May 2020

A Computational Framework for Axisymmetric Linear Elasticity and Parallel Iterative Solvers for Two-Phase Navier–Stokes

Alistair R. Bentley
Clemson University, alistairbntl@gmail.com

A COMPUTATIONAL FRAMEWORK FOR AXISYMMETRIC LINEAR ELASTICITY AND PARALLEL ITERATIVE SOLVERS FOR TWO-PHASE NAVIER–STOKES

A Thesis
Presented to
the Graduate School of
Clemson University

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

---

by
Alistair Bentley
May 2020

ABSTRACT

This dissertation explores ways to improve the computational efficiency of linear elasticity and the variable density/viscosity Navier–Stokes equations. While the approaches explored for these two problems are much different in nature, the end goal is the same - to reduce the computational effort required to form reliable numerical approximations.

The first topic considered is the axisymmetric linear elasticity problem. While the linear elasticity problem has been studied extensively in the finite-element literature, to the author's knowledge, this is the first study of the elasticity problem in an axisymmetric setting. Indeed, the axisymmetric nature of the problem means that a change of variables to cylindrical coordinates reduces a three-dimensional problem into a decoupled one-dimensional and two-dimensional problem. The change of variables to cylindrical coordinates, however, affects the functional form of the divergence operator and the definition of the inner products. To develop a computational framework for the linear elasticity problem in this context, a new projection operator is defined that is tailored to the cylindrical form of the divergence and inner products. Using this framework, a stable finite-element quadruple is derived for $k = 1, 2$. These computational rates are then validated with a few computational examples.

The second topic addressed in this work is the development of a new Schur complement approach for preconditioning the two-phase Navier–Stokes equations. Considerable research effort has been invested in the development of Schur complement preconditioning techniques for the Navier–Stokes equations, with the pressure-convection diffusion (PCD) operator and the least-squares commutator being among the most popular. Furthermore, more recently researchers have begun examining preconditioning strategies for variable density / viscosity Stokes and Navier–Stokes equations. This work contributes to recent work that has extended the PCD Schur complement approach for single phase flow to the variable phase case. Specifically, this work studies the effectiveness of a new two-phase PCD operator when applied to dynamic two-phase simulations that use the two-phase Navier–Stokes equations. To demonstrate the new two-phase PCD operators effectiveness, results are presented for standard benchmark problems, as well as parallel scaling results are presented for large-scale dynamic simulations for three-dimensional problems.

ACKNOWLEDGMENTS

I would like to extent a special thank you to my adviser Dr. Vincent Ervin for his patience and support throughout my entire graduate school experience at Clemson. I am especially grateful for the countless hours Dr. Ervin spent working closely with me throughout my PhD. Without his advice, insights and time this dissertation would not have been possible.

I would also like to thank Dr. Chris Kees for his mentorship and financial support. Many of the most practical skills I developed while writing this document are the result of working with Dr. Kees. Moreover, the workshops and conferences I attended as a result of working Dr. Kees were among the most enriching experiences I had as a PhD student.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation explores ways to improve the computational efficiency in approximating the solution to the axisymmetric linear elasticity problem and the two-phase Navier–Stokes fluid flow problem. Improving computational efficiency is an important part of scientific computing research. For one, when computations become more efficient, the space of solvable problems expands. Better efficiency also reduces the computational resources and time needed to solve a given problem. This is valuable because high-performance computing resources have a financial cost associated with their use. In addition, faster solvers allow end users to perform analysis for their applications more efficiently.

While this dissertation examines two different topics, both methods aim to reduce the effort needed to solve the linear systems of equations that arise from using the finite element method to approximate the solutions. In the axisymmetric linear elasticity case, the problem is recast in a way that reduces the size of the resulting linear system of equations. In the two-phase Navier–Stokes problem, a specialized preconditioner is introduced that improves the efficiency of the GMRES method.

Axisymmetric Linear Elasticity

Chapter 2 examines the axisymmetric linear elasticity problem. Various forms of the linear elasticity problem have been proposed and studied in the finite element literature (for example see [24, 26]). In this study, we restrict our analysis to the mixed formulation with weak symmetry (see [82, 83, 49, 70, 6]).

The goal of this work is to use the axisymmetric property of the solution to reduce the effort needed to compute a numerical approximation to the solution. In the axisymmetric setting, applying a change of variable from Cartesian to cylindrical coordinates allows the three-dimensional axisymmetric linear elasticity problem to be recast as a decoupled two-dimensional and one-dimensional problem. The linear systems that come from the decoupled systems can be solved in parallel and require less time and memory to solve individually that the fully coupled problem.

Using axisymmetry to reduce computational effort is not a new idea. Since the 1980s when Mercier and Raugel [69] undertook one of the earliest finite element analysis of axisymmetric problems, axisymmetric computational frameworks have been developed for many problems. Some important examples include Poisson's equation, Stokes equation and Darcy's equation [9, 10, 32, 35, 72, 20, 63, 43, 45, 46]. To the author's knowledge, however, no successful analysis has been undertaken for the axisymmetric linear elasticity problem.

A key challenge to developing a computational framework for the axisymmetric linear elasticity problem is the functional form of the axisymmetric divergence operator. In particular, the axisymmetric divergence operator does not map polynomial spaces into polynomial spaces. As a result, the standard approaches used to develop inf-sup stable finite elements in the Cartesian environment cannot be used. To address this challenge, a new projection operator is developed in this work that has been tailored specifically to handle the axisymmetric divergence operator.

Two Phase Navier–Stokes

Chapter 3 examines preconditioning methods for the two-phase Navier–Stokes equations. An important motivation for studying two-phase Navier–Stokes preconditioners are the dynamic free-surface models that arise in industrial applications. Free-surface models that reliably track the interface between air and water over time are an important part of modeling many complicated hydraulic processes such as waves crashing into coastal barriers, or flow dynamics following a failure in infrastructure.

In many cases, a complete free-surface model involves coupling several different physical models. Frequently a splitting scheme that uses the two-phase Navier–Stokes equations is used to model free-surface models. Moreover, it is often the case that solving the discrete two-phase Navier–Stokes equations forms a bottleneck when running the splitting scheme. Therefore, effective preconditioners for the two-phase Navier–Stokes equations are an important part of building a more efficient numerical method.

There is a large amount of literature about efficient Navier–Stokes preconditioners that includes techniques like the Augmented Lagrangian method, two-threshold incomplete LU factorizations, and Schur complement methods (for example see [18, 19, 62, 71, 40]). In addition to these methods, there is a significant body of literature on specific tools – like LU, multigrid, and ILU – that can be used to solve the subproblems that arise as part of a complete preconditioner [3, 1, 93, 84, 85, 14, 92, 60, 61, 39]. Much of the existing preconditioning work, however, has focused on the single-phase Navier–Stokes problem or on the variable viscosity Stokes problem [41, 40, 57, 75, 88, 29, 11, 51, 52, 68], rather than the two-phase Navier–Stokes equations. This work seeks to extend these methods to develop a preconditioner for a two-phase Navier–Stokes that is effective for dynamic free-surface models.

There are two key elements necessary to develop a scalable preconditioner for the two-phase Navier–Stokes equation. First, because the Schur complement operators that have been developed for single-phase flow do not generalize well into the multiphase setting, an appropriate approximation to the Schur complement operator must be developed for the two-phase Navier–Stokes equations. Second, this Schur complement approximation needs to be incorporated into a complete block preconditioner framework. This requires an in-depth knowledge of the discrete finite element problem as well as the linear solver tools need to solve each of the subproblems that arise as part of the complete block preconditioner.

Outline

The remainder of this dissertation proceeds as follows. Chapter 2 examines the axisymmetric linear elasticity problem. In Sections 2.1-2.3 of Chapter 2 we provide background, notation, and the weak formulation of the Cartesian axisymmetric linear elasticity problem. Next in Sections 2.4-2.6, we introduce the function spaces, notation and weak formulation of the axisymmetric linear elasticity problem with weak symmetry. In Sections 2.7-2.8, we introduce the discrete axisymmetric linear elasticity problem with weak symmetry and present a useful theorem that establishes a set of sufficient conditions for inf-sup stability. Sections 2.10-2.11 consider several different examples of finite element spaces that satisfy the sufficient conditions outlined in Section 2.8. Finally, Sections 2.12 - 2.14 present a formal error analysis, computational results and describe possible future research directions.
Chapter 3 examines preconditioning strategies for the solution of the approximating linear system for two-phase Navier–Stokes equations. In Section 3.1, we introduce the free-surface context in which the two-phase Navier–Stokes equations appear and survey some of the existing methods that have been developed to precondition the approximating linear system. Next, a detailed overview of the RANS2P free-surface model considered in this work is presented in Section 3.2. Following this, Sections 3.3-3.4 describe various details about the numerical methods used. This includes a discussion of the numerical techniques used to enforce boundary conditions, stabilization methods, and the numerical methods used to solve linear systems. Finally, Section 3.5 presents a number of numerical results demonstrating the effectiveness of the two-phase PCD preconditioner in a number of settings compared to other preconditioning strategies.

Chapter 2

Elasticity

2.1 Introduction

During the past twenty years, a number of papers have emerged in the numerical analysis literature investigating three-dimensional axisymmetric problems. This class of problem has attracted attention because a three-dimensional axisymmetric problem can be reduced to a two-dimensional problem when cylindrical coordinates are used (see Figure 2-1). Indeed, it is well recognized that the computational effort required to solve a two-dimensional problem is significantly less that the computational effort needed to solve a three-dimensional problem. It has also been noted that many problems that are not axisymmetric can be locally approximated as axisymmetric, permitting more opportunities for computational gains.

While axisymmetric problems have long appeared in engineering and mathematical literature, Mercier and Raugel undertook one of the earliest finite element analyses of

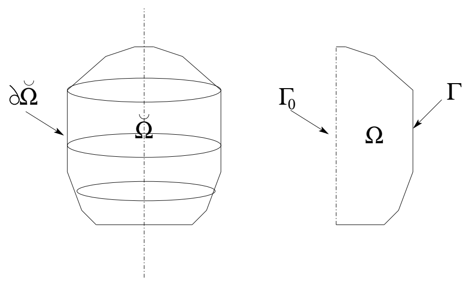


Figure 2-1: Axisymmetric Domain

these problems in the 1980s. In [69], the authors investigated the appropriate Sobolev spaces, projection properties and error analysis. In the late 1990s, Bernardi, Dauge and Maday [21] studied the axisymmetric form of a number of standard problems (including Laplace, Stokes and Maxwell equations) and introduced tools for analyzing axisymmetric spectral methods.

In the early 2000s, two important papers [9, 10] were published in which the authors studied the numerical approximation of the axisymmetric solution of the static and time dependent Maxwell equations. Following these papers, a number of studies analyzing different axisymmmetric problems began to appear. Notably, a computational framework for the axisymmetric Poisson equation was developed in [32] and a computational framework for div-curl systems was presented in [35]. More recently, [72] used finite element exterior calculus techniques to study the axisymmetric Hodge Laplace problem.

One area that has received a great deal of focus are axisymmetric fluid dynamics problems. In [20], axisymmetry was used to reduce the dimension of an eddy current model and in [4], a computational framework for axisymmetric Brinkman flows was developed. The axisymmetric Stokes and Darcy problems have also received a great deal of attention as discussed in [16, 63, 43, 45]. A coupled axisymmetric Stokes-Darcy problem was explored in [46].

Absent from this work, however, is a finite element analysis of the axisymmetric linear elasticity problem. This gap in the literature is notable because the linear elasticity problem appears in many applications. For example, a symmetric tank subjected to an internal pressure force (e.g. gas expanding from a changing temperature) can be modeled as an axisymmetric linear elasticity problem (see [66]). Mechanical systems that involve engine valve stems can also be modeled using axisymmetric linear elasticity [66]. See Figure 2-2.

The linear elasticity problem has been extensively studied and a number of excellent resources investigating the problem are available (see [24, 26]). A detailed description of the linear elasticity problem as well as a survey of the literature is presented in Section 2.3. At the outset, however, we comment that this work considers the mixed

(a) Propane Tank

(b) Engine Valve Stem

Figure 2-2: Applications of Axisymmetric Linear Elasticity Models [66]

form of the elasticity problem in which symmetry of the stress tensor is enforced weakly.

The nature of differential operators in cylindrical coordinates (e.g. the addition of a $\frac{1}{r}$ term) is an important reason that the linear elasticity problem becomes more challenging in the cylindrical setting. Indeed, a consequence of this radial scaling is that the gradient and divergence operators do not map polynomial spaces to polynomial spaces. This feature of cylindrical coordinates will be important in the construction of inf-sup stable finite elements.

The outline of this work is as follows. Section 2.2 introduces some important notation and operators that appear throughout our discussion on linear elasticity. Section 2.3 provides an overview of the general elasticity problem as well as the weak symmetry problem. In Sections 2.4 and 2.6 we introduce the weak form of the linear elasticity problem in the axisymmetric setting.

The key contributions of this work are presented in Sections 2.7 to 2.12, where the discrete axisymmetric problem is formally introduced, and then spaces that satisfy the saddle point theory of Brezzi [27, 28, 24] are introduced and explored. Finally, computational results are presented in Section 2.13 and some concluding remarks are presented in Section 2.14.

2.2 Notation

We start with an overview of the notation and definitions used in this Chapter. Bold Greek letters (e.g. $\boldsymbol{\sigma}$) will represent vectors, while bold Greek letters with an underline (e.g. $\underline{\boldsymbol{\sigma}}$) will denote tensors. For English letters, bold lowercase letters (e.g. $\mathbf{p}$) will denote vectors, while bold uppercase letters (e.g. $\mathbf{P}$) will denote tensors. Matrices will be represented with capital, non-bold letters (e.g. A).

Additionally, we let $\mathbb{R}^n$ denote the space of n dimensional real numbers, $\mathbb{M}^n$ denote the space of $n \times n$ dimensional real matrices, $\mathbb{S}^n$ denote the space of $n \times n$ dimensional real symmetric matrices and $\mathbb{K}^n$ denote the space of $n \times n$ dimensional real skew-symmetric matrices.

The domain is denoted as Ω and the boundary is denoted as $\partial\Omega$. Throughout this work, we assume that the coordinate system (either Cartesian or cylindrical) used to represent Ω is labeled in such a way that $z \geq 0$ for all $\mathbf{x} \in \Omega$. In addition, $\mathcal{T}_h$ will be used to denote a regular triangulation of Ω with element diameter h.

We will frequently use piecewise polynomials to define finite element spaces. To denote the space of degree k piecewise polynomials on a mesh $\mathcal{T}_h$ we use the notation $P_k(\mathcal{T}_h)$. If we are referencing a polynomial on a specific domain T or element $T \in \mathcal{T}_h$, we use $P_k(T)$. When referencing a vector or tensor space of polynomials, we use $(P_k(T))^n$ and $(P_k(T))^{n\times n}$ respectively.

The symmetric gradient operator, $\underline{\boldsymbol{\epsilon}}$, is applied to a vector $\mathbf{u}$ and returns the tensor $\underline{\boldsymbol{\epsilon}}(\mathbf{u})$ with components

$$\underline{\boldsymbol{\epsilon}}(\mathbf{u})_{ij} = \frac{1}{2}\left(\frac{\partial \mathbf{u}_i}{\partial x_j} + \frac{\partial \mathbf{u}_j}{\partial x_i}\right). \tag{2.1}$$

The divergence operator, $\nabla\cdot$, can be applied to vectors and tensors. When acting on a vector $\mathbf{v}$,

$$\nabla \cdot \mathbf{v} = \sum_{i=1}^{n} \frac{\partial \mathbf{v}_i}{\partial x_i}. \tag{2.2}$$

When acting on a tensor $\underline{\boldsymbol{\sigma}}$,

$$\nabla \cdot \underline{\boldsymbol{\sigma}} = \begin{pmatrix} \nabla \cdot \underline{\boldsymbol{\sigma}}_1 \\ \nabla \cdot \underline{\boldsymbol{\sigma}}_2 \end{pmatrix} \tag{2.3}$$

where $\underline{\boldsymbol{\sigma}}_i$ denotes row i of $\underline{\boldsymbol{\sigma}}$.

The trace operator, tr, is applied to tensors and represents the sum of the diagonal elements,

$$\mathrm{tr}(\underline{\boldsymbol{\sigma}}) = \sum_{i=1}^{n} \underline{\boldsymbol{\sigma}}_{ii}. \tag{2.4}$$

The mixed linear elasticity problem also requires some notation and operators that are not commonplace in finite element discussions. We take a moment to familiarize the reader with some of the less common operators and definitions that appear in the elasticity problem. The skew-symmetric part of a tensor $\underline{\boldsymbol{\sigma}}$ is defined as

$$as(\underline{\boldsymbol{\sigma}}) = \frac{1}{2}(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}^t) \tag{2.5}$$

where $\underline{\boldsymbol{\sigma}}^t$ is the transpose of $\underline{\boldsymbol{\sigma}}$. Furthermore, in two dimensions, $as(\underline{\boldsymbol{\sigma}})$ can be identified with a scalar value $q \in \mathbb{R}$ and the following operator

$$as(\underline{\boldsymbol{\sigma}}) = \mathcal{S}^2(q) = \begin{pmatrix} 0 & q \\ -q & 0 \end{pmatrix} \text{ where } q = \frac{1}{2}(\sigma_{12} - \sigma_{21}). \tag{2.6}$$

For vectors $\mathbf{a} = (a_1, a_2)^t$ and $\mathbf{b} = (b_1, b_2)^t$,

$$\mathbf{a} \otimes \mathbf{b} = \begin{pmatrix} a_1 b_1 & a_1 b_2 \\ a_2 b_1 & a_2 b_2 \end{pmatrix}. \tag{2.7}$$

If $\mathbf{w} = (w_1, w_2)^t$ and $\mathbf{v} = (v_1, v_2)^t$ are vectors, then the two-dimensional wedge product is

$$\mathbf{w} \wedge \mathbf{v} = w_1 v_2 - w_2 v_1. \tag{2.8}$$

For a tensor $\underline{\boldsymbol{\tau}}$ and vector $\mathbf{v}$, the wedge product is

$$(\underline{\boldsymbol{\tau}} \wedge \mathbf{v}) = \begin{pmatrix} \tau_{11} v_2 - \tau_{21} v_1 \\ \tau_{12} v_2 - \tau_{22} v_1 \end{pmatrix}. \tag{2.9}$$

Finally, if $\mathbf{x} = (x_1, x_2)^t$, then $\mathbf{x}^\perp = (x_2, -x_1)^t$.
Throughout this manuscript, some ideas are presented in Cartesian coordinates, while others are presented in cylindrical coordinates. As outlined in Section 2.4, these coordinate spaces motivate different inner product and bilinear forms. To distinguish between the cylindrical coordinate case and the Cartesian case, a c subscript will be attached to all Cartesian inner products and bilinear forms.

2.3 Elasticity Problem Description

Two formulations of the linear elasticity problem appear in the literature. The first is the *pure displacement formulation*, which for a domain $\Omega \subset \mathbb{R}^n$, is described by

$$\underline{\boldsymbol{\sigma}} = \mathcal{C}\underline{\boldsymbol{\epsilon}}(\mathbf{u}), \quad \nabla \cdot \underline{\boldsymbol{\sigma}} = \boldsymbol{f} \text{ in } \Omega. \tag{2.10}$$

Here $\underline{\boldsymbol{\sigma}}$ denotes the stress tensor, $\mathcal{C}$ is the stiffness tensor, $\mathbf{u}$ is the displacement, $\underline{\boldsymbol{\epsilon}}(\mathbf{u})$ is the symmetric gradient and $\mathbf{f}$ is an external body force. Since the stress tensor is symmetric, the stiffness tensor satisfies $\mathcal{C} : \mathbb{S}^{n\times n} \to \mathbb{S}^{n\times n}$ and takes the form $\mathcal{C}\underline{\boldsymbol{\tau}} = 2\mu(\underline{\boldsymbol{\tau}} + \lambda \operatorname{tr}(\underline{\boldsymbol{\tau}})\ I)$ for isotropic materials, where μ and λ are the Lamé constants with values dependent on the material properties.

With this formulation, one typically solves

$$\nabla \cdot \mathcal{C}\epsilon(\mathbf{u}) = \mathbf{f} \text{ in } \Omega \tag{2.11}$$

with appropriate boundary conditions for $\mathbf{u}$. Then one calculates the stress tensor $\underline{\boldsymbol{\sigma}}$ via $\underline{\boldsymbol{\sigma}} = \mathcal{C}\underline{\boldsymbol{\epsilon}}(\mathbf{u})$. For incompressible or nearly incompressible materials, however, the Lamé constant $\lambda \to \infty$. Examples of materials with nearly infinite Lamé constants include rubber, some saturated clays and some types of foam. As λ becomes large, the operator $\mathcal{C}$ becomes unbounded and the pure displacement formulation (2.10) becomes numerically unstable [6, 48]. One solution to this problem, is to recast (2.10) into a *mixed formulation* where the displacement vector and stress tensor are solved simultaneous.

Taking $\mathcal{A} = \mathcal{C}^{-1}$, (2.10) becomes

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \underline{\boldsymbol{\epsilon}}(\mathbf{u}), \quad \nabla \cdot \underline{\boldsymbol{\sigma}} = \boldsymbol{f} \text{ in } \Omega. \tag{2.12}$$

The compliance tensor $\mathcal{A} : \mathbb{S}^{n\times n} \to \mathbb{S}^{n\times n}$ is a bounded, symmetric postive definite operator that takes the form

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \frac{1}{2\mu}\left(\underline{\boldsymbol{\sigma}} - \frac{\lambda}{2\mu + m\lambda}\mathrm{tr}(\underline{\boldsymbol{\sigma}}))I\right) \tag{2.13}$$

for isotropic materials. To reflect that $\Omega \subset \mathbb{R}^3$, we take $m = 3$ throughout this work. For notational simplicity, we extend the domain of $\mathcal{A}$ to scalar functions. Then, $\mathcal{A}\sigma$ is given by (2.13) for $\sigma \in S^{1\times 1}$.

In general, the pure displacement formulation requires less computational effort than the mixed formulation. However, since it is unstable for nearly incompressible materials, it is also a less versatile formulation. Additionally, if the stress is a quantity of interest, post processing of the solution is required to obtain the stress. In this research, we focus on the mixed formulation.

Boundary conditions are necessary to fully specify the problem (2.12). Let $\partial\Omega =$

$\Gamma_1 \cup \Gamma_2$, and consider

$$\mathbf{u}|_{\Gamma_1} = \mathbf{0}, \text{ and } (\underline{\boldsymbol{\sigma}}\mathbf{n})|_{\Gamma_2} = \mathbf{g} \tag{2.14}$$

where $\mathbf{n}$ is the outward pointing unit normal vector on $\partial\Omega$. In the case where $\Gamma_1 = \partial\Omega$, the problem has a pure clamped displacement boundary condition. In the case where $\Gamma_2 = \partial\Omega$, the problem has a pure traction boundary condition. Unless specified otherwise, we will take $\partial\Omega = \Gamma_1$.

To specify the weak formulation of (2.12), we first define the functions spaces

$$\begin{aligned}
&L^2(\Omega) = \{v : \int_\Omega v^2 \, d\Omega < \infty\},\\
&\mathbf{L}^2(\Omega) = \{\mathbf{v} : v_i \in L^2(\Omega) \text{ for } i = 1, \cdots, n\},\\
&\underline{\mathbf{L}}^2(\Omega; \mathbb{S}^n) = \{\underline{\boldsymbol{\sigma}} \in \mathbb{S}^n : \sigma_{ij} \in L^2(\Omega) \text{ for } i, j = 1, \cdots, n\} \text{ and}\\
&\underline{\mathbf{H}}(\text{div}, \Omega; \mathbb{S}^n) = \{\underline{\boldsymbol{\sigma}} \in \underline{\mathbf{L}}^2(\Omega, \mathbb{S}^n) : \nabla \cdot \underline{\boldsymbol{\sigma}} \in \mathbf{L}^2(\Omega)\}.
\end{aligned}$$

One way of specifying the weak formulation of (2.12) is to multiply with test functions from $X = \underline{\mathbf{H}}(\text{div}, \Omega; \mathbb{S}^n)$ and $Q = \mathbf{L}^2(\Omega)$ and integrate by parts. In this way, the problem becomes: find $(\underline{\boldsymbol{\sigma}}, \mathbf{u}) \in X \times Q$ such that for all $(\underline{\boldsymbol{\tau}}, \mathbf{v}) \in X \times Q$

$$\begin{aligned}
\int_\Omega (\mathcal{A}\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} + \nabla \cdot \underline{\boldsymbol{\tau}} \, \mathbf{u}) \, d\Omega &= 0\\
\int_\Omega \nabla \cdot \underline{\boldsymbol{\sigma}} \cdot \mathbf{v} \, d\Omega &= \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega.
\end{aligned} \tag{2.15}$$

An alternative way of deriving a weak formulation to (2.12), is to view the pair $(\underline{\boldsymbol{\sigma}}, \mathbf{u}) \in X \times Q$ as the unique critical point of the Hellinger-Reissner functional

$$\mathcal{J}(\underline{\boldsymbol{\sigma}}, \mathbf{u}) = \int_\Omega \left(\frac{1}{2} \mathcal{A}\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\sigma}} + (\nabla \cdot \underline{\boldsymbol{\sigma}}) \cdot \mathbf{u} - \mathbf{f} \cdot \mathbf{u} \right) \, d\Omega. \tag{2.16}$$

Setting the variationals of $\mathcal{J}$ with respect to $\underline{\boldsymbol{\sigma}}$ and $\mathbf{u}$ to zero also yields the weak formulation (2.15).

The existence and uniqueness of a solution to (2.15) is shown in [23]. To approximate (2.15) with the finite element method, we need to choose approximation spaces $X_h \subset$

X and $Q_h \subset Q$. As is well known, the choice of such spaces is not arbitrary, but must satisfy the following stability conditions [8].

1. There exists a positive constant c_1 such that $\|\underline{\boldsymbol{\tau}}\|_{\underline{\mathbf{H}}(div,\Omega)} \leq c_1 \|\underline{\boldsymbol{\tau}}\|_{\underline{\mathbf{L}}^2(\Omega)}$ whenever $\underline{\boldsymbol{\tau}} \in X_h$ satisfies $\int_\Omega (\nabla \cdot \underline{\boldsymbol{\tau}}) \cdot \mathbf{v} \, d\Omega = 0$ for all $\mathbf{v} \in Q_h$.

2. There exists a $C > 0$ such that

$$\inf_{\boldsymbol{v} \in Q_h} \sup_{\underline{\boldsymbol{\tau}} \in X_h} \frac{\int_\Omega (\nabla \cdot \underline{\boldsymbol{\tau}}) \cdot \boldsymbol{v} \, d\Omega}{\|\boldsymbol{v}\|_Q \|\underline{\boldsymbol{\tau}}\|_X} \geq C.$$

Researchers have been studying ways to develop stable finite elements for (2.15) since the 1960s [90]. For many years, the only known stable finite elements to (2.15) were macro-elements in which the stress tensor was solved on a finer mesh than the displacement vector [7, 56, 91]. It was not until 2002 that Arnold and Winther developed a stable pair of piecewise polynomials with respect to a single triangulation of Ω [8]. These elements, however, carry a significant computational cost since the lowest order representation uses 24 degrees of freedom per triangle.
At its core, the challenge to creating a stable finite element scheme for (2.15) is that symmetry of the stress tensor represents the law of conservation of angular momentum, and imposing conservation laws exactly is difficult. As a result much of the research on finite elements approximations for elasticity has moved away from enforcing symmetry in a strong sense.

2.3.1 Weak Symmetry and the Cartesian Elasticity Problem

To avoid enforcing symmetry in the stress tensor strongly, a new Lagrangian multiplier can be added to (2.15) that weakly enforces symmetry in the stress tensor $\underline{\boldsymbol{\sigma}}$ [82, 83, 49, 70, 6]. The weak symmetry approach requires the introduction of a few new function spaces

$$\underline{\mathbf{L}}^2(\Omega;\mathbb{M}^n) = \{\underline{\boldsymbol{\sigma}} \in \mathbb{M}^n : \sigma_{ij} \in L^2(\Omega) \text{ for } i,j = 1,\cdots,n\},$$
$$\underline{\mathbf{L}}^2(\Omega;\mathbb{K}^n) = \{\underline{\boldsymbol{\sigma}} \in \mathbb{K}^n : \sigma_{ij} \in L^2(\Omega) \text{ for } i,j = 1,\cdots,n\} \text{ and}$$
$$\underline{\mathbf{H}}(\mathrm{div},\Omega,\mathbb{M}^n) = \{\underline{\boldsymbol{\sigma}} \in \underline{\mathbf{L}}^2(\Omega,\mathbb{M}^n) : \nabla\cdot\underline{\boldsymbol{\sigma}} \in \mathbf{L}^2(\Omega)\}.$$

Letting $X = \underline{\mathbf{H}}(\mathrm{div},\Omega,\mathbb{M}^n)$, $Q = \mathbf{L}^2(\Omega)$, and $W = \underline{\mathbf{L}}^2(\Omega;\mathbb{K}^n)$, we want to find $(\underline{\boldsymbol{\sigma}},\mathbf{u},\underline{\boldsymbol{\rho}}) \in X\times Q\times W$ such that for all $(\underline{\boldsymbol{\tau}},\mathbf{v},\underline{\boldsymbol{\xi}}) \in X\times Q\times W$

$$\int_\Omega (\mathcal{A}\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} + \nabla\cdot\underline{\boldsymbol{\tau}}\cdot\mathbf{u} + \underline{\boldsymbol{\tau}} : \underline{\boldsymbol{\rho}})\, d\Omega = 0 \tag{2.17}$$
$$\int_\Omega \nabla\cdot\underline{\boldsymbol{\sigma}}\cdot\mathbf{v}\, d\Omega = \int_\Omega \mathbf{f}\cdot\mathbf{v}\, d\Omega \tag{2.18}$$
$$\int_\Omega \underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\xi}}\, d\Omega = 0. \tag{2.19}$$

It also bares mentioning that the weak symmetry problem can be consider as the unique critical point of the modified Hellinger-Ressiner functional (recall (2.16))

$$\mathcal{J}_w(\underline{\boldsymbol{\sigma}},\mathbf{u},\rho) = \mathcal{J}(\underline{\boldsymbol{\sigma}},\mathbf{u}) + \int_\Omega \underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\rho}}\, d\Omega. \tag{2.20}$$

Again, setting the variationals of $\mathcal{J}$ with respect to $\underline{\boldsymbol{\sigma}}$, $\mathbf{u}$ and $\underline{\boldsymbol{\rho}}$ to zero also produces the weak formulation (2.17)-(2.19).
Defining the inner products

$$\begin{aligned} a(\cdot,\cdot) : X\times X \to \mathbb{R}, &\quad a(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\tau}}) := \int_\Omega \mathcal{A}\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}}\, d\Omega,\\ b(\cdot,\cdot) : Q\times X \to \mathbb{R}, &\quad b(\mathbf{u},\underline{\boldsymbol{\tau}}) := \int_\Omega (\nabla\cdot\underline{\boldsymbol{\tau}})\cdot\mathbf{u}\, d\Omega,\\ c(\cdot,\cdot) : W\times X \to \mathbb{R}, &\quad c(\underline{\boldsymbol{\rho}},\underline{\boldsymbol{\tau}}) := \int_\Omega \underline{\boldsymbol{\rho}} : \underline{\boldsymbol{\tau}}\, d\Omega \end{aligned} \tag{2.21}$$

and taking $A(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\tau}}) = a(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\tau}})$ and $B(\underline{\boldsymbol{\tau}},(\mathbf{u},\underline{\boldsymbol{\rho}})) = b(\mathbf{u},\underline{\boldsymbol{\tau}}) + c(\underline{\boldsymbol{\rho}},\underline{\boldsymbol{\tau}})$ clearly illustrates that (2.17) – (2.19) preserves a saddle point structure. Indeed, the problem can now be cast as finding $(\underline{\boldsymbol{\sigma}}, \mathbf{u}, \underline{\boldsymbol{\rho}}) \in X \times Q \times W$ such that

$$\begin{aligned} A(\underline{\boldsymbol{\sigma}}, \underline{\boldsymbol{\tau}}) + B(\underline{\boldsymbol{\tau}}, (\mathbf{u}, \underline{\boldsymbol{\rho}})) &= 0 \\ B(\underline{\boldsymbol{\sigma}}, (\mathbf{v}, \underline{\boldsymbol{\xi}})) &= (\mathbf{f}, \mathbf{v}) \end{aligned} \tag{2.22}$$

for all $(\underline{\boldsymbol{\tau}}, \mathbf{v}, \underline{\boldsymbol{\xi}}) \in X \times Q \times W$. If we let $V = \{\underline{\boldsymbol{\sigma}} \in X \mid B(\underline{\boldsymbol{\sigma}}, (\mathbf{u}, \underline{\boldsymbol{\rho}})) = 0$ for all $\mathbf{u} \in Q, \underline{\boldsymbol{\rho}} \in W\}$, then showing existence and uniqueness of (2.22) requires verifying that

- there exists a $C_1 > 0$ such that $A(\underline{\boldsymbol{\sigma}}, \underline{\boldsymbol{\sigma}}) \geq C_1 \|\underline{\boldsymbol{\sigma}}\|_X$ for all $\underline{\boldsymbol{\sigma}} \in V$, and

- there exists a $C_2 > 0$ such that

$$\inf_{\mathbf{v} \in Q, \underline{\boldsymbol{\xi}} \in W} \sup_{\underline{\boldsymbol{\tau}} \in X} \frac{B(\underline{\boldsymbol{\tau}}, (\mathbf{v}, \underline{\boldsymbol{\xi}}))}{\|\underline{\boldsymbol{\tau}}\|_X (\|\mathbf{v}\|_Q + \|\underline{\boldsymbol{\xi}}\|_W)} \geq C_2. \tag{2.23}$$

Since $\nabla \cdot \underline{\boldsymbol{\sigma}} = \mathbf{0}$ for all $\underline{\boldsymbol{\sigma}} \in V$ [24], applying a standard bounding argument to $A(\underline{\boldsymbol{\sigma}}, \underline{\boldsymbol{\sigma}})$ shows that $A(.,.)$ is cocercive in the X norm. Meanwhile, proving the inf-sup condition relies on the following lemma.

Lemma 1. *There exists a $C > 0$ such that for any $\mathbf{v} \in Q$ and $\underline{\boldsymbol{\rho}} \in W$, there exists a $\underline{\boldsymbol{\tau}} \in X$ whereby*

$$b(\underline{\boldsymbol{\tau}}, \mathbf{v}) + c(\underline{\boldsymbol{\tau}}, \underline{\boldsymbol{\rho}}) = \|\mathbf{v}\|_Q^2 + \|\underline{\boldsymbol{\rho}}\|_W^2 \tag{2.24}$$

and

$$\|\underline{\boldsymbol{\tau}}\|_X \leq C(\|\mathbf{v}\|_Q + \|\underline{\boldsymbol{\rho}}\|_W). \tag{2.25}$$

Proof. Here we outline the proof for the $\Omega \subset \mathbb{R}^2$ case. Let $\mathbf{v} \in Q$ and $\underline{\boldsymbol{\rho}} \in W$. The idea behind this proof is to construct two tensors $\underline{\boldsymbol{\tau}}_1, \underline{\boldsymbol{\tau}}_2 \in X$ such that $\underline{\boldsymbol{\tau}} = \underline{\boldsymbol{\tau}}_1 + \underline{\boldsymbol{\tau}}_2$ satisfies (2.24) and (2.25).

Tensor $\underline{\boldsymbol{\tau}}_1$ is constructed to satisfy

$$b(\underline{\boldsymbol{\tau}}_1, \mathbf{w}) = \int_\Omega (\nabla \cdot \underline{\boldsymbol{\tau}}_1) \cdot \mathbf{w} \, d\Omega = (\mathbf{v}, \mathbf{w}) \tag{2.26}$$

for all $\mathbf{w} \in Q$.

Meanwhile, tensor $\underline{\boldsymbol{\tau}}_2$ needs to be divergence free so (2.26) holds for $\underline{\boldsymbol{\tau}}$, while also satisfying

$$\begin{aligned} c(\underline{\boldsymbol{\tau}}, \underline{\boldsymbol{\xi}}) &= c(\underline{\boldsymbol{\tau}}_1, \underline{\boldsymbol{\xi}}) + c(\underline{\boldsymbol{\tau}}_2, \underline{\boldsymbol{\xi}}) \\ &= (as(\underline{\boldsymbol{\tau}}_1), \underline{\boldsymbol{\xi}}) + (as(\underline{\boldsymbol{\tau}}_2), \underline{\boldsymbol{\xi}}) \\ &= (\underline{\boldsymbol{\rho}}, \underline{\boldsymbol{\xi}}) \end{aligned} \tag{2.27}$$

for all $\underline{\boldsymbol{\xi}} \in W$. In other words, $(as(\underline{\boldsymbol{\tau}}_2), \underline{\boldsymbol{\xi}}) = (\underline{\boldsymbol{\rho}} - as(\underline{\boldsymbol{\tau}}_1), \underline{\boldsymbol{\xi}})$.

