + \nabla p_h, (\overline{\boldsymbol{u}}_h^{k+\frac{1}{2}} \cdot \nabla) \boldsymbol{u}_h^{k+1} + \nabla p_h^{k+1})_{L^2(K)}$$

which is not guaranteed to be strictly positive in general due to the different convective fields on either side of the inner products.

5.3 Numerical Results and Discussion

In this section we test our methods on two 2D and two 3D test cases. Both 2D and 3D cases were tested on FreeFem++ using a 2017 MacBook Pro.

5.3.1 2D Test Case 1: Flow Past a Step

In our first test, we solve the 2D Incompressible Navier-Stokes Equations at Reynolds numbers of 166 and 200[4] in the domain pictured in Figure 5.3.

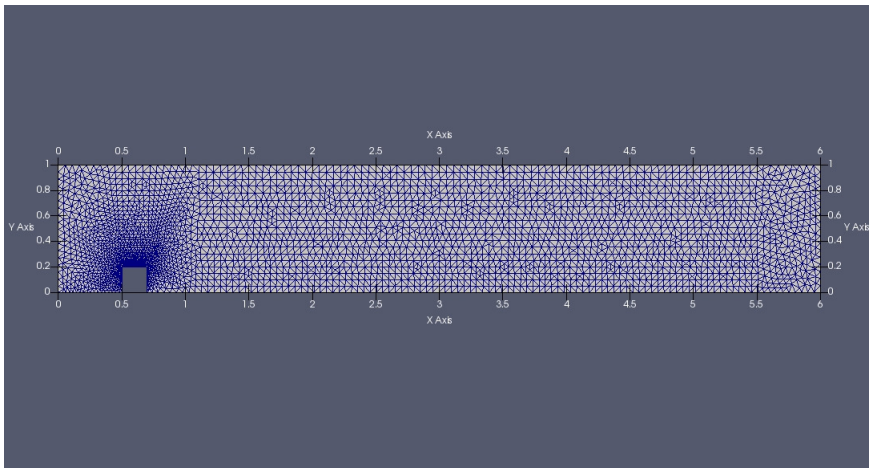

Figure 5.3: Domain for the Flow Past a Step test case.

The domain is a 6×1 rectangle with a $.2 \times .2$ rectangular step along the bottom starting at $x = 0.5$. The flow becomes disturbed and a recirculation region develops past the step as seen in Figure 5.4. This case can be regarded as a simple two-dimensional model of an artery where a vascular prosthesis has been deployed. We first computed a reference solution on a very fine mesh (38,280 DOF) using standard Picard iterations to a high level of accuracy (stopping tolerance of

[4]The characteristic length here is .2, the height of the step.

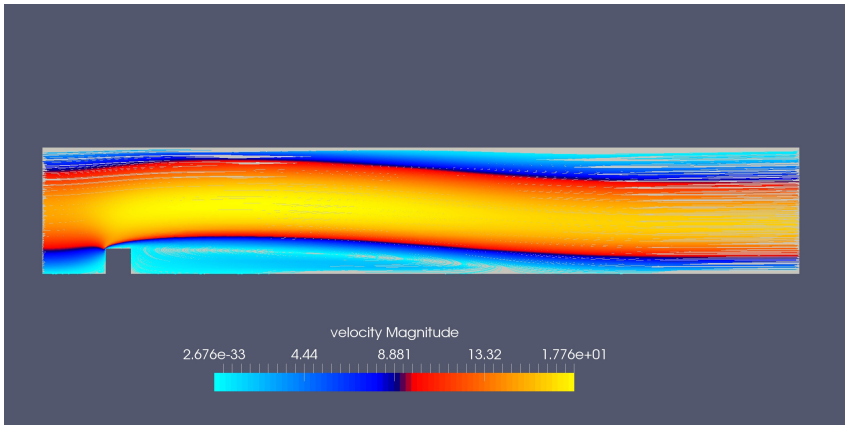

Figure 5.4: Velocity streamlines for test case, Re=200.

10^{-8} in the relative L^2 velocity norm between consecutive iterates). Next, we computed stabilized solutions on different mesh levels using the different techniques discussed, including both the streamline-diffusion type and strongly consistent methods.

For the purposes of comparison, we also computed solutions stabilized with the classical streamline diffusion and SUPG methods. For all computations, we used a Picard-type approach and iterated until the difference in velocity L^2 norm between iterations was below 1e-3. For all tests we used $\mathbb{P}^2/\mathbb{P}^1$ finite elements for velocity and pressure respectively, and set $\mu = .2$, $\delta_K = .1$. We report the results in the Table 5.1.

Flow Past a Step: Weakly Consistent Deconvolution Stabilization vs Streamline Diffusion						
Re	Ref. Iter.	DOF	Deconv. It.	$\|\boldsymbol{u}_D - \boldsymbol{u}\|_{L^2}/\|p_D - p\|_{L^2}$	SD Iter.	$\|\boldsymbol{u}_S - \boldsymbol{u}\|_{L^2}/\|p_S - p\|_{L^2}$
166	52	17292	23	.0029/.0133	30	.0035/.0143
166	52	28623	28	.0028/.0106	43	.0013/.0059
200	89	18192	28	.0046/.0186	40	.0051/.0139
200	89	28623	34	.0039/.0143	56	.0028/.0056
Flow Past a Step: Strongly Consistent Deconvolution Stabilization vs SUPG						
Re	Ref. Iter.	DOF	Deconv. It.	$\|\boldsymbol{u}_D - \boldsymbol{u}\|_{L^2}/\|p_D - p\|_{L^2}$	SUPG It.	$\|\boldsymbol{u}_S - \boldsymbol{u}\|_{L^2}/\|p_S - p\|_{L^2}$
166	52	17292	29	.0034/.0141	60	.0035/.0148
166	52	28623	39	.0016/.0058	52	.0013/.0048
200	89	17292	39	.0051/.0143	DNC	NA/NA
200	89	28623	55	.0028/.0056	85	.0019/.0059

Table 5.1: Numerical results, 2D flow past a step.

Our methods outperform the standard methods in all tests, requiring significantly fewer iterations to reach convergence in each instance. While the weakly consistent methods are known to suffer from reduced accuracy, this did not seem to have significant impact on this case except for Re=200 on the fine mesh, where weakly consistent deconvolution stabilization was less accurate than other methods. The above results establish that our scheme yields a more stable iteration, however faster convergence of nonlinear residuals $\|\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\|$ does not necessarily imply faster convergence to the reference solution.

Referring to Figures 5.5 and 5.6, we plot a comparison of the convergence of deconvolution stabilization vs SUPG for Re=200 for both the coarse (17,292 DOF) and moderate (28,623 DOF) meshes. It should be noted that standard Picard failed to converge at these mesh levels. We show the convergence to the reference solutions side-by-side with the nonlinear residuals. On the coarse mesh, we see that our method (blue) converges up to interpolation error in both velocity and pressure while for SUPG (red) the convergence becomes oscillatory, particularly for the pressure. This is reflected in the behavior of the nonlinear iteration residuals, which decrease monotonically for our method but oscillate and fail to drop below the tolerance level for SUPG.

On the fine mesh, we observe convergence to the solution for both methods, however convergence is much faster for the Deconvolution scheme. We still notice oscillations in the convergence for SUPG, however in this instance they merely slow the convergence and do not prevent it. We confirm here that our scheme does in fact converge more rapidly and monotonically than the standard schemes to the desired solution.

Remark: We must briefly clarify a point regarding this section (which applies to the future numerical tests as well). We recall that the reference solution is on a much finer mesh than our stabilized solutions, and therefore *we only expect*

convergence up to the level of interpolation error. This is indeed what we observe. In each instance for velocity and pressure we see the convergence 'bottoming out' and flattening after a certain point, indicating that the stabilized solution has converged up to interpolation error. 'Convergence' past this point is not meaningful. What we are interested in is the *rate* at which the respective methods reach convergence.

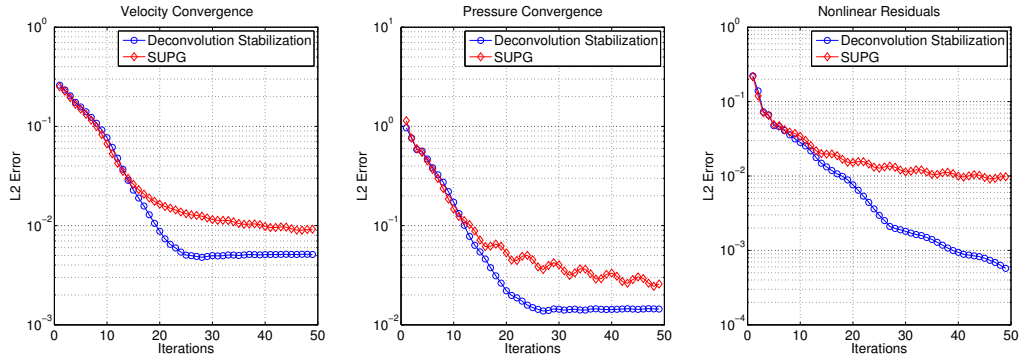

Figure 5.5: Convergence, Test Case 1. Re=200, 17292 DOF (Coarse).

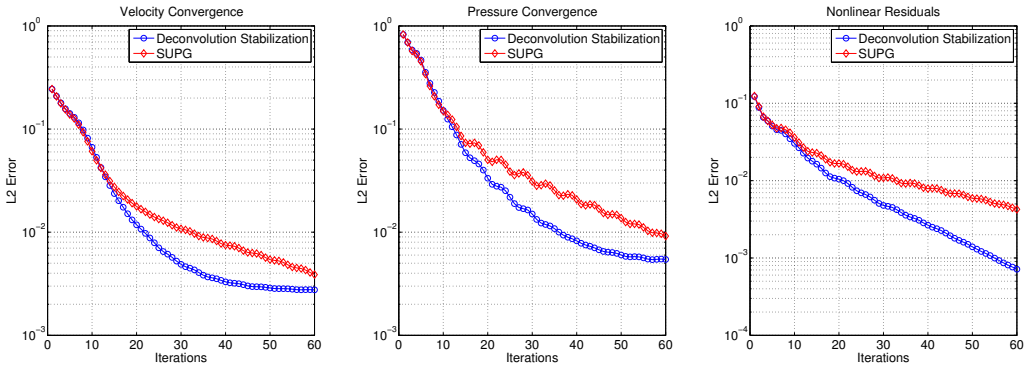

Figure 5.6: Convergence, Test Case 1. Re=200, 28623 DOF (Fine).

In (5.7) we show the velocity streamlines for the case $Re = 200$ colored according to the magnitude of our filtered field $\overline{\boldsymbol{u}}$. We verify that our filter properly identifies the recirculation region as requiring the most extra stabilization, despite it being the region of least convection as shown in Figure 5.4.

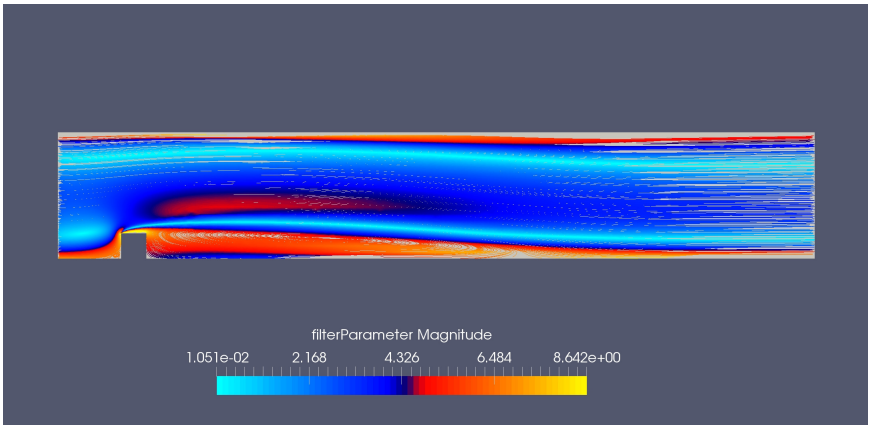

Figure 5.7: Streamlines for flow past a step, Re=200. Red regions indicate more stabilization.

5.3.2 2D Test Case 2: Flow in a Curved Pipe

We follow an identical procedure as before for our next 2D case. We compare our stabilized solutions for different mesh levels to a standard solution computed on a fine mesh (47,070 DOF) for Reynolds numbers 600 and 667 in the domain pictured in Figure 5.8. We report the results in Table 5.2. Although the two Reynolds

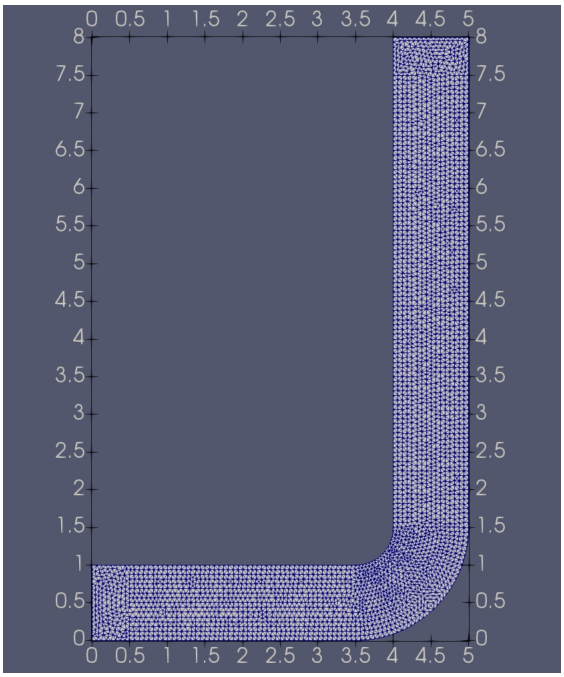

Figure 5.8: Domain for the 2D Curved Pipe test case.

numbers appear close together, the behavior of the flow is quite different. We found that for even slightly higher Reynolds numbers, such as 700, the solution is no longer steady, even on extremely fine meshes. This test is designed to showcase our stabilization method at the upper limit of the steady flow regime. Once again, our approach results in significantly fewer iterations to reach convergence while maintaining the same order of accuracy when compared to traditional stabilization techniques. For this test case, we do observe that the weakly consistent methods appear slightly less accurate when compared to the strongly consistent methods, particularly on the fine meshes.

Curved Pipe: Weakly Consistent Deconvolution Stabilization vs Streamline Diffusion						
Re	Ref. Iter.	DOF	Deconv. It.	$\|\boldsymbol{u}_D - \boldsymbol{u}\|_{L^2}/\|p_D - p\|_{L^2}$	SD Iter.	$\|\boldsymbol{u}_S - \boldsymbol{u}\|_{L^2}/\|p_S - p\|_{L^2}$
600	53	11879	21	.0192/.0448	44	.0222/.0458
600	53	23015	35	.0076/.0191	42	.00502/.0103
667	96	11879	30	.0308/.0934	54	.0251/.0571
667	96	23015	38	.0089/.0231	62	.0066/.0108
Curved Pipe: Strongly Consistent Deconvolution Stabilization vs SUPG						
Re	Ref. Iter.	DOF	Deconv. It.	$\|\boldsymbol{u}_D - \boldsymbol{u}\|_{L^2}/\|p_D - p\|_{L^2}$	SUPG It.	$\|\boldsymbol{u}_S - \boldsymbol{u}\|_{L^2}/\|p_S - p\|_{L^2}$
600	53	11879	36	.0166/.0440	49	.0163/.0435
600	53	23015	39	.0035/.0075	53	.0025/.00592
667	96	11879	61	.0187/.0453	85	.0215/.0580
667	96	23015	61	.0053/.0101	95	.0035/.0078

Table 5.2: Numerical results, 2D curved pipe.

The sharp increase in the required number of iterations (for all methods) for what appears to be a small increase in Reynolds number is expected based on the critical nature of this flow regime.

Remark: As mentioned before, our scheme also stabilizes the pressure and element spaces, enabling us to use non-LBB inf-sup compatible pairs for velocity and pressure (such as $\mathbb{P}^1/\mathbb{P}^1$). We compute the same 2D curved pipe test at Re=667 using $\mathbb{P}^1/\mathbb{P}^1$ on the same mesh level as our fine solution and report the results in Table 5.3.

This element space greatly reduces the degrees of freedom (16,032 rather than 47,070). The time per iteration for one (non-stabilized, $\mathbb{P}^2/\mathbb{P}^1$) fine mesh solution

Curved Pipe: Deconvolution Stabilization vs SUPG; Stabilized $\mathbb{P}^1/\mathbb{P}^1$						
Re	Ref. Iter.	DOF	Deconv. It.	$\|\boldsymbol{u}_D - \boldsymbol{u}\|_{L^2}/\|p_D - p\|_{L^2}$	SUPG It.	$\|\boldsymbol{u}_S - \boldsymbol{u}\|_{L^2}/\|p_S - p\|_{L^2}$
667	96	16032	59	.0170/.0469	71	.0163/.0432

Table 5.3: Numerical results, 2D curved pipe with stabilized $\mathbb{P}^1/\mathbb{P}^1$ elements.

was 2.64 seconds; for stabilized $\mathbb{P}^1/\mathbb{P}^1$ we observed 1.83 seconds per iteration. As expected, we lose accuracy with this element space, however. We again see faster convergence and similar error behavior using our method as compared to SUPG.

5.3.3 3D Test Case 1: Curved Pipe

Our next test case is a three-dimensional version of the curved pipe designed to demonstrate our method's applicability to 3D problems. We solve the 3D INS equations at Re=700 the domain pictured in Figure 5.9. For this test, we computed

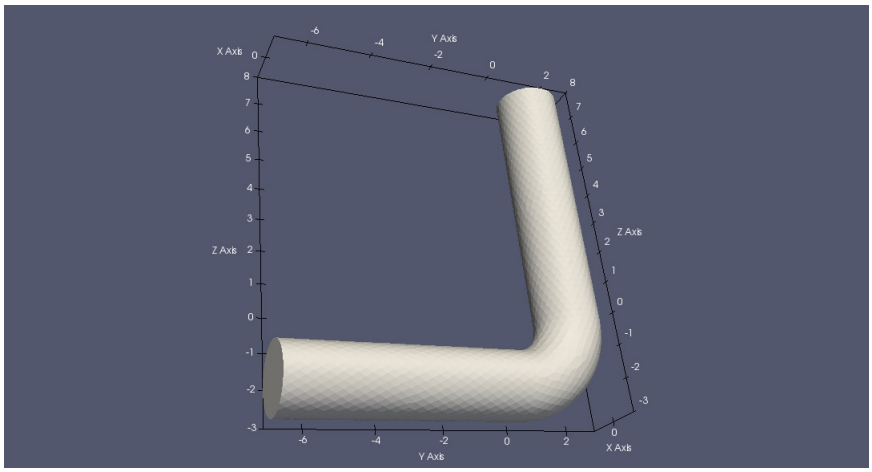

Figure 5.9: Domain for the 3D Curved Pipe test case.

a solution without stabilization using standard Picard iterations on a fine mesh (155,872 DOF). We then computed a stabilized solutions on a coarse mesh (61,391 DOF) using our strongly consistent method and SUPG stabilization. We validated our solutions by comparing the value of the pressure along the center plane of the pipe as pictured in Figure 5.10.

We used the same tolerance and values for δ and μ as before. We again ob-

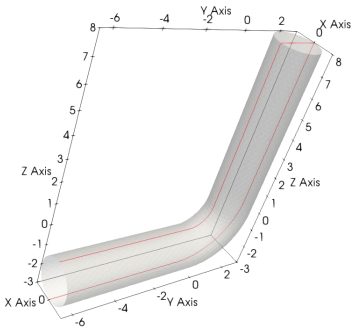

Figure 5.10: We compare the pressure along the pictured curve (red).

served improved performance; standard Picard required 34 iterations to converge on the fine mesh and did not converge on the coarse mesh. The coarse solution stabilized with SUPG converged in 47 iterations, while the solution stabilized with our techniques converged in 32 iterations (Figure 5.11). To assess the accuracy of our solution, we compare the pressure computed along the center of the vessel (pictured in Figure 5.10). We observe in Figure 5.12 that the stabilized solutions show good agreement with the fine mesh solution, with no significant difference between the Deconvolution and SUPG solutions.

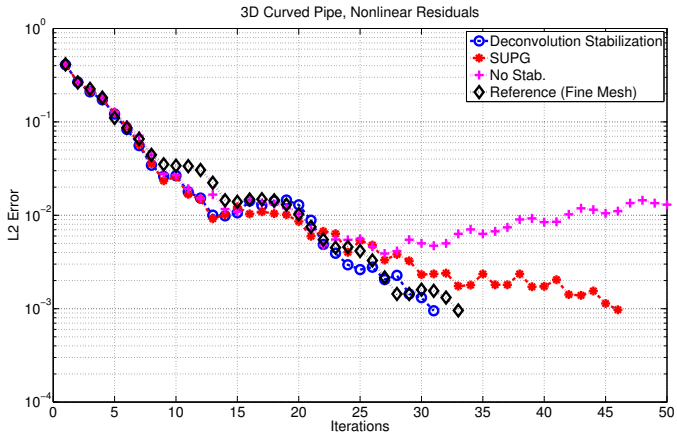

Figure 5.11: Convergence residuals for the 3D Curved pipe case; we see that without stabilization the solution does not converge on the coarse mesh.

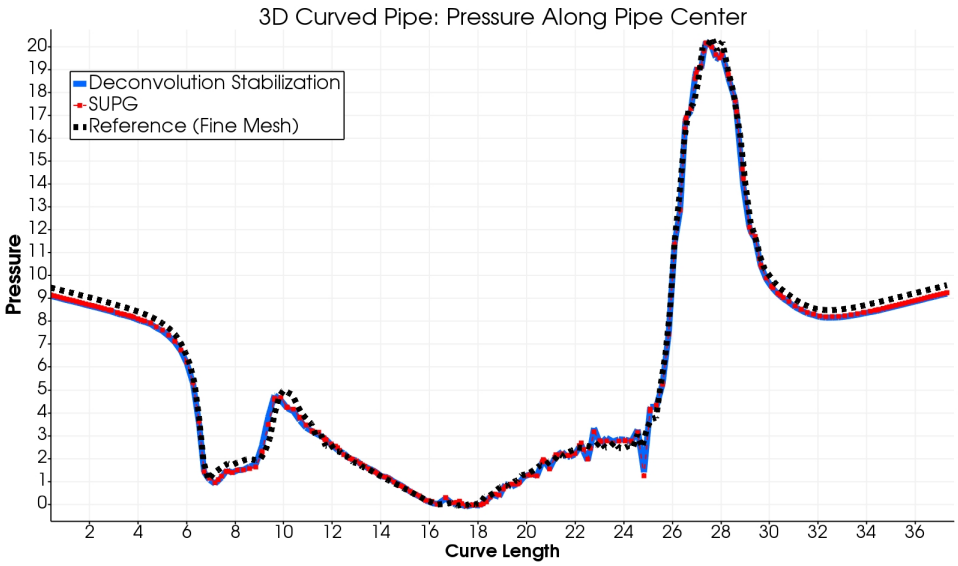

Figure 5.12: The pressure along the curve pictured in Figure 5.10.

5.3.4 3D Test Case 2: FDA Nozzle Benchmark, Re=500

For our final test case we consider the FDA Nozzle Benchmark case [1] for Re=500. We use an identical geometrical configuration as found in [90]. This test case differs from our previous test cases in that the primary instabilities arise from the high-convection; we seek to show that our stabilization method remains a viable choice in such instances. We note that our filtering method in no way excludes unstable high-convection regions, but merely aims to capture other, additional types of instability as well. Therefore, we do not necessarily expect our methods to outperform standard techniques; however we seek to verify that they remain competitive.

We set the density $\rho = 1.056$ g/cm^3 and viscocity $\nu = .035$ g/cm·s and prescribe a Poiseulle parabolic inflow profile with a flow rate of 5.2062 ml/s. We again set $\mu = .2$, $\delta_K = .1$. We monitor convergence by comparing the difference in velocity L^2 norm between consecutive iterations, and stop iterations when this falls below 1e-3. We will also monitor the pressure drop $p(0,0,-4) - p(0,0,0)$ between iterations to cross-validate the velocity convergence criterion.

We run our simulations with $\mathbb{P}^2/\mathbb{P}^1$ finite elements on a coarse mesh containing 52,895 tetrahedra (243,958 total DOF), an order of magnitude coarser than the meshes used in [90]. At this mesh level, a Picard iteration using standard Galerkin techniques fails to converge, necessitating the use of stabilization techniques. We compare our strongly consistent deconvolution scheme with SUPG in terms of both convergence and agreement with the measured solution. The convergence results are shown in Figure 5.13.

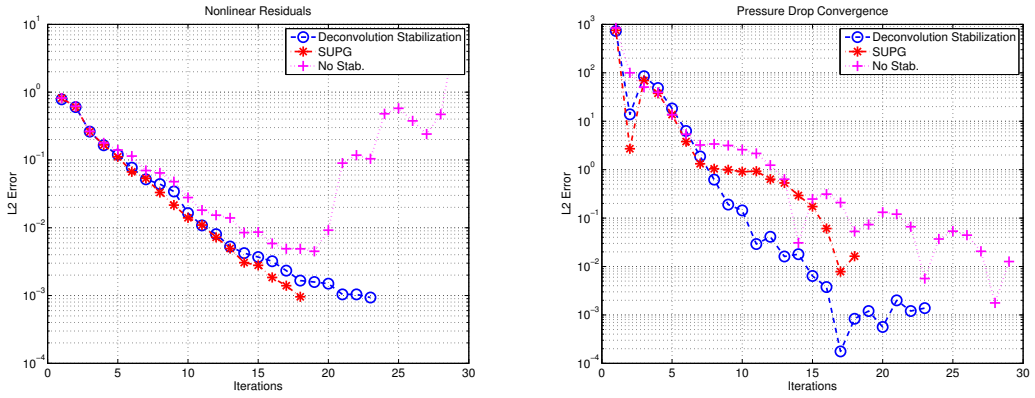

Figure 5.13: Left: FDA nonlinear residuals; Right: FDA pressure drop residuals

SUPG converges slightly faster than the deconvolution scheme (19 iterations compared to 24) in terms of our velocity residual, however we note that the pressure drop converges more steadily and rapidly for deconvolution filtering. We see that the iterations fail to converge without stabilization.