To build $\underline{\boldsymbol{\tau}}_1 \in X$, for $\mathbf{v} \in Q = L^2(\Omega)$ given, let $\mathbf{u} \in H^1(\Omega)$ solve Poisson's equation with homogeneous Dirichlet boundary conditions [49]

$$\begin{aligned} \Delta \mathbf{u} &= \mathbf{v} \text{ in } \Omega, \\ \mathbf{u} &= \mathbf{0} \text{ on } \partial\Omega. \end{aligned}$$

Let $\underline{\boldsymbol{\tau}}_1 = 2\underline{\boldsymbol{\epsilon}}(\mathbf{u}) - (\nabla \cdot \mathbf{u})I$. Then $\nabla \cdot \underline{\boldsymbol{\tau}}_1 = \Delta \mathbf{u} = \mathbf{v}$ and it follows that $\underline{\boldsymbol{\tau}}_1 \in X$. Further,

$$\|\underline{\boldsymbol{\tau}}_1\|_X \leq C_1 \|\mathbf{v}\|_Q. \tag{2.28}$$

Next, we build $\underline{\boldsymbol{\tau}}_2 \in X$. Let $\underline{\boldsymbol{\rho}} \in W$ be given and $p, \theta \in L^2(\Omega)$ be chosen so that

$$\underline{\boldsymbol{\rho}} = \begin{pmatrix} 0 & p \\ -p & 0 \end{pmatrix} = \mathcal{S}^2(p) \quad \text{and} \quad as(\underline{\boldsymbol{\tau}}_1) = S^2(\theta). \tag{2.29}$$

Then take s to be the mean value of $\theta - p$. That is,

$$s = \frac{1}{|\Omega|} \int_\Omega (\theta - p) \, d\Omega. \tag{2.30}$$

If we set $\beta = (\theta - p) - s$, then β has a mean value of zero over Ω (i.e. $\int_\Omega \beta \, d\Omega = 0$). Then, from [50], there exists $\mathbf{w} \in \mathbf{H}_0^1(\Omega)$ such that $\nabla \cdot \mathbf{w} = \beta$ and

$$\|\mathbf{w}\|_{\mathbf{H}_0^1(\Omega)} \leq C\|\beta\|_{L^2(\Omega)} \leq C(\|\mathbf{v}\|_Q + \|\underline{\boldsymbol{\rho}}\|_W). \tag{2.31}$$

With $\mathbf{w} = (w_1, w_2)^t$, we construct

$$\underline{\boldsymbol{\tau}}_2 = 2\begin{pmatrix} \text{curl } w_1 \\ \text{curl } w_2 \end{pmatrix} - 2\begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix} = 2\begin{pmatrix} (w_1)_y & -(w_1)_x \\ (w_2)_y & -(w_2)_x \end{pmatrix} - 2\begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix}. \tag{2.32}$$

Since

$$\nabla \cdot \underline{\boldsymbol{\tau}}_2 = 2\begin{pmatrix} (w_1)_{yx} - (w_1)_{xy} \\ (w_2)_{yx} - (w_2)_{xy} \end{pmatrix} - 2\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{2.33}$$

using (2.31), $\underline{\boldsymbol{\tau}}_2 \in X$. Also,

$$\begin{aligned} as(\underline{\boldsymbol{\tau}}_2) &= \begin{pmatrix} 0 & -\nabla \cdot \mathbf{w} - s \\ \nabla \cdot \mathbf{w} + s & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\beta - s \\ \beta + s & 0 \end{pmatrix} \\ &= \mathcal{S}^2(-(\beta + s)) = \mathcal{S}^2(p - \theta) = \underline{\boldsymbol{\rho}} - as(\underline{\boldsymbol{\tau}}_1). \end{aligned} \tag{2.34}$$

Taking $\underline{\boldsymbol{\tau}} = \underline{\boldsymbol{\tau}}_1 + \underline{\boldsymbol{\tau}}_2$, the desired properties are satisfied. First, using (2.33) for all $\mathbf{w} \in Q$

$$\begin{aligned} b(\underline{\boldsymbol{\tau}}, \mathbf{w}) &= b(\underline{\boldsymbol{\tau}}_1, \mathbf{w}) + b(\underline{\boldsymbol{\tau}}_2, \mathbf{w}) = b(\underline{\boldsymbol{\tau}}_1, \mathbf{w}) + 0 \\ &= (\mathbf{v}, \mathbf{w}). \end{aligned} \tag{2.35}$$

Second, for all $\underline{\boldsymbol{\xi}} \in W$,

$$\begin{aligned} c(\underline{\boldsymbol{\tau}}, \underline{\boldsymbol{\xi}}) &= c(\underline{\boldsymbol{\tau}}_1, \underline{\boldsymbol{\xi}}) + c(\underline{\boldsymbol{\tau}}_2, \underline{\boldsymbol{\xi}}) = (as(\underline{\boldsymbol{\tau}}_1), \underline{\boldsymbol{\xi}}) + (as(\underline{\boldsymbol{\tau}}_2), \underline{\boldsymbol{\xi}}) \\ &= (as(\underline{\boldsymbol{\tau}}_1), \underline{\boldsymbol{\xi}}) + (\underline{\boldsymbol{\rho}}, \underline{\boldsymbol{\xi}}) - (as(\underline{\boldsymbol{\tau}}_1), \underline{\boldsymbol{\xi}}) = (\underline{\boldsymbol{\rho}}, \underline{\boldsymbol{\xi}}). \end{aligned} \tag{2.36}$$

Taking $\mathbf{w} = \mathbf{v}$ and $\underline{\boldsymbol{\xi}} = \underline{\boldsymbol{\rho}}$ establishes (2.24).

To verify (2.25), first note that (2.28) implies

$$\|\underline{\boldsymbol{\tau}}_1\|_X \leq C_1(\|\mathbf{v}\|_Q + \|\underline{\boldsymbol{\rho}}\|_W). \tag{2.37}$$

In order to develop a similar bound for $\underline{\boldsymbol{\tau}}_2$, observe that

$$\sqrt{2}\,\|p\|_{L^2(\Omega)} = \left(\int_\Omega 2\,p^2\,d\Omega\right)^{\frac{1}{2}} = \left\|\begin{pmatrix} 0 & p \\ -p & 0 \end{pmatrix}\right\|_{L^2(\Omega)} = \|\mathcal{S}^2(p)\|_W = \|\underline{\boldsymbol{\rho}}\|_W \tag{2.38}$$

and

$$\|\theta\|^2_{L^2(\Omega)} = \int_\Omega (\tau_{12} - \tau_{21})^2\,d\Omega \leq 2\int_\Omega (\tau_{11}^2 + \tau_{12}^2 + \tau_{21}^2 + \tau_{22}^2)\,d\Omega \leq 2\,\|\underline{\boldsymbol{\tau}}_1\|_X^2, \tag{2.39}$$

where τ_{ij} are the elements of $\underline{\boldsymbol{\tau}}_1$.
Next, from (2.30), observe that

$$|s| \leq \frac{1}{|\Omega|}\int_\Omega |p - \theta|\,d\Omega \leq \frac{1}{|\Omega|}\int_\Omega |p| + |\theta|\,d\Omega \leq C_2(\|p\|_{L^2(\Omega)} + \|\theta\|_{L^2(\Omega)}). \tag{2.40}$$

It then follows from (2.38) and (2.39) that

$$|s| \leq C_3(\|\underline{\boldsymbol{\rho}}\|_W + \|\underline{\boldsymbol{\tau}}_1\|_X). \tag{2.41}$$

Therefore,

$$\|s\|_{L^2(\Omega)} = \left(\int_\Omega s^2\,d\Omega\right)^{\frac{1}{2}} \leq \left(\int_\Omega C_3^2(\|\underline{\boldsymbol{\rho}}\|_W + \|\underline{\boldsymbol{\tau}}_1\|_X)^2\,d\Omega\right)^{\frac{1}{2}} = C_3(\|\underline{\boldsymbol{\rho}}\| + \|\underline{\boldsymbol{\tau}}_1\|_X)\sqrt{|\Omega|}. \tag{2.42}$$

Lastly, using (2.32), (2.31), (2.42), (2.37) and the fact that $\nabla \cdot \underline{\boldsymbol{\tau}}_2 = 0$,

$$\begin{aligned} \|\underline{\boldsymbol{\tau}}_2\|_X &= \sqrt{\|\nabla \cdot \underline{\boldsymbol{\tau}}_2\|^2_{L^2(\Omega)} + \|\underline{\boldsymbol{\tau}}_2\|^2_{L^2(\Omega)}} = \|\underline{\boldsymbol{\tau}}_2\|_{L^2(\Omega)} \\ &\leq \left\| 2 \begin{pmatrix} (w_1)_y & -(w_1)_x \\ (w_2)_y & -(w_2)_x \end{pmatrix} \right\|_{L^2(\Omega)} + \left\| 2 \begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix} \right\|_{L^2(\Omega)} \\ &\leq C_4 \left(\|\nabla \mathbf{w}\|_{L^2(\Omega)} + \|s\|_{L^2(\Omega)} \right) \leq C_5 (\|\mathbf{v}\|_Q + \|\underline{\boldsymbol{\rho}}\|_W). \end{aligned} \tag{2.43}$$

Combining (2.37) and (2.43) with the fact that $\|\underline{\boldsymbol{\tau}}\|_X \leq \|\underline{\boldsymbol{\tau}}_1\|_X + \|\underline{\boldsymbol{\tau}}_2\|_X$ verifies (2.25). □

2.4 Axisymmetric Function Spaces

When the three dimensional axisymmetric linear elasticity problem is expressed in cylindrical coordinates, it can be expressed as a decoupled meridian and azimuthal problem. Changing the coordinate system from Cartesian to cylindrical, however, alters the algebraic form of differential operators and requires a new set of function spaces and notation. In this section, we introduce the key changes needed to present and discuss the meridian axisymmetric linear elasticity problem. Appendix A provides additional details on cylindrical coordinates and the procedure for decoupling the axisymmetric problem.

For axisymmetric vectors $\mathbf{u} = (u_r, u_z)^t$, we define the gradient operators ∇ and ∇_{axi} as

$$\nabla \mathbf{u} = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & \dfrac{\partial u_r}{\partial z} \\ \dfrac{\partial u_z}{\partial r} & \dfrac{\partial u_z}{\partial z} \end{pmatrix}, \text{ and } \nabla_{\text{axi}} \mathbf{u} = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & 0 & \dfrac{\partial u_r}{\partial z} \\ 0 & \frac{1}{r} u_r & 0 \\ \dfrac{\partial u_z}{\partial r} & 0 & \dfrac{\partial u_z}{\partial z} \end{pmatrix}. \tag{2.44}$$

Note that it is necessary to represent the gradient and axisymmetric gradient as tensors with different sizes because the non-constant nature of the cylindrical coordinate unit vectors creates additional terms in axisymmetric derivatives. However, in order to express the meridian problem using a two-dimensional formulation, we represent the tensor $\nabla_{\text{axi}}\mathbf{u}$ as an ordered pair made up of a tensor and a scalar function. That is

$$\nabla_{\text{axi}}\ \mathbf{u} = (\nabla\ \mathbf{u}, \frac{1}{r}u_r). \tag{2.45}$$

Next, for the axisymmetric vector $\mathbf{u} = (u_r, u_z)^t$, the divergence operators $\nabla\cdot$ and $\nabla_{\text{axi}}\cdot$ are defined as

$$\nabla\cdot\mathbf{u} = \frac{\partial u_r}{\partial r} + \frac{\partial u_z}{\partial z}, \text{ and } \nabla_{\text{axi}}\cdot\mathbf{u} = \frac{1}{r}\frac{\partial(r\ u_r)}{\partial r} + \frac{\partial u_z}{\partial z} = \frac{1}{r}u_r + \nabla_{rz}\cdot\mathbf{u}. \tag{2.46}$$

As alluded to in (2.45) and described in Appendix A, the stress tensor that appears in the meridian problem can be represented as $(\underline{\boldsymbol{\sigma}}, \sigma)$ where $\underline{\boldsymbol{\sigma}}$ denotes an $\mathbb{M}^2$ tensor function and σ represents a scalar function. The divergence of the meridian stress tensor is

$$\nabla_{\text{axi}}\cdot(\underline{\boldsymbol{\sigma}}, \sigma) = \begin{pmatrix} \nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}}_1 - \dfrac{1}{r}\sigma \\ \nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}}_2 \end{pmatrix}. \tag{2.47}$$

At times, the axisymmetric divergence operator will also be applied to an $\mathbb{M}^2$ tensor function $\underline{\boldsymbol{\sigma}}$, in which case

$$\nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}} = \begin{pmatrix} \nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}}_1 \\ \nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}}_2 \end{pmatrix}. \tag{2.48}$$

Note that for the skew symmetric component of $(\underline{\boldsymbol{\sigma}}, \sigma)$ we have

$$as((\underline{\boldsymbol{\sigma}}, \sigma)) = as(\underline{\boldsymbol{\sigma}}) = \mathcal{S}^2(q), \quad \text{where } \ q = \frac{1}{2}(\sigma_{12} - \sigma_{21}). \tag{2.49}$$

For a scalar function f, the axisymmetric Laplace operator Δ_{axi} takes the form

$$\Delta_{\text{axi}}f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) + \frac{\partial^2 f}{\partial z^2}. \tag{2.50}$$

For completeness, we note that for a vector function $\mathbf{u}$, the axisymmetric Laplace operator Δ_{axi} takes the form

$$\Delta_{\text{axi}}\mathbf{u} = \begin{pmatrix} \Delta_{\text{axi}}u_1 - \dfrac{1}{r^2}u_1 \\ \Delta_{\text{axi}}u_2 \end{pmatrix}. \tag{2.51}$$

The curl of an axisymmetric scalar function p is given by ∇_{ac} and is defined as

$$\nabla_{\text{ac}}\, p = \left(\frac{\partial p}{\partial z}, -\frac{1}{r}\frac{\partial (r\, p)}{\partial r} \right). \tag{2.52}$$

Note that ∇_{ac} returns a row-vector. For a vector function $\mathbf{p} = (p_r, p_z)^t$ we have

$$\nabla_{\text{ac}}\, \mathbf{p} = \begin{pmatrix} \nabla_{\text{ac}}\, p_r \\ \nabla_{\text{ac}}\, p_z \end{pmatrix}. \tag{2.53}$$

In addition to the divergence and curl, the cylindrical coordinate inner product also takes a different form from the Cartesian inner product. Indeed, consider the change of variables for a Cartesian function $\hat{p} \in L^2(\breve{\Omega})$ into cylindrical coordinates

$$\int_{\breve{\Omega}} \hat{p}^2\, d\breve{\Omega} = \iint_{\Omega} \int_{\theta=0}^{2\pi} p^2 r\, d\theta\, dr\, dz. \tag{2.54}$$

Notice the $r = r(\mathbf{x})$ scaling in the measure. In the axisymmetric setting, $p \equiv p(r, z)$ and the θ integral can be computed to give a factor of 2π. As this term is a constant factor in all such integrals arising, we omit it. To distinguish the cylindrical coordinate inner product from the Cartesian inner product, we use the following notation

$$\int_{\Omega} p\, q\, r\, dr\, dz = (p, q). \tag{2.55}$$

To account for this scaling in the inner product, we introduce the following function spaces

$$
\begin{aligned}
{}_{\alpha}L^2(\Omega) &= \{v : \int_{\Omega} v^2 r^{\alpha}\, dr\, dz < \infty\},\\
{}_{\alpha}\mathbf{L}^2(\Omega) &= \{\mathbf{v} \in \mathbb{R}^n : v_i \in {}_{\alpha}L^2(\Omega) \text{ for } i = 1, ..., n\},\\
{}_{\alpha}\underline{\mathbf{L}}^2(\Omega, \mathbb{M}^n) &= \{\underline{\boldsymbol{\sigma}} \in \mathbb{M}^n : \sigma_{ij} \in {}_{\alpha}L^2(\Omega) \text{ for } i = 1, \cdots n \text{ and } j = 1, \cdots n\},\\
{}_{\alpha}\underline{\mathbf{L}}^2(\Omega, \mathbb{K}^n) &= \{\underline{\boldsymbol{\sigma}} \in \mathbb{K}^n : \sigma_{ij} \in {}_{\alpha}L^2(\Omega) \text{ for } i = 1, \cdots n \text{ and } j = 1, \cdots n\}.
\end{aligned}
$$

The norms associated with these ${}_{\alpha}L^2$ spaces are

$$
\begin{aligned}
&\|v\|^2_{{}_{\alpha}L^2(\Omega)} = \int_{\Omega} v^2 r^{\alpha}\, dr\, dz, \quad \|\mathbf{v}\|^2_{{}_{\alpha}\mathbf{L}^2(\Omega)} = \sum_{i=1}^{n} \|v_i\|^2_{{}_{\alpha}L^2(\Omega)},\\
&\|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{L}}^2(\Omega,\mathbb{M}^n)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \|\sigma_{ij}\|^2_{{}_{\alpha}L^2(\Omega)} \quad \text{and} \quad \|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{L}}^2(\Omega,\mathbb{K}^n)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \|\sigma_{ij}\|^2_{{}_{\alpha}L^2(\Omega)}.
\end{aligned}
$$

In addition to the ${}_{\alpha}L^2$ spaces, the elasticity problem requires divergence spaces for the stress tensors. These spaces are

$$
\begin{aligned}
{}_{\alpha}\mathbf{H}(\text{div}_{\text{axi}}, \Omega) &= \{\mathbf{v} \in {}_{\alpha}\mathbf{L}^2(\Omega) : \nabla_{\text{axi}} \cdot \mathbf{v} \in {}_{\alpha}L^2(\Omega)\},\\
{}_{\alpha}\underline{\mathbf{H}}(\text{div}_{\text{axi}}, \Omega; \mathbb{M}^n) &= \{\underline{\boldsymbol{\sigma}} \in {}_{\alpha}\underline{\mathbf{L}}^2(\Omega; \mathbb{M}^n) : \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} \in {}_{\alpha}\mathbf{L}^2(\Omega)\},\\
{}_{\alpha}\underline{\mathbf{H}}(\text{div}_{\text{axi}}, \Omega; \mathbb{K}^n) &= \{\underline{\boldsymbol{\sigma}} \in {}_{\alpha}\underline{\mathbf{L}}^2(\Omega; \mathbb{K}^n) : \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} \in {}_{\alpha}\mathbf{L}^2(\Omega)\}.
\end{aligned}
$$

with norms

$$
\begin{aligned}
\|\mathbf{v}\|^2_{{}_{\alpha}\mathbf{H}(\text{div}_{\text{axi}},\Omega)} &= \|\nabla_{\text{axi}} \cdot \mathbf{v}\|^2_{{}_{\alpha}L^2(\Omega)} + \|\mathbf{v}\|^2_{{}_{\alpha}\mathbf{L}^2(\Omega)},\\
\|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{H}}(\text{div}_{\text{axi}},\Omega;\, \mathbb{M}^n)} &= \|\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\mathbf{L}^2(\Omega)} + \|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{L}}^2(\Omega;\mathbb{M}^n)},\\
\|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{H}}(\text{div}_{\text{axi}},\Omega;\, \mathbb{K}^n)} &= \|\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\mathbf{L}^2(\Omega)} + \|\underline{\boldsymbol{\sigma}}\|^2_{{}_{\alpha}\underline{\mathbf{L}}^2(\Omega;\mathbb{K}^n)}.
\end{aligned}
$$

Various forms of the weighted ${}_1H^k(\Omega)$ function space also appear in this work. To begin, the most important family of $H^k(\Omega)$ spaces that appear in the work are of the

$H^1(\Omega)$ variety. Specifically, let

$$
\begin{aligned}
{}_\alpha\mathbf{H}^1(\Omega) &= \{\mathbf{v} \in {}_\alpha\mathbf{L}^2(\Omega) : \nabla v_i \in {}_\alpha\mathbf{L}^2(\Omega),\ i = 1, \cdots, n\}, \\
{}_\alpha\underline{\mathbf{H}}^1(\Omega; \mathbb{M}^n) &= \{\underline{\boldsymbol{\sigma}} \in {}_\alpha\underline{\mathbf{L}}^2(\Omega; \mathbb{M}^n) : \nabla \sigma_{i,j} \in {}_\alpha\mathbf{L}^2(\Omega),\ i, j = 1, \cdots, n\},
\end{aligned}
$$

with norms

$$
\begin{aligned}
\|\mathbf{v}\|^2_{{}_\alpha\mathbf{H}^1(\Omega)} &= \sum_{i=1}^n \|\nabla v_i\|^2_{{}_\alpha\mathbf{L}^2(\Omega)} + \|\mathbf{v}\|^2_{{}_\alpha\mathbf{L}^2(\Omega)}, \\
\|\underline{\boldsymbol{\sigma}}\|^2_{{}_\alpha\underline{\mathbf{H}}^1(\Omega)} &= \sum_{i=1}^n \|\nabla \boldsymbol{\sigma}_i\|^2_{{}_\alpha\mathbf{L}^2(\Omega, \mathbb{M}^n)} + \|\underline{\boldsymbol{\sigma}}\|^2_{{}_\alpha\underline{\mathbf{L}}^2(\Omega; \mathbb{M}^n)}.
\end{aligned}
$$

More generally, one can define the function space $H^k(\Omega)$ for any $k \geq 1$. First, let ζ be a positive integer and v be ζ times differentiable. One can then define the vector

$$
\nabla^\zeta v = \left[\frac{\partial^\zeta v}{(\partial r)^\zeta}, \frac{\partial^\zeta v}{(\partial r)^{\zeta-1} \partial z}, \cdots, \frac{\partial^\zeta v}{(\partial r)(\partial z)^{\zeta-1}}, \frac{\partial^\zeta v}{(\partial z)^\zeta} \right].
$$

Then,

$$
\begin{aligned}
{}_\alpha H^k(\Omega) &= \{v \in {}_\alpha L^2(\Omega) : \nabla^\zeta v \in {}_\alpha\mathbf{L}^2(\Omega) \text{ for all } \zeta \leq k\}, \\
{}_\alpha\mathbf{H}^k(\Omega) &= \{\mathbf{v} \in {}_\alpha\mathbf{L}^2(\Omega) : \nabla^\zeta v_i \in {}_\alpha\mathbf{L}^2(\Omega) \text{ for all } \zeta \leq k \text{ and } i = 1, 2, \cdots n\}, \\
{}_\alpha\underline{\mathbf{H}}^k(\Omega; \mathbb{M}^n) &= \{\underline{\boldsymbol{\sigma}} \in {}_\alpha\mathbf{L}^2(\Omega; \mathbb{M}^n) : \nabla^\zeta \sigma_{i,j} \in {}_\alpha\mathbf{L}^2(\Omega) \text{ for all } \zeta \leq k \text{ and } i, j = 1, 2, \cdots n\},
\end{aligned}
$$

with norms

$$
\begin{aligned}
\|v\|^2_{{}_\alpha H^k(\Omega)} &= \|v\|^2_{{}_\alpha L^2(\Omega)} + \sum_{\zeta=1}^k \|\nabla^\zeta v\|^2_{{}_\alpha\mathbf{L}^2(\Omega)}, \\
\|\mathbf{v}\|^2_{{}_\alpha\mathbf{H}^k(\Omega)} &= \|\mathbf{v}\|^2_{{}_\alpha\mathbf{L}^2(\Omega)} + \sum_{i=1}^n \sum_{\zeta=1}^k \|\nabla^\zeta v_i\|^2_{{}_\alpha\mathbf{L}^2(\Omega)}, \\
\|\underline{\boldsymbol{\sigma}}\|^2_{{}_\alpha\underline{\mathbf{H}}^k(\Omega; \mathbb{M}^n)} &= \|\underline{\boldsymbol{\sigma}}\|^2_{{}_\alpha\underline{\mathbf{L}}^2(\Omega; \mathbb{M}^n)} + \sum_{i=1}^n \sum_{j=1}^n \sum_{\zeta=1}^k \|\nabla^\zeta \sigma_{ij}\|^2_{{}_\alpha\mathbf{L}^2(\Omega)}.
\end{aligned}
$$

Next we consider some subtle details related to function spaces containing axisymmetric derivative terms. To begin, recall from (2.44), that the gradient of the axisymmetric vector $\mathbf{v}$ has the form

$$\nabla_{\text{axi}}\mathbf{v} = (\nabla\mathbf{v}, \frac{1}{r}v_r), \tag{2.56}$$

which implies that

$$\|\nabla_{\text{axi}}\mathbf{v}\|^2_{{}_1L^2(\Omega)} = \int_\Omega \nabla_{\text{axi}}\mathbf{v} : \nabla_{\text{axi}}\mathbf{v}\; r\, d\Omega = \int_\Omega \nabla\mathbf{v} : \nabla\mathbf{v}\; r\, d\Omega + \int_\Omega \frac{1}{r}v_r^2\, d\Omega. \tag{2.57}$$

Therefore, in order that $\|\nabla_{\text{axi}}\mathbf{v}\|_{{}_1L^2(\Omega)} < \infty$, it is necessary for $v_r \in {}_1H^1(\Omega)$ and $v_r \in {}_{-1}L^2(\Omega)$. To denote this important subspace of ${}_1H^1(\Omega)$, we define

$${}_1V^1(\Omega) = \{v \in {}_1H^1(\Omega) : v \in {}_{-1}L^2(\Omega)\}. \tag{2.58}$$

For general k, let

$${}_1V^k(\Omega) = \{v \in {}_1H^k(\Omega) : v \in {}_{-1}L^2(\Omega)\}. \tag{2.59}$$

We also define a norm for $v \in {}_1V^k(\Omega)$ as

$$\|v\|_{{}_1V^k(\Omega)} = \left(\sum_{j=1}^{k} |v|^2_{{}_1H^j(\Omega)} + \|v\|^2_{{}_{-1}L^2(\Omega)}\right)^{\frac{1}{2}}. \tag{2.60}$$

It is also important to observe that unlike in the Cartesian setting, ${}_1\mathbf{H}^1(\Omega) \not\subset {}_1\mathbf{H}(\text{div}_{\text{axi}}, \Omega)$. When referencing a function space that has a vanishing trace along the boundary, we adopt the standard convention of including a zero subscript. For example,

$${}_1H^1_0(\Omega) = \{v \in {}_1H^1_0(\Omega) : v = 0 \text{ on } \Gamma\}. \tag{2.61}$$

It is important to highlight that Γ here does not include the rotation axis portion of the boundary of Ω as illustrated in Figure 2-1.

In the discussions that follow, we take $U = {}_1\mathbf{L}^2(\Omega)$, $Q = {}_1L^2(\Omega)$. As the merdian stress tensor is made up of a tensor and scalar component, we introduce the space $\mathbf{\Sigma}(\Omega)$ defined by

$$\mathbf{\Sigma}(\Omega) = \{(\underline{\boldsymbol{\sigma}}, \sigma) \in {}_1\underline{\mathbf{L}}^2(\Omega, \mathbb{M}^2) \times {}_1L^2(\Omega) \ : \ \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \in {}_1L^2(\Omega)\}. \tag{2.62}$$

Associated with $\mathbf{\Sigma}(\Omega)$ we have the norm

$$\|(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\mathbf{\Sigma}(\Omega)} = \left(\|\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma)\|^2_{{}_1\mathbf{L}^2(\Omega)} + \|\underline{\boldsymbol{\sigma}}\|^2_{{}_1\underline{\mathbf{L}}^2(\Omega;\, \mathbb{M}^2)} + \|\sigma\|^2_{{}_1L^2(\Omega)}\right)^{\frac{1}{2}}. \tag{2.63}$$

Additionally, we define $\boldsymbol{S}(\Omega) \subset \mathbf{\Sigma}(\Omega)$ by

$$\boldsymbol{S}(\Omega) = \{(\underline{\boldsymbol{\sigma}}, \sigma) \in \mathbf{\Sigma}(\Omega) : \underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}^1(\Omega;\ \mathbb{M}^2), \sigma \in {}_{-1}L^2(\Omega)\}, \tag{2.64}$$

with norm

$$\|(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\boldsymbol{S}(\Omega)} = \left(\|\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma)\|^2_{{}_1\mathbf{L}^2(\Omega)} + \|\underline{\boldsymbol{\sigma}}\|^2_{{}_1\underline{\mathbf{H}}^1(\Omega;\, \mathbb{M}^2)} + \|\sigma\|^2_{{}_{-1}L^2(\Omega)}\right)^{\frac{1}{2}}. \tag{2.65}$$

For convenience, when referencing Ω specifically, $\mathbf{\Sigma}(\Omega)$ and $\boldsymbol{S}(\Omega)$ will be denoted as $\mathbf{\Sigma}$ and $\boldsymbol{S}$.

2.5 Axisymmetric Strong Form

The strong form of the axisymmetric problem is the same as the problem described in (2.12) (restated here for convenience)

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \underline{\boldsymbol{\epsilon}}(\mathbf{u}), \quad \nabla \cdot \underline{\boldsymbol{\sigma}} = \boldsymbol{f} \text{ in } \Omega \tag{2.66}$$

with the boundary conditions described in (2.14).

For the complete three dimensional problem, the boundary consists of the surface of an three dimensional axisymmetric object. When the axisymmetric problem has been reduced to the two dimensional meridian problem in (r, z)-space, the symmetry axis Γ_0 is treated as a special boundary. To preserve the axisymmetry property of the solution, it is necessary to apply the following boundary conditions along Γ_0

$$\mathbf{u} \cdot \mathbf{n} = 0 \text{ and } \underline{\boldsymbol{\sigma}} \cdot \mathbf{n} = \mathbf{0} \text{ on } \Gamma_0. \tag{2.67}$$

The first condition prevents any normal displacement along the symmetry axis while the second condition prevents any normal stress from occurring along the symmetry axis.

2.6 Axisymmetric Weak Form

In this section we present the weak form of the axisymmetric meridian problem. This problem has many similarities with the Cartesian problem, however, new terms are introduced into the bilinear forms as a consequence of the change of variable from Cartesian to cylindrical coordinates. Details of the derivation can be found in Appendix A.
First, define the bilinear form $\tilde{a}(.,.) : \boldsymbol{\Sigma} \times \boldsymbol{\Sigma} \to \mathbb{R}$,

$$\tilde{a}((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\tau}}, \tau)) = (\mathcal{A}\underline{\boldsymbol{\sigma}}, \underline{\boldsymbol{\tau}}) + (\mathcal{A}\sigma, \tau) - \frac{1}{2\mu}\frac{\lambda}{2\mu + 3\lambda}((\sigma, \mathrm{tr}(\underline{\boldsymbol{\tau}})) + (\mathrm{tr}(\underline{\boldsymbol{\sigma}}), \tau)), \tag{2.68}$$

and the bilinear form $\tilde{b}(.,.) : \boldsymbol{\Sigma} \times U \to \mathbb{R}$,

$$\tilde{b}((\underline{\boldsymbol{\tau}}, \tau), \mathbf{u}) = (\nabla_{\mathrm{axi}} \cdot \underline{\boldsymbol{\tau}}, \mathbf{u}) - (\frac{\tau}{r}, u_r). \tag{2.69}$$

The axisymmetric meridian problem is then defined as: Given $\mathbf{f} \in {}_1\mathbf{L}^2(\Omega)$, find $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{u}) \in \boldsymbol{\Sigma} \times U$ such that

$$\tilde{a}((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\tau}}, \tau)) + \tilde{b}((\underline{\boldsymbol{\tau}}, \tau), \mathbf{u}) = 0 \tag{2.70}$$

$$\tilde{b}((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{2.71}$$

for all $((\underline{\boldsymbol{\tau}}, \tau), \mathbf{v}) \in \boldsymbol{\Sigma} \times U$.

For the weak symmetry version of the problem, (recall (2.19)), we define the bilinear form $\tilde{c}(.,.) : \boldsymbol{\Sigma} \times Q \to \mathbb{R}$

$$\tilde{c}((\underline{\boldsymbol{\sigma}}, \sigma), p) := (\underline{\boldsymbol{\sigma}}, \mathcal{S}^2(p)). \tag{2.72}$$

The axisymmetric meridian problem with weak symmetry is then: Given $\mathbf{f} \in {}_1\mathbf{L}^2(\Omega)$, find $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{u}, p) \in \boldsymbol{\Sigma} \times U \times Q$ such that

$$\tilde{a}((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\tau}}, \tau)) + \tilde{b}((\underline{\boldsymbol{\tau}}, \tau), \mathbf{u}) + \tilde{c}((\underline{\boldsymbol{\tau}}, \tau), p) = 0 \tag{2.73}$$

$$\tilde{b}((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{2.74}$$

$$\tilde{c}((\underline{\boldsymbol{\sigma}}, \sigma), q) = 0 \tag{2.75}$$

for all $((\underline{\boldsymbol{\tau}}, \tau), \mathbf{v}, q) \in \boldsymbol{\Sigma} \times U \times Q$.

2.6.1 Modified Weak Symmetry and the Axisymmetric Elasticity Problem

Of interest is to develop discrete inf-sup stable elements for the axisymmetric elasticity problem. In cylindrical coordinates, the divergence operator does not map polynomial spaces into polynomial spaces, so some of the standard techniques for verifying inf-sup stability cannot be used. Thus, to help establish a weak formulation for which stable triples of finite elements may be verified to satisfy the discrete inf-sup condition, we make two modifications to the problem (2.73)-(2.75).

To distinguish the modified weak symmetry problem, we introduce new notation. Previously, the bilinear forms that appear in the weak formulation of the elasticity problem were denoted with a tilde symbol (e.g. $\tilde{a}(\cdot, \cdot)$). The modified bilinear forms introduced next can be distinguished because the tilde will be removed (e.g. $a(\cdot, \cdot)$). In the case of $\tilde{b}(\cdot, \cdot)$, no modification will be made. However, to maintain notational consistency, we take $\tilde{b}(\cdot, \cdot) = b(\cdot, \cdot)$.

First, we add a grad-div stabilization term to $\tilde{a}(\cdot, \cdot)$ and define a new bilinear form

$a(\cdot,\cdot) : \mathbf{\Sigma} \times \mathbf{\Sigma} :\to \mathbb{R}$

$$a((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\tau}}, \tau)) = \tilde{a}((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\tau}}, \tau)) + \gamma(\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma), \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \tag{2.76}$$

where γ is the grad-div stabilization term. Unless specified otherwise, we take $\gamma = 1$. As discussed in Section 2.6.2, this stabilization term ensures that $a((\cdot,\cdot),(\cdot,\cdot))$ is coercive in the $\|\cdot\|_{\mathbf{\Sigma}}$ norm. Also, recall from (2.12) that in cylindrical coordinates, $\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) = \mathbf{f}$. Therefore, to account for the grad-div stabilization term in the constituent equation, $(\mathbf{f}, \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau))$ must also be added to the right hand side of (2.73).

For the second modification, recall that $\tilde{c}((\underline{\boldsymbol{\sigma}}, \sigma), q) = (\underline{\boldsymbol{\sigma}}, \mathcal{S}^2(q))$ and let $\mathbf{x} = (r, z)^t$. As described in Lemma 7 below,

$$\begin{aligned}\int_\Omega \underline{\boldsymbol{\sigma}} : \mathcal{S}^2(q)\; r\; d\,\Omega = &- \int_\Omega (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x})\; q\; r\; d\Omega \\ &+ \int_{\partial\Omega} (\underline{\boldsymbol{\sigma}} \cdot \mathbf{n}) \cdot \mathbf{x}^\perp q\; r\; ds - \int_\Omega \underline{\boldsymbol{\sigma}} : (\mathbf{x}^\perp \otimes \nabla q)\; r\; d\Omega \\ &- \int_\Omega \frac{1}{r} \sigma\; z\; q\; r\; d\Omega,\end{aligned} \tag{2.77}$$

or equivalently

$$\begin{aligned}\int_\Omega \underline{\boldsymbol{\sigma}} : \mathcal{S}^2(q)\; r\; d\Omega + &\int_\Omega (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x})\; q\; r\; d\Omega \\ &= \int_{\partial\Omega} (\underline{\boldsymbol{\sigma}} \cdot \mathbf{n}) \cdot \mathbf{x}^\perp q\; r\; ds - \int_\Omega \underline{\boldsymbol{\sigma}} : (\mathbf{x}^\perp \otimes \nabla q)\; r\; d\Omega \\ &- \int_\Omega \sigma\; z\; q\; d\Omega.\end{aligned} \tag{2.78}$$

In terms of establishing stable approximation elements via the construction of a suitable projection (see Theorem 1) it is much more convenient to use equation (2.78) than (2.77). To introduce (2.78) into the weak form, we add $\int_\Omega (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x})\; q\; r\; d\Omega$ to both sides of (2.75) giving

$$\tilde{c}((\underline{\boldsymbol{\sigma}}, \sigma), q) + \int_\Omega (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x})\; q\; r\; d\Omega = \int_\Omega (\mathbf{f} \wedge \mathbf{x})\; q\; r\; d\Omega, \tag{2.79}$$

where we have used the relationship $\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) = \mathbf{f}$ on the right hand side. To represent the left hand side of (2.79), we define a new bilinear form $c(\cdot,\cdot) : \boldsymbol{\Sigma} \times Q \to \mathbb{R}$ as

$$\begin{aligned} c((\underline{\boldsymbol{\sigma}}, \sigma), q) &:= \tilde{c}((\underline{\boldsymbol{\sigma}}, \sigma), q) + (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x}, q) \\ &= (\underline{\boldsymbol{\sigma}}, \mathcal{S}^2(q)) + (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \wedge \mathbf{x}, q). \end{aligned} \tag{2.80}$$

Therefore, (2.75) becomes

$$c((\underline{\boldsymbol{\sigma}}, \sigma), q) = (\mathbf{f} \wedge \mathbf{x}, q). \tag{2.81}$$

To maintain the saddle point structure of the weak formulation with the bilinear form $c(\cdot,\cdot)$, we need to add and subtract $(\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x}, p)$ to the left hand side of (2.73). To understand the affect of this modification on the weak formulation, first observe that

$$\begin{aligned} \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x} &= \begin{pmatrix} \dfrac{\partial \tau_{11}}{\partial r} + \dfrac{\partial \tau_{12}}{\partial z} + \dfrac{1}{r}(\tau_{11} - \tau) \\ \dfrac{\partial \tau_{21}}{\partial r} + \dfrac{\partial \tau_{22}}{\partial z} + \dfrac{1}{r}\tau_{21} \end{pmatrix} \wedge \begin{pmatrix} r \\ z \end{pmatrix} \\ &= z(\frac{\partial \tau_{11}}{\partial r} + \frac{\partial \tau_{12}}{\partial z} + \frac{1}{r}(\tau_{11} - \tau)) - r(\frac{\partial \tau_{21}}{\partial r} + \frac{\partial \tau_{22}}{\partial z} + \frac{1}{r}\tau_{21}) \\ &= (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \cdot \mathbf{x}^{\perp}. \end{aligned} \tag{2.82}$$

Therefore,

$$\begin{aligned} ((\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \wedge \mathbf{x}, p) &= \int_{\Omega} \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x} \; p \; r \; d\Omega \\ &= \int_{\Omega} (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \cdot \mathbf{x}^{\perp} \; p \; r \; d\Omega \\ &= b((\underline{\boldsymbol{\tau}}, \tau), \mathbf{x}^{\perp} \; p). \end{aligned} \tag{2.83}$$

This shows that $((\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x}, p)$ can be expressed as $b((\underline{\boldsymbol{\tau}}, \tau), \mathbf{x}^{\perp} p)$. As a result, the negative part of $((\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x}, p)$ that is used to balance the constituent equation enters into the expression as part of the bilinear form $b(\cdot,\cdot)$. That is,

$$b((\underline{\boldsymbol{\tau}},\tau),\mathbf{u}) - b((\underline{\boldsymbol{\tau}},\tau),\mathbf{x}^{\perp}p) = b((\underline{\boldsymbol{\tau}},\tau),\mathbf{u}-\mathbf{x}^{\perp}p). \tag{2.84}$$

To reflect the fact that the expression within the bilinear form $b(\cdot,\cdot)$ no longer depends only on the displacement $\mathbf{u}$, we define a new variable $\mathbf{w} = \mathbf{u} - \mathbf{x}^{\perp}p$. As we discuss further in Sections 2.12 and 2.13, once the true solution has be found, the true displacement $\mathbf{u} = \mathbf{w} + \mathbf{x}^{\perp}p$ can be accurately recovered during a post-processing step. As an additional comment, we must specify a boundary condition for the pseudo displacement $\mathbf{w}$. Since $p \in {}_1L^2(\Omega)$ and it was introduced to enforced the symmetry condition weakly, it is appropriate to impose the condition $p = 0$ on $\partial\Omega$. Therefore, the pure clamped boundary condition $\mathbf{u} = \mathbf{0}$ becomes $\mathbf{w} = \mathbf{0}$.
Therefore, an equivalent but modified version of the axisymmetric linear elasticity problem (2.73)-(2.75) can be expressed as: Given $\mathbf{f} \in {}_1\mathbf{L}^2(\Omega)$ find $((\underline{\boldsymbol{\sigma}},\sigma),\mathbf{w},p) \in \boldsymbol{\Sigma} \times U \times Q$ such that

$$a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau)) + b((\underline{\boldsymbol{\tau}},\tau),\mathbf{w}) + c((\underline{\boldsymbol{\tau}},\tau),p) = (\mathbf{f},\nabla_{\text{axi}}\cdot(\underline{\boldsymbol{\tau}},\tau)) \tag{2.85}$$

$$b((\underline{\boldsymbol{\sigma}},\sigma),\mathbf{v}) = (\mathbf{f},\mathbf{v}) \tag{2.86}$$

$$c((\underline{\boldsymbol{\sigma}},\sigma),q) = (\mathbf{f}\wedge\mathbf{x},q) \tag{2.87}$$

for all $((\underline{\boldsymbol{\tau}},\tau),\mathbf{v},q) \in \boldsymbol{\Sigma} \times U \times Q$.

2.6.2 Inf-sup stability of the modified weak problem

Next we establish a set of conditions on the spaces $\boldsymbol{\Sigma} \times U \times Q$ so that modified weak symmetry problem described in (2.85)-(2.87) are inf-sup stable. First we establish that $a(\cdot,\cdot)$ is coercive on $\boldsymbol{\Sigma} \times \boldsymbol{\Sigma}$.

Lemma 2. *The operator $a(.,.)$ defined in (2.76) is coercive. That is,*

$$a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\sigma}},\sigma)) \geq \gamma\|(\underline{\boldsymbol{\sigma}},\sigma)\|^2_{\boldsymbol{\Sigma}} \text{ where } \gamma = \min\{\frac{1}{2\mu}\frac{1}{2\mu+3\lambda},1\}. \tag{2.88}$$

Proof. We begin with the observation that

$$\begin{aligned} a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\sigma}},\sigma)) &= (\mathcal{A}\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\sigma}}) + (\mathcal{A}\sigma,\sigma) - \frac{1}{2\mu}\frac{\lambda}{2+3\lambda}[(\sigma,\mathrm{tr}(\underline{\boldsymbol{\sigma}})) + (\mathrm{tr}(\underline{\boldsymbol{\sigma}}),\sigma)] \\ &\quad + (\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma),\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma)) \\ &= \frac{1}{2\mu}(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\sigma}}) + \frac{1}{2\mu}(\sigma,\sigma) - \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda}(\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma,\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma) \\ &\quad + (\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma),\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma)). \end{aligned} \tag{2.89}$$

Next we must incorporate the $(\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma,\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma)$ term into (2.89) in a way that will allow us to obtain the $\boldsymbol{\Sigma}$ norm. To do so, we start by adding the inequalities

$$\sigma_{11}^2+\sigma_{22}^2 \geq 2\sigma_{11}\sigma_{22}, \quad \sigma_{22}^2+\sigma^2 \geq 2\sigma_{22}\sigma, \quad \sigma_{11}^2+\sigma^2 \geq 2\sigma_{11}\sigma \tag{2.90}$$

to get that $2(\sigma_{11}^2+\sigma_{22}^2+\sigma^2) \geq 2(\sigma_{11}\sigma_{22}+\sigma_{22}\sigma+\sigma_{11}\sigma)$. Adding additional positive terms to the left-hand side of this inequality gives

$$2(\sigma_{11}^2+\sigma_{22}^2+\sigma^2)+3(\sigma_{12}^2+\sigma_{21}^2) \geq 2(\sigma_{11}\sigma_{22}+\sigma_{22}\sigma+\sigma_{11}\sigma). \tag{2.91}$$

Since $(\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma)^2 = \sigma_{11}^2+\sigma_{22}^2+\sigma^2+2(\sigma_{11}\sigma_{22}+\sigma_{22}\sigma+\sigma_{11}\sigma)$ and $\mu,\lambda>0$,

$$\begin{aligned} \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda}(\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma,\mathrm{tr}(\underline{\boldsymbol{\sigma}})+\sigma) &\leq \frac{1}{2\mu}\frac{3\lambda}{2\mu+3\lambda}\int_\Omega(\sigma_{11}^2+\sigma_{12}^2+\sigma_{21}^2+\sigma_{22}^2+\sigma^2)\,r\,d\Omega \\ &= \frac{1}{2\mu}\frac{3\lambda}{2\mu+3\lambda}(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\sigma}}) + \frac{1}{2\mu}\frac{3\lambda}{2\mu+3\lambda}(\sigma,\sigma). \end{aligned} \tag{2.92}$$

Combining (2.89) and (2.92)

$$\begin{aligned} a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\sigma}},\sigma)) &\geq \frac{1}{2\mu}\frac{2\mu}{2\mu+3\lambda}\left((\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\sigma}})+(\sigma,\sigma)\right) + (\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma),\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma)) \\ &= \frac{1}{2\mu+3\lambda}\|(\underline{\boldsymbol{\sigma}},\sigma)\|^2_{{}_1\mathbf{L}^2(\Omega;\mathbb{M}^2\times\mathbb{R}^1)} + \|\nabla_{\mathrm{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma)\|^2_{{}_1\underline{\mathbf{L}}^2(\Omega)} \\ &\geq \gamma\|(\underline{\boldsymbol{\sigma}},\sigma)\|^2_{\boldsymbol{\Sigma}} \end{aligned} \tag{2.93}$$

where $\gamma = \min\{\frac{1}{2\mu+3\lambda},1\}$. □

Lemma 3. *The operator $a(\cdot,\cdot)$ is bounded. That is,*

$$a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau)) \leq \alpha \|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\boldsymbol{\Sigma}} \|(\underline{\boldsymbol{\tau}},\tau)\|_{\boldsymbol{\Sigma}} \tag{2.94}$$

for some $\alpha > 0$ and all $(\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau) \in \boldsymbol{\Sigma}$.

Proof. Using the Cauchy-Schwarz inequality,

$$\begin{aligned} a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau)) &= \frac{1}{2\mu}(\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\tau}}) + \frac{1}{2\mu}(\sigma,\tau) - \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda}(\operatorname{tr}(\underline{\boldsymbol{\sigma}})+\sigma, \operatorname{tr}(\underline{\boldsymbol{\tau}})+\tau) \\ &\quad + (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}},\sigma), \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}},\tau)) \\ &\leq \frac{1}{2\mu}\left(\|\underline{\boldsymbol{\sigma}}\|_{{}_1\underline{\mathbf{L}}^2(\Omega)} \|\underline{\boldsymbol{\tau}}\|_{{}_1\underline{\mathbf{L}}^2(\Omega)} + \|\sigma\|_{{}_1L^2(\Omega)} \|\tau\|_{{}_1L^2(\Omega)} \right) \\ &\quad + \|\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}},\sigma)\|_{{}_1\mathbf{L}^2(\Omega)} \|\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}},\tau)\|_{{}_1\mathbf{L}^2(\Omega)} \\ &\quad + \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda} \|\operatorname{tr}(\underline{\boldsymbol{\sigma}})+\sigma\|_{{}_1L^2(\Omega)} \|\operatorname{tr}(\underline{\boldsymbol{\tau}})+\tau\|_{{}_1L^2(\Omega)} \\ &\leq C\left(\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\boldsymbol{\Sigma}} \|(\underline{\boldsymbol{\tau}},\tau)\|_{\boldsymbol{\Sigma}} + \|\operatorname{tr}(\underline{\boldsymbol{\sigma}})+\sigma\|_{{}_1L^2(\Omega)} \|\operatorname{tr}(\underline{\boldsymbol{\tau}})+\tau\|_{{}_1L^2(\Omega)} \right). \end{aligned} \tag{2.95}$$

Further, for any $(\underline{\boldsymbol{\sigma}},\sigma) \in \boldsymbol{\Sigma}$,

$$\begin{aligned} \|\operatorname{tr}(\underline{\boldsymbol{\sigma}})+\sigma\|_{{}_1L^2(\Omega)} &= \int_\Omega (\operatorname{tr}(\underline{\boldsymbol{\sigma}})+\sigma)^2 r \, d\Omega \leq C\left((\underline{\boldsymbol{\sigma}},\underline{\boldsymbol{\sigma}}) + (\sigma,\sigma)\right) \\ &\leq C\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{{}_1\underline{\mathbf{L}}^2(\Omega;\mathbb{M}^2\times\mathbb{R}^1)} \leq C\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\boldsymbol{\Sigma}}. \end{aligned} \tag{2.96}$$

Combining (2.95) and (2.96) yields (2.94). □

2.6.3 Axisymmetric Meridian Continuous Inf-Sup Condition

Next we show that the inf-sup condition related to (2.85)-(2.87) is satisfied, i.e., there exists $C > 0$ such that

$$\inf_{\mathbf{u}\in U, p\in Q} \sup_{(\underline{\boldsymbol{\sigma}},\sigma)\in\boldsymbol{\Sigma}} \frac{b((\underline{\boldsymbol{\sigma}},\sigma),\mathbf{u}) + c((\underline{\boldsymbol{\sigma}},\sigma),p)}{\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\boldsymbol{\Sigma}}(\|\mathbf{u}\|_U + \|p\|_Q)} \geq C. \tag{2.97}$$

To establish the axisymmetric inf-sup condition (2.97), we follow a similar argument as presented in Lemma 1. However, some important modifications to the argument are necessary to account for the axisymmetric differential operators and function spaces. To help address these modifications, we first introduce Lemmas 4 - 5. Proof that the axisymmetric inf-sup condition (2.97) is satisfied is then presented in Lemma 6.

Lemma 4. *For $\beta \in {}_1L^2(\Omega)$ and $p \in {}_1L^2(\Omega)$ with $0 < \|p\|_{{}_1L^2(\Omega)} \leq 1$, there exists $\beta_s \in {}_1H^1_0(\Omega)$ such that*

$$(\beta_s, p) = (\beta, p) \text{ and } \|\beta_s\|_{{}_1H^1(\Omega)} \leq C\|\beta\|_{{}_1L^2(\Omega)}. \tag{2.98}$$

Proof. For β, p as given, consider the problem: Determine $\beta_s \in {}_1H^1_0(\Omega)$, $\lambda \in \mathbb{R}$ such that for all $v \in {}_1H^1(\Omega)$, $\mu \in \mathbb{R}$

$$\int_\Omega \nabla\beta_s \cdot \nabla v \; r \; dr \; dz + \int_\Omega \lambda \; v \; p \; r \; dr \; dz = \int_\Omega \beta \; v \; r \; dr \; dz \tag{2.99}$$

$$\int_\Omega \mu \; \beta_s \; p \; r \; dr \; dz = \int_\Omega \mu \; \beta \; p \; r \; dr \; dz. \tag{2.100}$$

For $a(\cdot,\cdot) : {}_1H^1_0(\Omega) \times {}_1H^1_0(\Omega) \to \mathbb{R}$ given by

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \; r \; dr \; dz \tag{2.101}$$

and $b(\cdot,\cdot) : {}_1H^1_0(\Omega) \times \mathbb{R} \to \mathbb{R}$ given by

$$b(v, \mu) = \int_\Omega \mu \; v \; p \; r \; dr \; dz = \mu(v, p), \tag{2.102}$$

it is straight forward to show that $a(\cdot,\cdot)$ is continuous and coercive and that $b(\cdot,\cdot)$ is continuous. Then, the existence and uniqueness of β_s and λ satisfying (2.99)-(2.100) depends upon if the inf-sup condition

$$\inf_{\mu\in\mathbb{R},\mu\neq 0} \sup_{v\in {}_1H^1_0(\Omega)} \frac{b(v, \mu)}{\|v\|_{{}_1H^1(\Omega)} \; |\mu|} \geq C > 0 \tag{2.103}$$

is satisfied.

To see that (2.103) is satisfied, choose $w \in {}_1H_0^1(\Omega)$ such that $(w,p) \neq 0$. Without loss of generality, we may assume that $(w,p) > 0$. Then, for $\mu > 0$,

$$\sup_{v \in {}_1H_0^1(\Omega)} \frac{b(v,\mu)}{\|v\|_{{}_1H^1(\Omega)}|\mu|} = \sup_{v \in {}_1H_0^1(\Omega)} \frac{\mu(v,p)}{\|v\|_{{}_1H^1(\Omega)}|\mu|} \geq \frac{\mu(w,p)}{\|w\|_{{}_1H^1(\Omega)}|\mu|} = \frac{(w,p)}{\|w\|_{{}_1H^1(\Omega)}} > 0. \tag{2.104}$$

For $\mu < 0$,

$$\sup_{v \in {}_1H_0^1(\Omega)} \frac{b(v,\mu)}{\|v\|_{{}_1H^1(\Omega)}|\mu|} = \sup_{v \in {}_1H_0^1(\Omega)} \frac{\mu(v,p)}{\|v\|_{{}_1H^1(\Omega)}|\mu|} \geq \frac{\mu(-w,p)}{\|-w\|_{{}_1H^1(\Omega)}|\mu|} = \frac{(w,p)}{\|w\|_{{}_1H^1(\Omega)}} > 0. \tag{2.105}$$

Therefore, (2.103) is satisfied, guaranteeing that (2.99)-(2.100) has a unique solution for β_s and μ. That $(\beta_s, p) = (\beta, p)$ follows directly from (2.100).

Next, we establish the stated bound for β_s. Taking $v = \beta_s$ and $\mu = \lambda$ in (2.99)-(2.100) and subtracting gives

$$\|\nabla\ \beta_s\|^2_{{}_1L^2(\Omega)} = (\beta, \beta_s) - \lambda(\beta, p). \tag{2.106}$$

Since $\beta_s \in {}_1H_0^1(\Omega)$, the Poincarè's inequality gives that there exists a $C_1 > 0$ such that

$$\|\beta_s\|^2_{{}_1H^1(\Omega)} \leq C_1 \|\nabla \beta_s\|^2. \tag{2.107}$$

Therefore, combining (2.106) and (2.107)

$$\begin{aligned} c_1\ \|\beta_s\|^2_{{}_1H^1(\Omega)} &\leq \|\nabla \beta_s\|^2_{{}_1L^2(\Omega)} \\ &\leq \|\beta\|_{{}_1L^2(\Omega)} \|\beta_s\|_{{}_1L^2(\Omega)} + |\lambda|\ \|\beta\|_{{}_1L^2(\Omega)} \|p\|_{{}_1L^2(\Omega)} \\ &\leq \frac{1}{2c_1} \|\beta\|^2_{{}_1L^2(\Omega)} + \frac{c_1}{2} \|\beta_s\|^2_{{}_1L^2(\Omega)} + |\lambda|\ \|\beta\|_{{}_1L^2(\Omega)} \|p\|_{{}_1L^2(\Omega)}. \end{aligned} \tag{2.108}$$

This implies that

$$\|\beta_s\|^2_{{}_1H^1(\Omega)} \leq \frac{1}{c_1^2}\|\beta\|^2_{{}_1L^2(\Omega)} + \frac{2}{c_1}|\lambda|\|\beta\|_{{}_1L^2(\Omega)}\|p\|_{{}_1L^2(\Omega)}. \tag{2.109}$$

Next, from the inf-sup condition (2.103), and using (2.99)

$$\begin{aligned} C\,|\lambda| &\leq \sup_{v\in {}_1H_0^1(\Omega)} \frac{\lambda(v,p)}{\|v\|_{{}_1H^1(\Omega)}} \\ &= \sup_{v\in {}_1H_0^1(\Omega)} \frac{(\beta,v) - \int_\Omega \nabla\beta_s\cdot\nabla v\; r\;dr\;dz}{\|v\|_{{}_1H^1(\Omega)}} \\ &\leq \sup_{v\in {}_1H_0^1(\Omega)} \frac{\|\beta\|_{{}_1L^2(\Omega)}\|v\|_{{}_1L^2(\Omega)} + \|\beta_s\|_{{}_1H^1(\Omega)}\|v\|_{{}_1H^1(\Omega)}}{\|v\|_{{}_1H^1(\Omega)}} \\ &\leq \|\beta\|_{{}_1L^2(\Omega)} + \|\beta_s\|_{{}_1H^1(\Omega)}. \end{aligned} \tag{2.110}$$

Combining (2.109) and (2.110) implies that

$$\begin{aligned} \|\beta_s\|^2_{{}_1L^2(\Omega)} &\leq \frac{1}{c_1^2}\|\beta\|^2_{{}_1L^2(\Omega)} + \frac{2}{c_1C}\|\beta\|^2_{{}_1L^2(\Omega)}\|p\|_{{}_1L^2(\Omega)} + \frac{2}{c_1C}\|\beta\|_{{}_1L^2(\Omega)}\|\beta_s\|_{{}_1H^1(\Omega)}\|p\|_{{}_1L^2(\Omega)} \\ &\leq \frac{1}{c_1^2}\|\beta\|^2_{{}_1L^2(\Omega)} + \frac{2}{c_1C}\|\beta\|^2_{{}_1L^2(\Omega)}\|p\|_{{}_1L^2(\Omega)} + \frac{1}{2}\|\beta_s\|^2_{{}_1H^1(\Omega)} \\ &\quad + \frac{2}{(c_1C)^2}\|\beta\|^2_{{}_1L^2(\Omega)}\|p\|^2_{{}_1L^2(\Omega)}. \end{aligned} \tag{2.111}$$

Rearranging (2.111) and using $\|p\|_{{}_1L^2(\Omega)} \leq 1$,

$$\begin{aligned} \|\beta_s\|^2_{{}_1H^1(\Omega)} &\leq \frac{2}{c_1^2}\|\beta\|^2_{{}_1L^2(\Omega)} + \frac{4}{c_1C}\|\beta\|^2_{{}_1L^2(\Omega)}\|p\|_{{}_1L^2(\Omega)} + (\frac{2}{c_1C})^2\|\beta\|^2_{{}_1L^2(\Omega)}\|p\|^2_{{}_1L^2(\Omega)} \\ &\leq C\|\beta\|_{{}_1L^2(\Omega)}. \end{aligned} \tag{2.112}$$

□

Lemma 5. *Assume $\breve{\Omega}$ is a bounded domain and $\partial\breve{\Omega}$ is C^3. Then, given $\beta_s^* \in {}_1H^1(\Omega)$*

with $(\beta_s^*, 1) = 0$*, there exists* $\mathbf{w} \in {}_1V^2(\Omega) \times {}_1H^2(\Omega)$ *satisfying*

$$\nabla_{axi} \cdot \mathbf{w} = \beta_s^* \text{ in } \Omega \tag{2.113}$$

with $\|\mathbf{w}\|_{{}_1V^2(\Omega)\times {}_1H^2(\Omega)} \leq C\ \|\beta_s^*\|_{{}_1H^1(\Omega)}$.

Proof. As $(\beta_s, 1) = 0$, consider g satisfying the Neumann problem

$$\Delta_{\text{axi}}\ g = \nabla_{\text{axi}} \cdot \nabla\ g = \beta_s^* \text{ in } \Omega \tag{2.114}$$

$$\frac{\partial g}{\partial n} = 0 \text{ on } \partial\Omega. \tag{2.115}$$

Equivalently, equations (2.114) and (2.115) can be expressed as the three dimensional Cartesian problem

$$\Delta_{(x,y,z)}\widehat{g} = \widehat{\beta}_s^* \text{ in } \breve{\Omega} \tag{2.116}$$

$$\frac{\partial\ \widehat{g}}{\partial n} = 0 \text{ on } \partial\breve{\Omega}. \tag{2.117}$$

By elliptic regularity [42], since $\widehat{\beta}_s^* \in H^1(\breve{\Omega})$, then $\widehat{g} \in H^3(\breve{\Omega})$ with $\|\widehat{g}\|_{H^3(\breve{\Omega})} \leq C\|\widehat{\beta}_s^*\|_{H^1(\breve{\Omega})}$. Then, $\widehat{\mathbf{w}} = \nabla_{(x,y,z)}\ g \in H^2(\breve{\Omega})$. The reduction formula [16] then gives $\mathbf{w} \in {}_1V^2(\Omega) \times {}_1H^2(\Omega)$, that satisfies (2.113) and $\|\mathbf{w}\|_{{}_1H^2(\Omega)} \leq C\ \|\beta_s\|_{H^1(\Omega)}$. □

Lemma 6. *For any* $\mathbf{v} \in U$ *and* $p \in Q$*, there exists a* $C > 0$ *and a* $(\underline{\boldsymbol{\tau}}, \tau) \in \mathbf{\Sigma}$ *such that*

$$b((\underline{\boldsymbol{\tau}}, \tau), \mathbf{v}) + c((\underline{\boldsymbol{\tau}}, \tau), p) = \|\mathbf{v}\|_U^2 + \|p\|_Q^2 \tag{2.118}$$

and

$$\|(\underline{\boldsymbol{\tau}}, \tau)\|_{\mathbf{\Sigma}} \leq C(\|\mathbf{v}\|_U + \|p\|_Q). \tag{2.119}$$

Proof. The approach used in this proof follows a similar outline to the one used in Lemma 1. Specifically, we will construct two tensor and scalar pairs $(\underline{\boldsymbol{\tau}}^1, \tau^1)$ and $(\underline{\boldsymbol{\tau}}^2, \tau^2)$ such that $(\underline{\boldsymbol{\tau}}, \tau) = (\underline{\boldsymbol{\tau}}^1, \tau^1) + (\underline{\boldsymbol{\tau}}^2, \tau^2)$ satisfies (2.118) and (2.119).

Let $\mathbf{v} \in U$ and $p \in Q$ be given. By a simple scaling argument, without loss of generality, we may assume that $\|\mathbf{v}\|_U + \|p\|_Y \leq 1$. For $\mathbf{v} = (v_1, v_2)^t$, there exist vectors $\mathbf{w}_1, \mathbf{w}_2 \in {}_1V^1(\Omega) \times {}_1H^1(\Omega)$ such that

$$\nabla_{\text{axi}} \cdot \mathbf{w}_1 = v_1 \text{ and } \nabla_{\text{axi}} \cdot \mathbf{w}_2 = v_2 \tag{2.120}$$

where $\|\nabla_{\text{axi}} \cdot \mathbf{w}_1\|_{{}_1L^2(\Omega)} + \|\mathbf{w}_1\|_{{}_1V^1(T)\times {}_1H^1(T)} \leq C\ \|v_1\|_{{}_1L^2(\Omega)}$ and
$\|\nabla_{\text{axi}} \cdot \mathbf{w}_2\|_{{}_1L^2(\Omega)} + \|\mathbf{w}_2\|_{{}_1V^1(T)\times {}_1H^1(T)} \leq C\ \|v_2\|_{{}_1L^2(\Omega)}$. To compute the vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, one can map the axisymmetric scalar functions v_1 and v_2 into 3D Cartesian space and solve scalar Laplace equations to obtain functions t_1 and t_2. The gradient functions $\widehat{\mathbf{w}}_1 = \nabla_{(x,y,z)} t_1$ and $\widehat{\mathbf{w}}_2 = \nabla_{(x,y,z)} t_2$ are then computed. Finally, the reduction mapping described in [16], can be used to map $\widehat{\mathbf{w}}_1$ and $\widehat{\mathbf{w}}_2$ to Ω and create $\mathbf{w}_1$ and $\mathbf{w}_2$.

Using $\mathbf{w}_1$ and $\mathbf{w}_2$, we then construct a matrix $\underline{\boldsymbol{\tau}}^1$, where

$$\underline{\boldsymbol{\tau}}^1 = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix}. \tag{2.121}$$

Thus, taking $(\underline{\boldsymbol{\tau}}^1, \tau^1) = (\underline{\boldsymbol{\tau}}^1, 0) \in (({}_1V^1(\Omega) \times {}_1H^1(\Omega))^2, {}_{-1}L^2(\Omega))$, one has that

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}^1, \tau^1) = \mathbf{v}, \quad \text{hence} \quad b((\underline{\boldsymbol{\tau}}^1, \tau^1), \mathbf{v}) = \|\mathbf{v}\|_U^2, \tag{2.122}$$

and

$$\|(\underline{\boldsymbol{\tau}}^1, \tau^1)\|_{\boldsymbol{\Sigma}} \leq \|(\underline{\boldsymbol{\tau}}^1, \tau)\|_{\boldsymbol{S}} \leq C\|\mathbf{v}\|_U \leq C(\|\mathbf{v}\|_U + \|p\|_Q). \tag{2.123}$$

To build $(\underline{\boldsymbol{\tau}}^2, \tau^2) \in \boldsymbol{\Sigma} \times S$, we first choose $\theta, \gamma \in {}_1L^2(\Omega)$ such that

$$\mathcal{S}^2(\theta) = as(\underline{\boldsymbol{\tau}}^1), \quad \text{and } \gamma = \frac{1}{2}(v_1 z - v_2 r) = \frac{1}{2}(\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}^1 \wedge \mathbf{x}). \tag{2.124}$$

Next, set $\beta = (\gamma + \theta - \frac{1}{2}p) \in {}_1L^2(\Omega)$. Note that

$$\begin{aligned} \|\theta\|^2_{{}_1L^2(\Omega)} &= \int_\Omega (\tau_{12} - \tau_{21})^2 \, r \, d\Omega \le 2 \int_\Omega (\tau_{11}^2 + \tau_{12}^2 + \tau_{21}^2 + \tau_{22}^2) \, r \, d\Omega \\ &\le 2\|(\underline{\boldsymbol{\tau}}^1, \tau^1)\|_{\boldsymbol{\Sigma}} \le C(\|\mathbf{v}\|^2 + \|p\|^2_Q) \end{aligned} \tag{2.125}$$

Also,

$$\|\gamma\|^2_{{}_1L^2(\Omega)} = \int_\Omega \frac{1}{4}(v_1 z - v_2 r)^2 \, r \, d\Omega \le \frac{1}{2}\int_\Omega (v_1^2 z^2 + v_2^2 r^2) \, r \, d\Omega \tag{2.126}$$

$$\le C \int_\Omega \mathbf{v} \cdot \mathbf{v} \, r \, d\Omega = C\|\mathbf{v}\|^2_U \tag{2.127}$$

where $C = \max_\Omega\{\dfrac{r^2}{2}, \dfrac{z^2}{2}\}$.

Therefore,

$$\begin{aligned} \|\beta\|_{{}_1L^2(\Omega)} &\le C(\|p\|_Q + \|\theta\|_{{}_1L^2(\Omega)} + \|\gamma\|_{{}_1L^2(\Omega)}) \\ &\le C(\|\mathbf{v}\|_U + \|p\|_Q). \end{aligned} \tag{2.128}$$

Using Lemma 4, we construct $\beta_s \in {}_1H^1(\Omega)$ such that $(\beta_s, p) = (\beta, p)$ and $\|\beta_s\|_{{}_1H^1(\Omega)} \le C \, \|\beta\|_{{}_1L^2(\Omega)}$. It then follows that

$$(\mathcal{S}^2(\beta_s), \mathcal{S}^2(p)) = (\mathcal{S}^2(\beta), \mathcal{S}^2(p)) = (\mathcal{S}^2(\gamma + \theta - \frac{1}{2}p), \mathcal{S}^2(p)). \tag{2.129}$$

Define $\beta_s^* = \beta_s - \bar{\beta}$ where $\bar{\beta} = \dfrac{1}{|\Omega|}\int_\Omega \beta_s r \, dr \, dz$. Observe that

$$\|\bar{\beta}\|_{{}_1L^2(\Omega)} = \left(\int_\Omega \left(\frac{1}{|\Omega|}\int_\Omega \beta_s r \, dr \, dz\right)^2 \, r dr \, dz\right)^{\frac{1}{2}} \le C \, \|\beta_s\|_{{}_1L^2(\Omega)} \le C \, \|\beta_s\|_{{}_1H^1(\Omega)}, \tag{2.130}$$

and

$$\begin{aligned}\|\beta_s^*\|_{{}_1H^1(\Omega)} &\leq \|\beta_s\|_{{}_1H^1(\Omega)} + \|\bar{\beta}\|_{{}_1L^2(\Omega)} \\ &\leq \|\beta_s\|_{{}_1H^1(\Omega)} + c_1\, \|\beta_s\|_{{}_1H^1(\Omega)} \\ &\leq C\, \|\beta_s\|_{{}_1H^1(\Omega)}.\end{aligned} \tag{2.131}$$

Moreover, by construction $(\beta_s^*, 1) = 0$. As a result, Lemma 5 ensures that a $\mathbf{w} \in {}_1V^2(\Omega) \times {}_1H^2(\Omega)$ exists that satisfies

$$\nabla_{\text{axi}} \cdot \mathbf{w} = \beta_s^* = \beta_s - \bar{\beta} \text{ in } \Omega, \tag{2.132}$$

and

$$\|\mathbf{w}\|_{{}_1V^2(\Omega)\times {}_1H^2(\Omega)} \leq C\|\beta_s^*\|_{{}_1H^1(\Omega)} \leq \|\beta\|_{{}_1L^2(\Omega)} \leq C\, (\|\mathbf{v}\|_U + \|p\|_Q). \tag{2.133}$$

Next, we can take

$$\underline{\boldsymbol{\tau}}^2 = 2\begin{pmatrix} \dfrac{\partial w_1}{\partial z} & -\dfrac{1}{r}\dfrac{\partial(r\; w_1)}{\partial r} - \dfrac{\partial w_2}{\partial z} \\ 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & -\bar{\beta} \\ 0 & 0 \end{pmatrix} \text{ and } \tau^2 = 2\, r\, \frac{\partial^2 w_2}{\partial z\, \partial z}. \tag{2.134}$$

It follows that

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}^2, \tau^2) = 2\begin{pmatrix} \dfrac{1}{r}\dfrac{\partial}{\partial r}(r\dfrac{\partial w_1}{\partial z}) - \dfrac{1}{r}\dfrac{\partial}{\partial z}\dfrac{\partial(r\; w_1)}{\partial r} - \dfrac{\partial^2 w_2}{\partial z \partial z} + \dfrac{\partial^2 w_2}{\partial z \partial z} \\ 0 \end{pmatrix} = \mathbf{0}, \tag{2.135}$$

hence

$$b((\underline{\boldsymbol{\tau}}^2, \tau^2), \mathbf{v}) = 0 \tag{2.136}$$

and

$$\|(\underline{\boldsymbol{\tau}}^2, \tau^2)\|_{\boldsymbol{\Sigma}} \leq C\, (\|\mathbf{v}\|_U + \|p\|_Q)\,. \tag{2.137}$$

Furthermore,

$$\mathrm{as}(\underline{\boldsymbol{\tau}}^2, \tau^2) = \begin{pmatrix} 0 & -\nabla_{\mathrm{axi}} \cdot \mathbf{w} - \bar{\beta} \\ \nabla_{\mathrm{axi}} \cdot \mathbf{w} + \bar{\beta} & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\beta_s \\ \beta_s & 0 \end{pmatrix}. \tag{2.138}$$

As a result, for $(\underline{\boldsymbol{\tau}}, \tau) = (\underline{\boldsymbol{\tau}}^1, \tau^1) + (\underline{\boldsymbol{\tau}}^2, \tau^2)$ we have using (2.122) and (2.136)

$$b((\underline{\boldsymbol{\tau}}, \tau), \mathbf{v}) = b((\underline{\boldsymbol{\tau}}^1, \tau^1), \mathbf{v}) + b((\underline{\boldsymbol{\tau}}^2, \tau^2), \mathbf{v}) = \|\mathbf{v}\|_U^2, \tag{2.139}$$

and

$$\begin{aligned} c((\underline{\boldsymbol{\tau}}, \tau), p) &= c((\underline{\boldsymbol{\tau}}^1, \tau^1), p) + c((\underline{\boldsymbol{\tau}}^2, \tau^2), p) \\ &= (\underline{\boldsymbol{\tau}}^1, \mathcal{S}^2(p)) + (\nabla_{\mathrm{axi}} \cdot \underline{\boldsymbol{\tau}}^1 \wedge \mathbf{x}, p) \\ &\qquad + (\underline{\boldsymbol{\tau}}^2, \mathcal{S}^2(p)) + (\nabla_{\mathrm{axi}} \cdot (\underline{\boldsymbol{\tau}}^2, \tau^2) \wedge \mathbf{x}, p) \\ &= (\underline{\boldsymbol{\tau}}^1, \mathcal{S}^2(p)) + (\nabla_{\mathrm{axi}} \cdot \underline{\boldsymbol{\tau}}^1 \wedge \mathbf{x}, p) + (\underline{\boldsymbol{\tau}}^2, \mathcal{S}^2(p)), \ \ (\text{using } \nabla_{\mathrm{axi}} \cdot (\underline{\boldsymbol{\tau}}^2, \tau^2) = 0) \\ &= (as(\underline{\boldsymbol{\tau}}^1), \mathcal{S}^2(p)) + 2(\gamma, p) + (as(\underline{\boldsymbol{\tau}}^2), \mathcal{S}^2(p)), \ \ (\text{using (2.124)}) \\ &= (\mathcal{S}^2(\theta), \mathcal{S}^2(p)) + (\mathcal{S}^2(\gamma), \mathcal{S}^2(p)) + (\mathcal{S}^2(-\beta_s), \mathcal{S}^2(p)), \ \ (\text{using (2.138)}) \\ &= (\mathcal{S}^2(\frac{1}{2}p), \mathcal{S}^2(p)) \ \ (\text{using (2.129)}) \\ &= \|p\|_Q^2. \end{aligned} \tag{2.140}$$

Thus, $(\underline{\boldsymbol{\tau}}, \tau)$ satisfies (2.118), and using (2.123) and (2.137),

$$\begin{aligned} \|(\underline{\boldsymbol{\tau}}, \tau)\|_{\boldsymbol{\Sigma}} &\leq \|(\underline{\boldsymbol{\tau}}^1, \tau^1)\|_{\boldsymbol{\Sigma}} + \|(\underline{\boldsymbol{\tau}}^2, \tau^2)\|_{\boldsymbol{\Sigma}} \\ &\leq C(\|\mathbf{v}\|_Q + \|p\|_Q). \end{aligned} \tag{2.141}$$

□

2.7 Discrete Axisymmetric Problem

Next we introduce a discrete version of the axisymmetric meridian problem described in equations (2.85)-(2.87). For the discrete approximation we assume the following setting for the approximation spaces.

$$\mathbf{\Sigma}_h := \Sigma_{h,\underline{\boldsymbol{\sigma}}} \times \Sigma_{h,\sigma} = \{(\underline{\boldsymbol{\sigma}}_h, \sigma_h) : \underline{\boldsymbol{\sigma}}_h \in \Sigma_{h,\underline{\boldsymbol{\sigma}}}, \sigma_h \in \Sigma_{h,\sigma}\} \subset \mathbf{\Sigma} \tag{2.142}$$

$$U_h \subset U \text{ where for all } (u_{h1}, u_{h2}) \in U_h,\ u_{h1} \in \Sigma_{h,\sigma} \tag{2.143}$$

$$Q_h \subset Q \text{ and } (Q_h \cup z\, Q_h) \subset \Sigma_{h,\sigma}. \tag{2.144}$$

Additionally, we assume that there exists a piecewise polynomial space $(\Theta_h)^2$ such that $((\Theta_h)^2, Q_h)$ is a stable axisymmetric Stokes pair satisfying: Given $\beta_S^* \in {}_1H^1(\Omega)$ with $(\beta_S^*, 1) = 0$, there exists $\mathbf{w}_h \in (\Theta_h)^2$ such that

$$(\nabla_{\text{axi}} \cdot \mathbf{w}_h, q_h) = (\beta_s^*, q_h), \text{ for all } q_h \in Q_h, \tag{2.145}$$

and

$$\|\mathbf{w}_h\|_{{}_1V^1(\Omega)\times{}_1H^1(\Omega)} + \left(\sum_{T\in\mathcal{T}_h} \left\| \frac{\partial^2 w_{h2}}{\partial z^2} \right\|^2_{{}_1L^2(T)} \right)^{\frac{1}{2}} \leq C\, \|\beta_s^*\|_{{}_1H^1(\Omega)} \tag{2.146}$$

(compare (2.145) and (2.146) with (2.132) and (2.133)).
The meridan problem becomes: Given $\mathbf{f} \in {}_1\mathbf{L}^2(\Omega)$ find $((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{w}_h, p_h) \in \mathbf{\Sigma}_h \times U_h \times Q_h$ such that

$$a((\underline{\boldsymbol{\sigma}}_h, \sigma_h), (\underline{\boldsymbol{\tau}}_h, \tau_h)) + b((\underline{\boldsymbol{\tau}}_h, \tau_h), \mathbf{w}_h) + c((\underline{\boldsymbol{\tau}}_h, \tau_h), p_h) = (\mathbf{f}, \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}_h, \tau_h)) \tag{2.147}$$

$$b((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{v}_h) = (\mathbf{f}, \mathbf{v}_h) \tag{2.148}$$

$$c((\underline{\boldsymbol{\sigma}}_h, \sigma_h), q_h) = (\mathbf{f} \wedge \mathbf{x}, q_h) \tag{2.149}$$

for all $((\underline{\boldsymbol{\tau}}_h, \tau_h), \mathbf{v}_h, q_h) \in \mathbf{\Sigma}_h \times U_h \times Q_h$.