Comparing our results to the benchmark data (labeled 243, 297, 468, 763, and 999 in accordance with [1]), we find our normalized velocity and pressure drop along the centerline (computed in the same manner as in [90], [10]) is in excellent agreement with the reference experimental data (Figures 5.14 and 5.15). We do not observe any noteworthy difference between SUPG and deconvolution filtering in terms of accuracy.

Qualitatively, we see that in this case the nonlinear filter does indeed identify high-convection regions as requiring stabilization, in addition to the recirculation

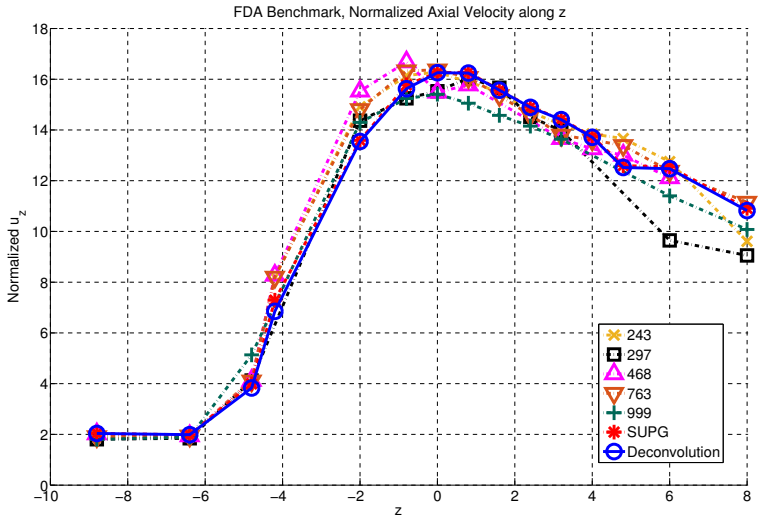

Figure 5.14: Normalized centerline velocity: FDA experimental datasets 243, 297, 468, 763, and 999 compared with computed solutions.

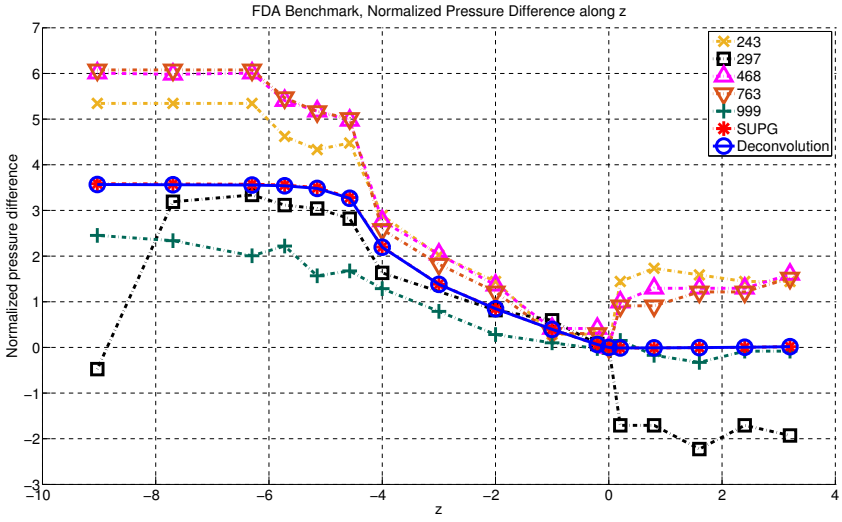

Figure 5.15: Pressure drop along centerline; FDA experimental datasets 243, 297, 468, 763, and 999 compared with computed solutions.

regions (Fig. 5.16). This shows that, while this method was not designed for stabilizing high-convection regions, it still can be used for this purpose. In particular, this suggests that our method remains a good choice for problems in which one encounters both convective and non-convective instabilities.

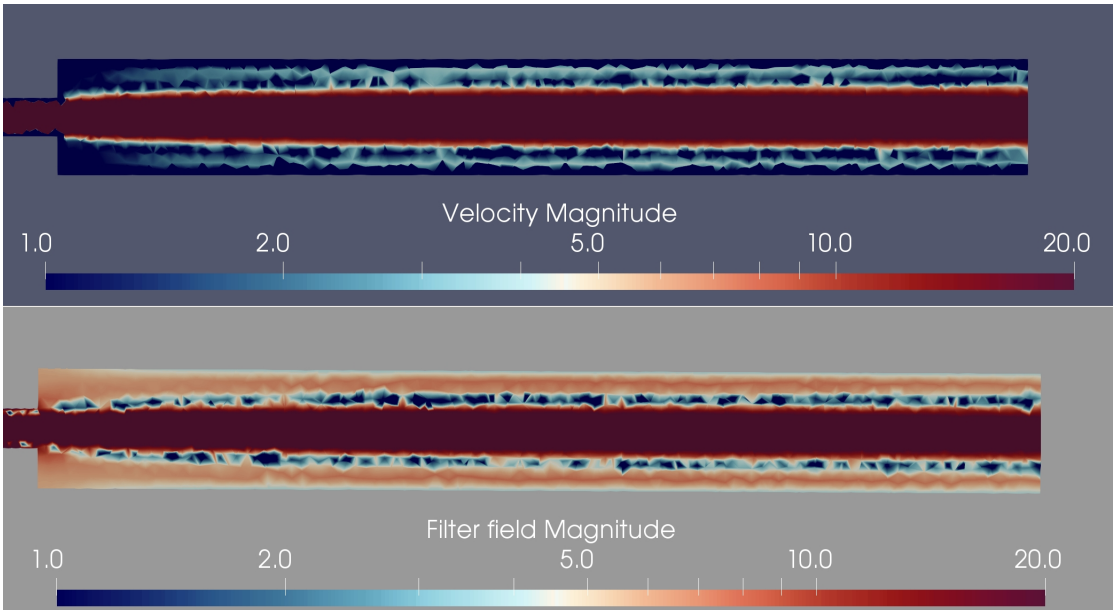

Figure 5.16: Top: velocity field $\boldsymbol{u}$. Bottom: filtered velocity field $\overline{\boldsymbol{u}}$.

Chapter 6

The Data-Deficient Boundary Condition Problem

In order for a CFD simulation to have any clinical relevance, it is essential that we achieve reliable results. Accuracy and reliability are distinct concepts. *Accuracy* ensures that the numerical solution to a given model problem is the correct one; however, if the underlying model is not physically realistic, an accurate solution to this problem will still fail to be clinically meaningful. *Reliability* refers to a model's ability to accurately reflect reality and produce results that are useful from the physical point of view. We may *verify* directly whether or not a given numerical solution is accurate by comparing it to a reference solution. To assess reliability, we do not verify but rather *validate* our numerical simulations by comparing the results against measured physical data.

Reliability is primarily an issue of modeling, while accuracy is primarily an issue of numerical approximation. To ensure reliable results, we must ensure that our underlying model is a satisfactory representation of physical reality. Within our domain, the incompressible Navier-Stokes equations given by (2.2.1) are a suitable physical model. That blood can accurately be represented as a Newtonian

fluid for large vessels is well-validated [40, 115, 121]. However, in order to solve these equations we must also prescribe appropriate boundary conditions. For (2.2.1) to provide an accurate physical model, these conditions must be assigned appropriately. Non-physical boundary conditions will lead to a non-physical solution.

In an ideal situation, we would be able to use reliable patient-specific data, such as flow rates, velocity fields, and pressure readings, to assign the most realistic possible boundary conditions. Unfortunately, this situation is not common in reality, and it is particularly uncommon for clinical hemodynamics problems. Often in these settings patient data is incomplete, unreliable, or even completely missing. We therefore must determine physically realistic boundary conditions to assign. Several techniques have been proposed for this, including approaches based on Kalman filtering and PDE-constrained optimization [32, 41, 11]. While these are effective, they have a large computational cost and often involve solving the underlying partial differential equation model multiple times. When computational time and resources are limited, these methods may not be viable.

In this section we will address the problem of determining reliable boundary conditions without incurring large additional computational costs. In particular, we would like to avoid methods that require multiple PDE solves, such as PDE-constrained optimization, while still incorporating any relevant known or estimated data into our problem. We will structure this section by first giving an overview of the defective-data problem in computational hemodynamics and the different situations and types of data one may encounter. While we will only go on to introduce novel approaches for a subset of these problems, we nonetheless feel that providing a comprehensive background is helpful for placing our work in proper context, as well as giving the reader a framework for areas of possible future development. After narrowing our scope to a specific problem, we will

then introduce a model problem arising from actual patient data in order to better elucidate the situation. We will then use this example to explain why common techniques are inadequate and to motivate and develop new techniques. We will conclude by validating our approach against clinical data to establish its efficacy.

The work presented in this chapter is based on collaborative efforts with Adrien Lefieux, Alessandro Veneziani, Don Giddens, and David Molony and is part of ongoing (as of this writing) research sponsored by the Emory University School of Medicine. Aspects of this work were presented by Adrien Lefieux at the 2017 SIAM Conference on Computational Science and Engineering [74].

6.1 Imposing and Determining Boundary Data

The proper selection and enforcement of boundary conditions is essential for reliability in computational hemodynamics. Unfortunately, in clinical settings there is typically a large gap between available data and a complete set of boundary data. Additionally, often the available data is subject to noise and measurement error, making its direct prescription potentially problematic even in the event that it is available. The main goal is therefore to determine appropriate boundary conditions in the absence of available data.

When dealing with boundary conditions, we face two issues: *what* boundary data to prescribe and *how* to prescribe them. Both issues are important. The question of how to prescribe boundary data refers to scheme used to enforce the data at a given boundary. In general, there are three types of boundary conditions: Dirichlet conditions, Neumann conditions, and Robin conditions. Which type of boundary condition is most appropriate depends on several factors including the type of data being enforced as well as computational and stability considerations. The question of *what* data to prescribe is less clear-cut and refers to the determina-

tion of appropriate boundary data when such data is unavailable or incomplete. Once again, which method is best will depend on whatever data is available and considerations based on the available computational power and time.

6.1.1 Imposing Boundary data

In this section we will briefly consider several important types of boundary data. In particular, we will discuss the enforcement of velocity boundary conditions, traction boundary conditions, and boundary conditions which incorporate both velocity and traction data. In practice, measured velocities and tractions are seldom known at the pointwise level, requiring us to instead surrogate these variables with related quantities that are more easily measured. We will detail possible strategies for surrogation and which type of boundary condition (Dirichlet, Neumann, or Robin) is most appropriate.

Velocity and Flow Rate data

Velocity data, if available, is the easiest type of data to work with in general. In the case that we have an accurate pointwise measurement of the velocity field, we may prescribe this data at a boundary using Dirichlet (pointwise) conditions.

Unfortunately this situation is rare in reality. A more common and realistic clinical scenario is one in which we have reliable measurements for the *flow-rate* of blood at a given vessel boundary. Flow rates represent a weak type of velocity data. Past a given boundary Γ, the flow rate Q tells us that

$$\int_{\Gamma} \boldsymbol{u} \cdot \boldsymbol{n} = Q \tag{6.1.1}$$

where $\boldsymbol{u}$ is our velocity vector. This data is *defective* in that it gives us information about the average flow behavior at a boundary, but does not provide information

about the pointwise behavior of the velocity. A stable and common way to enforce this information pointwise is through the construction of an inflow function $\boldsymbol{g}$, properly scaled so that:

$$\int_{\Gamma} \boldsymbol{g} \cdot \boldsymbol{n} = Q \tag{6.1.2}$$

We then enforce $\boldsymbol{u} = \boldsymbol{g}$ on Γ as a Dirichlet boundary condition. Near the boundary, this may cause inaccuracies; therefore it is common to add flow extensions in order to allow the impact of the artificial pointwise profile to subside near the region of interest. This approach is stable and reliable, but adding the flow extensions requires possibly nontrivial modification of the geometry, increasing the problem size and required preprocessing and computational time.

Another possible approach is to enforce flow rates using the Lagrange multiplier flux formulation as described in [39, 121]. One obtains this formulation by augmenting the Navier-Stokes system with (6.1.1) as an additional constraint, in effect enforcing the flow rate as a Neumann boundary condition [121]. This approach is arguably more natural than (6.1.2) as it makes no additional assumptions on the spatial profile of the velocity field, eliminating the need for flow extensions. However, this approach is known to cause stability issues arising from inflow through a section where a constant traction is prescribed (a mechanism similar to the well known backflow instabilities), and the addition of the extra constraint (6.1.1) makes the discrete problem more difficult to solve [94, 39, 31]. Techniques have been proposed to alleviatve one or both of these issues through penalization [130, 94], however these approaches require the weak prescription of a velocity field, for which a good choice may not be available given the data.