2.8 Discrete Axisymmetric Inf-Sup Condition

In this section, we introduce a framework for establishing inf-sup stability of the discrete axisymmetric problem. The approach we use is similar to that for Fortin's Lemma [24]. Given $\mathbf{u}_h \in U_h \subset U$, $p_h \in Q_h \subset Q$ we determine, as in the proof of Lemma 6, a $(\underline{\boldsymbol{\tau}}, \tau) = (\underline{\boldsymbol{\tau}}^1, \tau^1) + (\underline{\boldsymbol{\tau}}^2, \tau^2)$ such that the continuous inf-sup condition is satisfied. Then, using a suitably defined projection (see (2.161) - (2.163)), we obtain $(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \mathbf{\Sigma}_h$ such that

$$\frac{b((\underline{\boldsymbol{\tau}}_h, \tau_h), \mathbf{u}_h) + c((\underline{\boldsymbol{\tau}}_h, \tau_h), p_h)}{\|(\underline{\boldsymbol{\tau}}_h, \tau_h)\|_{\mathbf{\Sigma}}(\|\mathbf{u}_h\|_U + \|p_h\|_Q)} \geq C. \tag{2.150}$$

Helpful in this discussion is to define the restriction of the operators $b(\cdot,\cdot)$ and $c(\cdot,\cdot)$ to $T \in \mathcal{T}_h$ as:

$$b((\underline{\boldsymbol{\tau}}, \tau), \mathbf{u})_T = (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}, \mathbf{u})_T - (\frac{\tau}{r}, u_r)_T \tag{2.151}$$

$$c((\underline{\boldsymbol{\tau}}, \tau), p)_T = (as(\underline{\boldsymbol{\tau}}), \mathcal{S}^2(p))_T + ((\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \wedge \mathbf{x}, p)_T. \tag{2.152}$$

Next, we present the following useful identity for the operator $c(\cdot,\cdot)$.

Lemma 7. *For $T \in \mathcal{T}_h$,*

$$c((\underline{\boldsymbol{\tau}}, \tau), p)_T = \int_{\partial T} (\underline{\boldsymbol{\tau}} \cdot \mathbf{n}) \cdot \mathbf{x}^{\perp}\, p\, r\, ds - \int_T \underline{\boldsymbol{\tau}} : (\mathbf{x}^{\perp} \otimes \nabla\, p)\, r\, dT - \int_T \tau\, z\, p\, dT. \tag{2.153}$$

Proof. Let $\mathbf{x} = (r, z)$. Then

$$\begin{aligned}
\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) &= \nabla_{\text{axi}} \cdot \begin{pmatrix} \tau_{11} z - \tau_{21} r \\ \tau_{12} z - \tau_{22} r \end{pmatrix} \\
&= \frac{\partial}{\partial r}(\tau_{11} z - \tau_{21} r) + \frac{\partial}{\partial z}(\tau_{12} z - \tau_{22} r) + \frac{1}{r}(\tau_{11} z - \tau_{21} r) \\
&= z\frac{\partial \tau_{11}}{\partial r} - \tau_{21} - r\frac{\partial \tau_{21}}{\partial r} + \tau_{12} + z\frac{\partial \tau_{12}}{\partial z} - r\frac{\partial \tau_{22}}{\partial z} + \frac{z}{r}\tau_{11} - \tau_{21} \\
&= z(\frac{\partial \tau_{11}}{\partial r} + \frac{\partial \tau_{12}}{\partial z} + \frac{1}{r}\tau_{11}) - r(\frac{\partial \tau_{21}}{\partial r} + \frac{\partial \tau_{22}}{\partial z} + \frac{1}{r}\tau_{21}) + \tau_{12} - \tau_{21} \\
&= (\nabla_{\text{axi}} \cdot \begin{pmatrix} \tau_{11} & \tau_{12} \\ \tau_{21} & \tau_{22} \end{pmatrix}) \wedge \begin{pmatrix} r \\ z \end{pmatrix} + \begin{pmatrix} \tau_{11} & \tau_{12} \\ \tau_{21} & \tau_{22} \end{pmatrix} : \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\
&= (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}) \wedge \mathbf{x} + \underline{\boldsymbol{\tau}} : \mathbb{P}. \qquad (2.154)
\end{aligned}$$

Therefore,

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) - \frac{z}{r}\tau = \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau) \wedge \mathbf{x} + \underline{\boldsymbol{\tau}} : \mathbb{P}.$$

Next we multiply the left and right hand sides of (2.154) by $p\, r$ and integrate over T to yield

$$\begin{aligned}
\int_T \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}} \wedge \mathbf{x})\, p\, r\, dT - \int_T z\, \tau\, p\, dT &= \int_T (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \wedge \mathbf{x}\, p\, r\, dT + \int_K \underline{\boldsymbol{\tau}} : \mathbb{P}\, p\, r\, dT \\
&= ((\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}, \tau)) \wedge \mathbf{x}, p)_T + (as(\underline{\boldsymbol{\tau}}, \tau), \mathcal{S}(p))_T \\
&= c((\underline{\boldsymbol{\tau}}, \tau), p)_T. \qquad (2.155)
\end{aligned}$$

Note that we have used the relationship $\underline{\boldsymbol{\tau}} : \mathbb{P}\, p = as(\underline{\boldsymbol{\tau}}, \tau) : \mathcal{S}^2(p)$. Next, applying integration by parts to the first term on the left-hand side of (2.155) gives

$$\begin{aligned}
\int_T \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}} \wedge \mathbf{x})\, p\, r\, dT &= \int_T \nabla \cdot (r\, \underline{\boldsymbol{\tau}} \wedge \mathbf{x})\, p\, dT \\
&= \int_{\partial T} (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \mathbf{n}\, p\, r\, \partial s - \int_T (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \nabla p\ \ r\, dT. \qquad (2.156)
\end{aligned}$$

Then combining (2.155), and (2.156) yields

$$c(\underline{\boldsymbol{\tau}}, p) = \int_{\partial T} (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \mathbf{n}\; p\; r\; ds - \int_T (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \nabla p\;\; r\; dT - \int_T \tau\; z\; p\; dT. \tag{2.157}$$

Finally, since

$$(\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \nabla p = \underline{\boldsymbol{\tau}} : (\mathbf{x}^\perp \otimes \nabla p) \text{ and } (\underline{\boldsymbol{\tau}} \wedge \mathbf{x}) \cdot \mathbf{n} = (\underline{\boldsymbol{\tau}} \cdot \mathbf{n}) \cdot \mathbf{x}^\perp,$$

we have

$$c((\underline{\boldsymbol{\tau}}, \tau), p)_T = \int_{\partial T} (\underline{\boldsymbol{\tau}} \cdot \mathbf{n}) \cdot \mathbf{x}^\perp\; p\; r\; ds - \int_T \underline{\boldsymbol{\tau}} : (\mathbf{x}^\perp \otimes \nabla p)\; r\; dT - \int_T \tau\; z\; p\; dT. \tag{2.158}$$

□

Theorem 1. *Assume* $\mathbf{\Sigma}_h, U_h, Q_h$ *satisfy* (2.142)-(2.144). *If there exists a mapping* $\mathbf{\Pi}_h = \Pi_h \times \pi_h : (\boldsymbol{S} + (\nabla_{ac}((\Theta_h)^2) \times \Theta_h) \to \mathbf{\Sigma}_h$ *such that for all* $T \in \mathcal{T}_h$*:*

$$\text{if } (\underline{\boldsymbol{\sigma}}, \sigma) \in \boldsymbol{S}, \quad \|\Pi_h \times \pi_h(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\mathbf{\Sigma}(T)} \leq C\; \|(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\boldsymbol{S}(T)}, \tag{2.159}$$

$$(\underline{\boldsymbol{\sigma}}, \sigma) \in (\nabla_{ac}((\Theta_h)^2) \times \Theta_h), \quad \|\Pi_h \times \pi_h(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\mathbf{\Sigma}(T)} \leq C\; \|(\underline{\boldsymbol{\sigma}}, \sigma)\|_{\mathbf{\Sigma}(T)}, \tag{2.160}$$

and

$$\int_T (\underline{\boldsymbol{\tau}} - \Pi_h \underline{\boldsymbol{\tau}}) : (\nabla \mathbf{u}_h + \mathbf{x}^\perp \otimes \nabla q_h)\; r\; dT = 0 \quad \forall\; \mathbf{u}_h \in U_h,\; \forall\; q_h \in Q_h, \tag{2.161}$$

$$\int_\ell ((\underline{\boldsymbol{\tau}} - \Pi_h \underline{\boldsymbol{\tau}}) \cdot \mathbf{n}_K) \cdot (\mathbf{u}_h + \mathbf{x}^\perp q_h)\; r\; ds = 0 \quad \forall \textit{ edges } \ell,\; \forall\; \mathbf{u}_h \in U_h,\; \forall\; q_h \in Q_h, \tag{2.162}$$

$$\int_T \frac{1}{r}(\tau - \pi_h \tau)\; q_h\; z\; r\; dT = 0 \quad \forall\; q_h \in Q_h \tag{2.163}$$

then $\mathbf{\Sigma}_h \times U_h \times Q_h$ *are inf-sup stable.*

Proof. The approach to this proof is similar to that used in [23]. To begin, assume

$\mathbf{v}_h = (v_{h1}, v_{h2})^t \in U_h \subset {}_1\mathbf{L}^2(\Omega, \mathbb{R}^2)$ and $p_h \in Q_h \subset {}_1L^2(\Omega)$. As described in Lemma 6, there exists a tensor $(\underline{\boldsymbol{\tau}}, \tau) \in \boldsymbol{\Sigma}$ that satisfies the inf-sup condition. Moreover, as described in the proof of Lemma 6, $(\underline{\boldsymbol{\tau}}, \tau) = (\underline{\boldsymbol{\tau}}^1, \tau^1) + (\underline{\boldsymbol{\tau}}^2, \tau^2)$.

Recall from Lemma 6 that for $\mathbf{v}_h$ given, one can construct $\underline{\boldsymbol{\tau}}^1 \in ({}_1V^1(T) \times {}_1H^1(T))^2 \subset \boldsymbol{\Sigma}(T)$ such that

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}^1, 0) = \mathbf{v}_h \quad \text{ and } \quad \|(\underline{\boldsymbol{\tau}}^1, 0)\|_{\boldsymbol{S}} \leq C\left(\|\mathbf{v}_h\|_U + \|p_h\|_Q\right). \tag{2.164}$$

Note that $b((\underline{\boldsymbol{\tau}}^1, 0), \mathbf{u}_h) = (\mathbf{v}_h, \mathbf{u}_h)$, and using (2.161)-(2.162) with $q_h = 0$,

$$\begin{aligned} b((\underline{\boldsymbol{\tau}}^1, 0) - \boldsymbol{\Pi}_h(\underline{\boldsymbol{\tau}}^1, 0), \mathbf{v}_h) &= b((\underline{\boldsymbol{\tau}}^1 - \Pi_h\underline{\boldsymbol{\tau}}^1, 0), \mathbf{v}_h) \\ &= \sum_{T \in \mathcal{T}_h} (\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}^1 - \Pi_h\underline{\boldsymbol{\tau}}^1), \mathbf{v}_h)_T + (\frac{0}{r}, (\mathbf{v}_h)_1)_T = 0. \end{aligned} \tag{2.165}$$

Furthermore, from (2.159) and (2.164),

$$\|\boldsymbol{\Pi_h}(\underline{\boldsymbol{\tau}}^1, 0)\|_{\boldsymbol{\Sigma}} \leq C\|(\underline{\boldsymbol{\tau}}^1, 0)\|_{\boldsymbol{S}} \leq C(\|\mathbf{v}_h\|_U + \|p_h\|_Q). \tag{2.166}$$

Next, combining (2.158) with (2.161)-(2.162) when $\mathbf{u}_h = 0$ we have,

$$\begin{aligned} c((\underline{\boldsymbol{\tau}}^1, 0) - \boldsymbol{\Pi_h}(\underline{\boldsymbol{\tau}}^1, 0), p_h) &= \sum_{T \in \mathcal{T}_h} c((\underline{\boldsymbol{\tau}}^1 - \Pi_h\underline{\boldsymbol{\tau}}^1, 0), p_h)_T \\ &= \sum_{T \in \mathcal{T}_h} \left(\int_{\partial T} ((\underline{\boldsymbol{\tau}}^1 - \Pi_h\underline{\boldsymbol{\tau}}^1) \cdot \mathbf{n}_T) \cdot \mathbf{x}^\perp \; p_h \; r \; ds \right. \\ &\quad - \int_T (\underline{\boldsymbol{\tau}}^1 - \Pi_h\underline{\boldsymbol{\tau}}^1) : (\mathbf{x}^\perp \otimes \nabla \; p_h) \; r \; dT \\ &\quad \left. - \int_T \frac{1}{r} \; () \; z \; p_h \; r \; dT \right) = 0. \end{aligned} \tag{2.167}$$

Due to the $\frac{1}{r}\frac{\partial}{\partial r}(r \; w_1)$ entry in $\underline{\boldsymbol{\tau}}^2_{12}$ given in (2.134), $(\underline{\boldsymbol{\tau}}^2, \tau^2)$ may not lie in $\boldsymbol{S}$. As an alternative to using $(\underline{\boldsymbol{\tau}}^2, \tau^2)$, we use our assumptions that $((\Theta_h)^2 \times Q_h)$ is a stable axisymmetric Stokes pair to obtain $\mathbf{w}_h \in (\Theta_h)^2$ satisfying (2.145)-(2.146).

Note that as $w_{h1} \in {}_1V^1(\Omega)$, then $w_{h1}|_{r=0} = 0$ [16]. As w_{h1} is a piecewise polynomial, then on any triangle touching the axis of rotation, $r = 0$, we have $w_{h1} = r\ p(r,z)$ (where $p(r,z)$ is a polynomial in r and z). Thus, $\frac{1}{r}\frac{\partial}{\partial r}(r\ w_{h1}) = \frac{1}{r}w_{h1} + \frac{\partial}{\partial r}(w_{h1})$ is a polynomial in r and z on any triangle that touches the axis of rotation.

Analogous to (2.134), define

$$\underline{\boldsymbol{\tau}}_h^2 = 2\begin{pmatrix} \dfrac{\partial w_{h1}}{\partial z} & -\dfrac{1}{r}\dfrac{\partial}{\partial r}(r\ w_{h1}) - \dfrac{\partial w_{2h}}{\partial z} \\ 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & -\bar{\beta} \\ 0 & 0 \end{pmatrix} \text{ and } \tau_h^2 = 2\ r\ \frac{\partial^2\ w_{h2}}{\partial z^2}. \tag{2.168}$$

As in (2.135), (2.135) and (2.138)

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}_h^2, \tau_h^2) = \mathbf{0}, b((\underline{\boldsymbol{\tau}}_h^2, \tau_h^2), \mathbf{v}_h) = 0 \text{ and } \text{as}((\underline{\boldsymbol{\tau}}_h^2, \tau_h^2)) = \begin{pmatrix} 0 & -\beta_s \\ \beta_s & 0 \end{pmatrix}. \tag{2.169}$$

Also, from (2.146) and that $\|\beta_s^*\|_{{}_1H^1(\Omega)} \le C\left(\|\mathbf{v}_h\|_U + \|p_h\|_Q\right)$, (see (2.131) and (2.133))

$$\|(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2)\|_{\boldsymbol{\Sigma}} \le C\left(\|\mathbf{v}_h\|_U + \|p_h\|_Q\right). \tag{2.170}$$

Now, for $\boldsymbol{\Pi}_h(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2)$, proceeding as in (2.165); using (2.161)-(2.162) with $q_h = 0$ and (2.163) (and (2.143))

$$\begin{aligned} b((\underline{\boldsymbol{\tau}}_h^2, \tau_h^2) - \boldsymbol{\Pi}_h(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2), \mathbf{v}_h) &= b((\underline{\boldsymbol{\tau}}_h^2 - \Pi_h\underline{\boldsymbol{\tau}}_h^2, \tau_h^2 - \pi_h\tau_h^2), \mathbf{v}_h) \\ &= \sum_{T\in\mathcal{T}_h} \left(\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}_h^2 - \Pi_h\underline{\boldsymbol{\tau}}_h^2), \mathbf{v}_h\right)_T + (\frac{1}{r}(\tau_h^2 - \pi_h\tau_h^2), \mathbf{v}_{h1})_T \\ &= 0. \end{aligned} \tag{2.171}$$

Also, as in (2.167), and using (2.144),

$$
\begin{aligned}
c((\underline{\boldsymbol{\tau}}_h^2, \tau_h^2) - \mathbf{\Pi}_h(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2, p_h)) &= \sum_{T \in \mathcal{T}_h} c((\underline{\boldsymbol{\tau}}_h^2 - \Pi_h \underline{\boldsymbol{\tau}}_h^2), (\tau_h^2 - \pi_h \tau_h^2), p_h)_T \\
&= \sum_{T \in \mathcal{T}_h} \Bigg(\int_{\partial T} ((\underline{\boldsymbol{\tau}}_h^2 - \Pi_h \underline{\boldsymbol{\tau}}_h^2) \cdot \mathbf{n}_T) \cdot \mathbf{x}^\perp \; p_h \; r \; ds \\
&\quad - \int_T (\underline{\boldsymbol{\tau}}_h^2 - \Pi_h \underline{\boldsymbol{\tau}}_h^2) : (\mathbf{x}^\perp \otimes \nabla p_h) \; r \; dT \\
&\quad - \int_T (\tau_h^2 - \pi_h \tau_h^2) \; z \; p_h \; dT \Bigg) = 0.
\end{aligned}
\tag{2.172}
$$

Finally, with $(\underline{\boldsymbol{\tau}}_h, \tau_h) = (\Pi_h \underline{\boldsymbol{\tau}}^1, \pi_h \tau^1) + (\Pi_h \underline{\boldsymbol{\tau}}_h^2, \pi_h \tau_h^2)$,

$$
\sup_{(\underline{\boldsymbol{\sigma}}_h, \sigma_h) \in \mathbf{\Sigma}_h} \frac{b((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{v}_h) + c((\underline{\boldsymbol{\sigma}}_h, \sigma_h), p_h)}{\|(\underline{\boldsymbol{\sigma}}_h, \sigma_h)\|_{\mathbf{\Sigma}} \, (\|\mathbf{v}_h\|_U + \|p_h\|_Q)} \geq \frac{b((\underline{\boldsymbol{\tau}}_h, \tau_h), \mathbf{v}_h) + c((\underline{\boldsymbol{\tau}}_h, \tau_h), p_h)}{\|(\underline{\boldsymbol{\tau}}_h, \tau_h)\|_{\mathbf{\Sigma}} \, (\|\mathbf{v}_h\|_U + \|p_h\|_Q)}
$$

$$
\geq \frac{b((\Pi_h \underline{\boldsymbol{\tau}}^1, \pi_h \tau^1), \mathbf{v}_h) + c((\Pi_h \underline{\boldsymbol{\tau}}^1, \pi_h \tau^1), p_h) + b((\Pi_h \underline{\boldsymbol{\tau}}_h^2, \pi_h \tau_h^2), \mathbf{v}_h) + c((\Pi_h \underline{\boldsymbol{\tau}}_h^2, \pi_h \tau_h^2), p_h)}{(\|(\Pi_h \underline{\boldsymbol{\tau}}^1, \pi_h \tau^1)\|_{\mathbf{\Sigma}} + \|(\Pi_h \underline{\boldsymbol{\tau}}_h^2, \pi_h \tau_h^2)\|_{\mathbf{\Sigma}}) \, (\|\mathbf{v}_h\|_U + \|p_h\|_Q)}
$$

$$
\geq C \; \frac{b((\underline{\boldsymbol{\tau}}^1, \tau^1), \mathbf{v}_h) + c((\underline{\boldsymbol{\tau}}^1, \tau^1), p_h) + 0 + c((\underline{\boldsymbol{\tau}}_h^2, \tau_h^2), p_h)}{(\|(\underline{\boldsymbol{\tau}}^1, \tau^1)\|_{\boldsymbol{S}} + \|(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2)\|_{\mathbf{\Sigma}}) \, (\|\mathbf{v}_h\|_U + \|p_h\|_Q)}
$$

(using (2.165), (2.167), (2.170), (2.169), (2.172), (2.159) and (2.160))

$$
\geq \frac{\|\mathbf{v}_h\|_U^2 + \|p_h\|_Q^2}{(\|\mathbf{v}_h\|_U + \|p_h\|_Q + \|\mathbf{v}_h\|_U + \|p_h\|_Q)(\|\mathbf{v}_h\|_U + \|p_h\|_Q)}
$$

(using (2.122), (2.140), (2.123) and (2.170)

$\geq C.$

$\square$

Remarks

1. Concerning (2.159). Functions in $\mathbf{\Sigma}$ are not sufficiently regular to guarantee $\mathbf{\Pi}_h(\underline{\boldsymbol{\sigma}}, \sigma)$ is well defined. Specifically, for $\underline{\boldsymbol{\sigma}} \in {}_1L^2(\Omega, \mathbb{M}^2)$, $\underline{\boldsymbol{\sigma}}|_{\partial T}$ may not be well defined. The additional regularity required for $\mathbf{\Pi}_h$ to be well defined is reflected in the bound on the projection given by (2.159).

2. Concerning (2.160). As previously commented in the Proof of Theorem 1, $(\underline{\boldsymbol{\tau}}^2, \tau^2)$ defined in (2.134) may not be sufficiently smooth to guarantee that

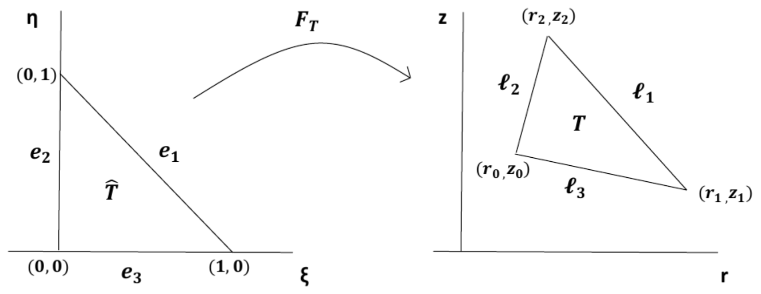


Figure 2-3: Reference Triangle

$\mathbf{\Pi}_h(\underline{\boldsymbol{\tau}}^2, \tau^2)$ is well defined. This lack of regularity is overcome by constructing $(\underline{\boldsymbol{\tau}}_h^2, \tau_h^2)$ using $\mathbf{w}_h$, a piecewise polynomial function.

3. Throughout the remainder of this document, we will denote the space $\boldsymbol{S} + (\nabla_{\text{ac}}((\Theta_h)^2)\times\Theta_h)$ as $\mathbf{\Sigma}^S$. Moreover, we denote the tensor and scalar components of $\mathbf{\Sigma}^S$ as $\Sigma^S_{h,\underline{\boldsymbol{\sigma}}}$ and $\Sigma^S_{h,\sigma}$. That is, $\mathbf{\Sigma}^S = \Sigma^S_{h,\underline{\boldsymbol{\sigma}}} \times \Sigma^S_{h,\sigma}$.

2.9 Mappings and $\mathcal{T}_h$

In Section 2.8 and Theorem 1, we introduced sufficient conditions to establish that a finite element space is inf-sup stable. Over the next several Sections, we will introduce a number of specific spaces that, subject to the specified projection being bounded, satisfy these conditions.

Before proceeding, we describe the different types of triangles T that can appear in $\mathcal{T}_h$. In addition, we present some useful properties for mapping functions between the physical domain T and the reference triangle $\widehat{T}$. Finally, we present the general form of several common integrals that appear in the discrete setting.

To start, the reference triangle $\widehat{T}$ is defined as the triangle with vertices $(0,0)$, $(1,0)$ and $(0,1)$. Moreover, every triangle $T \in \mathcal{T}_h$ has three coordinates (r_0, z_0), (r_1, z_1) and (r_2, z_2). We assume that the coordinates are always labeled in a counter-clockwise manner such that $r_0 \le r_1, r_2$. Further, an affine mapping F_T from the reference triangle $\widehat{T}$ (see Figure (2-3)) to the physical domain T exists and takes the form

$$\begin{pmatrix} r \\ z \end{pmatrix} = \begin{pmatrix} r_1 - r_0 & r_2 - r_0 \\ z_1 - z_0 & z_2 - z_0 \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} r_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} r_{10} & r_{20} \\ z_{10} & z_{20} \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} r_0 \\ z_0 \end{pmatrix}. \tag{2.173}$$

Observe that we have used the notational short hand $r_i - r_j = r_{ij}$, and $z_i - z_j = z_{ij}$. Associated with each affine mapping F_T is the determinant of the Jacobian matrix $|J_T| = |r_{10}z_{20} - z_{10}r_{20}|$.

Provided that the triangulation $\mathcal{T}_h$ is regular, every affine map F_T can be expressed as

$$\begin{pmatrix} r \\ z \end{pmatrix} = \begin{pmatrix} r_{10} & r_{20} \\ z_{10} & z_{20} \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} r_0 \\ z_0 \end{pmatrix} = h \begin{pmatrix} c_{10} & c_{20} \\ d_{10} & d_{20} \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} r_0 \\ z_0 \end{pmatrix}, \tag{2.174}$$

where $a_{min} \leq c_{10}, c_{20}, d_{10}, d_{20} \leq a_{max}$. Furthermore, the determinant of the Jacobian is $|J_T| = h^2(c_{10}d_{20} - d_{10}c_{20}) = h^2 J_D$ where $0 < jd_{min} \leq J_D \leq jd_{max}$.

For every regular triangulation $\mathcal{T}_h$ of an axisymmetric domain Ω with symmetry axis Γ_0, each triangle $T \in \mathcal{T}_h$ can be categorized as one of three types:

- Type I: $\partial T \cap \Gamma_0 = \mathbf{e}^*$ where $\mathbf{e}^*$ denotes an entire edge,

- Type II: $\partial T \cap \Gamma_0 = P_0$ where P_0 is a single point,

- Type III: $\partial T \cap \Gamma_0 = \emptyset$.

For each type of triangle, we can be more specific about the form of the affine mapping F_T. In the following, $\hat{r}$ and $\hat{z}$ represent the mapping of the variables r and z on the physical element T to the reference triangle $\widehat{T}$ as functions of ξ and η.

If T is Type I, then

$$\hat{r} = h\, c_{10}\, \xi \tag{2.175}$$

$$\hat{z} = (z_0 + h\, d_{10}\xi + h\, d_{20}\eta) \tag{2.176}$$

and

$$J_T = h \begin{pmatrix} c_{10} & 0 \\ d_{10} & d_{20} \end{pmatrix} = h\tilde{J}_T. \tag{2.177}$$

Since $c_{20} = 0$, it must be the case that $c_{10} > 0$ to ensure that T is well defined.
If T is Type II, then

$$\hat{r} = h\ (c_{10}\xi + c_{20}\eta) \tag{2.178}$$

$$\hat{z} = (z_0 + h\ d_{10}\xi + h\ d_{20}\eta) \tag{2.179}$$

and

$$J_T = h \begin{pmatrix} c_{10} & c_{20} \\ d_{10} & d_{20} \end{pmatrix} = h\tilde{J}_T. \tag{2.180}$$

In addition, since only one node lies on the symmetry axis, $c_{10}, c_{20} > 0$.
Finally, if T is Type III, then

$$\hat{r} = (r_0 + h\ c_{10}\xi + h\ c_{20}\eta) \tag{2.181}$$

$$\hat{z} = (z_0 + h\ d_{10}\xi + h\ d_{20}\eta) \tag{2.182}$$

and

$$J_T = h \begin{pmatrix} c_{10} & c_{20} \\ d_{10} & d_{20} \end{pmatrix} = h\tilde{J}_T \tag{2.183}$$

where $c_{10}, c_{20} \geq 0$ and $c_{10} + c_{20} > 0$.
In many cases, is it more convenient to work on the reference triangle $\widehat{T}$ than the physical domain T. However, it is important to recall that when mapping vector functions in ${}_1\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \mathbb{M}^2)$ between T and $\widehat{T}$, it is necessary to preserve normal components. Therefore, rather than using a standard affine mapping, we must use the contravariant Piola transformation [44, 17]. Let J_T be the Jacobian matrix associ-

ated with the affine mapping $F_T : \widehat{T} \to T$, then the Piola mapping of the function $\widehat{\mathbf{q}}$ (defined on the reference triangle) is

$$\mathcal{P}(\widehat{\mathbf{q}})(\mathbf{x}) := \frac{1}{|J_T|} J_T \, \widehat{\mathbf{q}}(\widehat{\mathbf{x}}), \text{ where } \mathbf{x} = F(\widehat{\mathbf{x}}). \tag{2.184}$$

The following Lemma describes some useful properties of the Piola map as it relates to the integration of ${}_1\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \mathbb{M}^2)$ functions.

Lemma 8. *Let* $\widehat{\underline{\boldsymbol{\tau}}}, \widehat{\underline{\boldsymbol{\sigma}}} \in {}_1\underline{\mathbf{H}}(\nabla_{axi}\cdot, \widehat{T}; \mathbb{M}^2)$ *and* $\widehat{\mathbf{v}} \in {}_1\mathbf{L}^2(\widehat{T})$, *and let* $\underline{\boldsymbol{\tau}} = \mathcal{P}(\widehat{\underline{\boldsymbol{\tau}}})$, $\underline{\boldsymbol{\sigma}} = \mathcal{P}(\widehat{\underline{\boldsymbol{\sigma}}})$, *and* $\mathbf{v} = \widehat{\mathbf{v}} \circ F^{-1}$

$$\int_T \underline{\boldsymbol{\tau}} : \underline{\boldsymbol{\sigma}} \, r \, dT = \int_{\widehat{T}} J_T \, \widehat{\underline{\boldsymbol{\tau}}}^t : J_T \, \widehat{\underline{\boldsymbol{\sigma}}}^t \, \frac{1}{|J_T|} \, \hat{r} \, d\widehat{T} \tag{2.185}$$

$$\int_{\partial T} (\underline{\boldsymbol{\tau}} \cdot \mathbf{n}) \cdot \mathbf{v} \, r \, ds = \int_{\partial \widehat{T}} (\widehat{\underline{\boldsymbol{\tau}}} \cdot \mathbf{n}) \cdot \widehat{\mathbf{v}} \, \hat{r} \, ds \tag{2.186}$$

Additional details and proofs can be found in [44, 24].

As a result of using polynomials as the discrete finite element approximation spaces, many of the integrals that appear in the finite element formulation have a similar structure. The next Lemma introduces an analytical solution for a common class of integrals that appear in the discrete finite element formulation of the axisymetric linear elasticity problem.

To begin, for convenience of notation, if $r_0 > 0$, let

$$\frac{\hat{r}}{r_0} = \frac{(r_1 - r_0)}{r_0}\xi + \frac{(r_2 - r_0)}{r_0}\eta + 1 = r_1^*\xi + r_2^*\eta + 1 \tag{2.187}$$

while if $r_0 = 0$, then

$$\hat{r} = r_1\xi + r_2\eta. \tag{2.188}$$

Since we assume that the coordinates of T are labeled such that $r_0 \le r_1, r_2$, it follows that $r_1^*, r_2^* \ge 0$. Thus, if we are calculating the integral of a function $f(r, z)$ on T

using the reference element $\widehat{T}$,

$$\int_T f(r,z)\, r\, dT = \begin{cases} r_0 \int_{\widehat{T}} \hat{f}(\xi,\eta)\ (r_1^*\xi + r_2^*\eta + 1)\ d\widehat{T} = r_0\ I(\hat{f}(\xi,\eta)) \text{ if } r_0 > 0 \\ \int_{\widehat{T}} \hat{f}(\xi,\eta)\ (r_1\xi + r_2\eta)\ d\widehat{T} = I(\hat{f}(\xi,\eta)) \text{ if } r_0 = 0. \end{cases} \tag{2.189}$$

where $d\widehat{T} = |J_T|\ d\xi\ d\eta$.

Lemma 9. *For integers $s \geq 0$ and $t \geq 0$,*

$$\begin{aligned} \int_0^1 \int_0^{1-\xi} \xi^s \eta^t (r_1\xi + r_2\eta + 1)\ d\eta\ d\xi &= \frac{s!\ t!}{(s+t+2)!}\left[\frac{r_1(s+1)}{(s+t+3)} + \frac{r_2(t+1)}{(s+t+3)} + 1\right] \\ &= \frac{s!\ t!}{(s+t+3)!}\left[r_1(s+1) + r_2(t+1) + (s+t+3)\right] \end{aligned} \tag{2.190}$$

and

$$\begin{aligned} \int_0^1 \int_0^{1-\xi} \xi^s \eta^t (r_1\xi + r_2\eta)\ d\eta\ d\xi &= \frac{s!\ t!}{(s+t+2)!}\left[\frac{r_1(s+1)}{(s+t+3)} + \frac{r_2(t+1)}{(s+t+3)}\right] \\ &= \frac{s!\ t!}{(s+t+3)!}\left[r_1(s+1) + r_2(t+1)\right]. \end{aligned} \tag{2.191}$$

Proof. First, for $\Gamma(\cdot)$ denoting the gamma function, note that

$$\int_0^1 \int_0^{1-\xi} \xi^s \eta^t\ d\eta\ d\xi = \int_0^1 \frac{\xi^s (1-\xi)^{t+1}}{t+1}\ d\xi = \frac{1}{t+1}\frac{\Gamma(s+1)\Gamma(t+2)}{\Gamma(s+t+3)} = \frac{s!\ t!}{(s+t+2)!}.$$

Therefore

$$\begin{aligned} &\int_0^1 \int_0^{1-\xi} \xi^s \eta^t (r_1\xi + r_2\eta + 1)\ d\eta\ d\xi \\ &= r_1 \int_0^1 \int_0^{1-\xi} \xi^{s+1}\eta^t\ d\eta\ d\xi + r_2 \int_0^1 \int_0^{1-\xi} \xi^s \eta^{t+1}\ d\eta\ d\xi + \int_0^1 \int_0^{1-\xi} \xi^s \eta^t\ d\eta\ d\xi \\ &= r_1 \frac{(s+1)!\ t!}{(s+t+3)!} + r_2 \frac{s!\ (t+1)!}{(s+t+3)!} + \frac{s!\ t!}{(s+t+2)!} \\ &= \frac{s!\ t!}{(s+t+2)!}\left(r_1 \frac{(s+1)}{(s+t+3)} + r_2 \frac{(t+1)}{(s+t+3)} + 1\right). \end{aligned}$$

which verifies (2.190). Removing the +1 from $(r_1\xi + r_2\eta + 1)$ yields (2.191). $\square$

Some useful integrals computed using Lemma 9 for $r_0 > 0$ are given below

$$\begin{aligned}
&\int_{\widehat{T}} \eta\ \hat{r}\ d\widehat{T} = \frac{1}{4!}\left[r_1^* + 2r_2^* + 4\right] && \int_{\widehat{T}} \xi\ \hat{r}\ d\widehat{T} = \frac{1}{4!}\left[2r_1^* + r_2^* + 4\right] \\
&\int_{\widehat{T}} \eta^2\ \hat{r}\ d\widehat{T} = \frac{2}{5!}\left[r_1^* + 3r_2^* + 5\right] && \int_{\widehat{T}} \xi\eta\ \hat{r}\ d\widehat{T} = \frac{1}{5!}\left[2r_1^* + 2r_2 + 5\right] \\
&\int_{\widehat{T}} \xi^2\ \hat{r}\ d\widehat{T} = \frac{2}{5!}\left[3r_1^* + r_2^* + 5\right] && \int_{\widehat{T}} \eta^3\ \hat{r}\ d\widehat{T} = \frac{3!}{6!}\left[r_1^* + 4r_2^* + 6\right] \\
&\int_{\widehat{T}} \xi\eta^2\ \hat{r}\ d\widehat{T} = \frac{2}{6!}\left[2r_1^* + 3r_2^* + 6\right] && \int_{\widehat{T}} \xi^2\eta\ \hat{r}\ d\widehat{T} = \frac{2}{6!}\left[3r_1^* + 2r_2^* + 6\right] \\
&\int_{\widehat{T}} \xi^3\ \hat{r}\ d\widehat{T} = \frac{3!}{6!}\left[4r_1 + r_2 + 6\right].
\end{aligned} \tag{2.192}$$

2.10 BDM_1 and BDM_2

The first finite element approximation space we consider combines two $\mathbf{BDM}_1$ polynomials to approximate $\Sigma^S_{h,\underline{\boldsymbol{\sigma}}}$ and P_1 to approximate $\Sigma^S_{h,\sigma}$.

Lemma 10. *Let $T \in \mathcal{T}_h$. The projection operators $\Pi_h : \Sigma^S_{h,\underline{\boldsymbol{\sigma}}} \to (\mathbf{BDM}_1(T))^2$ and $\pi_h : \Sigma^S_{h,\sigma} \to P_1(T)$ given by*

$$\int_\ell (\underline{\boldsymbol{\tau}} - \Pi_h\underline{\boldsymbol{\tau}}) \cdot \mathbf{n}_k \cdot \mathbf{p}_1\ r\ ds = 0 \text{ for all edges } \ell \in \partial T \text{ and } \mathbf{p}_1 \in (P_1(\ell))^2 \tag{2.193}$$

$$\int_T (\tau - \Pi_h\tau)\ p_1\ z\ dT = 0 \text{ for all } p_1 \in P_1(T) \tag{2.194}$$

are well defined and satisfy (2.161)-(2.163) *for*

$$\Sigma_{h,\underline{\boldsymbol{\sigma}}} = (\boldsymbol{BDM}_1)^2 \qquad \Sigma_{h,\sigma} = P_1 \qquad U_h = (P_0)^2 \qquad Q_h = P_0. \tag{2.195}$$

Proof. First we show that π_h is well defined. Since $\pi_h\tau \in P_1(T)$, the projection has 3 degrees of freedom, which matches the dimension of the trial space $P_1(T)$. Moreover, (2.194) is a well defined weighted L^2 projection because a relabeling of the coordinate system allows us to assume $z > 0$ with out loss of generality.

Next we show that Π_h is well defined. First note that $\Sigma_{h,\underline{\boldsymbol{\sigma}}} = (\mathbf{BDM}_1)^2$ has 12 degrees of freedom and $(P_1(\ell))^2$ has 4 degrees of freedom per edge. Therefore $\dim(\Sigma_{h,\underline{\boldsymbol{\sigma}}}) =$

$12 = 3 * \dim(P_1(\ell))^2$, which means that the number of unknowns in $\Pi_h\underline{\boldsymbol{\tau}}$ is equal to the number of constraints in (2.193). It follows that if $\underline{\boldsymbol{\tau}} = \mathbf{0}$ implies that $\Pi_h\underline{\boldsymbol{\tau}} = \mathbf{0}$, then the projection Π_h is well defined.

For ease of exposition, we first consider a single row of the tensor projection (2.193). In this case, for $\underline{\boldsymbol{\tau}} = (\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)^t$ the projection (2.193) takes the form

$$\int_\ell (\underline{\boldsymbol{\tau}}_s - \Pi_h\underline{\boldsymbol{\tau}}_s) \cdot \mathbf{n}_k \; p_1 \; r \; ds = 0 \text{ for } s = 1, 2. \tag{2.196}$$

Next, observe that the function $\Pi_h\underline{\boldsymbol{\tau}}_s \cdot \mathbf{n}_k \; p_1 \; r$ is a cubic polynomial. Recalling that a degree n Gauss quadrature rule integrates polynomials of degree $2n-1$ exactly, we select two Gauss quadrature points $\{q_i^{\ell_k}\}_{i=1}^2$ on each edge ℓ_k for $k = 1, 2, 3$.

For $\ell_k \in \partial K$, define a basis for $P_1(\ell_k)$ so that

$$p_1^{\ell_k}(x) = \begin{cases} 1 \text{ if } x = q_1^{\ell_k} \\ 0 \text{ if } x = q_2^{\ell_k} \end{cases} \quad \text{and} \quad p_2^{\ell_k}(x) = \begin{cases} 0 \text{ if } x = q_1^{\ell_k} \\ 1 \text{ if } x = q_2^{\ell_k} \end{cases}. \tag{2.197}$$

Next, let $\{\boldsymbol{\phi}_i^{\ell_k}\}$ be a basis for $\mathbf{BDM}_1$ [44] such that

$$(\boldsymbol{\phi}_i^{\ell_m} \cdot \mathbf{n})(q_j^{\ell_n}) = \delta_{(i,j),(\ell_m,\ell_n)} \text{ for } i, j = 1, 2 \text{ and } m, n = 1, 2, 3.$$

That is, the normal component of the basis functions satisfy a Lagrangian property at the boundary quadrature points. Since $\Pi_h\underline{\boldsymbol{\tau}}_s \in \mathbf{BDM}_1$, it can be written

$$\Pi_h\underline{\boldsymbol{\tau}}_s = \sum_{k=1}^{3}\sum_{i=1}^{2} \alpha_i^{\ell_k}\boldsymbol{\phi}_i^{\ell_k}.$$

Next, suppose that $\underline{\boldsymbol{\tau}}_s = \mathbf{0}$, then taking the basis function $p_1^{\ell_k}$ for $\ell_k \in \partial K$ and using

(2.193) and Gaussian quadrature gives

$$
\begin{aligned}
0 = \int_{\ell_k} \Pi_h \underline{\boldsymbol{\tau}}_s \cdot \mathbf{n}\, p_1\, r\, ds &= \sum_{j=1}^{2} (\Pi_h \underline{\boldsymbol{\tau}}_s \cdot \mathbf{n})(q_j^{\ell_k}) \cdot p_1^{\ell_k}(q_j^{\ell_k})\, r(q_j^{\ell_k})\, w(q_j^{\ell_k}) \\
&= \alpha_1^{\ell_k}\, p_1^{\ell_k}(q_1^{\ell_k})\, r(q_1^{\ell_k}) w(q_1^{\ell_k}) + \alpha_2^{\ell_k}\, p_1^{\ell_k}(q_2^{\ell_k})\, r(q_2^{\ell_k}) w(q_2^{\ell_k}) \\
&= \alpha_1^{\ell_k} r(q_1^{\ell_k}) w(q_1^{\ell_k}).
\end{aligned}
$$

In the case where $r(q_1^{\ell_k}) \neq 0$, this implies $\alpha_1^{\ell_k} = 0$. If, however, $r(q_1^{\ell_k}) = 0$, then $\alpha_1^{\ell_k}$ and $\beta_1^{\ell_k}$ must be zero. Otherwise, the normal stress along the axis of symmetry will be non-zero implying that the solution is not axisymmetric. A similar argument can be used to show the rest of the α terms are zero as well. This then illustrates that the vector projections from (2.196) are well defined.

Finally, to show that (2.193) is well defined, we can extend the basis for $P_1(\ell_k)$ from (2.197) to $(P_1(\ell_k))^2$ by taking

$$
\left\{ \begin{pmatrix} p_1^{\ell_k} \\ 0 \end{pmatrix}, \begin{pmatrix} p_2^{\ell_k} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ p_1^{\ell_k} \end{pmatrix}, \begin{pmatrix} 0 \\ p_2^{\ell_k} \end{pmatrix} \right\}. \tag{2.198}
$$

Using this basis, the arguments presented above for the vector case can be applied to each row of (2.193) to show that Π_h is well defined.

Lastly, we verify that the spaces given in (2.195) satisfy the conditions outlined in (2.161)-(2.163). Since gradients of the piecewise constant spaces U_h and Q_h are zero on each element T, (2.161) is satisfied trivially. Next, observe that the test space of (2.193) includes all $\mathbf{p}_1 \in (P_1(\ell_k))^2$ for $k = 1, 2, 3$, while (2.162) only requires that the projection is satisfied on a subspace of $(P_1(\ell_k))^2$. Finally, since $P_0(T) \subset P_1(T)$, (2.194) ensures that (2.163) is satisfied. □

Next, we consider approximating the tensor $\Sigma^{\boldsymbol{S}}_{h,\underline{\boldsymbol{\sigma}}}$ with $(\mathbf{BDM}_2)^2$ and $\Sigma^{\boldsymbol{S}}_{h,\sigma}$ with P_2.

Lemma 11. *Let $T \in \mathcal{T}_h$. The projection operators $\Pi_h : \Sigma^{\boldsymbol{S}}_{h,\underline{\boldsymbol{\sigma}}} \to (\mathbf{BDM}_2(T))^2$ and*

$\pi_h : \Sigma^{\boldsymbol{S}}_{h,\sigma} \to P_2(T)$ given by

$$\int_T (\underline{\boldsymbol{\tau}} - \Pi_h \underline{\boldsymbol{\tau}}) : (\underline{\mathbf{p_0}} + \mathbf{x}^\perp \otimes \mathbf{p_0})\, r\, dT = 0 \quad \forall\ \underline{\mathbf{p_0}} \in (P_0(T))^{2\times 2} \quad \forall\ \mathbf{p_0} \in (P_0(T))^2 \tag{2.199}$$

$$\int_\ell (\underline{\boldsymbol{\tau}} - \Pi_h \underline{\boldsymbol{\tau}}) \cdot \mathbf{n}_k \cdot \mathbf{p}_2\, r\, ds = 0 \quad \forall \textit{ edges } \ell \quad \forall\ \mathbf{p}_2 \in (P_2(\ell))^2 \tag{2.200}$$

$$\int_T (\tau - \pi_h \tau)\, p_2\, z\, dT = 0 \textit{ for all } p_2 \in P_2(T) \tag{2.201}$$

are well defined and satisfy (2.161)-(2.163) *for*

$$\Sigma_{h,\underline{\boldsymbol{\sigma}}} = (\boldsymbol{BDM}_2)^2 \qquad \Sigma_{h,\sigma} = P_2 \qquad U_h = (P_1)^2 \qquad Q_h = P_1. \tag{2.202}$$

Proof. First we show that π_h is well defined. Since $\pi_h \tau \in P_2(T)$, the projection has 6 degrees of freedom, which matches the dimension of the trial space $P_2(T)$. Moreover, (2.201) is a well defined weighted L^2 projection because a relabeling of the coordinate system allows us, with out loss of generality, to assume $z > 0$.

Next we verify that Π_h is well defined. First observe that the number of constraints defined by Π_h is the same as number of degrees of freedom in $(\mathbf{BDM}_2(T))^2$. The space $(\mathbf{BDM}_2(T))^2$ has dimension 24. Moreover, $(P_2(\ell))^2$ has 6 degrees of freedom per edge for a total of 18 boundary degrees of freedom, while $(P_0(T))^{2\times 2}$ and $(P_0(T))^2$ have four and two degrees of freedom, respectively. Therefore, $\dim((P_2(\ell))^2) + \dim((P_0(T))^{2\times 2}) + \dim((P_0(T))^2 = 18 + 4 + 2 = 24 = \dim(\mathbf{BDM}_2(T))^2$.

Second we verify that the projection is injective. That is, if

$$\int_T \Pi_h \underline{\boldsymbol{\tau}} : (\underline{\mathbf{p_0}} + \mathbf{x}^\perp \otimes \mathbf{p_0})\, r\, dT = 0 \quad \forall\ \underline{\mathbf{p_0}} \in (P_0(T))^{2\times 2} \quad \forall\ \mathbf{p_0} \in (P_0(T))^2 \tag{2.203}$$

$$\int_\ell \Pi_h \underline{\boldsymbol{\tau}} \cdot \mathbf{n}_k \cdot \mathbf{p}_2\, r\, ds = 0 \quad \forall \text{ edges } \ell \quad \forall\ \mathbf{p}_2 \in (P_2(\ell))^2 \tag{2.204}$$

then $\Pi_h \underline{\boldsymbol{\tau}} = \mathbf{0}$. In other words, the kernel of Π_h is $\{\mathbf{0}\}$.

We can represent $\Pi_h \underline{\boldsymbol{\tau}}$ in terms of the basis for $(\mathbf{BDM}_2(\widehat{T}))^2$, where $\mathbf{BDM}_2(\widehat{T})$ is the reference element representation presented in [44, Section 4.2]. This $\mathbf{BDM}_2(\widehat{T})$ basis is expressed in terms of edge and interior element functions. Using equation (2.204) with three Gauss quadrature points[1] and an argument analogous to that used in the proof of Lemma 10, it follows that all 18 of the $\mathbf{BDM}_2(\widehat{T})$ edge basis functions must equal zero.

Therefore, the only possible non-zero basis functions on $\widehat{T}$ are the interior element functions

$$\underline{\phi}_1 = \frac{\sqrt{2}}{(g_2 - g_1)}(1-\xi-\eta)\begin{pmatrix} g_2\xi \\ (g_2-1)\eta \end{pmatrix} \quad \underline{\phi}_2 = \frac{1}{(g_2-g_1)}\xi\begin{pmatrix} g_2\xi+\eta-g_2 \\ (g_2-1)\eta \end{pmatrix}$$
$$\underline{\phi}_3 = \frac{1}{(g_2-g_1)}\eta\begin{pmatrix} (g_2-1)\xi \\ \xi+g_2\eta-g_2 \end{pmatrix} \tag{2.205}$$

where $g_1 = 1/2 - \sqrt{3}/6$ and $g_2 = 1/2 + \sqrt{3}/6$ are the Gaussian quadrature points on $[0,1]$. Thus, $\widehat{\Pi_h\underline{\boldsymbol{\tau}}}$, the representation of $\Pi_h\underline{\boldsymbol{\tau}}$ on $\widehat{T}$, must have the form $\widehat{\Pi_h\underline{\boldsymbol{\tau}}} = \begin{pmatrix} \underline{\phi_\alpha}^t \\ \underline{\phi_\beta}^t \end{pmatrix}$ where

$$\underline{\phi_\alpha}^t = \alpha_1\underline{\phi_1}^t + \alpha_2\underline{\phi_2}^t + \alpha_3\underline{\phi_3}^t \quad \text{and} \quad \underline{\phi_\beta}^t = \beta_1\underline{\phi_1}^t + \beta_2\underline{\phi_2}^t + \beta_3\underline{\phi_3}^t. \tag{2.206}$$

It remains to show that $\alpha_i = \beta_i = 0$ for $i = 1,2,3$. To do so, we consider the matrix representation of equation (2.199). Indeed, the functions (2.205) can be used to construct the six trial functions of (2.199), while the test space of (2.199) has dimension 6, and is spanned by the functions

$$\underline{\psi}_i = \begin{pmatrix} \delta_{i1} & \delta_{i2} \\ \delta_{i3} & \delta_{i4} \end{pmatrix} + \begin{pmatrix} \eta\,\delta_{i5} & \eta\,\delta_{i6} \\ -\xi\delta_{i5} & -\xi\delta_{i6} \end{pmatrix} \text{ for } i = 1,\cdots,6 \text{ and } \delta_{ij} \in \mathbb{R} \text{ for } i,j = 1,2,\cdots 6. \tag{2.207}$$

Taking ψ_i as the test function for row i, the resulting matrix representation of equation (2.199) is presented in (2.211) where $I(\cdot)$ is defined in (2.189).

To illustrate how the elements of (2.211) are calculated, we consider the first row of

[1]Recall the quadrature rule using three Gauss quadrature points is exact for polynomials of degree less than or equal to 5

(2.211). Recalling (2.189), Lemma 9 and (2.192), the entries of the first row are

$$\begin{aligned}
I(g_2(1-\xi-\eta)\xi) &= \int_{\widehat{T}} g_2(1-\xi-\eta)\ \xi\ (r_1^*\xi + r_2^*\eta + 1)\ d\widehat{T} \\
&= g_2 \left[\int_{\widehat{T}} (\xi - \xi^2 - \eta\xi)\ (r_1^*\xi + r_2^*\eta + 1)\ d\widehat{T}\right] \\
&= g_2 \left([\frac{1}{4!}(2r_1^* + r_2^* + 4) - \frac{2}{5!}(3r_1^* + r_2^* + 5) - \frac{1}{5!}(2r_1^* + 2r_2^* + 5)\right) \\
&= g_2 \left[\frac{1}{5!}(2r_1^* + r_2^* + 5)\right] \\
I(\xi(g_2\xi + \eta - g_2)) &= \int_{\widehat{T}} (g_2\xi^2 + \xi\eta - g_2\xi)(r_1^*\xi + r_2^*\eta + 1)\ d\widehat{T} \\
&= \left(\frac{2g_2}{5!}(3r_1^* + r_2^* + 5) + \frac{1}{5!}(2r_1^* + 2r_2^* + 5) - \frac{g_2}{4!}(2r_1^* + r_2^* + 4)\right) \\
&= 2(1-2g_2)r_1^* + (2-3g_2)r_2^* + 5(1-2g_2) \\
I((g_2-1)\eta\xi) &= \int_{\widehat{T}} (g_2-1)\eta\xi(r_1^*\xi + r_2^*\eta + 1)\ d\widehat{T} \\
&= (g_2-1)\left[\frac{1}{5!}(2r_1^* + 2r_2^* + 5)\right]
\end{aligned}$$

with the remaining columns equaling zero. A similar procedure can be used to find the remaining terms in the system. The complete entires of the matrix expressed in terms of the coordinates of the triangle T are shown in (2.212) which we denote M_T. Taking the determinate of (2.212) yields

$$\begin{aligned}
|M_T| = \frac{1}{36}&(r_1^* + r_2^* + 3)(2r_1^* + r_2^* + 5)(r_1^* + 2r_2^* + 5) \\
&(2r_1^* + 2r_2^* + 5)((r_1^*)^2 + 4r_1^*r_2^* + (r_2^*)^2 + 10r_1^* + 10r_2^* + 15).
\end{aligned}$$

Since $r_1^*, r_2^* \geq 0$, it follows that $|M_T| > 0$ implying that the matrix representation of the projection operator is full rank. Therefore, the null space of Π_h is $\{\mathbf{0}\}$ and Π_h is well defined.

Finally, we verify that the spaces given in (2.202) satisfy the conditions outlined in (2.161)-(2.163). Observe that for $U_h = (P_1)^2$ and $Q_h = P_1$ the test space of (2.161) is the set

$$\left\{\begin{pmatrix}\delta_1 + z\delta_5 & \delta_2 + z\delta_6 \\ \delta_3 - r\delta_5 & \delta_4 - r\delta_6\end{pmatrix} \mid \ \forall\, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6 \in \mathbb{R}\right\}, \tag{2.208}$$

which is the same as the test space described in (2.199). Furthermore, Theorem 1 requires that (2.162) is satisfied on a subset of

$$\left\{\begin{pmatrix}s_1 + s_2 s + s_3 s^2 \\ s_4 + s_5 s + s_6 s^2\end{pmatrix} \mid \ \forall\, s_1, s_2, s_3, s_4, s_5, s_6 \in \mathbb{R}\right\} \tag{2.209}$$

for all ℓ. Since the boundary integral (2.200) is satisfied for all quadratic polynomials on all ℓ, this condition is also satisfied. Lastly, since $P_1(T) \subset P_2(T)$, (2.201) ensures that (2.163) is satisfied. $\square$

2.11 Axisymmetric Elasticity Projection for General k

In this section, we begin to lay the groundwork for establishing a projection $\boldsymbol{\Pi_h} = \Pi_h \times \pi_h : \boldsymbol{\Sigma}^S \to \boldsymbol{\Sigma}_h$ that satisfies the conditions of Theorem 1 for a general polynomial order $k \geq 3$. To this end, we establish a convenient functional representation for polynomial tensors with zero divergence and a vanishing normal component along element boundaries. While we leave the development of the projection $\boldsymbol{\Pi_h}$ to future work, these functional representations should help to inform the development of $\boldsymbol{\Pi_h}$.

Lemma 12. *Functions in the set* $S_0 := \{\bar{\phi}(r,z) = \begin{pmatrix}\phi_1(r,z) \\ \phi_2(r,z)\end{pmatrix} \in (P_k(r,z))^2 : \nabla_{axi} \cdot \bar{\phi} = 0\}$ *can be expressed as*

$$\phi_1 = \sum_{m=1}^{k} \sum_{j=0}^{m-1} a_{m,j}\, r^{m-j}\, z^j \quad , \quad \phi_2 = \sum_{m=0}^{k} b_{m,0} r^m + \sum_{m=1}^{k} \sum_{j=1}^{m} b_{m,j}\, r^{m-j} z^j \tag{2.210}$$

$$\begin{array}{llllllll}
m=0 & 1 \\
m=1 & r & z \\
m=2 & r^2 & rz & z^2 \\
m=3 & r^3 & r^2z & rz^2 & z^3 \\
m=4 & r^4 & r^3z & r^2z^2 & rz^3 & z^4 \\
m=k & r^k & r^{k-1}z & \cdots & \cdots & \cdots & rz^{k-1} & z^k
\end{array}$$

(a) $P_k(r,z)$ polynomial basis

$$\begin{array}{llllllll}
m=0 & 0 \\
m=1 & 1 & 0 \\
m=2 & 2r & z & 0 \\
m=3 & 3r^2 & 2rz & z^2 & 0 \\
m=4 & 4r^3 & 3r^2z & 2rz^2 & z^3 & 0 \\
m=k & k\,r^{k-1} & (k-1)\,r^{k-2}z & \cdots & \cdots & \cdots & z^{k-1} & 0
\end{array}$$

(b) $\dfrac{\partial}{\partial r}P_k(r,z)$ polynomial basis

$$\begin{array}{llllllll}
m=0 & 0 \\
m=1 & 0 & 1 \\
m=2 & 0 & r & 2z \\
m=3 & 0 & r^2 & 2rz & 3z^2 \\
m=4 & 0 & r^3 & 2r^2z & 3rz^2 & 4z^3 \\
m=k & 0 & r^{k-1} & \cdots & \cdots & \cdots & (k-1)\,rz^{k-2} & k\,z^{k-1}
\end{array}$$

(c) $\dfrac{\partial}{\partial z}P_k(r,z)$ polynomial basis

$$\begin{array}{llllllll}
m=0 & a_{0,0} \\
m=1 & a_{1,0} & a_{1,1} \\
m=2 & a_{2,0} & a_{2,1} & a_{2,2} \\
m=3 & a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\
m=4 & a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\
m=k & a_{k,0} & a_{k,1} & \cdots & \cdots & \cdots & a_{k,(k-1)} & a_{k,k}
\end{array}$$

(d) $P_k(r,z)$ polynomial coefficients

Figure 2-4: Bivariate Polynomial Map

$$\begin{pmatrix} I(g_2(1-\xi-\eta)\ \xi) & I(\xi(g_2\xi+\eta-g_2)) & I((g_2-1)\eta\xi) & 0 & 0 & 0 \\ I((g_2-1)(1-\xi-\eta)\ \eta) & I((g_2-1)\eta\xi) & I(\eta(\xi+g_2\eta-g_2) & 0 & 0 & 0 \\ 0 & 0 & 0 & I(g_2(1-\xi-\eta)\ \xi) & I(\xi(g_2\xi+\eta-g_2)) & I((g_2-1)\eta\xi) \\ 0 & 0 & 0 & I((g_2-1)(1-\xi-\eta)\ \eta) & I((g_2-1)\eta\xi) & I(\eta(\xi+g_2\eta-g_2)) \\ I((1-\xi-\eta)g_2\xi\eta) & I(\xi(g_2\xi+\eta-g_2)\eta) & I(\eta(g_2-1)\xi\eta) & I(-(1-\xi-\eta)g_2\xi^2) & I(-\xi(g_2\xi+\eta-g_2)\xi)) & I(-\eta(g_2-1)\xi^2) \\ I((1-\xi-\eta)(g_2-1)\eta^2) & I(\xi(g_2-1)\eta^2) & I(\eta(\xi+g_2\eta-g_2)\eta) & I(-(1-\xi-\eta)(g_2-1)\eta\xi) & I(-\xi(g_2-1)\eta\xi) & I(-\eta(\xi+g_2\eta-g_2)\xi) \end{pmatrix} \tag{2.211}$$

$$\begin{pmatrix} g_2(2r_1^*+r_2^*+5) & 2(1-2g_2)r_1^*+(2-3g_2)r_2^*+5(1-2g_2) & (g_2-1)(2r_1^*+2r_2^*+5) & 0 & 0 & 0 \\ (g_2-1)(r_1^*+2r_2^*+5) & (g_2-1)(2r_1^*+2r_2^*+5) & ((2-3g_2)r_1^*+2(1-2g_2)r_2^*+5(1-2g_2) & 0 & 0 & 0 \\ 0 & 0 & 0 & g_2(2r_1^*+r_2^*+5) & 2(1-2g_2)r_1^*+(2-3g_2)r_2^*+5(1-2g_2) & (g_2-1)(2r_1^*+2r_2^*+5) \\ 0 & 0 & 0 & (g_2-1)(r_1^*+2r_2^*+5) & (g_2-1)(2r_1^*+2r_2^*+5) & (2-3g_2)r_1^*+2(1-2g_2)r_2^*+5(1-2g_2) \\ g_2(r_1^*+r_2^*+3) & (2-3g_2)r_1^*+(3-4g_2)r_2^*+3(2-3g_2) & (g_2-1)(2r_1^*+3r_2^*+6) & -g_2(3r_1^*+r_2^*+6) & -[(3-6g_2)r_1^*+(2-3g_2)r_2^*+6(1-2g_2)] & -(g_2-1)(3r_1^*+2r_2^*+6) \\ (g_2-1)(r_1^*+3r_2^*+6) & (g_2-1)(2r_1^*+3r_2^*+6) & (2-3g_2)r_1^*+3(1-2g_2)r_2^*+6(1-2g_2) & -(g_2-1)(r_1^*+r_2^*+3) & -(g_2-1)(3r_1^*+2r_2^*+6) & -(3-4g_2)r_1^*+(2-3g_2)r_2^*+3(2-3g_2) \end{pmatrix} \tag{2.212}$$

where $a_{m,m} = 0$ *for* $0 \leq m \leq k$, *and* $b_{m,(j+1)} = \dfrac{-(1+m-j)}{(j+1)} a_{m,j}$ *for* $1 \leq m \leq k$ *and* $0 \leq j \leq m-1$.

Proof. For $\bar{\phi} \in (P_k(r,z))^2$, let

$$\bar{\phi}(r,z) = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} a_{0,0} + a_{1,0}r + a_{1,1}z + \cdots + a_{k,0}r^k + \cdots + a_{k,j}r^{k-j}z^j + \cdots + a_{k,k}z^k \\ b_{0,0} + b_{1,0}r + b_{1,1}z + \cdots + b_{k,0}r^k + \cdots + b_{k,j}r^{k-j}z^j + \cdots + b_{k,k}z^k \end{pmatrix}. \tag{2.213}$$

Alternatively, in summation notation,

$$\phi_1 = \sum_{m=0}^{k}\sum_{j=0}^{m} a_{m,j}\ r^{m-j}\ z^j \text{ and } \phi_2 = \sum_{m=0}^{k}\sum_{j=0}^{m}\ b_{m,j}\ r^{m-j}\ z^j. \tag{2.214}$$

Next we identify the restrictions on ϕ_1 and ϕ_2 needed to ensure that $\nabla_{\text{axi}} \cdot \bar{\phi} = 0$. From Figure 2-4, observe that

$$\begin{aligned} \frac{1}{r}\phi_1 &= \sum_{m=1}^{k}\sum_{j=0}^{m-1}\ a_{m,j}\ r^{(m-j-1)}\ z^j + \frac{1}{r}(a_{0,0} + a_{1,1}z + a_{2,2}z^2 + \cdots + a_{k,k}z^k), \\ \frac{\partial \phi_1}{\partial r} &= \sum_{m=1}^{k}\sum_{j=0}^{m-1}\ (m-j)\ a_{m,j}\ r^{m-j-1}\ z^j, \\ \frac{\partial \phi_2}{\partial z} &= \sum_{m=1}^{k}\sum_{j=1}^{m}\ j\ b_{m,j}\ r^{m-j}\ z^{j-1} = \sum_{m=1}^{k}\sum_{j=0}^{m-1}\ (j+1)\ b_{m,(j+1)}\ r^{m-j-1}\ z^j. \end{aligned} \tag{2.215}$$

Furthermore,

$$\begin{aligned}
\nabla_{\text{axi}} \cdot \bar{\phi} &= \frac{1}{r}\phi_1 + \frac{\partial \phi_1}{\partial r} + \frac{\partial \phi_2}{\partial z} \\
&= \frac{1}{r}(a_{0,0} + a_{1,1}z + a_{2,2}z^2 + \cdots + a_{k,k}z^k) + \sum_{m=1}^{k}\sum_{j=0}^{m-1} \ a_{m,j} \ r^{(m-j-1)} \ z^j \\
&+ \sum_{m=1}^{k}\sum_{j=0}^{m-1} (m-j) \ a_{m,j} \ r^{(m-j-1)} \ z^j + \sum_{m=1}^{k}\sum_{j=0}^{m-1} (j+1) \ b_{m,(j+1)} \ r^{(m-j-1)} \ z^j \\
&= \frac{1}{r}\sum_{m=0}^{k} a_{m,m} z^m + \sum_{m=1}^{k}\sum_{j=0}^{m-1} [(1+m-j)a_{m,j} + (j+1)b_{m,(j+1)}] r^{m-j-1} z^j .
\end{aligned} \tag{2.216}$$

The condition $\nabla_{\text{axi}} \cdot \bar{\phi} = 0$, together with the linear independence of $\{\frac{1}{r}, \frac{1}{r}z, \cdots, \frac{1}{r}z^k, r^{(m-j-1)}z^j\}$ for $1 \le m \le k$ and $0 \le j \le m-1$ implies that

$$a_{0,0} = a_{1,1} = a_{2,2} = \cdots = a_{k,k} = 0 \quad , \quad b_{m,(j+1)} = \frac{-(1+m-j)}{(j+1)} a_{m,j} \tag{2.217}$$

for $1 \le m \le k$ and $0 \le j \le m-1$. □

Corollary 1. *The set* $S_0 := \{\bar{\phi}(r,z) \in (P_k(r,z))^2 : \nabla_{axi} \cdot \bar{\phi} = 0\}$ *has dimension* $\dfrac{(k+2)(k+1)}{2}$.

Proof. Using the representation of S_0 from Lemma 12, ϕ_1 has $\dfrac{k^2+k}{2}$ degrees of freedom. Meanwhile, the only independent degrees of freedom from ϕ_2 correspond to the $b_{m,0}$ terms of which there are $k+1$. Thus, $\dim S_0 = \dfrac{k^2+k}{2} + (k+1) = \dfrac{(k+2)(k+1)}{2}$. □

Lemma 13. *The sets* $S_0 := \{\bar{\phi}(r,z) \in (P_k(r,z))^2 : \nabla_{axi} \cdot \bar{\phi} = 0\}$ *and* $S_1 := \{\nabla_{ac}\, r\, p(r,z) : p(r,z) \in P_k(r,z)\}$ *are equal.*

Proof. To prove this result, we show that $S_1 \subset S_0$ and then $S_0 \subset S_1$. To show that

$S_1 \subset S_0$, consider $\bar{\psi} \in S_1$. Then there exists $p \in P_k(r,z)$ such that

$$\bar{\psi} = \nabla_{ac}\ r\ p(r,z) = \begin{pmatrix} \dfrac{\partial\ r\ p(r,z)}{\partial z} \\ -\dfrac{1}{r}\dfrac{\partial\ r^2\ p(r,z)}{\partial r} \end{pmatrix} = \begin{pmatrix} r\ \dfrac{\partial\ p(r,z)}{\partial z} \\ -2\ p(r,z) - r\dfrac{\partial p(r,z)}{\partial r} \end{pmatrix} \tag{2.218}$$

so $\bar{\psi} \in (P_k(r,z))^2$. Furthermore,

$$\begin{aligned} \nabla_{\text{axi}} \cdot \bar{\psi} = \nabla_{\text{axi}} \cdot \nabla_{ac}(r\ p(r,z)) &= \begin{pmatrix} \dfrac{1}{r}\dfrac{\partial}{\partial r}(r\cdot) \\ \dfrac{\partial}{\partial z} \end{pmatrix} \cdot \begin{pmatrix} \dfrac{\partial}{\partial z} r\ p(r,z) \\ -\dfrac{1}{r}\dfrac{\partial(r^2\ p(r,z))}{\partial r} \end{pmatrix} \\ &= \frac{1}{r}\frac{\partial}{\partial r} r^2 \frac{\partial p(r,z)}{\partial z} - \frac{1}{r}\frac{\partial}{\partial z}\frac{\partial(r^2 p(r,z))}{\partial r} = \frac{1}{r}\frac{\partial}{\partial r} r^2 \frac{\partial p(r,z)}{\partial z} - \frac{1}{r}\frac{\partial}{\partial r} r^2 \frac{\partial p(r,z)}{\partial z} = 0. \end{aligned} \tag{2.219}$$

Note that we have used the fact that $r^2 p(r,z)$ is a polynomial to interchange the order of differentiation. Therefore, $\bar{\psi} \in S_0$.

To show that $S_0 \subset S_1$, consider $\bar{\phi} = (\phi_1 \quad \phi_2)^t \in S_0$. Recall from Lemma 12 that $\bar{\phi}$ has the form

$$\phi_1 = \sum_{m=1}^{k}\sum_{j=0}^{m-1} a_{m,j}\ r^{m-j}\ z^j \quad \text{and} \quad \phi_2 = \sum_{m=0}^{k} b_{m,0} r^m + \sum_{m=1}^{k}\sum_{j=1}^{m} b_{m,j}\ r^{m-j} z^j. \tag{2.220}$$

Thus, to show that $\bar{\phi} \in S_1$, we construct a polynomial $p(r,z) \in P_k(r,z)$ so that $\bar{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \nabla_{ac}\ r\ p(r,z) = \begin{pmatrix} \dfrac{\partial(r\ p(r,z))}{\partial z} \\ -\dfrac{1}{r}\dfrac{\partial(r^2 p(r,z))}{\partial r} \end{pmatrix}$. Expressing $p(r,z)$ in summation notation

$$p(r,z) = \sum_{m=0}^{k}\sum_{j=0}^{m}\ p_{m,j}\ r^{m-j}\ z^j. \tag{2.221}$$

Therefore,

$$r\frac{\partial p(r,z)}{\partial z} = r\sum_{m=1}^{k}\sum_{j=0}^{m-1} p_{m,(j+1)}\ (j+1)\ r^{m-j-1}\ z^{j} = \sum_{m=1}^{k}\sum_{j=0}^{m-1} p_{m,(j+1)}\ (j+1)\ r^{m-j}\ z^{j}. \tag{2.222}$$

Then, from (2.220), $\phi_1 = r\dfrac{\partial p(r,z)}{\partial z}$ implies

$$p_{m,(j+1)}(j+1) = a_{m,j} \text{ for } 1 \le m \le k \text{ and } 0 \le j \le m-1. \tag{2.223}$$

Using (2.223), $p(r,z)$ must have the form

$$\begin{aligned} p(r,z) &= \sum_{m=0}^{k} p_{m0}\ r^{m} + \sum_{m=1}^{k}\sum_{j=0}^{m-1} p_{m,(j+1)} r^{m-(j+1)} z^{j+1} \\ &= \sum_{m=0}^{k} p_{m0}\ r^{m} + \sum_{m=1}^{k}\sum_{j=0}^{m-1} \frac{a_{m,j}}{j+1} r^{m-(j+1)} z^{j+1}. \end{aligned} \tag{2.224}$$

Next, we set $\phi_2 = -2p(r,z) - r\dfrac{\partial p(r,z)}{\partial r}$. Starting with (2.224),

$$\frac{\partial\ p(r,z)}{\partial r} = \sum_{m=1}^{k} p_{m0}\ m\ r^{m-1} + \sum_{m=2}^{k}\sum_{j=0}^{m-1} a_{m,j}\frac{m-(j+1)}{j+1} r^{(m-1)-(j+1)} z^{j+1}. \tag{2.225}$$

Therefore,

$$\begin{aligned}
\phi_2 = -2\,p(r,z) - r\frac{\partial p}{\partial r} &= -2\sum_{m=0}^{k} p_{m0} r^m - 2\sum_{m=1}^{k}\sum_{j=0}^{m-1}\frac{a_{m,j}}{j+1} r^{m-(j+1)} z^{j+1} \\
&\quad - \sum_{m=1}^{k} p_{m0}\; m\; r^m - \sum_{m=2}^{k}\sum_{j=0}^{m-1} a_{m,j}\frac{m-(j+1)}{j+1} r^{m-(j+1)} z^{j+1} \\
&= -\sum_{m=0}^{k} p_{m0}(m+2) r^m \\
&\quad - \sum_{m=1}^{k}\sum_{j=0}^{m-1} a_{m,j}\left(\frac{1+m-j}{j+1}\right) r^{(m-1)-j} z^{j+1} \\
&= -\sum_{m=0}^{k} p_{m0}(m+2) r^m + \sum_{m=1}^{k}\sum_{j=1}^{m} b_{m,j} r^{m-j} z^j
\end{aligned} \tag{2.226}$$

where in the final step we have used (2.217). Thus, comparing (2.226) with (2.220), the choices $p_{m0} = \dfrac{-b_{m,0}}{(m+2)}$ and (2.223) gives a polynomial $p(r,z) \in P_k(r,z)$ such that for $\bar{\phi}(r,z) \in S_2$, $\bar{\phi}(r,z) = \nabla_{ac}\; r\; p(r,z)$. □

Lemma 14. *Let T denote a triangle with no edge lying on the line $r = 0$. For $k \geq 3$ let*

$$\begin{aligned}
&S_0 = \{\bar{\phi} \in (P_k(r,z))^2 : \nabla_{axi}\cdot\bar{\phi} = 0, \quad \bar{\phi}\cdot\mathbf{n} = 0 \text{ on } \partial T\}, \\
&\text{and } S_1 = \{\nabla_{ac}\; r\; b_T\; \psi_{k-3}\},
\end{aligned} \tag{2.227}$$

where b_T is the cubic bubble function on T and $\psi_{k-3} \in P_{k-3}(r,z)$. Then $S_0 = S_1$.

Proof. We present a standard inclusion argument to prove Lemma 14.

First, suppose $\bar{\phi} \in S_0$ and recall from Lemma 13 that

$$\{\bar{\phi} \in (P_k(r,z))^2 : \nabla_{\text{axi}}\cdot\bar{\phi} = 0\} = \{\nabla_{\text{ac}}\; r\; p(r,z) : p(r,z) \in P_k(r,z)\}. \tag{2.228}$$

Since $\bar{\phi}\cdot\mathbf{n} = 0$, then there exists $\tilde{p}(r,z) \in P_k(r,z)$, such that $(\nabla_{\text{ac}}\; r\; \tilde{p}(r,z))\cdot\mathbf{n} = 0$

on ∂T. Noting that $r \neq 0$, we have that $\tilde{p}(r,z)$ must satisfy

$$\begin{aligned} &\begin{pmatrix} \dfrac{\partial}{\partial z} r\,\tilde{p} \\ \dfrac{-1}{r}\dfrac{\partial}{\partial r}(r^2\tilde{p}) \end{pmatrix} \cdot \mathbf{n} = 0 \text{ on } \partial T \quad \Rightarrow \frac{1}{r}\begin{pmatrix} \dfrac{\partial}{\partial z}(r^2\tilde{p}) \\ \dfrac{\partial}{\partial r}(-r^2\tilde{p}) \end{pmatrix} \cdot \mathbf{n} = 0 \text{ on } \partial T \\ \Rightarrow &\frac{1}{r}\begin{pmatrix} \dfrac{\partial}{\partial r}(r^2\tilde{p}) \\ \dfrac{\partial}{\partial z}(r^2\tilde{p}) \end{pmatrix} \cdot \mathbf{t} = 0 \text{ on } \partial T \qquad \Rightarrow \frac{1}{r}\frac{d}{ds}(r^2\tilde{p}) = 0 \text{ on } \partial T \end{aligned} \tag{2.229}$$

where $\mathbf{t}$ is a unit tangent vector on ∂T and s is the arclength parameterization variable for ∂T. An important implication of (2.229), is that $r^2\tilde{p}(r,z)$ is a constant function on ∂T.

In fact, (2.229) implies that $\tilde{p}(r,z) = 0$ on ∂T. To see why, we must consider two scenarios: (i) when r^2 is not constant along any edge of ∂T, and (ii) when $r^2 = a^2 \neq 0$ along an edge of ∂T.