We may also employ the method suggested in [41], a variational approach in

which given a flow rate Q is enforced by minimizing:

$$\left| \int_\Gamma \boldsymbol{u} \cdot \boldsymbol{n} - Q \right|^2 \tag{6.1.3}$$

such that $\boldsymbol{u}$ is a solution of the incompressible Navier-Stokes equations. While this approach is effective, it has a heavy computational burden as it may involve solving the underlying Navier-Stokes problem multiple times. This may render this approach to impractical in clinical settings.

Traction and Pressure data

Traction data refers to data measuring the traction force $\boldsymbol{T}$ defined by:

$$-\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n} + p\boldsymbol{n} = \boldsymbol{T} \qquad \text{on } \Gamma \tag{6.1.4}$$

where $\boldsymbol{n}$ is the outward-pointing unit normal vector to Γ. We will show later that these conditions are very easy to enforce due to the weak formulation of our problem.

Actual pointwise data on the traction force is quite uncommon, so as in the case of velocity data, we most commonly surrogate traction with a closely related spatially indeterminate quantity. Note that in our applications typically $\nu << 1$ and our boundary Γ is typically such that $\boldsymbol{u}$ enters or exits perpendicularly to Γ. In this case, the term $-\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n}$ is either zero or small, allowing us to use a pressure measurement P as a surrogate for $\boldsymbol{T}$ as follows [39]:

$$-\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n} + p\boldsymbol{n} = P\boldsymbol{n} \qquad \text{on } \Gamma \tag{6.1.5}$$

This is important because while traction data is in general hard to obtain, blood pressure is among the most commonly available clinical measurements.

Pressure (and by extension traction) boundary data is most easily enforced as a Neumann condition. If the data is assigned at an outflow, these conditions can be easily assigned without much difficulty, provided that the flow is sufficiently regular. However, when assigned as an inflow this approach may again suffer from a lack of stability caused by flow reversal near the boundary [94]. We may also encounter flow reversal instability at outflows arising from disturbed flow patterns at the outlet reentering the domain. Recent work has made progress for stabilizing the flow reversal in both cases, but it should be noted that the goal of the stabilization is not the same for inflows and outflows. At outflows, the flow reversal represents a physical solution and one does not seek to eliminate them, but to obtain a stable solution incorporating their presence [35, 78, 36, 31]. At inflows, as in [94, 31], the flow reversal is non-physical and stabilization procedures focus on eliminating them.

As with flow rates, one may also employ a variational approach wherein a given P is enforced by minimizing [41]:

$$\left| \int_\Gamma (p\boldsymbol{n} - P\boldsymbol{n}) \right|^2 \tag{6.1.6}$$

subject to the constraint that p is a pressure field obtained by solving the incompressible Navier-Stokes equations.

Velocity/Flow Rate and Traction/Pressure data

If both velocity and traction data is available at a boundary Γ, ideally we would like to incorporate both pieces of information into our boundary conditions for maximum accuracy. For instance, we may want to enforce:

$$-\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n} + p\boldsymbol{n} + \theta \boldsymbol{u} = \boldsymbol{D} \qquad \text{on } \Gamma \tag{6.1.7}$$

where θ is an appropriate parameter. This condition incorporates information about both the traction and the velocity. Since it is constructed using both the velocity and its derivatives, it is a Robin boundary condition.

We recall from the previous subsections that such pointwise information on the traction and velocity is not common and is most commonly surrogated. The pressure P and flow rate Q are more easily obtainable and may be regarded as reasonable surrogates for velocity and pressure respectively. Defining $\boldsymbol{g}$ as in (6.1.2), we may surrogate (6.1.7) with

$$-\nu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\boldsymbol{n} + p\boldsymbol{n} + \boldsymbol{u} = P\boldsymbol{n} + \boldsymbol{g} \qquad \text{on } \Gamma. \tag{6.1.8}$$

Assuming that the $\boldsymbol{u}$ is dominated by its normal component $(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n}$ on Γ (as is common in our applications), another possible surrogate of (6.1.7) is given by:

$$-\nu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\boldsymbol{n} + p\boldsymbol{n} + \theta(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n} = P\boldsymbol{n} + \frac{Q}{A_\Gamma}\boldsymbol{n} \qquad \text{on } \Gamma \tag{6.1.9}$$

where θ is a suitable parameter and A_Γ is the area of the outlet. The same backflow instability concerns addressed for traction boundary conditions also apply in this instance, as similar energy arguments apply [31, 48].

As with flow rates and tractions, approaches based on variational minimization are also possible, obtained by minimizing:

$$\left|\int_\Gamma (p\boldsymbol{n} - P\boldsymbol{n})\right|^2 + \left|\int_\Gamma \theta\boldsymbol{u}\cdot\boldsymbol{n} - \frac{Q}{A_\Gamma}\right|^2 \tag{6.1.10}$$

subject to the constraint that $(\boldsymbol{u}, p)$ are velocity and pressure fields solving the incompressible Navier-Stokes equations respectively.

6.1.2 Determining Boundary data

The issue of how to determine boundary data to enforce when the given data is incomplete or missing is a less straightforward task. Which method is most appropriate again depends on the nature of the available measured data (if it exists at all) as well as computational time and power considerations.

A common approach is to assign Robin-type boundary conditions by using a *Windkessel-type* model to determine appropriate values of P and Q in (6.1.8) or (6.1.9). This approach is based on a physiological interpretation of arterial flow as behaving like an electrical circuit [127]. The models used to determine P and Q vary in complexity, ranging from simple algebraic equations to systems of differential equations [40, 98, 67]. This approach is physically well-justified and requires very little *a priori* information to implement. However, a common denominator for all of these approaches is that they are inherently unsteady and as such they are not suitable for steady problems. As discussed in Chapter 2, steady problems are attractive for clinical applications for a number of reasons and accordingly, we require a method for determining boundary data consistent with the steady setting.

Empirical approaches based on physical models represent a possible such method. These sorts of methods generally exploit some relationship between geometric information and a suitable value that can be assigned as boundary data. For example, [24, 27, 84, 60] establish empirical relationships between vessel diameter and flow rate based on measured *in vivo* data and minimization principles. While these are easy to implement and suitable for steady problems, they are constructed using large patient pools and may fail to give reliable results at the individual level. In some instances, they may be inconsistent with the geometry. We also note that geometric reconstructions are subject to noise and inaccuracy, naturally causing problems when using such reconstructions to determine flow

conditions. Figure 6.1 gives a flow chart detailing the process for determining and assigning boundary conditions.

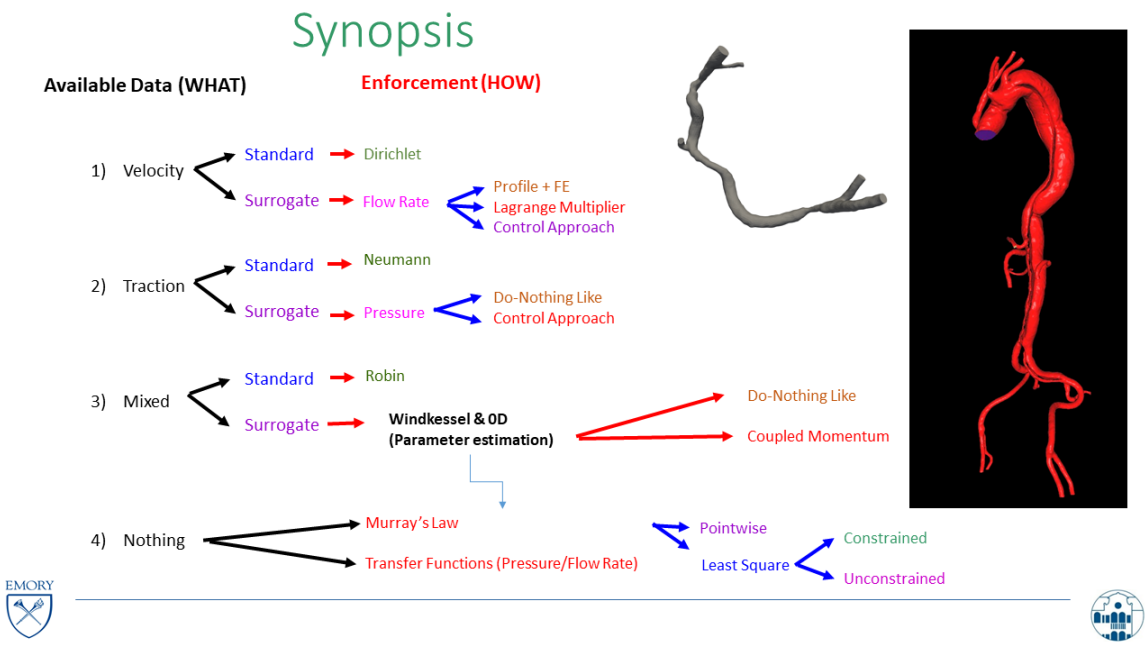

Figure 6.1: Flow chart: the determination and assignment of boundary conditions. *Image courtesy of Alessandro Veneziani.*

6.2 Determining Boundary Conditions: A Model Problem

We motivate our discussion by using a concrete example to better illustrate the problem. Consider the following domain Ω, reconstructed from a CT of a patient's right coronary artery (RCA) in Figure 6.2.

The boundary is defined by:

$$\partial\Omega = \Gamma_{wall} \cup \Gamma_{in} \cup \{\Gamma_i\}_{i=1}^{5}$$

where Γ_{wall} denotes the vessel wall, while Γ_{in} and $\Gamma_{i=1:5}$, labeled in Figure (6.2),

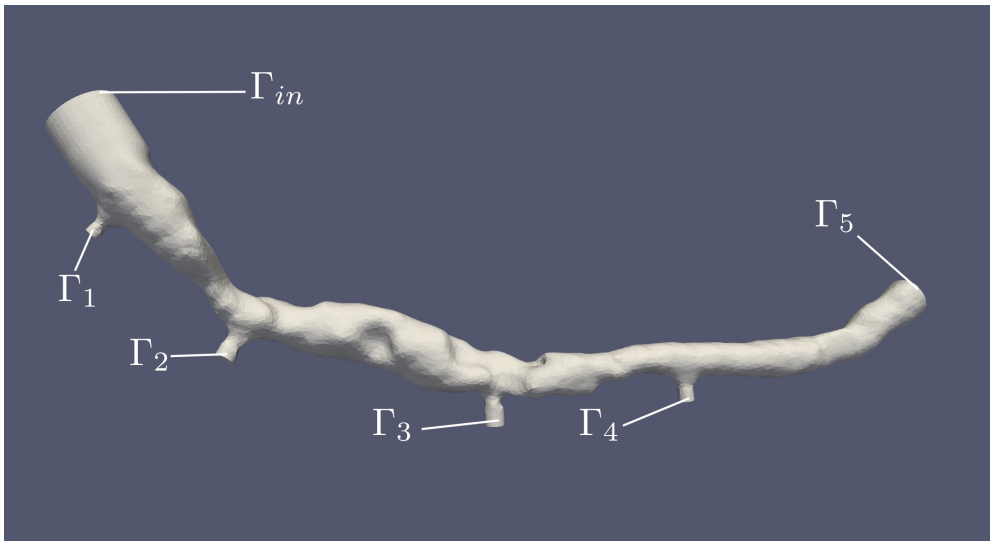

Figure 6.2: Reconstructed right coronary artery illustrating the problem of data deficiency.

denote the inlet and outlets for the blood flow respectively. The goal of this simulation is to obtain an accurate measure of the blood pressure distal to the stenosis (P_d) at the outlet Γ_5. Our given data consists of an estimate of the inflow rate Q_{in} derived from literature values and existing clinical measurements of other patients, and blood pressure anterior to the stenosis P_a at the inlet Γ_{in}.

For the boundary Γ_{in}, we may use the methods discussed in Section 6.1 to enforce our flow estimate Q_{in} as a boundary condition. For the purposes of stability and simplicity, we will construct a parabolic inflow function $\boldsymbol{g}_{in}$, properly scaled so that:

$$\int_{\Gamma_{in}} \boldsymbol{g}_{in} \cdot \boldsymbol{n} = -Q_{in} \tag{6.2.1}$$

where the negative sign is in order to enforce the condition as an inflow, rather than an outflow. Parabolic profiles are a good choice for circular or nearly-circular outlets, as is the case here; for other geometries a different type of flow profile may be more appropriate. The negative sign in (6.2.1) enforces the condition as an inflow, rather than an outflow. Along the vessel wall we enforce the no slip

boundary condition $\boldsymbol{u} = \mathbf{0}$. Note that our conditions for Γ_{in} and Γ_{wall} are both of Dirichlet type. Our model problem then reads:

$$
\begin{aligned}
-\nu\nabla\cdot\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T\right)+\rho\left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u}+\nabla p &= \boldsymbol{f} \quad \text{in } \Omega \\
\nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega \\
\boldsymbol{u} &= \boldsymbol{g}_{in} \quad \text{on } \Gamma_{in} \\
\boldsymbol{u} &= \mathbf{0} \quad \text{on } \Gamma_{wall}
\end{aligned}
\qquad , \tag{6.2.2}
$$

Referring to (6.2.2), we see that the problem is not well-defined. In particular, we do not have any boundary conditions prescribed for the outlets Γ_i, and based on our given data, there is no obvious way to assign them. Our problem is now to determine suitable boundary conditions for the outflows without incurring large additional computational costs while preserving the reliability of the solution.