First, suppose that r^2 is not constant on any edge of ∂T. Since $\tilde{p}(r,z) \in P_k(r,z)$,

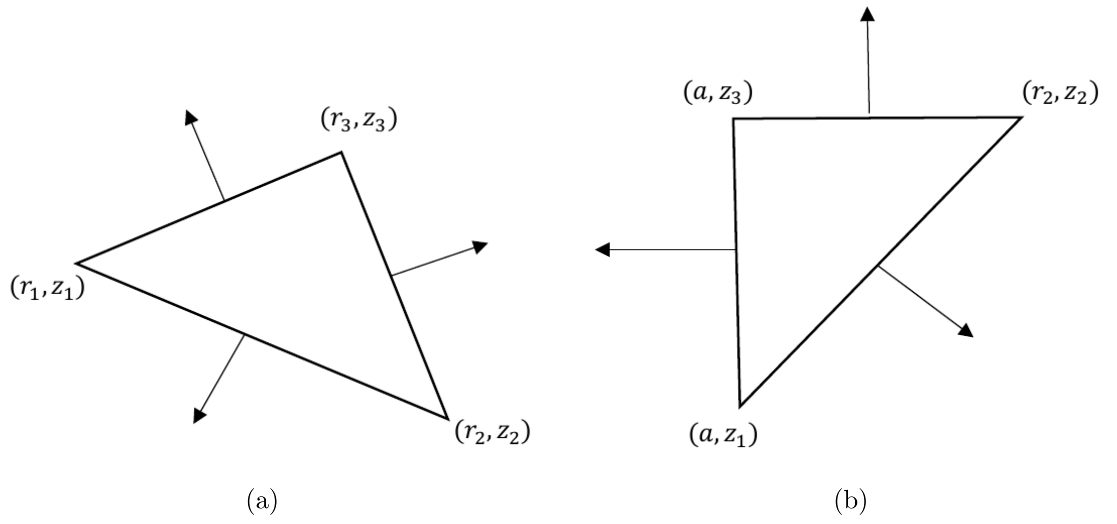


Figure 2-5: Example of a triangle where r^2 is not constant along any edge and a triangle that is constant along an edge.

the only way that $r^2\tilde{p}(r,z)$ can be constant on ∂T is for $\tilde{p}(r,z) = 0$ on ∂T.

Second, suppose that $r = a$ along the edge ℓ^k of ∂T. As T is a non-degenerate triangle, r cannot be constant along the other two edges ℓ^i and ℓ^j of ∂T. Thus,

$\tilde{p}(r,z) = 0$ on edges ℓ^i and ℓ^j, as described for case (i). Along ℓ^k, we have the representation $r^2\tilde{p}(r,z) = a^2\tilde{p}(a,z) = a^2\tilde{p}(z)$, where $\tilde{p}(z)$ is a polynomial in z of degree $\leq k$. Therefore, on ℓ^k, $0 = \frac{d}{ds}(r^2 p) = a^2 \frac{d}{dz}\tilde{p}(z)$ which shows that $\tilde{p}(z)$ is constant on ℓ^k. Since $r^2 p(r,z)$ is continuous on ∂T, $r^2 p(r,z) = 0$ at the triangle vertices where ℓ^k intersects with ℓ^i and ℓ^j. Therefore, since $a \neq 0$, $\tilde{p}(z) = \tilde{p}(a,z) = \tilde{p}(r,z) = 0$ on ℓ^k from which it follows that $\tilde{p}(r,z) = 0$ on ∂T.

A consequence of $\tilde{p}(r,z) = 0$ on ∂T is that the cubic bubble function b_T can be factored from $\tilde{p}(r,z)$. That is, $\tilde{p}(r,z) = b_T\ \psi_{k-3}$ for some $\psi_{k-3} \in P_{k-3}(r,z)$. Using this representation, $\bar{\phi} = \text{curl}_{\text{ac}}\ r\ b_T\ \psi_{k-3}$ for some $\psi_{k-3} \in P_{k-3}(r,z)$. Therefore, $\bar{\phi} \in S_1$ and $S_0 \subset S_1$.

Next, let $\bar{\phi} \in S_1$, so that

$$\bar{\phi} = \text{curl}_{ac}\ r\ b_T\ \psi_{k-3} = \begin{pmatrix} \dfrac{\partial(r\ b_T\ \psi_{k-3})}{\partial z} \\ \dfrac{-1}{r}\dfrac{\partial(r^2\ b_T\ \psi_{k-3})}{\partial r} \end{pmatrix} = \begin{pmatrix} r\ \dfrac{\partial(b_T\ \psi_{k-3})}{\partial z} \\ \dfrac{-1}{r}\ \left(2\ r\ b_T\ \psi_{k-3} + r^2\dfrac{\partial(b_T\ \psi_{k-3})}{\partial r}\right) \end{pmatrix}. \tag{2.230}$$

Observe that $\bar{\phi} \in (P_k(r,z))^2$ and

$$\begin{aligned}\nabla_{\text{axi}} \cdot \bar{\phi} &= \frac{1}{r}\left[\frac{\partial}{\partial r} r^2 \frac{\partial(b_T\ \psi_{k-3})}{\partial z}\right] - \frac{\partial}{\partial z}\left[2\ b_T\ \psi_{k-3} + r\frac{\partial(b_T\ \psi_{k-3})}{\partial r}\right] \\ &= \frac{1}{r}\left[2\ r\frac{\partial(b_T\ \psi_{k-3})}{\partial z} + r^2\ \frac{\partial^2(b_T\ \psi_{k-3})}{\partial z\ \partial r}\right] - \left[2\frac{\partial(b_T\ \psi_{k-3})}{\partial z} + r\frac{\partial^2(b_T\ \psi_{k-3})}{\partial r\ \partial z}\right] = 0.\end{aligned} \tag{2.231}$$

Since $b_T = 0$ on ∂T, the function $r^2\ b_T\ \psi_{k-3} = 0$ on ∂T. Therefore, the tangential directional derivative

$$\begin{pmatrix} \dfrac{\partial(r^2\ b_T\ \psi_{k-3})}{\partial r} \\ \dfrac{\partial(r^2\ b_T \psi_{k-3})}{\partial z} \end{pmatrix} \cdot \mathbf{t}|_{\partial K} = 0 \ \text{ which implies } \ \frac{1}{r}\begin{pmatrix} \dfrac{\partial(r^2\ b_T\ \psi_{k-3})}{\partial z} \\ \dfrac{-\partial(r^2\ b_T \psi_{k-3})}{\partial r} \end{pmatrix} \cdot \mathbf{n}|_{\partial K} = 0. \tag{2.232}$$

Rearranging this expression to match (2.230) gives

$$\bar{\phi} \cdot \mathbf{n}|_{\partial K} = \begin{pmatrix} r \, \dfrac{\partial (b_T \, \psi_{k-3})}{\partial z} \\ \dfrac{-1}{r} \left(2 \, r \, b_T \, \psi_{k-3} + r^2 \dfrac{\partial (b_T \, \psi_{k-3})}{\partial r} \right) \end{pmatrix} \cdot \mathbf{n}|_{\partial K} = 0. \tag{2.233}$$

Therefore $\bar{\phi} \in S_0$, $S_1 \subset S_0$ and $S_1 = S_0$. □

Corollary 2. *Let T denote an arbitrary triangle with one edge lying on the line $r = 0$. For $k \geq 2$ let*

$$\begin{aligned} S_0 :&= \{\bar{\phi} \in (P_k(r,z))^2 : \nabla_{axi} \cdot \bar{\phi} = 0, \quad \bar{\phi} \cdot \mathbf{n} = 0 \text{ on } \partial K\}, \\ \text{and } S_1 :&= \{\nabla_{ac} \, b_T \, \psi_{k-2}\}, \end{aligned} \tag{2.234}$$

where b_T is the cubic bubble function on T and $\psi_{k-2} \in P_{k-2}(r,z)$. Then $S_0 = S_1$.

Proof. Following a similar approach as described in the proof of Lemma 14, suppose $\bar{\phi} \in S_0$ and recall from Lemma 13 that

$$\{\bar{\phi} \in (P_k(r,z))^2 : \nabla_{\text{axi}} \cdot \bar{\phi} = 0\} = \{\text{curl}_{ac} \, r \, p(r,z) : p(r,z) \in P_k(r,z)\}.$$

Since $\bar{\phi} \cdot \mathbf{n} = 0$, then there exists $\tilde{p}(r,z) \in P_k(r,z)$, such that $(\text{curl}_{ac} \, r \, \tilde{p}(r,z)) \cdot \mathbf{n} = 0$ on ∂T. That is,

$$\begin{pmatrix} r \dfrac{\partial}{\partial z} \tilde{p}(r,z) \\ -2 \, \tilde{p}(r,z) - r \dfrac{\partial \tilde{p}}{\partial r} \end{pmatrix} \cdot \mathbf{n} = 0 \text{ on } \partial K. \tag{2.235}$$

Let ℓ_i for $i = 1,2,3$ denote the edges of ∂T, where ℓ_1 represents the edge along the line $r = 0$ (see Figure 2-6). As described in the proof of Lemma 14, $\tilde{p}(r,z) = 0$ along ℓ_2 and ℓ_3. For edge ℓ_1, the normal vector is $\mathbf{n} = (-1,0)^t$. Since $r \dfrac{\partial}{\partial z} \tilde{p}(r,z)|_{\ell_1} = 0$, (2.235) holds for any choice of $\tilde{p}(r,z)$.

Thus, $(r \, \tilde{p}(r,z)) \in P_{k+1}(r,z)$ and $r \, \tilde{p}(r,z)|_{\ell_i} = 0$ for $i = 1,2,3$. If the bubble function b_T is factored out, then $r \, \tilde{p}(r,z) = b_T \, \psi_{k-2}$ for some $\psi_{k-2} \in P_{k-2}(r,z)$. Thus, $S_0 \subset S_1$. The proof that $S_1 \subset S_0$ follows in a similar manner as in the proof of Lemma 14. □

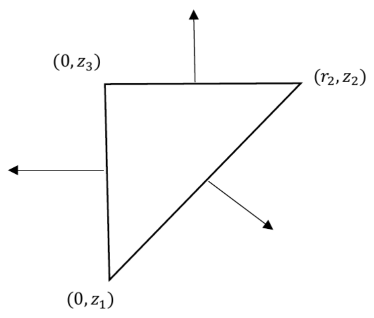


Figure 2-6: Example of a triangle with an edge where $r = 0$.

2.12 Error Analysis

In this section, for $\boldsymbol{\Sigma}_h \times U_h \times Q_h$ satisfying the inf-sup condition

$$\inf_{\mathbf{w}_h \in U_h, p_h \in Q_h} \sup_{(\underline{\boldsymbol{\sigma}}_h, \sigma_h) \in \boldsymbol{\Sigma}_h} \frac{b((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{w}_h) + c((\underline{\boldsymbol{\sigma}}_h, \sigma_h), p_h)}{(\|(\underline{\boldsymbol{\sigma}}_h, \sigma_h)\|_{\boldsymbol{\Sigma}})(\|\mathbf{w}_h\|_U + \|p_h\|_Q)} \geq \beta > 0, \tag{2.236}$$

we present an error analysis for the solution to the discrete linear elasticity problem (2.147)-(2.149). For notational compactness, we take

$$B((\underline{\boldsymbol{\sigma}}_h, \sigma_h), (\mathbf{v}_h, p_h)) = b((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{v}_h) + c((\underline{\boldsymbol{\sigma}}_h, \sigma_h), p_h). \tag{2.237}$$

Before moving forward, we take a moment to consider a reformulation of the problems described in (2.85)-(2.87) and (2.147)-(2.149) to simplify the error analysis. For $A : V \to V'$, $B : V \to M$, $B' : M' \to V'$, consider the well posed general saddle point system problem: Given $f \in V'$ and $g \in M$, find $u \in V$ and $q \in M$ such that

$$Au + B'q = f \tag{2.238}$$

$$Bu = g. \tag{2.239}$$

for $f \in A'$ and $g \in M$. In the context of (2.147)-(2.149), $V := \boldsymbol{\Sigma}_h$, $M := U_h \times Q_h$, A comes from the left most term of (2.147), B and B' come from the expression (2.237), $f = \mathbf{f}$ and $g = \mathbf{f} + \mathbf{f} \wedge \mathbf{x}$ (which equals the sum of the right hand sides of (2.148)-(2.149)).

Since the system is well posed, B is surjective [42]. Taking $u = \phi + u_g$, yields an equivalent problem: Given $\tilde{f} \in V^{'}$, find $\phi \in V$ and $q \in M$ such that find $\phi \in V$ and $q \in M$ such that

$$A\phi + B^{'}q = \tilde{f} := f - Au_g \tag{2.240}$$

$$B\phi = 0. \tag{2.241}$$

Note that for ϕ and its approximation ϕ_h, with $u_h = \phi_h + u_g$, $\|\phi - \phi_h\| = \|u - u_h\|$. Hence for the error analysis, we will assume $g = 0$.
Recall that operator $a(\cdot,\cdot) : \mathbf{\Sigma}_h \times \mathbf{\Sigma}_h \to \mathbb{R}$ as defined in (A.33) is both coercive and continuous (see Lemma 2 and 3). That is,

$$a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\sigma}},\sigma)) = \|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\mathbf{\Sigma}}^2 \geq \gamma > 0 \text{ for all } (\underline{\boldsymbol{\sigma}},\sigma) \in \mathbf{\Sigma}_h, \tag{2.242}$$

$$a((\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau)) \leq \alpha\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\mathbf{\Sigma}}\|(\underline{\boldsymbol{\tau}},\tau)\|_{\mathbf{\Sigma}} \tag{2.243}$$

for some $\alpha > 0$ and all $(\underline{\boldsymbol{\sigma}},\sigma),(\underline{\boldsymbol{\tau}},\tau) \in \mathbf{\Sigma}$. We also note that $B((\cdot,\cdot),(\cdot,\cdot))$ is continuous since

$$\begin{aligned} B((\underline{\boldsymbol{\sigma}},\sigma),(\mathbf{v},q)) &= b((\underline{\boldsymbol{\sigma}},\sigma),\mathbf{v}) + c((\underline{\boldsymbol{\sigma}},\sigma),q) \\ &= (\mathbf{v},\nabla_{\text{axi}}\cdot\underline{\boldsymbol{\sigma}}) - (v_r,\frac{\sigma}{r}) + (\underline{\boldsymbol{\sigma}},\mathcal{S}^2(q)) + (\nabla_{\text{axi}}\cdot(\underline{\boldsymbol{\sigma}},\sigma)\wedge\mathbf{x},q) \\ &\leq C_1\|\mathbf{v}\|_{{}_1\mathbf{L}^2(\Omega)}\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\mathbf{\Sigma}} + C_2\|q\|_{{}_1L^2(\Omega)}\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\mathbf{\Sigma}} \\ &\leq \beta\|(\underline{\boldsymbol{\sigma}},\sigma)\|_{\mathbf{\Sigma}}(\|\mathbf{v}\|_U + \|q\|_Q) \end{aligned} \tag{2.244}$$

for all $(\underline{\boldsymbol{\sigma}},\sigma) \in \mathbf{\Sigma}$, $\mathbf{v} \in U$ and $q \in Q$ where $C_1, C_2, \beta > 0$.
The discrete null space of the operator $B((\cdot,\cdot),(\cdot,\cdot))$ is defined as

$$Z_h = \{(\underline{\boldsymbol{\tau}}_h,\tau_h) \in \mathbf{\Sigma}_h : B((\underline{\boldsymbol{\tau}}_h,\tau_h),(\mathbf{v}_h,q_h) = 0 \text{ for all } \mathbf{v}_h \in U_h \text{ and } q_h \in Q_h\}. \tag{2.245}$$

Since $B((\underline{\boldsymbol{\tau}}_h,\tau_h),(\mathbf{v}_h,q_h)) = 0$ only holds on the discrete subspaces U_h and Q_h, $Z_h \not\subset Z$. This observation motivates the following theorem which bounds the error $\underline{\boldsymbol{\sigma}}_h$ in terms of the spaces U_h, Q_h and Z_h.

Theorem 2. *Let* $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{w}, p)$ *solve* (2.85)-(2.87) *and* $(\underline{\boldsymbol{\sigma}}_h, \sigma_h)$ *solve* (2.147)-(2.149). *If* $\boldsymbol{\Sigma}_h \subset \boldsymbol{\Sigma}$, $U_h \subset U$, $Q_h \subset Q$, *and* Z_h *is defined as in* (2.245), *then*

$$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} \leq C\Big(\inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \inf_{\mathbf{v}_h \in U_h} \|\mathbf{w} - \mathbf{v}_h\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q \Big), \tag{2.246}$$

where $C > 0$, *the constant, is independent of* h.

Proof. Let $(\underline{\boldsymbol{\sigma}}_h, \sigma_h) \in Z_h$ be the unique solution to

$$a((\underline{\boldsymbol{\sigma}}_h, \sigma_h), (\underline{\boldsymbol{\tau}}_h, \tau_h)) = (\mathbf{f}, \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}_h, \tau_h)) \text{ for all } (\underline{\boldsymbol{\tau}}_h, \tau_h) \in Z_h, \tag{2.247}$$

as ensured by the Lax-Milgram Theorem (provided that $\mathbf{f}$ lives in the dual space of ${}_1\mathbf{H}(\text{div}_{\text{axi}}, \Omega; \mathbb{R}^2)$). To develop an error bound, for $(\underline{\boldsymbol{\sigma}}_h, \sigma_h)$, we must compare it with the true solution $(\underline{\boldsymbol{\sigma}}, \sigma)$. Noting again that $Z_h \not\subset Z$, from (2.85)-(2.87) the true solution $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{w}, p)$ satisfies

$$a((\underline{\boldsymbol{\sigma}}, \sigma), (\underline{\boldsymbol{\xi}}_h, \xi_h)) = (\mathbf{f}, \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\xi}}_h, \xi_h)) - B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w}, p)) \text{ for all } (\underline{\boldsymbol{\xi}}_h, \xi) \in \boldsymbol{\Sigma}_h. \tag{2.248}$$

Subtracting (2.247) from (2.248)

$$a((\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h), (\underline{\boldsymbol{\xi}}_h, \xi_h)) = -B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w}, p)) \text{ for all } (\underline{\boldsymbol{\xi}}_h, \xi_h) \in Z_h. \tag{2.249}$$

From (2.245) it then follows that for all $(\underline{\boldsymbol{\xi}}_h, \xi_h) \in Z_h, \mathbf{v}_h \in U_h, q_h \in Q_h$

$$a((\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h), (\underline{\boldsymbol{\xi}}_h, \xi_h)) = -B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w}, p)) + B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{v}_h, q_h)). \tag{2.250}$$

Next, adding and subtracting $(\underline{\boldsymbol{\tau}}_h, \tau_h) \in Z_h$ in $a(\cdot, \cdot)$, (2.250) becomes

$$\begin{aligned} a((\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h), (\underline{\boldsymbol{\xi}}_h, \xi_h)) = &- a((\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h), (\underline{\boldsymbol{\xi}}_h, \xi_h)) \\ &- B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w} - \mathbf{v}_h, p - q_h)). \end{aligned} \tag{2.251}$$

Choosing $(\underline{\boldsymbol{\xi}}_h, \xi_h) = (\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h) \in Z_h$, and using the coercivity and continuity of $a(\cdot, \cdot)$ (described in (2.242), (2.243)) and the continuity of $B((\cdot, \cdot), (\cdot, \cdot))$ (described in (2.244)) we obtain

$$\begin{aligned} 0 < \ &\gamma \|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|^2_{\boldsymbol{\Sigma}} \\ &\leq \alpha \ \|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|_{\boldsymbol{\Sigma}} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} \\ &\quad + \beta \ \|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|_{\boldsymbol{\Sigma}} \left(\|\mathbf{w} - \mathbf{v}_h\|_U + \|p - q_h\|_Q\right). \end{aligned} \tag{2.252}$$

Dividing through by $\gamma \|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|_{\boldsymbol{\Sigma}}$ gives

$$\|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|_{\boldsymbol{\Sigma}} \leq \frac{\alpha}{\gamma} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \frac{\beta}{\gamma} \left(\|\mathbf{w} - \mathbf{v}_h\|_U + \|p - q_h\|_Q\right). \tag{2.253}$$

Next, applying the triangle inequality, for an arbitrary element $(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h$,

$$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} \leq \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \|(\underline{\boldsymbol{\tau}}_h - \underline{\boldsymbol{\sigma}}_h, \tau_h - \sigma_h)\|_{\boldsymbol{\Sigma}}. \tag{2.254}$$

Since $(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h$, $\mathbf{v}_h \in U_h$ and $q_h \in Q_h$ are arbitrary, combining (2.253) and (2.254) we get

$$\begin{aligned} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} \leq &(1 + \frac{\alpha}{\gamma}) \inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in Z_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} \\ &+ \frac{\beta}{\gamma} \left(\inf_{\mathbf{v}_h \in U_h} \|\mathbf{w} - \mathbf{v}_h\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q \right). \end{aligned} \tag{2.255}$$

In order to lift the approximation of $(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)$ from the infinimum over Z_h to the infinimum over $\boldsymbol{\Sigma}_h$, we use the inf-sup condition (2.236). A equivalent property to the spaces $\boldsymbol{\Sigma}_h \times U_h \times W_h$ satisfying (2.236) is the existence of a projection $\Pi_h : \boldsymbol{\Sigma} \to \boldsymbol{\Sigma}_h$

satisfying

$$B(((\underline{\boldsymbol{\tau}},\tau)-\Pi_h(\underline{\boldsymbol{\tau}},\tau)),(\mathbf{v}_h,q_h))=0 \text{ for all } (\mathbf{v}_h,q_h)\in U_h\times Q_h \tag{2.256}$$

and

$$\|\Pi_h(\underline{\boldsymbol{\tau}},\tau)\|_{\mathbf{\Sigma}}\le C_\Pi\|(\underline{\boldsymbol{\tau}},\tau)\|_{\mathbf{\Sigma}}, \tag{2.257}$$

where $C_\Pi>0$ is a constant that is independent of h.
Let $(\underline{\boldsymbol{\xi}}_h,\xi_h)\in\mathbf{\Sigma}_h$, and introduce $(\underline{\boldsymbol{\rho}}_h,\rho_h)\in\mathbf{\Sigma}_h$ satisfying

$$(\underline{\boldsymbol{\rho}}_h,\rho_h)=\Pi_h(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\xi}}_h,\sigma-\xi_h) \text{ where } \|(\underline{\boldsymbol{\rho}}_h,\rho_h)\|_{\mathbf{\Sigma}}\le C_\Pi\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\xi}}_h,\sigma-\xi_h)\|_{\mathbf{\Sigma}}. \tag{2.258}$$

Taking $(\underline{\boldsymbol{\tau}}_h,\tau_h)=(\underline{\boldsymbol{\xi}}_h+\underline{\boldsymbol{\rho}}_h,\xi_h+\rho_h)$

$$\begin{aligned}B((\underline{\boldsymbol{\tau}}_h,\tau_h),(\mathbf{w}_h,q_h))&=B((\underline{\boldsymbol{\xi}}_h,\xi_h),(\mathbf{v}_h,q_h))+B((\underline{\boldsymbol{\rho}}_h,\rho_h),(\mathbf{v}_h,q_h))\\&=B((\underline{\boldsymbol{\xi}}_h,\xi_h),(\mathbf{v}_h,q_h))+B((\underline{\boldsymbol{\sigma}},\sigma),(\mathbf{v}_h,q_h))-B((\underline{\boldsymbol{\xi}}_h,\xi_h),(\mathbf{v}_h,q_h))\\&=B((\underline{\boldsymbol{\sigma}},\sigma),(\mathbf{v}_h,q_h))=0,\end{aligned} \tag{2.259}$$

which implies that $(\underline{\boldsymbol{\tau}}_h,\tau_h)\in Z_h$. Note that the final equality in this expression is a result of the saddle point reformulation (2.240)-(2.241). Next, using $(\underline{\boldsymbol{\tau}}_h,\tau_h)=(\underline{\boldsymbol{\xi}}_h+\underline{\boldsymbol{\rho}}_h,\xi_h+\rho_h)$

$$\begin{aligned}\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\tau}}_h,\sigma-\tau_h)\|_{\mathbf{\Sigma}}&\le\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\xi}}_h,\sigma-\xi_h)\|_{\mathbf{\Sigma}}+\|(\underline{\boldsymbol{\rho}}_h,\rho_h)\|_{\mathbf{\Sigma}}\\&\le(1+C_\Pi)\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\xi}}_h,\sigma-\xi_h)\|_{\mathbf{\Sigma}}.\end{aligned} \tag{2.260}$$

Finally, taking infinimums over the appropriate spaces on the left and right sides gives the result

$$\inf_{(\underline{\boldsymbol{\tau}}_h,\tau_h)\in Z_h}\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\tau}}_h,\sigma-\tau_h)\|_{\mathbf{\Sigma}}\le(1+C_\Pi)\inf_{(\underline{\boldsymbol{\xi}}_h,\xi_h)\in\mathbf{\Sigma}_h}\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\xi}}_h,\sigma-\xi_h)\|_{\mathbf{\Sigma}}. \tag{2.261}$$

Combining (2.255) and (2.261) we obtain

$$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} \le C(\inf_{\underline{\boldsymbol{\tau}}_h, \tau_h \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} \\ + \inf_{\mathbf{v}_h \in U_h} \|\mathbf{w} - \mathbf{v}_h\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q). \tag{2.262}$$

□

With error bounds for the stress space established, the following theorem establishes error bounds for the displacement and skew-symmetry approximations.

Theorem 3. *For* $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{w}, p)$ *satisfying* (2.85)-(2.87) *and* $((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{w}_h, p_h)$ *satisfying* (2.147)-(2.149) *there exists* $C > 0$*, independent of* h*, such that*

$$\|\mathbf{w} - \mathbf{w}_h\|_U + \|p - p_h\|_Q \\ \le C \left(\inf_{\underline{\boldsymbol{\tau}}_h, \tau_h \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \inf_{\mathbf{v}_h \in U_h} \|\mathbf{w} - \mathbf{v}_h\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q \right). \tag{2.263}$$

Proof. Subtracting equations (2.147) from (2.85) gives

$$B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w} - \mathbf{w}_h, p - p_h)) = -a((\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h), (\underline{\boldsymbol{\xi}}_h, \xi_h)) \tag{2.264}$$

for all $(\underline{\boldsymbol{\xi}}_h, \xi_h) \in \boldsymbol{\Sigma}_h$.

For any $\mathbf{v}_h \in U_h$ and $q_h \in Q_h$, the inf-sup condition (2.236) gives

$$\beta\left(\|\mathbf{w}_h - \mathbf{v}_h\|_U + \|p_h - q_h\|_Q\right) \le \sup_{(\underline{\boldsymbol{\xi}}_h, \xi_h) \in \boldsymbol{\Sigma}_h} \frac{|B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w}_h - \mathbf{v}_h, p_h - q_h))|}{\|(\underline{\boldsymbol{\xi}}_h, \xi_h)\|_{\boldsymbol{\Sigma}}} \\ \le \sup_{(\underline{\boldsymbol{\xi}}_h, \xi_h) \in \boldsymbol{\Sigma}_h} \left(\frac{|B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w_h} - \mathbf{w}, p_h - p))|}{\|(\underline{\boldsymbol{\xi}}_h, \xi_h)\|_{\boldsymbol{\Sigma}}} + \frac{|B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w} - \mathbf{v}_h, p - q_h))|}{\|(\underline{\boldsymbol{\xi}}_h, \xi_h)\|_{\boldsymbol{\Sigma}}} \right) \\ \le \sup_{(\underline{\boldsymbol{\xi}}_h, \xi_h) \in \boldsymbol{\Sigma}_h} \left(\frac{|-a((\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h), (\underline{\boldsymbol{\xi}}_h, \xi_h))|}{\|(\underline{\boldsymbol{\xi}}_h, \xi_h)\|_{\boldsymbol{\Sigma}}} + \frac{|B((\underline{\boldsymbol{\xi}}_h, \xi_h), (\mathbf{w} - \mathbf{v}_h, p - q_h))|}{\|(\underline{\boldsymbol{\xi}}_h, \xi_h)\|_{\boldsymbol{\Sigma}}} \right) \\ \le \max\{\alpha, \beta\}(\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} + \|\mathbf{w} - \mathbf{v}_h\|_U + \|p - q_h\|_Q), \tag{2.265}$$

where in the last step we have used the continuity of $a(\cdot,\cdot)$ and $B(\cdot,\cdot)$.
Combining (2.265) with the triangle inequality gives

$$\begin{aligned} &\|\mathbf{w}-\mathbf{w}_h\|_U + \|p-p_h\|_Q \\ &\le \|\mathbf{w}-\mathbf{v}_h\|_U + \|\mathbf{v}_h-\mathbf{w}_h\|_U + \|p-q_h\|_Q + \|q_h-p_h\|_Q \\ &\le C(\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\sigma}}_h, \sigma-\sigma_h)\|_{\boldsymbol{\Sigma}} + \|\mathbf{w}-\mathbf{v}_h\|_U + \|p-q_h\|_Q). \end{aligned} \tag{2.266}$$

As $\mathbf{v}_h \in U_h$ and $q_h \in Q_h$ are arbitrary, (2.263) follows from (2.266) and (2.262). $\square$

Combining Theorems 2 and 3 we have the following.

Corollary 3. *Let* $((\underline{\boldsymbol{\sigma}},\sigma),\mathbf{w},p) \in \boldsymbol{\Sigma}\times U\times Q$ *be the solution of* (2.85)-(2.87) *and* $((\underline{\boldsymbol{\sigma}}_h,\sigma_h),\mathbf{w}_h,p_h) \in \boldsymbol{\Sigma}_h\times U_h\times Q_h$ *the solution of* (2.147)-(2.149), *then*

$$\begin{aligned} &\|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\sigma}}_h, \sigma-\sigma_h)\|_{\boldsymbol{\Sigma}} + \|\mathbf{w}-\mathbf{w}_h\|_U + \|p-p_h\|_Q \\ &\quad \le C(\inf_{(\underline{\boldsymbol{\tau}}_h,\tau_h)\in\boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}}-\underline{\boldsymbol{\tau}}_h, \sigma-\tau_h)\|_{\boldsymbol{\Sigma}} + \inf_{\mathbf{v}_h\in U_h}\|\mathbf{w}-\mathbf{v}_h\|_U + \inf_{q_h\in Q_h}\|p-q_h\|_Q). \end{aligned} \tag{2.267}$$

Using Corollary 3, and additional smoothness assumptions, we can now form an error bound in terms of the mesh parameter h. First observe that for the axisymmetric BDM$_k$ interpolation operator $\tilde{\rho}_h : {}_1\mathbf{H}^1(\Omega) \to \text{BDM}_k(\mathcal{T}_h)$ as defined in [45], if $\mathbf{u} \in {}_1\mathbf{H}^{k+1}(\Omega)$, then for some $C > 0$,

$$\|\mathbf{u}-\tilde{\rho}_h(\mathbf{u})\|_{{}_1L^2(\Omega)} \le C\, h^{k+1}|\mathbf{u}|_{{}_1H^{k+1}(\Omega)}. \tag{2.268}$$

In addition, if $\nabla_{\text{axi}}\cdot\mathbf{u} \in {}_1H^k(\Omega)$ where $\left(\Sigma_{T\in\mathcal{T}_h}|\nabla_{\text{axi}}\cdot\tilde{\rho}_h(\mathbf{u})|^2_{{}_1H^{k+1}(T)}\right)^2 < C_1$, then for some $C > 0$,

$$\|\nabla_{\text{axi}}\cdot\mathbf{u} - \nabla_{\text{axi}}\cdot\tilde{\rho}_h(\mathbf{u})\|_{{}_1L^2(\Omega)} \le Ch^k. \tag{2.269}$$

Combining the results and assumptions of (2.268) and (2.269), if $\mathbf{u} \in {}_1\mathbf{H}^{k+1}(\Omega)$ and $\nabla_{\text{axi}}\cdot\mathbf{u} \in {}_1H^k(\Omega)$ where $\left(\Sigma_{T\in\mathcal{T}_h}|\nabla_{\text{axi}}\cdot\tilde{\rho}_h(\mathbf{u})|^2_{{}_1H^{k+1}(T)}\right) < C_1$, then there exists $C > 0$

such that

$$\|\mathbf{u} - \tilde{\rho}_h \, \mathbf{u}\|_{{}_1\mathbf{H}(\mathrm{div},\Omega)} \leq C \, h^k. \tag{2.270}$$

Under analogous assumptions, this result can be extended to the tensor case, where $\tilde{\boldsymbol{\rho}}_h : {}_1\underline{\mathbf{H}}^1(\Omega) \to (\mathrm{BDM}_k(\mathcal{T}_h))^2$ represents the BDM_k interpolation operator applied to the rows of a tensor so that

$$\|\underline{\boldsymbol{\sigma}} - \tilde{\boldsymbol{\rho}}_h \underline{\boldsymbol{\sigma}}\|_{{}_1\underline{\mathbf{H}}(\mathrm{div},\Omega)} \leq C \, h^k. \tag{2.271}$$

Next we present a result from [16] which bounds the Clément operator Λ_h^k. The Clément operator Λ_h^k maps ${}_1L^2(\Omega)$ into the space of degree k Lagrangian finite elements on the mesh $\mathcal{T}_h$. Indeed, as stated in Corollary 2 of Theorem 1 in [16], for $v \in {}_1H^{k+1}(\Omega)$, there exists a C independent of h such that

$$\|v - \Lambda_h^k v\|_{{}_1L^2(\Omega)} \leq C h^{k+1} |v|_{{}_1H^{k+1}(\Omega)}. \tag{2.272}$$

As with the BDM interpolation $\tilde{\rho}_h$, the bound for Λ_h^k can be extended to vector and tensor functions.
The next corollary introduces error bounds in terms of the mesh parameter h for the $k = 1, 2$ cases.

Corollary 4. *Assume that $\mathbf{\Pi}_h$ of Lemma 10 or 11 satisfies* (2.159)-(2.160). *If $(\underline{\boldsymbol{\sigma}}, \sigma, \mathbf{w}, p) \in {}_1\underline{\mathbf{H}}^k(\Omega) \times ({}_{-1}L^2(\Omega) \cap {}_1H^k(\Omega)) \times {}_1\mathbf{H}^k(\Omega) \times {}_1H^k(\Omega)$ solves* (2.85)-(2.87) *and $(\underline{\boldsymbol{\sigma}}_h, \sigma_h, \mathbf{w}_h, p_h) \in (BDM_k)^2(\mathcal{T}_h) \times P_k(\mathcal{T}_h) \times (P_{k-1}(\mathcal{T}_h))^2 \times P_{k-1}(\mathcal{T}_h)$ solves* (2.147)-(2.149) *for $k = 1, 2$, then*

$$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} + \|\mathbf{w} - \mathbf{w}_h\|_U + \|p - p_h\|_Q \leq C \, h^k. \tag{2.273}$$

Proof. From Corollary 3,

$$\begin{aligned}&\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}} + \|\mathbf{u} - \mathbf{u}_h\|_U + \|p - p_h\|_Q \\ &\leq C\Big(\inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \inf_{\mathbf{v}_h \in U_h} \|\mathbf{u} - \mathbf{v_h}\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q \Big)\end{aligned}. \tag{2.274}$$

The BDM error bounds from (2.271), (2.272) gives

$$\inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \Sigma_h \times S_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} \leq C\ h^k. \tag{2.275}$$

In addition, using a vector generalization of (2.272)

$$\inf_{\mathbf{v}_h \in U_h} \|\mathbf{u} - \mathbf{v}_h\|_U \leq \|\mathbf{u} - \Lambda_h^{k-1}\mathbf{u}\|_{{}_1L^2(\Omega)} \leq C\ h^k |\mathbf{u}|_{{}_1\mathbf{H}^k(\Omega)} \tag{2.276}$$

and

$$\inf_{q_h \in W_h} \|\mathcal{S}^2(p - q_h)\|_Q \leq C_1 \|p - \Lambda_h^k\ p\|_{{}_1L^2(\Omega)} \leq C\ h^k |p|_{{}_1H^k(\Omega)}. \tag{2.277}$$

Combining (2.274), (2.275), (2.276) and (2.277) gives the result. □

To conclude this section, we establish an error bound for the true displacement $\mathbf{u}$. At this point, error bounds have been established in terms of the pseudo displacement variable $\mathbf{w}$. Recall from Section 2.6.1, however, that $\mathbf{w} = \mathbf{u} - \mathbf{x}^{\perp} p$.

Corollary 5. *Let* $((\underline{\boldsymbol{\sigma}}, \sigma), \mathbf{w}, p) \in \boldsymbol{\Sigma} \times U \times Q$ *be the solution of* (2.85)-(2.87) *and* $((\underline{\boldsymbol{\sigma}}_h, \sigma_h), \mathbf{w}_h, p_h) \in \boldsymbol{\Sigma}_h \times U_h \times Q_h$ *the solution of* (2.147)-(2.149)*. Furthermore, let* $\mathbf{u} = \mathbf{w} + \mathbf{x}^{\perp} p$ *denote the true displacement, and* $\mathbf{u}_h = \mathbf{w}_h + \mathbf{x}^{\perp} p_h$ *denote the discrete approximation to the true displacement. There exists a* $C > 0$ *independent of* h*, such that*

$$\begin{aligned}\|\mathbf{u} - \mathbf{u}_h\|_U \leq C\Big(&\inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} \\ &+ \inf_{\mathbf{v}_h \in U_h} \|\mathbf{w} - \mathbf{v_h}\|_U + \inf_{q_h \in Q_h} \|p - q_h\|_Q \Big).\end{aligned} \tag{2.278}$$

Proof. For a bounded domain Ω, observe that

$$\|\mathbf{x}^{\perp}(p - p_h)\|_U \leq C_{\mathbf{x}^{\perp}} \|p - p_h\|_Q, \tag{2.279}$$

where the constant $C_{\mathbf{x}^{\perp}} > 0$ is independent of h. Therefore, using Theorem 3 we have that

$$\begin{aligned} \|\mathbf{u} - \mathbf{u}_h\|_U &= \|(\mathbf{w} - \mathbf{w}_h) + \mathbf{x}^{\perp}(p - p_h)\|_U \leq \|\mathbf{w} - \mathbf{w}_h\|_U + \|\mathbf{x}^{\perp}(p - p_h)\|_U \\ &\leq C\Big(\inf_{(\underline{\boldsymbol{\tau}}_h, \tau_h) \in \boldsymbol{\Sigma}_h} \|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\tau}}_h, \sigma - \tau_h)\|_{\boldsymbol{\Sigma}} + \inf_{\mathbf{v}_h \in U_h} \|\mathbf{u} - \mathbf{v_h}\|_U \\ &\qquad + \inf_{q_h \in Q_h} \|p - q_h\|_Q \Big). \end{aligned} \tag{2.280}$$

□

2.13 Computational Results

To verify our theoretical results, we next consider two computational experiments. A square domain, $[0, 1] \times [0, 1]$, is used. We consider the displacement solution

$$\mathbf{u}(r, z) = \begin{pmatrix} 4r^3(1 - r)z(1 - z) \\ -4r^3(1 - r)z(1 - z) \end{pmatrix}. \tag{2.281}$$

To avoid confusion, this $\mathbf{u}$ represents the true displacement solution, not the pseudo-displacement, $\mathbf{w}$, solution that is described in Section 2.6.1.