6.2.1 Traction-free Boundary Conditions

The naive approach would be to assign so-called *traction-free* boundary conditions at each Γ_i. This is perhaps the most common method for dealing with outflows [56], and was used for many of the simulations shown in this work. These are often referred to as *do-nothing* boundary conditions as they are imposed automatically by a weak formulation of the problem. We demonstrate how this occurs by writing (6.2.2) in its weak form.

Let $\Gamma_D = \Gamma_{wall} \cup \Gamma_{in}$ be the Dirichlet portion of $\partial\Omega$ and $\Gamma_N = \cup_{i=1}^{5}\Gamma_i$ be the outflows (where we do not have boundary data). Proceeding from similar arguments as shown before, we choose an appropriate functional space $V \times Q$ for velocity and pressure respectively and recast the problem as the following: Find $\boldsymbol{u} \in V$

and $p \in Q$ such that for all $\boldsymbol{v} \in V$, $q \in Q$:

$$\begin{aligned} -\int_\Omega \nu \nabla \cdot \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) \cdot \boldsymbol{v} + \int_\Omega \rho \left(\boldsymbol{u} \cdot \nabla\right) \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega \nabla p \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \\ \int_\Omega (\nabla \cdot \boldsymbol{u}) q = 0 \end{aligned} \tag{6.2.3}$$

We can introduce a lifting function to enforce the Dirichlet boundary conditions on Γ_D (see e.g. [99, 102]). We then apply integration by parts to obtain:

$$\begin{aligned} &\nu \int_\Omega \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) : \nabla \boldsymbol{v} + \int_\Omega \rho \left(\boldsymbol{u} \cdot \nabla\right) \boldsymbol{u} \cdot \boldsymbol{v} \\ &\quad - \int_\Omega p (\nabla \cdot \boldsymbol{v}) - \nu \int_{\Gamma_N} \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) \boldsymbol{n} \cdot \boldsymbol{v} + \int_{\Gamma_N} p\, \boldsymbol{n} \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}, \\ &\quad \int_\Omega (\nabla \cdot \boldsymbol{u}) q = 0 \end{aligned} \tag{6.2.4}$$

where $\boldsymbol{n}$ is the outward-facing unit normal vector to the boundary Γ_N. The highlighted terms above arise automatically from the application of integration by parts to (6.2.3). We see that by adding a term of the form:

$$\int_{\Gamma_i} \phi\, \boldsymbol{n} \cdot \boldsymbol{v} \tag{6.2.5}$$

to the right-hand side of the first equation in (6.2.4), one may then exploit the presence of the highlighted terms to enforce the condition:

$$-\nu (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \boldsymbol{n} + p \boldsymbol{n} = \phi \boldsymbol{n} \quad \text{on } \Gamma_i \tag{6.2.6}$$

in a weak (average) sense over Γ_i.

Traction-free boundary conditions are assigned by simply adding no terms of the form (6.2.5) to (6.2.2), which is why they are also called 'do-nothing' conditions. However, rather than 'doing nothing', as that name would suggest, one is

actually enforcing the condition:

$$\int_{\Gamma_i} \left(-\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n} + p\boldsymbol{n} \right) \cdot \boldsymbol{v} = 0 \tag{6.2.7}$$

at each i. Since $\nu << 1$ in our applications, we can assume the contributions of $\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \boldsymbol{n}$ are negligible and hence we may regard traction-free boundary conditions as enforcing at each i:

$$\int_{\Gamma_i} p\boldsymbol{n} \cdot \boldsymbol{v} = 0 \tag{6.2.8}$$

Therefore, assigning traction-free boundary conditions at each outlet in the above case is equivalent to enforcing that the mean pressure at each outlet is identically zero, and more problematically, that the pressure drop is the same at each outlet [56]. Such a flow configuration is obviously non-physical for our model problem; we do not expect the mean pressure drop across each branch to be the same, particularly when defects such as stenoses, known to cause pressure drops, are present [40, 87]. In order to produce a reliable and physically accurate solution to (6.2.2), this method is not viable and we must develop a more sophisticated approach.

Briefly, we also note that one can prescribe a given pressure P_i at an outlet Γ_i by letting $\phi = P_i$ in (6.2.5) [39]. However, given that the distal pressure is generally unknown (and in fact in this case what we are trying to compute), this is not helpful for our model problem. Indeed, for most clinical applications, pressure distal to defects is difficult to measure non-invasively.

6.2.2 Physical Models and Murray's Law

Without any patient data to rely on, and traction-free conditions not being an option, we must use a different method. As suggested in Section 6.1.2, a possible and sensible technique may be to employ some sort of physical model to assign our boundary conditions. In hemodynamics, *Murray's Law* is a commonly-employed model for these types of problems.

Murray's Law for a branching lumen vessel states that for a parent vessel of radius r with n daughter branches with radii r_1, r_2, ..., r_n the following relationship holds between the branch radii:

$$r^\kappa = r_1^\kappa + r_2^\kappa + ... + r_n^\kappa \tag{6.2.9}$$

Murray's law is derived by minimizing the work done by laminar flow through a vessel, and in its original formulation had $\kappa = 3$ [84, 112]. Later work has suggested different values for κ, with the flow regime and geometrical considerations causing κ to vary between $7/3$ and 3, with the lower values corresponding to turbulent flow regimes [120, 60]. Since we are assuming steady non-turbulent flow, we will use the original exponent of $\kappa = 3$ in our computations; however we will continue to denote the exponent as κ in the interest of consistency and generality. Many experiments have been conducted to validate Murray's law, with the experiments confirming this relationship, though some deviation is to be expected [57, 112, 126].

A consequence of Murray's law is that it predicts a constant relationship Q_i/r_i^κ between the radius r_i and flow rate Q_i for all branches Γ_i [112]. As we know Q_{in} and we can obtain the radii r_i for each branch from our geometry Ω, from this relationship we obtain the following estimate Q_i for the outflow rate at a branch

Γ_i:

$$Q_i = \left(\frac{r_i}{r_{in}}\right)^{\kappa} Q_{in} \tag{6.2.10}$$

This suggests a possible approach of assigning outflow conditions at each Γ_i with parabolic outflow functions $\boldsymbol{g}_i$ scaled such that:

$$\int_{\Gamma_i} \boldsymbol{g}_i \cdot \boldsymbol{n} = Q_i \tag{6.2.11}$$

with the right-hand side positive in order to enforce this as an outflow.

While such an approach is attractive and physically well-justified, it cannot be applied in general. While experiments confirm that Murray's law holds approximately, as mentioned previously, discrepancies are often present. Arterial disease, which affects a significant portion of our cases of interest, is known to cause vessel deformation leading to deviation from Murray's law [112, 111]. Lastly, as our geometry is based on a reconstruction of an artery from a CT, we expect additional noise and error to induce some additional discrepancies. Therefore, the Q_i predicted by (6.2.10) will not satisfy:

$$\sum_{i=1}^{5} Q_i = Q_{in} \tag{6.2.12}$$

in general; that is, these boundary conditions are incompatible with the incompressibility constraint, which requires the sum total of outflows to equal the sum total of inflows. While this approach is certainly a step forward from traction-free boundary conditions, it is not suitable for our problem (6.2.2) in its current form.

6.2.3 Unilateral Least-Squares Minimized Murray's Law

As seen in the previous section, Murray's Law offers us a physically valid approach for relating known quantities (branch radii r_i and inflow rate Q_{in}) to unknown quantities (Q_i) and is known experimentally to hold at least approximately [57, 112, 126]. From this, we can obtain a complete set of boundary conditions for our model problem (6.2.2). Unfortunately, this law cannot be applied directly as normal deviations are expected in addition to the further discrepancies from the physical model caused by the geometric reconstruction, causing an incompatibility with the incompressibility constraint. We therefore would like to modify Murray's Law accordingly in order to recover incompressibility.

The most immediate and obvious approach is to use a *Least-Squares minimization*: Find $\boldsymbol{Q} = [Q_1, Q_2, ..., Q_5]$ such that:

$$\begin{aligned} \arg\min_{\boldsymbol{Q}} \sum_{i=1}^{5} \left(Q_i - \left(\frac{r_i}{r_{in}} \right)^{\kappa} Q_{in} \right)^2 \\ \text{s.t. } \sum_{i=1}^{5} Q_i = Q_{in} \end{aligned} \tag{6.2.13}$$

This seems to satisfy all of our requirements; however there is a serious problem with this formulation. The method (6.2.13) allows for both positive and negative Q_i in general. This makes it unsuitable for determining our boundary conditions, as it may assign backflows at some of our outflow boundaries. As such flow behavior is non-physical for coronary arteries, we must ensure that each Q_i is an outflow by enforcing that $Q_i \geq 0$ for each i[1].

Taking this into account, our method for determining boundary conditions for (6.2.2) is given by the following algorithm:

Algorithm 6.2.1 ***Unilateral Least-Squares Minimized Murray's Law.*** *For a prob-*

[1]Backflows do occur in other vessels, in particular the Aorta, and when applying this approach to these vessels we may not necessarily wish to enforce $Q_i > 0$.

lem on a branching lumen vessel with a parent branch of radius r_{in} and n daughter branches with radius r_i, $i = 1 : n$, given an inflow rate Q_{in} the outflow rates Q_i at each daughter branch Γ_i are determined by solving the following problem: Find $\boldsymbol{Q} = [Q_1, Q_2, ..., Q_n]$ such that:

$$\begin{aligned} \arg\min_{\boldsymbol{Q}} \sum_{i=1}^{n} \left(Q_i - \left(\frac{r_i}{r_{in}} \right)^{\kappa} Q_{in} \right)^2 \\ s.t. \ \sum_{i=1}^{n} Q_i = Q_{in}, \\ Q_i \geq 0 \quad \forall i \end{aligned} \tag{6.2.14}$$

The computation of (6.2.1) is non-trivial, however software packages exist for solving such problems. We used the Matlab subroutine `lsqlin`, part of the Matlab Optimization toolbox [80].

6.3 Validation on Clinical Data

As detailed in the previous section, we developed the Unilateral Least-Squares Minimized Murray's Law (6.2.1) in order to efficiently solve the model pressure drop problem shown in Section 6.1 in a clinical setting. We validated the algorithm by solving the problem on a dataset obtained from clinical data consisting of $n = 7$ patients, with the data coming from three different locations. We found that our methods produced excellent agreement with the measured clinical data, while remaining simple, easy to implement, and computationally inexpensive. We will organize this section by briefly describing our study design and its objectives and methodology, presenting information regarding our dataset, and concluding with a presentation of our results.

6.3.1 Study Design

This study aims to validate the Unilateral Least-Squares Murray's Law (6.2.1) method for assigning boundary conditions for patient-specific CFD simulations on stenosed coronary arteries. Our goal is to successfully estimate the distal pressure P_d past the stenosis. For each patient, we use a CT to reconstruct the right-coronary artery (RCA). We reconstructed the main vessel, as well as the side branches; however we note that the number of reconstructed side branches varied on each case based on the CT. We included every side branch that we were able to successfully reconstruct. All geometric reconstructions were done using the freely available software Segment [52].

For each patient, we have a measurement of the pressure anterior to the stenosis P_a. We set the the blood density $\rho = 1.06$ g/cm^3 and kinematic viscocity $\nu = .035$ g/cm.s based on [40, 87]. For our inflow rate Q_{in} we used a value 5 ml/s for each patient. This estimate is based on the flow rate measurements found in [114, 87], adjusted to account for nitroglycerin dilation of the vessels during the CT, which causes an increase in vessel size of approximately 23 percent [38].

The boundary conditions were assigned following the procedures outlined in the preceding section. We enforced the inflow with Dirichlet boundary conditions by constructing a parabolic inflow function $\boldsymbol{g}_{in}$ scaled to match our Q_{in} as described in (6.2.1). At the vessel walls we assign the no-slip boundary condition $\boldsymbol{u} = \boldsymbol{0}$.

For the outflows, we determine the outflow rate Q_i at each branch Γ_i with the Least-Squares Minimized Murray's Law algorithm (6.2.1). We enforce the flow rates for the outflow branches Γ_1, Γ_2, ..., Γ_{n-1}, with weighted parabolic outflow profiles $\boldsymbol{g}_i$ constructed as shown in (6.2.11). We may leave the outlet of interest Γ_n as a traction-free boundary condition. As the inflow and all other outflows are enforced strongly with Dirichlet profiles, the incompressibility condition ensures

that the outflow rate Q_n at Γ_n will be satisfied according to (6.2.1) [39, 56, 121].

This leads to the following problem:

Problem 6.3.1 ***Problem for Validation Study:*** *For the clinical validation of (6.2.1), for each patient in our data pool with n total outflow branches, we solve the following stationary incompressible Navier-Stokes problem:*

$$
\begin{aligned}
-\nu\nabla\cdot\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T\right)+\rho\left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u}+\nabla p &= \boldsymbol{f} \quad \textit{in } \Omega \\
\nabla\cdot\boldsymbol{u} &= 0 \quad \textit{in } \Omega \\
\boldsymbol{u} &= \boldsymbol{g}_{in} \quad \textit{on } \Gamma_{in}, \quad \int_{\Gamma_{in}} \boldsymbol{g}_{in}\cdot\boldsymbol{n} = -Q_{in} \\
\boldsymbol{u} &= \boldsymbol{g}_i \quad \textit{on } \Gamma_i, \ \int_{\Gamma_i} \boldsymbol{g}_i\cdot\boldsymbol{n} = Q_i, \ i = 1:n-1 \\
\boldsymbol{u} &= \boldsymbol{0} \quad \textit{on } \Gamma_{wall}
\end{aligned}
$$

with $\nu = .035$ g/cm.s, ρ=1.06 g/cm^3, Q_{in}=5 ml/s and Q_i determined according to Unilateral Least-Squares Minimized Murray's Law (6.2.1). We then compare our computed pressure distal to the stenosis P_d^{comp} with the measured value P_d^{meas} for the same parameter.

For the discretization and solution of the Problem 6.3.1, we used the Finite Element method with Taylor-Hood $\mathbb{P}^2/\mathbb{P}^1$ elements for velocity pressure respectively. To solve the nonlinear problem we used Picard iterations with Grad-Div stabilization ($\xi = 1$) for additional stability (4.1.2), terminating the iterations when relative difference between consecutive Picard iterations for the pressure drop $p|_{\Gamma_{in}} - p|_{\Gamma_n}$ (values taken from the centroid of each surface) dropped below 1e-3. As our meshes were relatively small, we used sparse direct solvers to solve the monolithic problem at each iteration. Computations were performed using the software FreeFem++ on a 2017 MacBook Pro.

We conclude this section by noting that, although the anterior pressure P_a

is our known patient data, we do not directly use this information during our computation of (6.3.1). However, we do in fact use this information to recover our variable of interest, the distal pressure P_d^{comp}. Note briefly that the pressure p only appears in (6.3.1) under the gradient operator and therefore we are technically computing a pressure field $p + C$, with p unique up to an additive constant C, which is in turn determined by the Neumann boundary conditions. We may easily recover P_d^{comp} by applying the formula:

$$P_d^{comp} = P_a - \left(p|_{\Gamma_{in}} - p|_{\Gamma_n}\right) \tag{6.3.1}$$

In our particular case, $C = 0$ as we applied homogenous Neumann (traction-free) boundary conditions on Γ_n; however the above formula (6.3.1) is independent of C.

6.3.2 Patient Data Pool

Our dataset consisted of $n = 7$ patients from three different data sites: A ($n = 2$), B ($n = 4$), and C ($n = 1$). For each patient we had the measured pressure anterior to the stenosis P_a and measured pressure distal to the stenosis P_d^{meas}. We provide the patient numbering, pressure measurements, and number of branches in Table 6.1 (note that P_a and P_d^{meas} are rounded to nearest mmHg). In Table 6.2 we provide the measurements of the vessel branches for each case, which we then use as an input to the Unilateral Least-Squares Minimized Murray's Law (6.2.1) to compute the flow splittings provided in Table 6.3.

Each case is a right coronary artery (RCA) with the inlet branch Γ_{in} defined where the artery exits from the Aorta. Flow extensions (respecting vessel size) were added to Γ_{in} in order to allow the parabolic flow profile to develop before entering the main vessel. Each outlet branch was treated similarly to the example

Patient Data, Validation Study				
Case number	Data Site	num. of outflows	P_a (mmHg)	P_d^{meas} (mmHg)
1	A	5	100	88
2	A	3	88	74
3	B	3	70	60
4	B	4	92	89
5	B	6	95	86
6	B	4	118	92
7	C	4	80	74

Table 6.1: Patient information for validation study.

Branch radii (cm), Validation Study							
	Case Number						
Vessel Branch	1	2	3	4	5	6	7
Γ_{in}	.229	.239	.202	.166	.226	.221	.186
Γ_1	.047	.053	.078	.077	.050	.091	.104
Γ_2	.066	.072	.103	.035	.040	.101	.092
Γ_3	.067	.219	.151	.038	.034	.137	.107
Γ_4	.050	-	-	.136	.107	.147	.148
Γ_5	.136	-	-	-	.054	-	-
Γ_6	-	-	-	-	.148	-	-

Table 6.2: Validation study, patient vessel branch radii (in cm).

Flow Splitting (% of $Q_{in} = 5$ ml/s), Validation Study							
	Case Number						
Vessel Branch	1	2	3	4	5	6	7
Γ_1	15.26%	7.45%	18.94%	18.43%	10.80%	14.39%	17.77%
Γ_2	16.81%	9.07%	26.47%	9.17%	10.25%	16.96%	12.33%
Γ_3	16.90%	83.48%	54.59%	9.49%	10.03%	31.63%	19.16%
Γ_4	15.47%	-	-	62.91%	20.21%	37.01%	50.73%
Γ_5	35.55%	-	-	-	11.04%	-	-
Γ_6	-	-	-	-	37.66%	-	-

Table 6.3: Validation study, flow splittings computed by (6.2.1). Note that these are percentages of $Q_{in} = 5$ for each case.

shown in Figure 6.2, with the branch being truncated several millimeters after its splitting from the main vessel. Flow extensions were added on a case-by-case basis depending on how dissimilar the specific branch was from a circle. The location of the final outlet Γ_n on the main vessel was determined based on the clinical measurement location of P_d^{meas}.

6.3.3 Results

The study produced strong results and supports the conclusion that (6.2.1) is an effective and inexpensive way to prescribe boundary conditions for our problem. We found broad agreement between P_d^{comp} and P_d^{meas}, with the results being reported in Table 6.4 (P_d^{comp} and P_d^{meas} rounded to nearest mmHg).

Patient Data, Validation Study Results		
Case number	P_d^{meas} (mmHg)	P_d^{comp} (mmHg)
1	88	85
2	74	71
3	60	59
4	89	85
5	86	89
6	92	102
7	74	74

Table 6.4: Results for validation study.

In Figure 6.3 we display our results using two plots common in medical statistics: a line-of-agreement plot (left) and a Bland-Altman plot (right). These plots are designed to assess the agreement of two different measurement techniques. In our case, these measurements are P_d^{meas} and P_d^{comp}. The line-of-agreement plot is constructed by plotting one measurement along the y-axis and the competing measurement along the x-axis, with stronger agreement between the two resulting in clustering near the line of agreement $y = x$. Bland-Altman plots are constructed by plotting the the mean of the two measurements along the x-axis and the dif-

ference between the two measurements along the y-axis, with stronger agreement resulting in clustering near the line $y = 0$. A more thorough description of these plotting methods can be found in [2].

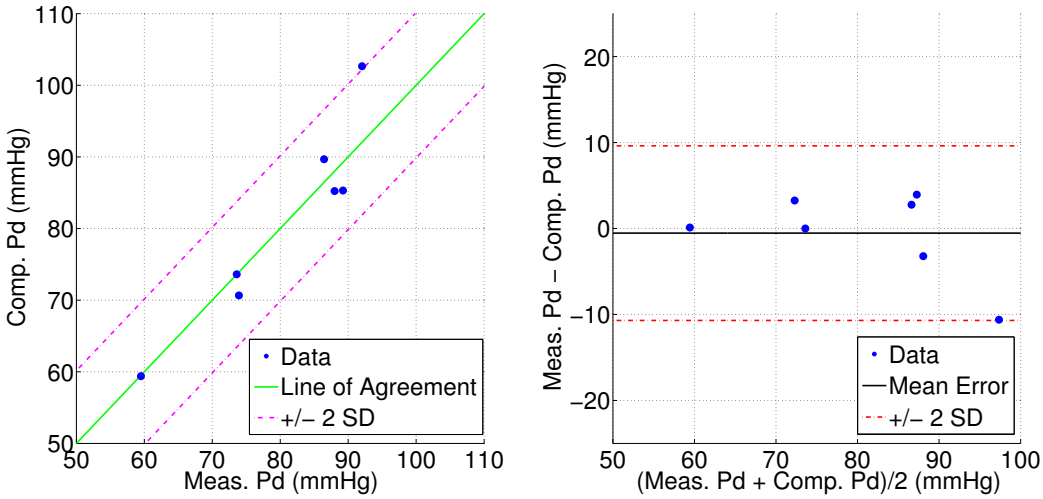

Figure 6.3: Agreement between P_d^{meas} and P_d^{comp} for the validation study.

The plots show strong agreement between P_d^{comp} and P_d^{meas}, with the data points clustered near the line of agreement on the left and near the line $y = 0$ on the right. The mean signed difference across all seven cases was -.53 mmHg, suggesting that our method did not produce any systemic error. This is significant in Bland-Altman analysis, as mean signed difference far from zero indicates systematic disagreement between the two measurement methods [2]. The mean unsigned difference was 3.41 mmHg with a standard deviation of 5.07 mmHg. This is in line with expected variability for blood pressure across different measurement times and methods from the available medical literature [69, 85, 89].

We acknowledge that our sample size of $n = 7$ is small and that further validation across a larger number of patients is still required. Additionally, future work in this direction should incorporate more patient and geometry-specific information into the inflow assumption. In this study we used a constant inflow value for all cases, however more realistic inflow values will likely increase the

overall accuracy of our evaluation. Despite the small patient size and the use of several simplifying assumptions, we nonetheless observe compelling results from this study. The results obtained seem to strongly support the effectiveness of (6.2.1) as an accurate and inexpensive way to prescribe boundary conditions in data-deficient settings.

Chapter 7

Conclusions and Future Research

Recent improvements in medical imaging, image reconstruction, numerical methods for computational fluid dynamics, as well as a general increase in computing power, have now made CFD a potentially valuable clinical tool in hemodynamics. However, several important challenges remain before its large-scale adoption can be fully realized, with many of these difficulties requiring new approaches in CFD. Broadly, these concerns can be categorized into three groups: efficiency, stability, and reliability.

Efficiency

Much of the difficulty in clinical CFD arises from the restrictive demands on computational turnaround time. While unsteady simulations have long been employed for hemodynamics problems in other research settings, the time scale required to compute a full unsteady simulation is often large. In order for clinical CFD simulations to be competitive with *in-vivo* tests, ideally simulations should be completed within minutes, or hours at the most. We feel that the use of steady simulations as a surrogate for the unsteady time average is a possible way to reduce the computational time frame to a clinically viable level, though the ab-

sence of a time derivative has long been known to make this problem difficult numerically.

We introduced methods to counter this difficulty by extending the popular and efficient class of inexact algebraic factorization methods for unsteady problems (shown in e.g. [123, 125, 100, 101]) to the steady setting. We found that these methods retain many of the advantages of their unsteady analogues: in particular, they are easy to implement and amenable to preconditioning, with the associated preconditioners being effective, easy to implement, cheap to apply, and crucially, do not require any updating during the nonlinear process.

We developed this class of solvers using two approaches: one in which we introduced a semi-explicit discretization of the problem (Chapter 3) and one in which we used grad-div stabilization (Chapter 4). Both classes of methods are effective but have their drawbacks. The methods from Chapter 3 often require under-relaxation delaying their convergence somewhat, while the methods from Chapter 4 require the introduction of a grad-div parameter. The grad-div parameter ξ is known to cause additional numerical difficulties when large, and though we found that large values of ξ are not usually necessary, it may nonetheless not be a viable choice for some problems.

Though these types of methods were presented separately, the development and validation of methods combining both approaches is a potential area for future work. Introducing small-grad div parameters into the methods in Chapter 3 make the approximate Schur complement even easier to solve and allow one to use a slightly larger parameter α, thereby reducing the amount of necessary underrelaxation. The other way around is also true, in that using the semi-explicit formulations from Chapter 3 may allow one to reduce the necessary size of ξ. Investigating what formulations ultimately allow for the maximization of the positive aspects of each class of methods while minimizing the drawbacks may be a

worthy topic for future investigation.

Stability

The use of steady solvers requires the introduction of nonlinear iterative schemes whose convergence may be numerically delicate. In particular, small scale dynamics, such as recirculating regions of flow, originating from aspects of the geometry are known to delay or in some cases even prevent convergence. In hemodynamics, such flow behavior may occur as a result of physical pathologies, such as aneurysms or stenoses, or spuriously due to irregularities in the reconstructed geometry arising from noisy input data.

Stabilization techniques based on adding numerical viscosity to convection-dominated regions are well-established [20, 59, 42, 102, 99]. Such methods are effective when the instabilities arise from convection, but in cases such as the ones described earlier, where the instabilities come from recirculating flow and similar phenomena, the instability is not caused by convection-domination and these effectiveness of these approaches may be limited.