The solution has been selected to be consistent with homogenous Dirichlet conditions while maintaining a large enough polynomial degree to observe the order of convergence. Based on $\mathbf{u}$, the true solution for $\underline{\boldsymbol{\sigma}}$ is derived from the relationship

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \underline{\boldsymbol{\epsilon}}(\mathbf{u}), \quad \text{where} \quad \mathcal{A}\underline{\boldsymbol{\sigma}} = \frac{1}{2\mu}\left(\underline{\boldsymbol{\sigma}} - \frac{\lambda}{2\mu + 3\lambda}\operatorname{tr}(\underline{\boldsymbol{\sigma}})\mathbf{I}\right). \tag{2.282}$$

The values of λ and μ vary based on the example.

Example 1

In our first example, we consider the parameters $\mu = \frac{1}{2}$ and $\lambda = 0$. Therefore, based on (2.282), the true symmetric stress tensor is

$$\underline{\boldsymbol{\sigma}} = \begin{pmatrix} 4r^2(4r-3)(z-1)z & 2r^2(r(r-1)(2z-1)-(4r-3)(z-1)z) \\ * & -4r^3(r-1)(2z-1) \end{pmatrix} \tag{2.283}$$

$$\sigma = 4r^2(1-r)z(1-z) \tag{2.284}$$

and the divergence of the stress tensor is

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) = \begin{pmatrix} 2r(2r^3 - 2r^2(4z-1) + r(32z^2 - 26z - 3) - 18(z-1)z) - 4r(1-r)z(1-z) \\ -2r(4r^3 + r^2(1-10z) + 4r(4z^2 - 2z - 1) - 9(z-1)z) \end{pmatrix}. \tag{2.285}$$

Presented in Table 2.1-2.1 are results of the simulation with the grad-div parameter (see (2.76)) $\gamma = 1$. We note that the convergence rate for the displacement reflects the true displacement, not the pseudo displacement. Computations were performed using the approximation elements $\text{BDM}_1 - \text{disc}P1 - \text{disc}P_0 - \text{disc}P_0$ (shown in Table 2.1), and $\text{BDM}_2 - \text{disc}P_2 - \text{disc}P_1 - \text{disc}P_1$ (shown in Table 2.2)

Example 2

For the second example, we consider the parameters $\mu = \frac{1}{2}$ and $\lambda = 1$. Therefore, based on (2.282), the true symmetric stress tensor is

$$\underline{\boldsymbol{\sigma}} = \begin{pmatrix} 4r^2(-2r^2z + r^2 + 9rz^2 - 7rz - r - 7z^2 + 7z) & 2r^2(2r^2z - r^2 - 4rz^2 + 2rz + r + 3z^2 - 3z) \\ * & 4r^2(-4r^2z + 2r^2 + 5rz^2 - rz - 2r - 4z^2 + 4z) \end{pmatrix} \tag{2.286}$$

$$\sigma = 4r^2(-2r^2z + r^2 + 6rz^2 - 4rz - r - 5z^2 + 5z) \tag{2.287}$$

and the divergence of the stress tensor is

$$\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) = \begin{pmatrix} 2r(2r^3 - 24r^2z + 10r^2 + 60rz^2 - 42rz - 9r - 32z^2 + 32z) \\ -2r(8r^3 + r^2(7-30z) + 4r(4z^2 + 2z - 3) - 9(z-1)z) \end{pmatrix}. \tag{2.288}$$

Table 2.1: Example 1 : Axisymmetric Elasticity Convergence Rates for BDM1 - disc P1 - discP0 - discP0 finite elements with grad-div stabilization parameter $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\mathbf{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\text{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	5.444E-01	1.0	2.679E-02	1.0	1.263E-01	1.0
$\frac{1}{6}$	3.584E-01	1.0	1.800E-02	1.0	8.268E-02	1.0
$\frac{1}{8}$	2.673E-01	1.0	1.353E-02	1.0	6.129E-02	1.0
$\frac{1}{10}$	2.132E-01	1.0	1.083E-02	1.0	4.867E-02	1.0
$\frac{1}{12}$	1.774E-01	–	9.029E-03	–	4.036E-02	–
Pred.		1.0		1.0		1.0

Table 2.2: Example 1 : Axisymmetric Elasticity Convergence Rates for BDM2 - discP2 - discP1 - discP1 finite elements with grad-div stabilization parameter $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\mathbf{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\text{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	6.797E-02	2.0	8.381E-03	1.9	1.602E-02	2.1
$\frac{1}{6}$	3.061E-02	2.0	3.915E-03	1.9	6.753E-03	2.1
$\frac{1}{8}$	1.730E-02	2.0	2.238E-03	2.0	3.647E-03	2.1
$\frac{1}{10}$	1.109E-02	2.0	1.442E-03	2.0	2.264E-03	2.1
$\frac{1}{12}$	7.711E-03	–	1.005E-03	–	1.536E-03	–
Pred.		2.0		2.0		2.0

Presented in Table 2.3-2.4 are results of the simulation with the grad-div parameter (see (2.76)) $\gamma = 1$. We note that the convergence rate for the displacement reflects the true displacement, not the pseudo displacement. Computations were performed using the approximation elements $\text{BDM}_1 - \text{disc}P1 - \text{disc}P_0 - \text{disc}P_0$ (shown in Table 2.3) and $\text{BDM}_2 - \text{disc}P_2 - \text{disc}P_1 - \text{disc}P_1$ (shown in Table 2.4).

The computational results from Example 1 and Example 2 are consistent with the theoretically predicted results from Lemma 10, Lemma 11, Corollary 4, and Corollary 5.

Table 2.3: Example 2 : Axisymmetric Elasticity Convergence Rates for BDM1 - discP1 - discP0 - discP0 finite elements with grad-div stabilization parameter $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\mathrm{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	1.273E+00	1.0	2.908E-02	1.0	1.912E-01	1.1
$\frac{1}{6}$	8.444E-01	1.0	1.911E-02	1.1	1.200E-01	1.1
$\frac{1}{8}$	6.308E-01	1.0	1.410E-02	1.0	8.636E-02	1.1
$\frac{1}{10}$	5.034E-01	1.0	1.115E-02	1.0	6.727E-02	1.1
$\frac{1}{12}$	4.189E-01	–	9.227E-03	–	5.508E-02	–
Pred.		1.0		1.0		1.0

Table 2.4: Example 2 : Axisymmetric Elasticity Convergence Rates for BDM2 - discP2- discP1 - discP1 finite elements with grad-div stabilization parameter $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\mathrm{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	1.169E-01	1.9	8.660E-03	1.9	2.202E-02	2.2
$\frac{1}{6}$	5.344E-02	1.9	3.986E-03	2.0	8.882E-03	2.2
$\frac{1}{8}$	3.053E-02	2.0	2.263E-03	2.0	4.652E-03	2.2
$\frac{1}{10}$	1.974E-02	2.0	1.454E-03	2.0	2.823E-03	2.2
$\frac{1}{12}$	1.380E-02	–	1.011E-03	–	1.881E-03	–
Pred.		2.0		2.0		2.0

2.14 Conclusion and Future Work

In this work, we develop a computational framework for the axisymmetric linear elasticity problem with weak symmetry. Provided the projection bounds (2.159)-(2.160) are satisfied, Lemmas 10, and 11 establish that the finite element spaces $(((\mathbf{BDM}_1)^2 \times P_1) \times (P_0)^2 \times P_0)$ and $(((\mathbf{BDM}_2)^2 \times P_2) \times (P_1)^2 \times P_1)$ are inf-sup stable with error bounds as stated in Corollary 4. Computational examples presented in Section 2.13 support these results. It remains an open question whether for general k, if $(((\mathbf{BDM}_k)^2 \times P_k) \times (P_{k-1})^2 \times P_{k-1})$ form an inf-sup finite element for this problem.

In the Cartesian setting, the spaces $((\mathbf{BDM}_k)^2 \times (P_{k-1})^2 \times P_{k-1})$ form an inf-sup stable triple for the linear elasticity problem with weak symmetry [23]. Moreover, in the axisymmetric setting (provided the projection bound is satisfied), the convergence order for the $k = 1, 2$ cases matches the Cartesian result. Therefore, it maybe a reasonable conjecture that the finite element $(((\mathbf{BDM}_k)^2 \times P_k) \times (P_{k-1})^2 \times P_{k-1})$ is inf-sup stable for the axisymmetric problem.

To test this conjecture, Tables 2.5 and 2.6 present convergence results for $(((\mathbf{BDM}_3)^2, P_3) \times P_2 \times P_2)$. Encouragingly, the convergence rates match the theoretically expected rate given the polynomial approximation order.

It is not clear how one can prove that $(((\mathbf{BDM}_k)^2, P_k) \times (P_{k-1})^2 \times P_{k-1})$ is inf-sup stable. One problem lies in showing that a projection operator that satisfies the properties of Theorem 1 exists. Notably, any projection operator must account for the functional form of the axisymmetric divergence. Specifically, relative to the Cartesian divergence, the axisymmetric divergence includes the term $\frac{1}{r}p_k(z)$, introducing an additional $k + 1$ degrees of freedom.

The form of the divergence relates to another important difference between the axisymmetric and Cartesian BDM$_k$ space. In the Cartesian setting, the basis functions of BDM$_k$ can be grouped into three categories. The first are functions with a non-zero normal component. In the standard BDM$_k$ space, these functions are associated with the degrees of freedom

$$\int_{\ell} \mathbf{u} \cdot \mathbf{n}\, p(s)\, ds = 0 \qquad \text{for all } \ell \in \partial K, \text{ and } p(s) \in R_{\partial K}(s). \tag{2.289}$$

The second set of basis functions are those with a zero normal component but non-zero divergence. These functions are associated with the degrees of freedom

$$\int_K \mathbf{u} \cdot \nabla p\, dK = 0 \qquad \text{for all } p \in P_{k-1}(K). \tag{2.290}$$

To see why, integrate by parts to get

$$\int_K \mathbf{u} \cdot \nabla p\, dK = \int_{\partial K} \mathbf{u} \cdot \mathbf{n}\, p\, \partial K - \int_K \nabla \cdot \mathbf{u}\, p\, dK = 0 \tag{2.291}$$

where $\mathbf{u} \cdot \mathbf{n} = 0$ implies that $\nabla \cdot \mathbf{u} = 0$ and hence $\mathbf{u} = 0$.
The third set of basis functions are those with a zero normal component and a zero divergence. These functions are associated with the degrees of freedom

$$\int_K \mathbf{u} \cdot \mathbf{q}\, dK = 0 \tag{2.292}$$

where $\mathbf{q} \in \Phi = \{\mathbf{w} \mid \mathbf{w} \in H(\text{div}; K), \mathbf{w} \cdot \mathbf{n} = 0 \text{ and } \nabla \cdot \mathbf{w} = 0\}$. From here, it can be shown that functions $\mathbf{q} \in \Phi$ have the form $\mathbf{q} = \text{curl}(b_K\, p)$ where $p \in P_{k-2}(x, y)$.
In contrast, the equivalent axisymmetric bubble function representation from Lemma 14 shows that functions with zero normal component and zero divergence have the form $\mathbf{q} = \nabla_{\text{ac}}(r\ b_K\ p)$ where $p \in P_{k-3}(r, z)$. This indicates that the space remaining once the boundary and non-vanishing divergence functions are removed is notably smaller than in the Cartesian case.

Table 2.5: Example 1 : Axisymmetric Elasticity Convergence Rates for BDM3 - discP2 - discP2 - discP2 finite elements with $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\text{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	6.363E-03	2.9	1.357E-03	2.9	1.699E-03	3.0
$\frac{1}{6}$	1.930E-03	3.0	4.229E-04	2.9	5.031E-04	3.0
$\frac{1}{8}$	8.219E-04	3.0	1.815E-04	3.0	2.109E-04	3.0
$\frac{1}{10}$	4.230E-04	3.0	9.367E-05	3.0	1.074E-04	3.0
$\frac{1}{12}$	2.456E-04	–	5.443E-05	–	6.185E-05	–

Table 2.6: Example 2 : Axisymmetric Elasticity Convergence Rates for BDM3 - discP2 - discP2 - discP2 finite elements with $\gamma = 1$.

h	$\|(\underline{\boldsymbol{\sigma}} - \underline{\boldsymbol{\sigma}}_h, \sigma - \sigma_h)\|_{\boldsymbol{\Sigma}}$	Cvg. Rate	$\|\mathbf{u} - \mathbf{u}_h\|_U$	Cvg. Rate	$\|\text{as}(\underline{\boldsymbol{\sigma}}_h)\|_Q$	Cvg. Rate
$\frac{1}{4}$	1.155E-02	3.0	1.359E-03	2.9	1.454E-03	3.1
$\frac{1}{6}$	3.459E-03	3.0	4.233E-04	2.9	4.192E-04	3.1
$\frac{1}{8}$	1.465E-03	3.0	1.816E-04	3.0	1.733E-04	3.1
$\frac{1}{10}$	7.517E-04	3.0	9.370E-05	3.0	8.742E-05	3.1
$\frac{1}{12}$	4.356E-04	–	5.445E-05	–	5.002E-05	–

Chapter 3

Two-Phase Navier–Stokes Preconditioners

3.1 Introduction

Dynamic free-surface models are often used to simulate physical processes that appear in industrial applications. For example, in fluid dynamics simulations free-surface models track the interface between air and water over time. Reliably tracking fluid interfaces over time is an important element of modeling many complicated hydraulic processes such as waves crashing into coastal barriers, the stress forces affecting a bridge, or flow dynamics following a catastrophic failure in infrastructure (e.g., a dam break).

Often, several different physical models are coupled to form a complete free-surface model, and a splitting scheme is then used to approximate the solution numerically. The variable density/viscosity Navier–Stokes equations – which describe the evolution of fluid velocity – are an important component of these splitting schemes. In many cases, the computational effort needed to numerically approximate the Navier–Stokes equations forms a bottleneck in the splitting scheme, which limits the size of the simulation that practitioners can perform. Therefore, developing efficient methods to solve the variable density/viscosity Navier–Stokes equations is important in extending the size and scope of simulation that can be performed with free-surface models.

Indeed, because of the importance in industrial applications, a lot of research has been done to develop efficient methods for solving the Navier–Stokes equations. One technique used to improve the performance of the Navier–Stokes equations is to use a projection style schemes like those developed by Chorin and Temam (see [53]). These techniques involving splitting the Navier–Stokes equation to first solve an advection-diffusion equation to obtain a non-mass conserving approximation. Next, a pressure correction is made to make the velocity mass conserving. While these methods have the advantage of reducing simulation run times, there are important challenges about the appropriate boundary conditions in the pressure correction step as well as difficulties in applying the methods to higher order approximation schemes. For these reasons, it remains important to find faster ways of solving the fully coupled Navier–Stokes problem.

While significant progress has been made, finding efficient solvers for the fully coupled Naiver–Stokes equation remain a challenge, particularly for high Reynolds number flow. Typically, the simulation size is too large for direct solvers to be practical. Therefore, most research has focused on preconditioned Krylov based iterative techniques like GMRES [79]. Examples of preconditioning methods that have been purposed include augmented Lagrangian, algebraic multigrid and incomplete LU-factorization [18, 19, 62, 89, 71].

In this work, we focus our attention on Schur complement preconditioners, a strategy that makes use of the physical structure of the Navier–Stokes equations. Schur complement preconditioners have been studied extensively, for example see [41, 40, 57, 75, 88]. Until recently, however, these methods have only been successfully applied to the constant density-viscosity form of the Naiver–Stokes equations.

In [25] a new pressure convection diffusion (PCD) Schur complement approximation for the variable density/viscosity Navier–Stokes equation is presented. While other work has examined variable viscosity problems in the Stokes context [29, 11, 51, 52, 68], this new PCD operator represents the first scalable Schur complement preconditioner designed specifically for the variable density/viscosity Naiver–Stokes equations.

This research extends the development of the variable density/viscosity PCD operator in several ways. First, numerical experiments are conducted using the RANS2P module of Proteus (`http://proteustoolkit.org`) – an industrial software package used to simulation free-surface fluid dynamics problems. As such, this work demonstrates that the the variable density/viscosity PCD operator provides a meaningful improvement relative to other Navier–Stokes preconditioners used in industrial applications. Second, numerical experiments are run for large-scale three dimensional problems on high-performance computers with thousands of computational nodes, demonstrating that the variable density/viscosity PCD preconditioner is effective in a large scale modern computing environment.

The first several sections of this chapter provide useful background material. Section 3.2 offers a brief overview of the continuous conservative level-set method used to solve dynamic two-phase flow problems in the RANS2P module. Section 3.3 highlights the discrete nonlinear Navier-Stokes equations that arise at each time step of the discrete level-set approach. In addition, this section describes the stabilization method used in the RANS2P model, the enforcement of boundary conditions, and the approach to linearizing the discrete Navier–Stokes equations. Section 3.4 discusses how the linear system of equations that arise from the discrete nonlinear Navier–Stokes equations are solved. This section includes a discussion of direct methods and the need for iterative Krylov methods to solve the large linear systems of equations that arise in Proteus. An overview of algebraic multigrid methods is also given, as these methods form an important tool in larger preconditioning strategies. Finally, we present the details of the current Additive Schwarz preconditioner used in Proteus and introduce the variable density-viscosity approach described in [25].

Finally, Section 3.5 presents numerical results of the variable density/viscosity PCD preconditioner used solve fluid dynamics problems in Proteus. In the first part of this section, results are presented for two static benchmark problems that appear frequently in the Navier–Stokes literature – a cavity problem and a channel flow with a step. Next, results are presented for a two-dimensional dynamic free-surface model that simulates a column of water collapsing under the force of gravity. Finally, the last experiment explores a three-dimensional dambreak simulation that uses several million unknowns and is solved with high-performance computing resources and thousands of processors.

3.2 The RANS2P Free-Surface Model

This section describes the weak formulation of the Navier–Stokes equations and the level-set method used in the Reynolds Averaged Two-Phase Navier–Stokes (RANS2P) module of the Proteus toolkit (`http://proteustoolkit.org`). For ease of exposition, we present the method in $\mathbb{R}^2$, but the method extends in a straight forward way to $\mathbb{R}^3$. Additionally, we use bold letters to denote vector quantities.

We begin with a brief overview of the RANS2P level set model. At each discrete time step, the RANS2P module first solves the Navier–Stokes equations to generate a fluid velocity profile across the simulation domain. This solution is then coupled with a level set model to track the interface between the air and water phases. In the discrete setting, however, this approach can lead to unacceptable mass conservation errors. Therefore, RANS2P also generates a mass conserving approximation of the volume fraction across the domain. Finally, the mass conserving volume fraction is coupled with the level set model to produce an adjustment to create a mass conserving level set function. A number of details are omitted in this discussion, but interested readers can find a complete description in [58].

3.2.1 Two Phase Domain

Consider a domain $\Omega \subset \mathbb{R}^2$ occupied by two immiscible fluids — air and water. Let Ω_w denote the segment of the domain containing water and Ω_a the segment of the domain containing air. $\Gamma := \overline{\Omega}_w \cap \overline{\Omega}_a$ describes the fluids' interface and $\Omega = \Omega_a \cup \Omega_b \cup \Gamma$. The non-interface boundaries of Ω_a and Ω_w are defined as $\partial\Omega_i = \overline{\overline{\Omega_i} \setminus (\Omega_i \cup \Gamma)}$ for $i = a, w$ and $\partial\Omega = \partial\Omega_a \cup \partial\Omega_w$.

The air and water phases of the domain can be described with a level set function ϕ

such that

$$\Omega_w = \{\mathbf{x} : \phi(\mathbf{x}, t) < 0\},\ \Omega_a = \{\mathbf{x} : \phi(\mathbf{x}, t) > 0\} \text{ and } \Gamma = \{\mathbf{x} : \phi(\mathbf{x}) = 0\}. \tag{3.1}$$

3.2.2 Navier–Stokes Equations

For a given time step, the RANS2P module first solves the two-phase Navier–Stokes equations. In this section, we present the continuous weak formulation of the Navier–Stokes equations. The discrete form of the equations is described in Section 3.3.

Consider a variable density and viscosity Navier-Stokes model for incompressible fluids in $[0 \times T] \times \Omega$. To describe the two-phase nature of the problem, let ρ_a, ρ_w and ν_a, ν_w denote the density and kinematic viscosity of the air and water respectively. We then define ρ, ν and μ as

$$\rho = \rho_a H(\phi) + \rho_w(1 - H(\phi))\ ,\ \nu = \nu_a H(\phi) + \nu_w(1 - H(\phi)), \tag{3.2}$$

$$\text{and } \mu = \rho_a \nu_a H(\phi) + \rho_w \nu_w (1 - H(\phi)), \tag{3.3}$$

where H is the Heaviside function

$$H(\phi) = \begin{cases} 1 \text{ if} & \phi > 0 \\ \frac{1}{2} \text{ if} & \phi = 0 \\ 0 \text{ if} & \phi < 0 \end{cases} \cdot \tag{3.4}$$

The dynamic viscosity Navier–Stokes equations are

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho\, \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nabla \cdot (2\mu \nabla^s \mathbf{u}) = \rho\, \mathbf{g} \text{ in } \Omega_w \cup \Omega_a, \tag{3.5}$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega_w \cup \Omega_a, \tag{3.6}$$

where $\mathbf{u} = \begin{pmatrix} u & v \end{pmatrix}^t$ is the velocity, p is the pressure, μ is the dynamic viscosity, ρ is the density, $\mathbf{g}$ is the gravitational acceleration and $\nabla^s \mathbf{u} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^t)$ is the symmetric gradient tensor.

Along the dynamic fluid interface, $\Gamma(t) = \overline{\Omega}_w \cap \overline{\Omega}_a$, we have

$$\mathbf{u}_w - \mathbf{u}_a = \mathbf{0}, \tag{3.7}$$

$$(\underline{\boldsymbol{\sigma}}_w - \underline{\boldsymbol{\sigma}}_a) \cdot \mathbf{n} = \mathbf{f}, \tag{3.8}$$

where $\mathbf{n}$ is the outward normal vector for the water phase, $\underline{\boldsymbol{\sigma}}_i = -p_i \mathbf{I} + 2\mu_i \nabla^s \mathbf{u}_i$ is the stress tensor for $i = a, w$, and $\mathbf{f}$ denotes the surface tension force between the fluids. Along the fluid interface, we use $\mathbf{u}_w$ to denote the velocity of the water phase and $\mathbf{u}_a$ to denote the velocity of the air phase. We assume

$$\mathbf{f} = \gamma \kappa \, \mathbf{n}. \tag{3.9}$$

where γ is the air-water surface tension coefficient and κ is the mean curvature of Γ. Boundary conditions for (3.5)-(3.6), can be either Dirichlet or Neumann, where

$$\mathbf{u} = \mathbf{w} \text{ on } \partial\Omega_D, \tag{3.10}$$

$$\mathbf{n} \cdot \underline{\boldsymbol{\sigma}} = \mathbf{h} \text{ on } \partial\Omega_N \tag{3.11}$$

and $\underline{\boldsymbol{\sigma}} \cdot \mathbf{n} = \begin{pmatrix} (\underline{\boldsymbol{\sigma}}_1) \cdot \mathbf{n} \\ (\underline{\boldsymbol{\sigma}}_2) \cdot \mathbf{n} \end{pmatrix}$.

To define the variational form, let $\mathbf{V}_T(0, T; \mathbf{V}(\Omega))$ denote an appropriate function space for the vector-quantity velocity across time and space, and $M_T(0, T; M(\Omega))$ denote an appropriate function space for the pressure across time and space.

For equations (3.5) and (3.6), a weak formulation requires finding $\mathbf{u} \in \mathbf{V}_T(0, T; \mathbf{V}(\Omega))$ and $p \in M_T(0, T; M(\Omega))$ such that for $t \in (0, T]$

$$\begin{aligned} \int_\Omega \rho \, \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \, d\Omega + \int_\Omega \rho \, (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} \, d\Omega &= -\int_\Omega \nabla p \cdot \mathbf{v} \, d\Omega - \int_\Omega (2\mu \nabla^s \mathbf{u}) \cdot \nabla \mathbf{v} \, d\Omega \\ &+ \int_\Omega \rho \, \mathbf{g} \cdot \mathbf{v} \, d\Omega + \int_{\partial\Omega} (2\mu \nabla^s \mathbf{u} \cdot \mathbf{n}) \cdot \mathbf{v} \, \partial\Omega + \gamma \int_\Gamma \kappa \, \mathbf{n}_\Gamma \cdot \mathbf{v} \, d\Gamma, \end{aligned} \tag{3.12}$$

$$\int_\Omega \mathbf{u} \cdot \nabla q \, d\Omega = -\int_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n}) \, q \, \partial\Omega, \tag{3.13}$$

for all $\mathbf{v} \in \mathbf{V}(\Omega)$ and $q \in M(\Omega)$. Note that we assume that the pressure is constant across Γ.

This weak form differs from the standard variational form of Navier-Stokes in that (3.13) represents an integration by parts resulting from (3.6) instead of integrating the pressure gradient ∇p term in the momentum equation.

3.2.3 Level set transport equation

Once the fluids' velocity $\mathbf{u}$ is solved, the level set function ϕ is updated via a transport equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \text{ in } \Omega \tag{3.14}$$

which implicitly describes the time evolution of the fluids' interface in Ω.

3.2.4 Redistancing

Once the level set has been updated, the Eikonal equation is used to redistance ϕ to ϕ_d so that

$$\|\nabla \phi_d\| = 1 \text{ in } \Omega, \tag{3.15}$$

$$\phi_d = 0 \text{ on } \Gamma. \tag{3.16}$$

Note that Γ is given by $\phi = 0$, where ϕ satisfies (3.1).

3.2.5 Volume fraction

Next, the linear scalar conservation of fluid mass equation

$$\frac{\partial \widehat{H}}{\partial t} + \nabla \cdot (\widehat{H}\mathbf{u}) = 0 \text{ in } \Omega \tag{3.17}$$

$$\widehat{H}(\mathbf{x}, t) = H(\phi) \text{ on } \partial\Omega \tag{3.18}$$

is solved for $\widehat{H}$ and the velocity $\mathbf{u}$ comes from the solution of the Navier–Stokes equations.

3.2.6 Mass correction

Finally, the volume fraction and signed distance functions are coupled through the nonlinear equation

$$\kappa\Delta\phi^{'} = H(\phi_d + \phi^{'}) - \widehat{H} \text{ in } \Omega \tag{3.19}$$

$$\nabla\phi^{'} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega \tag{3.20}$$

and solved for $\phi^{'}$ to enforce mass conservation on each subdomain of the triangulation. In the discrete setting, the mass conserving adjustment $\phi^{'}$ prevents the accumulation of temporal error that appears in the level set approximation. See [58] for more details.

3.3 Numerical Methods for Navier-Stokes

In this section we outline the discrete nonlinear Navier-Stokes equations that arises in the discrete level-set method. To establish the fluid phases, we assume there is a continuous level-set phase function ϕ (recall (3.1)) from the previous time step.

3.3.1 Discrete Navier-Stokes

Assume that the physical domain Ω and its boundary $\partial\Omega$ are fixed during $[0, T]$. Next, assume the time interval $[0, T]$ is partitioned into a sequence of subintervals $0 = t_0 < \cdots < t_n < t_{n+1} < \cdots < t_N = T$, with time intervals $\Delta t_n = t_n - t_{n-1}$. In practice, this partition is done dynamically during the simulation.

At each time step, the domain Ω is partitioned into an unstructured mesh $\mathcal{T}_h$ of N_e simplex elements K_i. Each element K_i has diameter h_i and boundary ∂K_i. The mesh $\mathcal{T}_h$ also partitions $\partial\Omega$ into n_e segments $\partial\Omega_i$. When calculating element or boundary integrals over the entire domain, we use the notation

$$\int_{\Omega'} = \sum_{i=1}^{N_e} \int_{K_i} \quad \text{and} \quad \int_{\partial\Omega'} = \sum_{i=1}^{n_e} \int_{\partial\Omega_i} . \tag{3.21}$$

For the mesh partitioning $\mathcal{T}_h$, we define our finite element spaces as continuous degree k polynomials of equal order

$$\mathbf{V}^h = \{\mathbf{u}^h \in \mathbf{V}(\Omega) \cap C^0(\Omega) : \mathbf{u}^h|_{K_i} \in (P^k(K_i))^2 \text{ for all } K_i \in \mathcal{T}_h\} \tag{3.22}$$

$$M^h = \{w^h \in M(\Omega) \cap C^0(\Omega) : w^h|_{K_i} \in P^k(K_i) \text{ for all } K_i \in \mathcal{T}_h\}. \tag{3.23}$$

Let $\{\boldsymbol{\phi}_i\}_{i=1}^{N_v}$ and $\{\psi_i\}_{i=1}^{N_p}$ denote bases for $\mathbf{V}^h$ and M^h respectively. When referring to the components of an element of $\{\boldsymbol{\phi}_i\}_{i=1}^{N_v}$, we use the notation $\boldsymbol{\phi}_i = \begin{pmatrix} \phi_i^u & \phi_i^v \end{pmatrix}^t$.
The RANS2P module allows the user to specify the polynomial order k for the velocity and pressure spaces. A frequently used element pair is $k = 1$ for the velocity and pressure (commonly referred to as P1-P1). This choice requires pressure stabilization terms, which are discussed in Section 3.3.2.
At each time step t_n, we use the discrete solution from the previous m time steps to approximate the temporal derivative, $\frac{\partial \mathbf{v}_n}{\partial t}$, with a backward differentiation formula (BDF)

$$\frac{\partial \mathbf{v}_n}{\partial t} \approx \mathbf{D}_t \mathbf{v}_n = \alpha \mathbf{v}_n + \sum_{i=1}^{m} \beta^i \mathbf{v}_{n-i} \tag{3.24}$$

where the subscript n indicates the time of the solution variable. That is, $\mathbf{u}_n$ indicates the discrete solution at time t_n. Also, note that for $m = 1$, $\alpha = \frac{1}{\Delta t_n}$, and $\beta^1 = \frac{-1}{\Delta t_n}$ (3.24) represents the backward Euler formula.
Using $\mathbf{V}^h$ and M^h, we can define the discrete approximation to the continuous prob-

lem (3.12) and (3.13) at time t_n as: find $\mathbf{u}_n^h \in \mathbf{V}^h$ and $p_n^h \in M^h$ such that

$$\begin{aligned}\int_\Omega \rho\, \mathbf{D}_t \mathbf{u}_n^h \cdot \mathbf{v}^h \, d\Omega - \int_\Omega \rho(\mathbf{u}_n^h \cdot \nabla \mathbf{u}_n^h) \cdot \mathbf{v}^h \, d\Omega = -\int_\Omega \nabla p_n^h \cdot \mathbf{v}^h \, d\Omega \\ - \int_\Omega (2\mu \nabla^s \mathbf{u}_n^h) : \nabla \mathbf{v}^h \, d\Omega + \int_\Omega \mathbf{g} \cdot \mathbf{v}^h \, d\Omega + \int_{\partial\Omega} (2\mu \nabla^s \mathbf{u}_n^h \cdot \mathbf{n}) \cdot \mathbf{v}^h \, \partial\Omega\end{aligned} \tag{3.25}$$

$$\int_\Omega \mathbf{u}_n^h \cdot \nabla q^h \, d\Omega = -\int_{\partial\Omega} (\mathbf{u}_n^h \cdot \mathbf{n})\, q^h \, \partial\Omega \tag{3.26}$$

for all $\mathbf{v}^h \in \mathbf{V}_h$ and $q^h \in M_h$. Implementation of the boundary conditions are discussed in Section 3.3.3.

3.3.2 Stabilization

It is well known that discrete formulations such as (3.25)-(3.26) often produce non-physical approximations unless the discretization parameters, Δt and h, are taken "sufficiently small." However, for high Reynolds' number flow taking the discretization parameters "sufficiently small" may lead to an approximation scheme that is not computationally tractable. Moreover, pressure stabilization is needed when finite element pairs are not inf-sup stable (e.g. P1-P1). In this section, we outline the stabilization terms used to address these issues in Proteus' RANS2P module. For more information on the stabilization methods discussed herein see [54, 33, 86].

RANS2P uses a variation of algebraic subgrid scale (ASGS) stabilization. ASGS stabilization uses weighted element integrals of the strong residual tested against an adjoint differential operator to stabilize (3.25)-(3.26). As described in [54], ASGS stabilization assumes that the subgrid component of the discrete solution (e.g. the part of the solution that is too fine to be captured by the mesh) can be determined analytically on each element using the strong residuals of the discrete solution, adjoint operators, and Green's functions. The resulting correction for the subgrid component of the solution can then be added to the discrete weak formulation to provide the ASGS stabilization. To illustrate, consider the strong residuals of equation (3.6)

$$r_p = \frac{\partial u_n^h}{\partial x} + \frac{\partial v_n^h}{\partial y} = \nabla \cdot \mathbf{u}_n^h. \tag{3.27}$$

Note that if there was a mass source, it would be part of this term. For the two components of the vector equation (3.5), the strong residuals are defined as

$$r_u = \rho\,\mathbf{D}_t u_n^h + \rho\,\mathbf{u}_{n-1}^h \cdot \nabla u_n^h + \frac{\partial p_n^h}{\partial x} - \nabla \cdot (\mu \nabla u_n^h) - \rho\, g_1, \tag{3.28}$$

$$r_v = \rho\,\mathbf{D}_t v_n^h + \rho\,\mathbf{u}_{n-1}^h \cdot \nabla v_n^h + \frac{\partial p_n^h}{\partial y} - \nabla \cdot (\mu \nabla v_n^h) - \rho\, g_2. \tag{3.29}$$

For linear elements, the diffusion terms in (3.28) and (3.29) are dropped because the second derivatives of linear functions vanish (we assume linear finite elements for the rest of this section). Also, (3.28) and (3.29) use different forms of the advection and diffusion operators from (3.5). Finally, the solution from the previous time step is used to calculate the advective velocity field in (3.28) and (3.29).
Next we define the adjoint operator $\mathcal{L}^*$. Rather than acting on the solution functions $\mathbf{u}^h$ and p^h, $\mathcal{L}^*$ acts on the test functions $\mathbf{v}^h$ and q^h. The components of $\mathcal{L}^*$ are

$$\begin{aligned} \mathcal{L}^*_{uu}\mathbf{v}^h &= \rho\,(-\mathbf{u}_{n-1}^h \cdot \nabla v_1^h + g_1 v_1^h), & \mathcal{L}^*_{vv}\mathbf{v}^h &= \rho\,(-\mathbf{u}_{n-1}^h \cdot \nabla v_2^h + g_2 v_2^h), \\ \mathcal{L}^*_{up} q^h &= -\frac{\partial q^h}{\partial x}, & \mathcal{L}^*_{vp} q^h &= -\frac{\partial q^h}{\partial y}, \\ \mathcal{L}^*_{pu}\mathbf{v}^h &= -\frac{\partial v_1^h}{\partial x}, & \mathcal{L}^*_{pv}\mathbf{v}^h &= -\frac{\partial v_2^h}{\partial y}. \end{aligned} \tag{3.30}$$

Each element $K_i \in \mathcal{T}_h$ also has the stabilization weighting terms

$$\tau_v(K_i) = \left(\frac{4\nu}{h_i^2} + \frac{2\|\mathbf{u}_{n-1}^h\|_2}{h_i} + |\alpha| \right)^{-1}, \tag{3.31}$$

$$\tau_p(K_i) = \rho \left(4\nu + 2\|\mathbf{u}_{n-1}^h\|_2 h_i + |\alpha| h_i^2 \right). \tag{3.32}$$

Recall that α is the coefficient from the BDF formula (3.24) and h_i is the diameter of element K_i.

Finally, the stabilization terms

$$\int_{\Omega'} \tau_v(K_i)\ (r_u\ \mathcal{L}^*_{uu} + r_v\ \mathcal{L}^*_{vv})\mathbf{v}_h\ dK_i + \int_{\Omega'} \tau_p(K_i)\ r_p\ (\mathcal{L}^*_{pu} + \mathcal{L}^*_{pv})\mathbf{v}_h\ dK_i, \tag{3.33}$$

are added to (3.25) and

$$\int_{\Omega'} \tau_v(K_i)\ (r_u\ \mathcal{L}^*_{up} + r_v\ \mathcal{L}^*_{vp})q_h\ dK_i, \tag{3.34}$$

is added to (3.26).

A separate numerical diffusion term is also used to account for discontinuity capturing. Specifically,

$$\int_{\Omega'} q^*\nabla\mathbf{u}^h : \nabla\mathbf{v}^h\ d\Omega \quad \text{where} \quad q^* = C_{dc}\sqrt{(r^i_u)^2 + (r^i_v)^2}\ h_i^2 \tag{3.35}$$

is added to the conservation of momentum equation (3.25), where $C_{dc} > 0$ is a constant and r^i_u and r^i_v denote the residuals (3.28) and (3.29) on the element K_i.

The formulas for τ_v, τ_p and q^* above are the simplest versions used in RANS2P. While the details are omitted, Proteus has more sophisticated methods for calculating these quantities that account for elements with wide aspect ratios. For more information on metric based approaches to calculating the τ_v, τ_p and q^* quantities see [55] and [80].

3.3.3 Proteus Boundary Conditions

In this section, we describe the implementation of boundary conditions in Proteus. As discussed in Section 3.2.2, to be fully specified, the Navier-Stokes equations require boundary conditions. In this work, we consider Dirichlet and Neumann boundary conditions where

$$\mathbf{u} = \mathbf{w} \text{ on } \partial\Omega_D, \text{ and } (-p\mathbf{I} + 2\mu\nabla^s\mathbf{u})\cdot\mathbf{n} = \mathbf{h} \text{ on } \partial\Omega_N. \tag{3.36}$$

The boundary conditions for most problems include a mixture of Dirichlet and Neumann boundary conditions (e.g., $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$). Pure Dirichlet problems (e.g., $\partial\Omega = \partial\Omega_D$) can also occur, in which case the pressure solution is only specified up to a constant.