We introduced new stabilization approaches based on these methods, but modified such that numerical viscosity is not added in the high-convection regions, but in areas where the flow is irregular and small-scale dynamics are present, which may be low-convection regions in general. In order to accomplish this, we used methods inspired by differential-filtering techniques used in turbulence modeling (see e.g. [72, 71, 10, 15]). Though our stabilization methods do not model turbulence, the filters used for turbulence modeling are nonetheless useful in identification of disturbed flow regions, which we then use for applying artificial viscosity in a manner similar to the standard stabilization methods. We introduced both consistent and strongly consistent versions of our method and demonstrated its effectiveness with two- and three-dimensional numerical tests.

For future work, an obvious direction is to investigate the approaches introduced Chapter 5 with different choices of filters. The determination of optimality criteria for of the relevant filtering parameters is another potential topic of interest. A rigorous convergence proof for the strongly consistent version of the method is also desired.

Reliability

The last concern which we addressed is perhaps the most important, the development of reliable methods. In this setting, *reliability* refers to the ability of a simulation to provide results that are physically meaningful, allowing clinicians to use the results to inform treatment decisions for specific patients. Due to the lack of reliable patient-specific data often found in clinical settings, this problem is an extremely challenging one as it requires the proper surrogation and determination of boundary conditions from a limited set of information.

While approaches based on variational problems have been proposed [41], these suffer from high numerical costs which make their use in this setting difficult due to aforementioned restrictions on computational time. Other methods, such as Windkessel-type models, are effective for unsteady settings, but the lack of time-dependence prevents their use for steady problems. We introduced a novel approach, suitable for steady problems, that does not require any variational minimization techniques. This method incorporates geometric information with a widely-used physical model, Murray's Law, to determine boundary data. We applied this approach to a patient-specific dataset and confirmed that it gave reliable results.

There is ample room for future work in this area. The method we proposed is valuable not only in and of itself, but also as a potential framework for future methods. Given a physical model, patient input data, and geometrical infor-

mation, the same framework could also be to develop analogous techniques for different physical models. This may have relevance outside of hemodynamics and could extend to CFD in general.

Regarding the specific method (6.2.13) introduced in Chapter 6, further validation on a larger patient pool and for more general vessels, such as the carotid arteries or other parts of the coronary tree, are necessary. Investigating different choices of exponent in Murray's law and how it affects the performance of (6.2.13) is also potentially worthwhile. Additionally, it may be worth investigating the use of (6.2.13) with different methods for imposing boundary conditions, such as the Lagrange-multiplier flux formulations shown in [121, 39]. Finally, for cases in which pressure and flow data is available, an extension of the method for Robin-type boundary conditions may also be useful.

Bibliography

[1] Computational Fluid Dynamics Round Robin Study, https://fdacfd.nci.nih.gov/.

[2] D.G. Altman and J. M. Bland. Measurement in Medicine: The Analysis of Method Comparison Studies. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32(3):307–317, 1983.

[3] Aneurisk-Team. AneuriskWeb project website, http://ecm2.mathcs.emory.edu/aneuriskweb. Web Site, 2012.

[4] L. Antiga, P. Joaquim, and D. Steinman. From image data to computational domains. In L. Formaggia, A. Quarteroni, and A. Veneziani, editors, *Cardiovascular Mathematics*, pages 123–175. Springer, 2009.

[5] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Lordache, A. Remuzzi, and D. Steinman. An image-based modeling framework for patient-specific computational hemodynamics. *Medical and Biological Engineering and Computing*, 46:1097–1112, 2008.

[6] Y Bazilevs, VM Calo, JA Cottrell, TJR Hughes, A Reali, and G Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1-4):173–201, 2007.

[7] Roland Becker. *Weighted error estimators for finite element approximations of the incompressible Navier-Stokes equations*. IWR, 1998.

[8] M. Benzi and M. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Sci. Comput.*, 28:2095–2113, 2006.

[9] Michele Benzi, Gene H Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[10] L Bertagna, A Quaini, and A Veneziani. Deconvolution-based nonlinear filtering for incompressible flows at moderately large reynolds numbers. *International Journal for Numerical Methods in Fluids*, 14(8):463–488, 2016.

[11] C. Bertoglio, P. Moireau, and J. F. Gerbeau. Sequential parameter estimation for fluid-structure problems. Application to hemodynamics. *International Journal for Numerical Methods in Biomedical Engineering*, 28(4):434–455, 2012.

[12] J Borggaard, T Iliescu, and JP Roop. A bounded artificial viscocity large eddy simulation model. *SIAM J. Numer. Anal.*, 47:622–645, 2009.

[13] S. Börm and S. Le Borne. $\mathcal{H}$-LU factorization in preconditioners for augmented Lagrangian and grad-div stabilized saddle point systems. *International Journal for Numerical Methods in Fluids*, 68:83–98, 2012.

[14] G. Bosch and W. Rodi. Simulation of vortex shedding past a square cylinder with different turbulence models. *International Journal for Numerical Methods in Fluids*, 28:601–616, 1998.

[15] AL Bowers and LG Rebholz. Numerical study of a regularization model for incompressible flow with deconvolution-based adaptive nonlinear filtering. *Comput. Methods Appl. Mech. Eng.*, 258:1–12, 2013.

[16] M. Braack and E. Burman. Local projection stabilization for the Oseen problem and its interpretation as a variational multiscale method. *SIAM J. Numer. Anal.*, 43(6):2544–2566, 2006.

[17] M. Braack, E. Burman, V. John, and G. Lube. A numerical method for solving incompressible viscous flow problems. *Computer Methods in Applied Mechanics and Engineering*, 2(1):12–26, 1967.

[18] Haim Brezis. *Operateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert*, volume 5. Elsevier, 1973.

[19] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[20] Alexander N Brooks and Thomas JR Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer methods in applied mechanics and engineering*, 32(1-3):199–259, 1982.

[21] Abigail Brouwers, Benjamin Cousins, Alexander Linke, and Leo Rebholz. New connections between finite element formulations of the Navier-Stokes equations. Technical Report 1516, WIAS, Berlin, 2010.

[22] M. Case, V. Ervin, A. Linke, and L. Rebholz. A connection between Scott-Vogelius elements and grad-div stabilization. *SIAM Journal on Numerical Analysis*, 49(4):1461–1481, 2011.

[23] M.A. Castro, J.R. Cebral, and C.M. Putman. Computational fluid dynamics modeling of intracranial aneurysms: Effects of parent artery segmentation on intra-aneurysmal hemodynamics. *American Journal of Neuroradiology*, 27:1703–1709, 2006.

[24] J.R. Cebral, M.A. Castro, C.M. Putman, and N. Alperin. Flow-area relationship in internal carotid and vertebral arteries. *Physiological Measurements*, 29:585–594, 2008.

[25] JR. Cebral, F. Mut, D. Sforza, R. Lohner, E. Scrivano, R. Lylyk, and CM. Putman. Clinical application of image-based cfd for cerebral aneurysms. *International Journal for Numerical Methods in Biomedical Engineering*, 27(7):977–992, 2011.

[26] T. Charnyi, T. Heister, M. Olshanskii, and L. Rebholz. On conservation laws of Navier-Stokes Galerkin discretizations. *Journal of Computational Physics*, 337:289–308, 2017.

[27] C. Chnafa, P. Bouillot, O. Brina, D.M.A. Delattre M.I. Vargas, K.O. Lovblad, V.M. Pereira, and D.A. Steinman. Vessel calibre and flow splitting relationships at the internal carotid artery terminal bifurcation. *Physiological Measurements*, 38:2044–2057, 2017.

[28] Alexandre Joel Chorin. Stabilized finite element methods for the generalized Oseen problem. *Comput. Methods Appl. Mech. Eng.*, 196:853–866, 2007.

[29] Burt Cohen. FDA Approves HeartFlow FFR_{CT}: Non-Invasive Method for Determining Coronary Ischemia, 2014.

[30] Wouter Couzy. *Spectral element discretization of the unsteady Navier-Stokes equations and its iterative solution on parallel computers*. Ph.D., EPFL, 1995.

[31] J. Craske and M.V. Reeuwijk. Robust and accurate open boundary conditions for incompressible turbulent jets and plumes. *Computers and Fluids*, 86:284–297, 2013.

[32] M. D'Elia, L. Mirabella, T. Passerini, M. Perego, M. Piccinelli, C. Vergara, and A. Veneziani. Applications of variational data assimilation in computational hemodynamics. In D. Ambrosi, A. Quarteroni, and G. Rozza, editors, *Modeling of Physiological Flows*. Springer, 2011.

[33] S. Deparis, G. Grandperrin, and A. Quarteroni. Parallel preconditioners for the unsteady Navier-Stokes equations and applications to hemodynamics simulations. *Computers and Fluids*, 92:253–273, 2014.

[34] M.D. Deshpande, D. Giddens, and R.F. Mabon. Steady laminar flow through modeled vascular stenoses. *Journal of Biomechanics*, 9(4):165–174, 1976.

[35] S. Dong, G.E. Karniadakis, and C. Chryssostomidis. A robust and accurate outflow boundary condition for incompressible flow simulations on severely-truncated unbounded domains. *Journal of Computational Physics*, 261:83–105, 2013.

[36] S. Dong and J. Shen. A pressure correction scheme for generalized form of energy-stable open boundary conditions for incompressible flows. *Journal of Computational Physics*, 291:254–278, 2015.

[37] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2014.

[38] R. Feldman, C. J. Pepine, and C. R. Conti. Magnitude of Dilation of Large and Small Coronary Arteries by Nitroglycerin. *Circulation*, 64:324–333, 1981.

[39] 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(1):376–401, 2006.

[40] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics*. Springer, 2009.

[41] 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(6):2769–2794, 2008.

[42] Leopoldo P Franca and Thomas JR Hughes. Convergence analyses of Galerkin least-squares methods for symmetric advective-diffusive forms of the Stokes and incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 105(2):285–298, 1993.

[43] Alain Gauthier, Fausto Saleri, and Alessandro Veneziani. A fast preconditioner for the incompressible Navier Stokes equations. *Computing and Visualization in Science*, 6(2-3):105–112, 2004.

[44] Paola Gervasio, Fausto Saleri, and Alessandro Veneziani. Algebraic fractional-step schemes with spectral methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 214(1):347–365, 2006.

[45] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[46] R. Glowinski. Finite element methods for incompressible viscous flow. In P.G. Ciarlet and J.-L. Lions, editors, *Handbook of numerical analysis*, volume 9. North-Holland, 2003.

[47] BD. Gogas, B. Yang, T. Passerini, A. Veneziani, M. Piccinelli, G. Esposito, E. Rasoul-Arzrumly, M. Awad, G. Mekonnen, OY. Hung, B. Holloway,

M. McDaniel, D. Giddens, SB. King, and H. Samady. Computational fluid dynamics applied to virtually deployed drug-eluting coronary bioresorbable scaffolds: Clinical translations derived from a proof-of-concept. *Global Cardiology Science and Practice*, 56(4):428–436, 2014.

[48] V. Gravemeier, A. Comerford, L. Yosihara, M. Ismail, and W. Wall. A novel formulation for neumann inflow boundary conditions in biomechanics. *International Journal for Numerical Methods in Biomedical Engineering*, 28(5):560–573, 2012.

[49] J-L Guermond and L Quartapelle. Calculation of incompressible viscous flows by an unconditionally stable projection FEM. *Journal of Computational Physics*, 132(1):12–33, 1997.

[50] J-L Guermond and L Quartapelle. On the approximation of the unsteady Navier–Stokes equations by finite element projection methods. *Numerische Mathematik*, 80(2):207–238, 1998.

[51] Nathan D. Heavner. Locally chosen grad-div stabilization parameters for finite element discretizations of incompressible flow problems. 2014.

[52] E. Heiberg, J. Sjogren, M. Ugander, M. Carlsson, H. Engblom, and H. Arheden. Design and Validation of Segment- a Freely Available Software for Cardiovascular Image Analysis. *BMC Medical Imaging*, 10(1), 2010.

[53] T. Heister and G. Rapin. Efficient augmented Lagrangian-type preconditioning for the Oseen problem using grad-div stabilization. *Int. J. Numer. Meth. Fluids*, 71:118–134, 2013.

[54] H.V. Henderson and S.R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23:53–60, 1981.

[55] Martin Ofstad Henriksen and Jens Holmen. Algebraic splitting for incompressible Navier–Stokes equations. *Journal of Computational Physics*, 175(2):438–453, 2002.

[56] J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 22(5):325–352, 1996.

[57] K. Horsfield and G. Cumming. Angles of branching and diameters of branches in the human bronchial tree. *Bulletin of Mathematical Biophysics*, 29:245–259, 1967.

[58] T. J R Hughes and A. Brooks. Multi-dimensional upwind scheme with no crosswind diffusion. *American Society of Mechanical Engineers, Applied Mechanics Division, AMD*, 34, 1979.

[59] Thomas JR Hughes, Leopoldo P Franca, and Marc Balestra. A new finite element formulation for Computational Fluid Dynamics: Circumventing the Babuška-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59(1):85–99, 1986.

[60] Y. Huo and G.S. Kassab. Intraspecific scaling laws of vascular trees. *J R Soc Interface*, 9:190–200, 2012.

[61] G. Iaccarino, A. Ooi, P.A. Durbin, and M. Behnia. Reynolds averaged simulation of unsteady separated flow. *Int. J. Heat Fluid Flow*, 24:147–156, 2003.

[62] E. Jenkins, V. John, A. Linke, and L. Rebholz. On the parameter choice in grad-div stabilization for the Stokes equations. *Advances in Computational Mathematics*, 40(2):491–516, 2014.

[63] Volker John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier Stokes equations. *International Journal for Numerical Methods in Fluids*, 40(6):775–798, 2002.

[64] Volker John, Petr Knobloch, and Julia Novo. Finite elements for scalar convection-dominated equations and incompressible flow problems: a never ending story? *Computing and Visualization in Science*, 2018.

[65] Volker John and Gunar Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids*, 37(8):885–903, 2001.

[66] Ohannes A Karakashian. On a Galerkin-Lagrange multiplier method for the stationary Navier-Stokes equations. *SIAM Journal on Numerical Analysis*, 19(5):909–923, 1982.

[67] H.J. Kim, I.E. Vignon-Clementel, J.S. Coogan, C.A. Figueroa, K.E. Jansen, and C.A. Taylor. Patient-Specific Modeling of Blood Flow and Pressure in Human Coronary Arteries. *Annals of Biomedical Engineering*, 38:3195–3209, 2010.

[68] D.N. Ku and D. Giddens. Pulsatile flow in a model carotid bifurcation. *Arteriosclerosis, Thrombosis, and Vascular Biology*, 3(1):31–39, 1983.

[69] M. E. Lacruz, A. Kluttig, O. Kuss, D. Tiller, D. Medenwald, S. Nuding, K. H. Greiser, S. Frantz, and J. Haerting. Short term blood pressure variability-variation between arm side, body position, and successive measurements: a population-based cohort study. *BMC Cardiovascular Disorders*, 17(1):31, 2017.

[70] W. Layton. *An Introduction to the Numerical Analysis of Viscous Incompressible Flows*. SIAM, Philadelphia, 2008.

[71] W. Layton, C. Manica, M. Neda, and L. Rebholz. Numerical analysis and computational testing of a high accuracy Leray-deconvolution model of turbulence. *Numerical Methods for Partial Differential Equations*, 24(2):555–582, 2008.

[72] W Layton, LG Rebholz, and C Trenchea. Modular nonlinear filter stabilization of methods for higher reynolds numbers flow. *J. Math. Fluid Mech.*, 14:325–354, 2012.

[73] William J Layton and Leo G Rebholz. *Approximate deconvolution models of turbulence: analysis, phenomenology and numerical analysis*, volume 2042. Springer Science & Business Media, 2012.

[74] A. Lefieux and A. Viguerie. On Optimization Approaches to Boundary Conditions Setting in Computational Hemodynamics. SIAM Conference on Computational Science and Engineering, 2017.

[75] T. Leonard, L. Y. M. Gicquel, N. Gourdain, and F. Duchaine. Steady/Unsteady Reynolds-Averaged Navier-Stokes and Large Eddy Simulations of a Turbine Blade at High Subsonic Outlet Mach Number. *J. Turbomach*, 137(4), 2015.

[76] J Leray. Essai sur le mouvement d'un fluide visqueux emplissant l'espace. *Acta. Math*, 63:193–248, 1934.

[77] A. Linke, M. Neilan, L. Rebholz, and N. Wilson. A connection between coupled and penalty projection timestepping schemes with FE spatial discretization for the Navier-Stokes equations. *Journal of Numerical Mathematics*. , to appear.

[78] J. Liu. Open and traction boundary conditions for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 228(19):254–278, 2009.

[79] G. Lube and G. Matthies. On streamline-diffusion methods for inf-sup stable discretisations of the generalised Oseen problem. *Submitted to IMA J. Numer. Anal.*, 2007.

[80] MathWorks. Global Optimization Toolbox: User's Guide (R2017b). `https://www.mathworks.com/help/pdf_doc/optim/optim_tb.pdf`, 2017 (accessed February 5, 2018).

[81] G. Matthies, P. Skrzypcz, and L. Tobiska. A unified convergence analysis for local projection stabilisations applied to the Oseen problem. Technical Report 44, Otto-von-Guericke-Universität Magdeburg, 2006.

[82] L. Mirabella, C.M. Haggerty, T. Passerini, M. Piccinelli, A.J. Powell, P.J. Del Nido, A. Veneziani, and A.P. Yoganathan. Treatment planning for a TCPC Test Case: A Numerical Investigation Under Rigid and Moving Wall Assumptions. *International Journal for Numerical Methods in Biomedical Engineering*, 29(2):197–216, 2013.

[83] B. Mohammadi and O. Pironneau. *Analysis of the K-Epsilon turbulence model*. Wiley, 1994.

[84] C. D. Murray. The Physiological Principle of Minimum Work. I. The Vascular System and the Cost of Blood Volume. *Proc. Natl. Acad. Sci. USA*, 12(3):207–214, 1926.

[85] V. M. Musini and J. M. Wright. Factors Affecting Blood Pressure Variability: Lessons Learned from Two Systematic Reviews of Randomized Controlled Trials. *PLoS One*, 4(5), 2009.

[86] Guido Nabh. *On high order methods for the stationary incompressible Navier-Stokes equations*. Ph.D., Interdisziplinares Zentrum fur Wiss. Rechnen der Univ. Heidelberg, 1998.

[87] Wilmer W. Nichols and Michael F. O'Rourke. *McDonald's Blood Flow in Arteries*. Oxford University Press, 2005.

[88] M. A. Olshanskii and A. Reusken. Grad-Div stabilization for the Stokes equations. *Math. Comp.*, 73:1699–1718, 2004.

[89] G. Pannarale, G. Bebb, S. Clark, A. Sullivan, C. Foster, and A. J. S. Coats. Bias and Variabiity in Blood Pressure Measurement With Ambulatory Recorders. *Hypertension*, 22:591–598, 1993.

[90] T. Passerini, Quaini A., U. Villa, A. Veneziani, and S. Canic. Validation of an open source framework for the simulation of blood flow in rigid and deformable vessels. *International Journal for Numerical Methods in Biomedical Engineering*, 29(11):1192–1213, 2013.

[91] J Blair Perot. An analysis of the fractional step method. *Journal of Computational Physics*, 108(1):51–58, 1993.

[92] M. Piccinelli, L. Mirabella, T. Passerini, E. Haber, and A. Veneziani. 4D image-based CFD simulation of a Compliant Blood Vessel. Technical Report TR-2010-27, Emory University, 2010.

[93] S.B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[94] A. Porpora, P. Zunino, C. Vergara, and M. Piccinelli. Numerical treatment of boundary conditions to replace lateral branches in hemodynamics. *International Journal for Numerical Methods in Biomedical Engineering*, 28:1165–1183, 2012.

[95] Andreas Prohl. *Projection and quasi-compressibility methods for solving the incompressible Navier-Stokes equations*. Springer, 1997.

[96] Luigi Quartapelle. *Numerical solution of the incompressible Navier-Stokes equations*, volume 113. Birkhäuser, 2013.

[97] A. Quarteroni, A. Veneziani, and L. Formaggia. *Complex Systems in Biomedicine*. Springer, 2006.

[98] A. Quarteroni, A. Veneziani, and C. Vergara. Geometric multiscale modeling of the cardiovascular system, between theory and practice. *Comp. Meth. Appl. Mech. Eng.*, 302:192–252, 2016.

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

[100] Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. Analysis of the Yosida method for the incompressible Navier–Stokes equations. *Journal de Mathématiques Pures et Appliquées*, 78(5):473–503, 1999.

[101] Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. Factorization methods for the numerical approximation of Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 188(1):505–526, 2000.

[102] Alfio Quarteroni and Alberto Valli. *Numerical approximation of partial differential equations*, volume 23. Springer Science & Business Media, 2008.

[103] David Randall. Reynolds Averaging, 2018. URL: `https://kiwi.atmos.colostate.edu/group/dave/pdf/Reynolds_Averaging.pdf`. Last visited on 2018/30/01.

[104] Rolf Rannacher. *On Chorin's projection method for the incompressible Navier-Stokes equations*, pages 167–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[105] L. Rebholz and M. Xiao. On reducing the splitting error in Yosida methods for the Navier-Stokes equations with grad-div stabilization. *Computer Methods in Applied Mechanics and Engineering*, 294:259–277, 2015.

[106] O. Reynolds. On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Philosophical Transactions of the Royal Society of London*, 186:123–164, 1895.

[107] Hans-Gorg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations, Second Edition*. Springer, 2008.

[108] Fausto Saleri and Alessandro Veneziani. Pressure correction algebraic splitting methods for the incompressible navier–stokes equations. *SIAM journal on numerical analysis*, 43(1):174–194, 2005.

[109] S Salsa. *Partial differential equations in action, from modeling to theory*. Springer, 2009.

[110] Michael Schäfer, Stefan Turek, Franz Durst, Egon Krause, and Rolf Rannacher. *Benchmark computations of laminar flow around a cylinder*. Springer, 1996.

[111] A. Schoenberger, N. Urbanek, S. Toggweiler, R. Seelos, P. Jamshidi, T. Resink, and P. Erne. Deviation from Murray's law is associated with a higher degree of calcification in coronary bifurcations. *Artherosclerosis*, 221(1):124–130, 2012.

[112] T. Sherman. On Connecting Large Vessels to Small: The Meaning of Murray's Law. *Journal of General Physiology*, 78(4):431–453, 1981.

[113] P.R. Spalart. Strategies for turbulence modeling and simulations. *Int. J. Heat Fluid Flow*, 21:252–263, 2000.

[114] P. Spiller, F.K. Schmiel, B. Politz, M. Block, U. Fermor, W. Hackbarth, J. Jehle, R. Korfer, and H. Pannek. Measurement of systolic and diastolic flow rates in the coronary artery system by x-ray densitometry. *Circulation*, 68:337–347, 1983.

[115] CA. Taylor and MT. Draney. Experimental and computational methods in cardiovascular mathematics. *Annual Review of Fluid Mechanics*, 36:197–231, 2004.

[116] R. Temam. *Navier-Stokes equations*. Elsevier, North-Holland, 1991.

[117] Roger Temam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (ii). *Archive for Rational Mechanics and Analysis*, 33(5):377–385, 1969.

[118] Y. Tominaga. Flow around a high-rise building using steady and unsteady RANS CFD: Effect of large-scale fluctuations on velocity statistics. *J. Wing Eng. Ind. Aerodyn.*, 142:93–103, 2015.

[119] Stefan Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approache*, volume 6. Springer Science & Business Media, 1999.

[120] H. B. M. Uylings. Optimization of diameters and bifurcation angles in lung and vascular tree structures. *Bulletin of Mathematical Biology*, 39(5):509–520, 1977.

[121] A. Veneziani and C. Vergara. Flow rate defective boundary conditions in haemodynamics simulations. *International Journal for Numerical Methods in Fluids*, 47(8–9):803–816, 2004.

[122] A. Veneziani and A. Viguerie. Algebraic splitting methods for steady incompressible Navier-Stokes equations at moderate Reynolds numbers. *Computer Methods in Applied Mechanics and Engineering*, 330:271–291, 2018.

[123] Alessandro Veneziani. Block factorized preconditioners for high-order accurate in time approximation of the Navier-Stokes equations. *Numerical Methods for Partial Differential Equations*, 19(4):487–510, 2003.

[124] Alessandro Veneziani. A note on the consistency and stability properties of yosida fractional step schemes for the unsteady Stokes equations. *SIAM Journal on Numerical Analysis*, 47(4):2838–2843, 2009.

[125] Alessandro Veneziani and Umberto Villa. ALADINS: An ALgebraic splitting time ADaptive solver for the Incompressible Navier–Stokes equations. *Journal of Computational Physics*, 238:359–375, 2013.

[126] E. R. Weibel. Morphometrics of the lung. *Handbook of Physiology*, 1(3):285–307, 1981.

[127] N. Westerhof, J.W. Lankhaar, and B. E. Westerhof. The arterial windkessel. *Medical and Biological Engineering and Computing*, 47(2):131–141, 2009.

[128] D.C. Wilcox. *Turbulence Modeling for CFD, third edition*. DCW Industries, 2006.

[129] D.A. Zelicourt, B.N. Steele, and A.P. Yoganathan. Advances in Computational Simulations for Interventional Treatments and Surgical Planning.

In K.B. Chandard, H.S. Udaykumar, and J.M. Reinhardt, editors, *Image-Based Computational Modeling of the Human Circulatory and Pulmonary Systems: Methods and Applications*, pages 343–373. 2011.

[130] P. Zunino. Numerical approximation of incompressible flows with net flux defective boundary conditions by means of penalty techniques. *Comp. Meth. Appl. Mech. Eng.*, 198:3026–3038, 2009.