First we explore how Proteus enforces Dirichlet boundary conditions strongly and weakly. While the strong enforcement of Dirichlet boundary conditions is easier to implement, in [15] it is shown that in the presence of unresolved boundary layers, weakly enforced Dirichlet conditions often produce better solution approximation properties.

After our discussion on the enforcement of Dirichlet boundary conditions, we conclude with a discussion on the advective flux condition that appears in the conservation of mass equation and the enforcement of the Neumann boundary condition.

In some cases, the enforcement of boundary conditions will vary depending on whether the boundary segment represents and inflow or an outflow region. When it is necessary to distinguish these regions we use

$$\begin{aligned}
\partial\Omega^+ &= \{\mathbf{x} \in \partial\Omega : \mathbf{u}\cdot\mathbf{n} < 0\} \\
\partial\Omega^0 &= \{\mathbf{x} \in \partial\Omega : \mathbf{u}\cdot\mathbf{n} = 0\} \\
\partial\Omega^- &= \{\mathbf{x} \in \partial\Omega : \mathbf{u}\cdot\mathbf{n} > 0\}
\end{aligned} \tag{3.37}$$

where $\partial\Omega^+$ is the inflow region, $\partial\Omega^0$ is the characteristic boundary and $\partial\Omega^-$ is the outflow region.

Strongly Enforced Dirichlet Conditions

When Dirichlet conditions are enforce strongly, it is equivalent to requiring that the velocity approximation come from the space

$$\mathbf{V}^h(\Omega) = \{\mathbf{u}^h \mid \mathbf{u} \in \mathbf{V}^h \ , \ \mathbf{u}^h = \mathbf{w} \text{ on } \partial\Omega_D\}. \tag{3.38}$$

To enforced this condition, the row corresponding to every Dirichlet boundary degree of freedom is replaced with zeros except in the column of the unknown, which is replaced with a one. The value of the right-hand-side vector is then set to the value of $\mathbf{w}$ at the point corresponding to that degree of freedom. This has the effect of setting the degree of freedom to the value specified by the boundary condition.

As an additional note, matrix entries in the column corresponding to the Dirichlet degree of freedom are not removed. This is because removing column values is an expensive operation for the compressed sparse row (CSR) sparse matrix representation. Therefore, using strong Dirichlet boundary conditions can introduce non-symmetry into the global matrix. Since the Navier–Stokes equations are not symmetric, however, this is not a concern in this context.

Weakly Enforced Dirichlet Boundary Conditions

Dirichlet Boundary Conditions

Rather than require the solution to satisfy certain boundary conditions strongly, weakly enforced Dirichlet boundary conditions use penalty terms in the finite element formulation (see [15] and [5] for more details). A drawback of this approach is that the weak formulation becomes more complicated and a penalty parameter must be chosen. However, in the case of under-resolved boundary layers it has been demonstrated that weakly enforced boundary conditions produce more accurate solution approximations than those from strong enforcement [15].

To enforce the Dirichlet boundary conditions weakly on the velocity, the following boundary integral terms are added to the weak formulation (3.25)-(3.26)

$$\int_{\partial\Omega_D} \gamma(\mathbf{u}^h - \mathbf{w}) \cdot \mathbf{v}^h \, ds \tag{3.39}$$

$$+ \int_{\partial\Omega_D \cap \partial\Omega^+} \left((2\mu \nabla^s \mathbf{v}^h \cdot \mathbf{n}) + (\mathbf{u} \cdot \mathbf{n}) \, \mathbf{v}^h \right) \cdot (\mathbf{w} - \mathbf{u}^h) \, ds \tag{3.40}$$

$$+ \int_{\partial\Omega_D \cap (\partial\Omega^+)^c} (2\mu \nabla^s \mathbf{v}^h \cdot \mathbf{n}) \cdot (\mathbf{w} - \mathbf{u}^h) \, ds \tag{3.41}$$

where $(\partial\Omega^+)^c = \partial\Omega^0 \cup \partial\Omega^-$ and $\gamma = \dfrac{C_b |2\mu|}{h}$ for a penalty constant $C_b > 0$. The first equation (3.39) can be viewed as a penalty term. Indeed, along the Dirichlet boundary, deviations of the solution $\mathbf{u}^h$ away from the boundary condition $\mathbf{w}$ are penalized.

Next we describe the adjoint diffusive flux term (i.e., $2\mu\nabla^s\mathbf{v}\cdot\mathbf{n}$) that appears in (3.40) and (3.41). This adjoint flux term is derived from the diffusive term that appears in the conservation of momentum equation (3.25). Applying integration by parts gives

$$\begin{aligned}\int_\Omega (2\mu\nabla^s\mathbf{u}^h) : \nabla\mathbf{v}^h \, d\Omega = &\int_{\partial\Omega_D} \mathbf{w}\cdot(2\mu\nabla\mathbf{v}^h\cdot\mathbf{n})\, ds \\ &+ \int_{\partial\Omega_N} \mathbf{u}^h\cdot(2\mu\nabla\mathbf{v}^h\cdot\mathbf{n})\, ds - \int_\Omega \mathbf{u}^h\cdot(\nabla^s\cdot(2\mu\nabla\mathbf{v}^h))\, d\Omega\end{aligned} \tag{3.42}$$

where we have set $\mathbf{u}^h = \mathbf{w}$ on the Dirichlet boundary. Applying integration by parts again, this time without substituting the Dirichlet boundary condition, gives

$$\begin{aligned}\int_\Omega \mathbf{u}^h\cdot(\nabla^s\cdot(2\mu\nabla\mathbf{v}^h))\, d\Omega = &\int_{\partial\Omega_D} \mathbf{u}^h\cdot(2\mu\nabla\mathbf{v}^h\cdot\mathbf{n})\, ds \\ &+ \int_{\partial\Omega_N} \mathbf{u}^h\cdot(2\mu\nabla\mathbf{v}^h\cdot\mathbf{n})\, ds - \int_\Omega (2\mu\nabla^s\mathbf{u}^h)\cdot\nabla\mathbf{v}^h\, d\Omega.\end{aligned} \tag{3.43}$$

Combining equations (3.42) and (3.43), we get the adjoint diffusive flux term along the entire Dirichlet boundary

$$0 = \int_{\partial\Omega_D} (\mathbf{w}-\mathbf{u}^h)\cdot(2\mu\nabla\mathbf{v}^h\cdot\mathbf{n})\, ds. \tag{3.44}$$

This integral is then separated into the inflow region and the outflow / characterisitic boundary to give the diffusion adjoint terms in (3.40) and (3.41).

The last term to consider is the advective flux term (i.e. $(\mathbf{u}\cdot\mathbf{n})\ \mathbf{v}^h)$) that appears on the inflow boundary in equation (3.40) . As constructed, the advection term from the conservation of momentum equation (3.25) is non-linear

$$\int_\Omega \rho\,(\mathbf{u}^h\cdot\nabla\mathbf{u}^h)\cdot\mathbf{v}^h\, d\Omega. \tag{3.45}$$

The approach to linearizing the discrete Navier-Stokes equations is described in more detail in Section 3.3.4. For our purpose, however, assume that the the first $\mathbf{u}^h$ that appears in (3.45) has been replaced with a known solenoidal vector field $\mathbf{a}$ (e.g., the solution approximation from a previous time step). Applying integration by parts,

$$\begin{aligned}\int_\Omega \rho(\mathbf{a}\cdot\nabla\mathbf{u}^h)\cdot\mathbf{v}^h\,d\Omega &= \int_\Omega \rho\,\left(a_1\frac{\partial u_1^h}{\partial x}v_1^h + a_2\frac{\partial u_1^h}{\partial y}v_1^h + a_1\frac{\partial u_2^h}{\partial x}v_2^h + a_2\frac{\partial u_2^h}{\partial y}v_2^h\right)\,d\Omega\\ &= \int_\Omega \rho\,(\mathbf{v}^h\,\mathbf{a}^t) : \nabla\mathbf{u}^h\,d\Omega\\ &= \int_{\partial\Omega}\rho\,(\mathbf{a}\cdot\mathbf{n})\,(\mathbf{u}^h\cdot\mathbf{v}^h)\,d\Omega - \int_\Omega \nabla\cdot(\rho\,\mathbf{v}^h\,\mathbf{a}^t)\cdot\mathbf{u}^h\,d\Omega.\end{aligned} \tag{3.46}$$

Following a procedure analogous to that used for the adjoint diffusion terms, we get the advective flux condition

$$0 = \int_{\partial\Omega_D}\rho\,(\mathbf{a}\cdot\mathbf{n})\cdot\mathbf{v}^h(\mathbf{w}-\mathbf{u}^h). \tag{3.47}$$

To illustrate why this advective flux penalty condition is only included along the Dirichlet inflow boundary (3.40), suppose we are interested in solving the homogenous advection equation

$$\mathbf{w}\cdot\nabla u = 0 \tag{3.48}$$

where $\mathbf{w}$ is a conservative vector field and u is a scalar function of several variables. Consider a characteristic curve

$$\mathbf{c}(s) = \begin{cases} x = x(s) \\ y = y(s)\end{cases} \quad \text{for } 0 \le s \le s_1 \tag{3.49}$$

associated with the vector field $\mathbf{w}$ (i.e., $\frac{d\mathbf{c}}{ds} = \mathbf{w}(\mathbf{c}(s))$). Then, using (3.48) the solution u satisfies

$$\frac{d}{ds}(u(\mathbf{c}(s))) = \frac{\partial u}{\partial x}\frac{dx}{ds} + \frac{\partial u}{\partial y}\frac{dy}{ds} = \frac{d\mathbf{c}}{ds} \cdot \nabla u = \mathbf{w} \cdot \nabla u = 0. \qquad (3.50)$$

In other words, the solution u is constant along the characteristic curve $\mathbf{c}(s)$. Suppose that $\mathbf{c}(s)$ begins at an inflow boundary Γ_{in}. Since the solution u is constant along the characteristic curve, $u(c(0)) = u(c(s_1))$, the inflow condition specifies exactly the outflow boundary condition.

In the presence of diffusion, an outflow boundary condition can be specified that is not consistent with the advection process. If, however, the equation is advection dominated, the solution may exhibit strange behavior at the outflow boundary. To avoid unnatural boundary layer solution profiles when weakly enforcing the velocity, we only incorporate the advective terms along the inflow boundary. For additional discussion of these issues see [40, 15].

Advective Flux Boundary Conditions

In Proteus, the first step in setting a Neumann type boundary condition is to specify a Dirichlet boundary condition on the pressure unknown. Care must be taken to specify this pressure condition correctly. For example, for an outflow boundary that includes air and water, the pressure value must be consistent with the hydrostatic pressure from gravity, as well as the varying densities of the fluid phases. Once an appropriate value for the pressure condition is established, the boundary condition can be enforced strongly or weakly, following analogous procedures described for the Dirichlet velocity conditions.

Once the pressure term has been specified, a diffusive flux condition must be set to enforce the Neumann condition

$$(-p\mathbf{I} + 2\mu\nabla^s\mathbf{u}) \cdot \mathbf{n} = \mathbf{h} \text{ on } \partial\Omega_N. \qquad (3.51)$$

In other words, if one wishes to enforce the normal stress $\mathbf{h}$ along the boundary with a specified pressure p^*, then the diffusive flux boundary condition must be specified as

$$2\mu\nabla\mathbf{u}^h \cdot \mathbf{n} = \mathbf{h} + p^*\mathbf{I} \cdot \mathbf{n} = \sigma_d. \tag{3.52}$$

Free Slip vs No Slip Boundary Conditions

In addition to weak and strong enforcement of boundary conditions, we also consider the difference between free slip and no slip. Both free slip and no slip boundary conditions occur on the characteristic boundary (i.e. $\mathbf{u} \cdot \mathbf{n} = 0$, recall (3.37)). In the case of free slip, no restriction is placed on the tangential flow component. For no slip boundary conditions, however, the tangential component is enforced to be zero (i.e. $\mathbf{u} \cdot \mathbf{t} = 0$).

3.3.4 Nonlinear Solver

The discrete two-phase Navier-Stokes problem at time t_n was outlined in Sections 3.3 - 3.3.3. Because of the advection term, the Naiver–Stokes equations are nonlinear and must be solved using a iterative method which we outline in this section.
Applying standard finite element techniques to (3.25), (3.26) together with the stabilization terms and boundary condition terms presented in Sections 3.3.2 and 3.3.3, results in a nonlinear system of equations $\mathbf{F}(\mathbf{x}) = 0$ at each time step t_n, where $\mathbf{x} = \begin{pmatrix} \mathbf{u} & p \end{pmatrix}^t$. The solution to this problem, is $\mathbf{x}^*$ satisfying

$$\mathbf{F}(\mathbf{x}^*) = \mathbf{0}, \tag{3.53}$$

that can be solved using a fixed point Newton iteration.
To approximate $\mathbf{x}^*$, we use the multivariate Newton's method. Recall that Newton's method is a fixed point iteration, that creates a sequence $\{\mathbf{x}_0, \mathbf{x}_1, \cdots\}$ of approximations to the true solution $\mathbf{x}^*$. Note that in Section 3.3.2, the subscript was used to identify the time step. In this section, subscripts are used to indicate the Newton

iteration. If it is necessary to refer to the solution from the previous time step, we use the notation $\mathbf{x}^*_{-1}$. Provided the initial guess $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$ and $\mathbf{F}$ is sufficiently regular, this sequence converges to $\mathbf{x}^*$ quadratically.

Newton's method follows from the linear approximation of the vector function $\mathbf{F}(\mathbf{x}) = \Big(f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_n(\mathbf{x})\Big)^t$ centered at a point $\mathbf{x}_k$

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{F}(\mathbf{x}_k) + D\mathbf{F}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \tag{3.54}$$

where $D\mathbf{F}(\mathbf{x}_k)$ is the Jacobian matrix with entries

$$[D\mathbf{F}(\mathbf{x}_k)]_{i,j} = \left[\frac{\partial f_i(\mathbf{x}_k)}{\partial x_j}\right]_{i,j}. \tag{3.55}$$

It follows that if $\mathbf{F}(\mathbf{x}_k) + D\mathbf{F}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{0}$, then $\mathbf{F}(\mathbf{x}_{k+1}) \approx \mathbf{0}$ where

$$\mathbf{x}_{k+1} = \mathbf{x}_k - D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k). \tag{3.56}$$

That is, each Newton iteration generates an updated solution approximation $\mathbf{x}_{k+1}$ using the previous iterate $\mathbf{x}_k$ and the solution update $-D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k) = \Delta\mathbf{x}_k$.

The initial guess for the Newton iteration $\mathbf{x}_0$, comes from the solution at the previous time step $\mathbf{x}^*_{-1}$, except for the initial time step $t_0 = 0$ where $\mathbf{x}_0 = \mathbf{0}$. Provided the time steps are sufficiently close together, these initial guesses will be in the Newton solver's radius of convergence. The Newton iteration will continue until the ℓ_2 norm of the solution update decreases below a small threshold such as 10^{-6}.

For most problems of interest, computing $D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k)$ using a direct method to find $\Delta\mathbf{x}_k$ is impractical. Instead, the sparse linear system $D\mathbf{F}(\mathbf{x}_k)\Delta\mathbf{x} = -\mathbf{F}(\mathbf{x}_k)$ is solved using an iterative Krylov technique like GMRES. It is the process of solving the linear system $D\mathbf{F}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{F}(\mathbf{x}_k)$ that constitutes the majority of the work for each Newton iteration (see Section 3.4). While the numerical values of the Jacobian matrix change at each Newton iteration, the linear system $D\mathbf{F}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{F}(\mathbf{x}_k)$, always has the block structure

$$D\mathbf{F}(\mathbf{x}_k)\Delta\mathbf{x}_k = \begin{pmatrix} \mathbf{A} & B^T \\ B & C \end{pmatrix} \Delta\mathbf{x}_k = -\mathbf{F}(\mathbf{x}_k), \tag{3.57}$$

where $D\mathbf{F}(\mathbf{x}_k)$ is a sparse matrix.

3.4 Linear Solvers and Preconditioners

In this section, we discuss techniques to solve the large, sparse linear systems of equations that arise from the discrete linearized Navier–Stokes equations (e.g., (3.57)). Specifically, we explore two distinct approaches to solving linear systems - direct LU-factorization based methods and iterative Krylov subspace methods. Assuming exact computational arithmetic, direct methods produce exact answers but require computationally expensive numerical algorithms. Iterative methods, in contrast, can produce computationally efficient and reliable solution approximations, but usually require well designed preconditioners.

Before delving into the details of different linear solver techniques, we first introduce some important terminology. To describe linear solver performance, we often refer to the complexity of an algorithm. Complexity describes how the computational effort required to solve a problem grows as the problem size increases. This idea is frequently described using $\mathcal{O}$ notation.

Definition 1. *Let $f(n)$ denote the number of floating point operations required to solve a linear system and let $g(n)$ denote a real valued function of n. We say that $f(n) = \mathcal{O}(g(n))$ as $n \to \infty$ if there exists integers n_0 and M such that*

$$|f(n)| \leq M\ g(n) \text{ for all } n \geq n_0. \tag{3.58}$$

Typically, $g(n)$ refers to a polynomial degree. For example, LU-factorization is a $\mathcal{O}(n^3)$ algorithm. This means that the number of floating point operations required to perform an LU decomposition to an $n \times n$ matrix can be bounded by a cubic polynomial. Another example are large sparse linear systems, which when solved using effectively preconditioned iterative methods, can exhibit $\mathcal{O}(n)$ complexity. Note that as the problem size increases, the work required to solve a system with direct solvers grows significantly faster than for effectively preconditioned iterative methods. Related to the algorithmic complexity of a linear solver is scalability. Indeed, when assessing linear solver performance, we consider two measures of scalability: (i) how a solver performs when a computational mesh is refined, and (ii) how a solver performs when the number of processors used to compute the solution increases.

The first notion of scalability arises when one solves the same problem to a higher level of precision. Generating an increasingly accurate solution requires solving a larger system of approximating equations. The change in simulation run time that results from running the problem on an increasing refined mesh is referred to as scaling with mesh size or scaling with h. Ideally, if the problem size doubles, then the amount of time it takes an algorithm to finish would double.

The second notion of scalability arises when more processors are used to solve a linear system of equations. This type of processor scaling has two flavors: strong and weak. Strong parallel scaling refers to the change in run time when the number of processors is increases but the problem size remains constant. Weak parallel scaling refers to the change in run time when a problem size grows but the work per processor remains constant. For example, if we double the number of unknowns but use twice as many processors, it would be ideal for the amount of time needed to complete the simulation to remain unchanged.

3.4.1 Sparse Direct Solvers

Direct solvers are a fundamental technique for solving linear systems of equations. Indeed, direct methods are more robust numerically than Krylov iterative algorithms [3] and thus offer a useful benchmark for comparison. Furthermore, for reasonable problem sizes, direct methods are often faster. Direct methods are also a useful tool for physics based preconditioners that utilize the block structure of a matrix and require solving a series of smaller localized problems. Finally, direct solvers form the

basis for the development of incomplete factorization based preconditioners [67].
While important, the drawbacks of direct linear solvers for large systems of equations are well understood. The algorithmic complexity of the standard LU factorization is $\mathcal{O}(n^3)$. As a result, the number of operations required to use a direct solver become prohibitively expensive as the size of a linear system is increased. Moreover, the LU-factorization of a sparse matrix usually generates dense matrices that require an infeasible amount of storage. Finally, direct solver algorithms are inherently serial in nature which makes it difficult to realize scaling advantages from distributing work across parallel computing resources.
Significant research has been dedicated to minimizing the bottlenecks that arise from direct solvers, particularly when the underlying matrix structure is sparse and distributed across multiple processors. When considering the linear systems that arise from the discretization of differential equations in three-dimensional geometries with problem size n, state-of-the-art direct methods typically require $\mathcal{O}(n^2)$ work and $\mathcal{O}(n^{\frac{4}{3}})$ memory [31]. This n^2 work requirement still becomes prohibitively expensive. The algorithms used to solve sparse systems of equations directly are typically more complicated than standard LU-factorization methods because fill-in must be limited as much as possible. Most sparse, distributed direct linear solver algorithms apply the following four steps:

1. the degrees-of-freedom are reordered into a structure that minimizes fill-in, or to arrange the system of unknowns in a more convenient manner, such as triangular dependent structure;

2. the nonzero structure of the factorization is established and the relevant data structures are initialized;

3. the LU factorization is computed;

4. the LU factorization is then used to calculate the solution.

Each step in the direct linear solver algorithm is very involved. Rather than provide an overview of the steps here, we refer the reader to [36] and the extensive references cited therein. A number of excellent software packages are designed to develop and implement state-of-the-art sparse matrix solvers that have parallel implementations (see [2, 65]). In this work, we use the direct sparse matrix solver SuperLU_Dist.

3.4.2 Krylov Solvers

The algorithmic complexity, memory limitations and serial nature of direct solvers limit their effectiveness in solving the large nonsymmetric sparse linear systems of equations that occur in large scale fluid problems. Thus, our goal is to improve performance using the iterative preconditioned GMRES method. In this section, we provide a brief survey of the preconditioned GMRES and flexible GMRES algorithms used in our work. Further details can be found in [78].
The k-th Krylov space of the linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$ is defined as

$$\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0) := \text{span}\{\mathbf{r}_0, \mathcal{A}\mathbf{r}_0, \cdots, \mathcal{A}^{k-1}\mathbf{r}_0\} \tag{3.59}$$

where $\mathbf{r}_0 = \mathbf{b} - \mathcal{A}\mathbf{x}_0$ for some initial guess $\mathbf{x}_0$. To create a numerically stable space with nice algebraic properties, the Arnoldi algorithm is used to construct an orthonormal basis $V_k = [\mathbf{v}_1, \cdots, \mathbf{v}_k]$ of $\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$. Typically, a modified Gram-Schmidt orthogonalization [87] is used to improve the stability of this orthogonalization step.
Starting with an initial guess $\mathbf{x}_0$ and the corresponding residual $\mathbf{r}_0 = \mathbf{b} - \mathcal{A}\mathbf{x}_0$, the k-th step of the GMRES iteration identifies the vector $\mathbf{x}_k$ in the space $\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$ that minimizes the residual $\mathbf{r}_k = \mathbf{b} - \mathcal{A}\mathbf{x}_k$. If the residual norm $\|\mathbf{r}_k\|_2$ is not sufficiently small, the process is repeated with the enlarged Krylov space $\mathcal{K}_{k+1}(\mathcal{A}, \mathbf{r}_0)$ until a desired residual tolerance is reached.

GMRES Complexity

At each GMRES iteration, a new orthonormal basis vector is added to the previous Krylov space. This requires saving all previous Krylov vectors and performing a Gram-Schmidt orthogonalization across a growing set of vectors. Thus, the GMRES algorithm will only scale if the number of GMRES iterations remains constant as the size of the approximating linear system increases.

Unfortunately, in most cases, as the size of the approximating linear system increases (i.e., the mesh is refined), the spectrum and condition number of $\mathcal{A}$ increase. This negatively impacts the GMRES performance which generally works best when the eigenvalues of $\mathcal{A}$ are clustered and bounded away from 0 [40]. Thus, applying GMRES directly rarely produces useful results, motivating the need for preconditioning.

Preconditioners can be applied to the GMRES algorithm from the right or left side (there is also a split variant that we do not consider here). In left preconditioning, the preconditioner $\mathcal{P}$ is applied from the left side of the equation, creating the equivalent system

$$\mathcal{P}^{-1}\mathcal{A}\mathbf{x} = \mathcal{P}^{-1}\mathbf{b}. \tag{3.60}$$

to which the GMRES algorithm is then applied.

Right preconditioning, meanwhile, is applied to the vector of unknowns $\mathbf{x}$ and has the algebraic form

$$\mathcal{A}\mathcal{P}^{-1}\mathbf{u} = \mathbf{b} \quad \text{where} \quad \mathcal{P}\mathbf{x} = \mathbf{u}. \tag{3.61}$$

Note that it is not necessary to compute $\mathcal{P}\mathbf{x}$ directly. Indeed, the initial residual can be computed using the initial approximation $\mathbf{x}_0$ (e.g., $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 = \mathbf{b} - A\mathcal{P}^{-1}\mathbf{u}_0$). Moreover, during the GMRES iteration, the Krylov basis is constructed from preconditioned residual vectors. That is,

$$\mathcal{K}_k := \operatorname{span}\{\mathbf{r}_0, \mathcal{A}\mathcal{P}^{-1}\mathbf{r}_0, \cdots, (\mathcal{A}\mathcal{P}^{-1})^{k-1}\mathbf{r}_0\}. \tag{3.62}$$

Thus, once the algorithm is complete, it returns the minimum vector $\mathbf{u}$. To obtain the true the solution approximation $\mathbf{x}$ we simply apply $\mathcal{P}^{-1}$ one more time (e.g., $\mathbf{x} = \mathcal{P}^{-1}\mathbf{u}$).

For two reasons we only consider right preconditioning in this work. First, right preconditioning is easily adapted to the flexible GMRES algorithm. This offers a big advantage when building a block preconditioner that uses nest Krylov solves as part of a global scheme. Second, right preconditioned GMRES minimizes over the true residual vector while left preconditioned GMRES minimizes over the preconditioned residual vector. As a result, right preconditioning allows for the direct comparison of residual performance across different preconditioners.

3.4.3 Algebraic Multigrid

Multigrid Basics

Since the 1980s, multigrid methods have been a fundamental tool for solving the linear systems of equations that occur when differential equations are discretized [85]. Multigrid methods can be used as stand alone solvers for linear systems of equations, or as effective preconditioners for iterative Kryolv subspace methods such as GMRES. Another common use of multigrid methods is to approximate solutions to the various subproblems that emerge from the block Schur complement preconditioner outlined in Section 3.4.4 [1].

Multigrid methods can be divided into two classes - geometric and algebraic methods. As the name suggests, geometric multigrid (GMG) methods are developed from the physical geometry (e.g., mesh) of the problem. In contrast, algebraic multigrid (AMG) methods only use information from the linear system of equations to develop a solver. As a result of their algebraic nature, AMG methods can be applied to problems defined on unstructured meshes. As unstructured meshes are used in many applications of interest, we focus on AMG methods.

The literature on multigrid methods is extensive and growing, so we do not attempt to provide a complete summary of these methods. Rather, the reader can refer to [93, 84, 85] and the extensive reference listed therein. In this section, we instead aim to provide a brief summary and mention some specific issues related to parallel implementation that relate to our current work.

Consider the linear system $A\mathbf{x} = \mathbf{b}$, and a solution approximation $\mathbf{x}_k$. The residual of this system is given by

$$\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k. \tag{3.63}$$

It has been observed that running a relaxation method like Jacobi or Gauss-Seidel will quickly smooth (i.e., force to zero) the high-frequency error of the residual vector $\mathbf{r}_k$, while removing the low frequency errors very slowly. The idea of multigrid methods is to move the residuals to a coarser mesh where the low-frequency errors become high-frequency errors relative to the coarser mesh and can be smoothed further. This process continues until a coarse enough mesh is reached that the linear system can be solved quickly with a direct method. The solution is then propagated back up the mesh hierarchy to form a solution on the original mesh.

There are many steps involved in this process. First, one must decide what type of smoother to apply. One must then specify the coarser meshes and define operators that can restrict the problem to the coarser meshes and interpolate these results back to the refined meshes. These steps can be broken into four main components: coarsening operators, interpolation operators, restriction operators and smoothers. We take a moment to highlight some key features of each of these ingredients relevant to our work.

Relaxation or smoothing step - the first step of the multigrid method is to smooth or relax the solution to dampen the high frequency part of the error, making it easier to approximate on a coarser mesh. A foundational result in multigrid convergence analysis shows that in order for a multigrid method to be effective, the smoothing operation must be selected in way that is consistent with the interpolation operator [85]. For GMG methods, the geometric nature of the coarsening typically defines the form of the interpolation operator. Thus, for difficult problems, a great deal of effort is needed to develop effective smoothers. For AMG methods, however, a simple smoothing operator is often used, while considerable effort is spent building an effective coarsening and interpolation strategy. Therefore, AMG methods often employ basic algorithms like a single pass of a Jacobi or Gauss-Seidel iteration. The

Gauss-Seidel approach, however, has the downside that the smoothing operation is inherently serial. Some parallel algorithms like hybrid and polynomial smoothers have been developed and shown promise as scalable smoothers [94, 12]. The default smoother option in the AMG library *hypre* [47] is a hybrid Gauss-Seidel smoother.

Coarsening operator - a major challenge for AMG methods is choosing an effective coarsening method. In this setting it is convenient to think of the coefficient matrix as describing a graph with the non-zero entries denoting the non-zero entires denoting weighted edges between nodes. Indeed, coarsening algorithms must select nodes that will reduce the problem size while also allowing the construction of an effective interpolator. Most coarsening algorithms consider strongly connected nodes, where one says i strong depends on j or j strongly influences i if

$$|a_{ij}| \geq \alpha \max_{k \neq i} |a_{ik}|$$

for some $0 < \alpha < 1$. Notably, the larger the α, the stronger the implied connection between two nodes.

Nodes that strongly influence a number of other points can then be selected as potential candidates to preserve on the coarser mesh. To further restrict the space of coarsening nodes, several heuristics are used (see [94]) to ensure these nodes will in turn have good interpolation properties while not including too many nodes for coarsening. Perhaps the most common of these approaches is the Ruge-Struben (RS) coarsening method.

While this general approach has proven effective, the algorithms are serial in nature and are of limited use for 3D problems. To address this, the PMIS and HMIS parallel algorithms have been proposed and shown to produce reasonably scalable results [38]. One downside of these methods, however, is that they generally do not ensure that the coarse mesh will support an effective interpolant.

Even though PMIS and HMIS methods can effectively reduce complexity, for many 3D problems, aggressive coarsening algorithms may be needed to maintain efficiency [84]. The concept underlying these methods is to weaken the definition of strongly connected nodes. Naturally, this additional reduction in complexity typically causes the AMG solver to be less effective.

Restriction operators - once the solution approximation has been smoothed, the residual must be transfered to a coarser mesh. These restriction operators typically use a weighted average of values on the finer mesh to approximate values on the coarser mesh. At this point, the discrete differential operator must also be transfered to the coarse mesh.

Interpolation or Prolongation operators - once a solution update has been computed on a coarse mesh, it must be be interpolated back to the fine mesh using the solution values from the coarse grid nodes. The most effective interpolating strategy will then depend on the coarsening strategy used. For serial coarsening strategies like the RS method, every fine grid point will be strongly connected with one of the nodes on the coarse mesh. Therefore a distance one interpolating strategy such as classical interpolation (or direct) is an appropriate choice [37].

For the parallel coarsening strategies like PMIS and HMIS or aggressive coarsening strategies, many nodes on the fine mesh will not strongly depend on a coarsen grid point. In this case, the interpolator will need to find a strongly connected point through an intermediary point. A class of algorithms known as long-range interpolation strategies have been designed for this purpose and includes methods like multipass interpolation, extended interpolation and extended-i interpolation [37].

When analyzing the effectiveness of a multigrid method, it is helpful to decompose the process into two stages: the setup stage and the solve stage. In the setup stage, the mesh coarsening is performed, and then the coarse matrices, interpolation and restriction operators are built. Here we need to consider two sources of complexity - operation complexity (the amount of memory needed to store the coarse information relative to the refined matrix) and stencil size (number of degrees of freedom per row in coarse matrices).

Once the setup phase has been completed, the multigrid method follows a sequence of recursive coarse grid corrections. Once the coarsest mesh has been reached, an LU decomposition is used to calculate the solution and the method then propagates the solution back to the finest mesh. As such, multigrid methods require making a trade-off between setup time and solution time. Typically, the more effort ones spends setting up the coarse mesh and its components the faster and more effective the solve phase will be.

In this study, we use the PETSc [13] wrapper for the external *hypre* [47] for all AMG calculations.

3.4.4 Preconditioning Strategy 2: Block Schur Complement Preconditioners

The linear systems of equations that arise in dynamic free-surface models for the two-phase Navier-Stokes equations have a saddle point structure (see Section 3.3.4). Block Schur complement preconditioners [40] are one popular strategy for preconditioning saddle point systems. To illustrate, consider a block LU factorization of the saddle point matrix $\mathbf{F}$,

$$\mathbf{F} = \begin{pmatrix} \mathbf{A} & B^T \\ B & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ B\mathbf{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathbf{A} & B^T \\ 0 & -S \end{pmatrix} = \mathcal{L}\,\mathcal{U} \tag{3.64}$$

where $S = B\mathbf{A}^{-1}B^T - C$.

Equation (3.64) implies that $\mathbf{F}\,\mathcal{U}^{-1} = \mathcal{L}$. Since $\mathcal{L}$ is a triangular matrix with ones along the diagonal, it has a single eigenvalue of one. This suggests that $\mathcal{U}$ maybe be a powerful preconditioner. Indeed, in [81] it is shown that GMRES converges in two iterations when

$$\mathcal{U}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{A}^{-1}B^T S^{-1} \\ 0 & -S^{-1} \end{pmatrix} \tag{3.65}$$

is used as a right preconditioner.

In practice, directly applying $\mathcal{U}^{-1}$ is not feasible because it requires knowledge of the dense matrices $\mathbf{A}^{-1}$ and S^{-1}. Instead, block Schur complement preconditioners approximate $\mathcal{U}^{-1}$ via

$$\mathcal{P}^{-1} = \begin{pmatrix} \widehat{\mathbf{A}}^{-1} & \widehat{\mathbf{A}}^{-1}B^T\widehat{S}^{-1} \\ 0 & -\widehat{S}^{-1} \end{pmatrix} \tag{3.66}$$

where $\widehat{\mathbf{A}}^{-1}$ and $\widehat{S}^{-1}$ approximate the action of $\mathbf{A}^{-1}$ and S^{-1} respectively. In the rest of this section, we describe several different approaches to approximating $\mathbf{A}$ and S. Figure 3-1 outlines the work flow required to apply (3.66).

3.4.5 Approximating A

The quality of the approximation of the operator $\widehat{\mathbf{A}}^{-1}$ has an important bearing on the quality of the preconditioner $\mathcal{P}^{-1}$. A great deal of research on developing effective saddle point preconditioners focus on the more challenging $\widehat{S}^{-1}$ problem. However, for large problems, a reliable scalable approximation to $\mathbf{A}^{-1}$ is equally important.
There are a number of ways one can approximate the action of $\mathbf{A}^{-1}$ including a distributed direct solver, the classical additive Schwarz method or a multigrid based approach. While each method has inherent advantages, the need to approximate $\mathbf{A}$ frequently and quickly for large problems suggests the multigrid approach offers the

Navier–Stokes Schur Complement Right Preconditioner Overview

Right preconditioned GMRES and FGMRES requires computing the preconditioned quantities $(v_{\mathbf{u}}^{\mathcal{P}}, v_p^{\mathcal{P}})$ from the vector $(v_{\mathbf{u}}, v_p)$. This requires computing

$$\mathcal{P}^{-1}\begin{pmatrix} v_{\mathbf{u}} \\ v_p \end{pmatrix} = \begin{pmatrix} \widehat{\mathbf{A}}^{-1} & \widehat{\mathbf{A}}^{-1}B^T\widehat{S}^{-1} \\ 0 & -\widehat{S}^{-1} \end{pmatrix}\begin{pmatrix} v_{\mathbf{u}} \\ v_p \end{pmatrix} = \begin{pmatrix} v_{\mathbf{u}}^{\mathcal{P}} \\ v_p^{\mathcal{P}} \end{pmatrix}. \tag{3.67}$$

The procedure for finding $v_{\mathbf{u}}^{\mathcal{P}}$ and $v_p^{\mathcal{P}}$ requires three steps.
First, **Apply** $\widehat{S}^{-1}v_p = -v_p^{\mathcal{P}}$.
Second, **Apply** $\widehat{\mathbf{A}}^{-1}v_{\mathbf{u}} = \mathbf{u}_1$ **and** $-\widehat{\mathbf{A}}^{-1}B^T v_p^{\mathcal{P}} = \mathbf{u}_2$.
Third, **Add** $\mathbf{u}_1 + \mathbf{u}_2 = v_{\mathbf{u}}^{\mathcal{P}}$.

Figure 3-1: Navier–Stokes Schur Complement Preconditioner Workflow

best choice.

The first option we considered was a distributed sparse direct solver (e.g. SuperLU_DIST [64] or MUMPS [2]) that applies $\mathbf{A}^{-1}$ exactly. By utilizing a sparse matrix structure and distributing work across multiple processors, this method is capable of achieving some parallel speed up. That said, the serial nature of direct solvers does limit the capacity for speed up. Furthermore, since the work involved with direct solvers grows cubicly with the number of unknowns, distributed sparse direct solvers cannot achieve scaling with the mesh size. Thus, while providing a simple and reliable option for small problems, the effectiveness of this approach is limited for large scale simulations.

Approximating $\mathbf{A}^{-1}$ in a scalable way requires using an iterative method. Regardless of the iterative method used, to achieve scaling with respect to mesh size, an effective preconditioner for $\mathbf{A}$ is necessary. Without an effective preconditioner, the iterative method will become less effective as the mesh is refined. It is also important that the preconditioner is scalable across multiple processors. To satisfy these requirements, we used a multigrid preconditioner.

A naive approach is to use an algebraic multigrid preconditioner directly on the entire system $\mathbf{A}$. As we shall see in the examples presented, however, this method typically does not scale well for advective dominated problems. Instead, the block structure of $\mathbf{A}$ can be used to develop a more effective preconditioner. Arranging the degrees of freedom of $\mathbf{A}$ in terms of velocity component (e.g., in two dimensions u and v) gives the block matrix

$$\mathbf{A} = \begin{pmatrix} A_{uu} & A_{uv} \\ A_{vu} & A_{vv} \end{pmatrix}. \tag{3.68}$$

Since the off diagonal blocks are a result of the symmetric gradient term, for advection dominated flows, the main features of the system is preserved even after these blocks are removed. Thus, the block diagonal matrix

$$\widehat{\mathbf{A}}^{-1} = \begin{pmatrix} A_{uu}^{-1} & 0 \\ 0 & A_{vv}^{-1} \end{pmatrix} \tag{3.69}$$

should serve as a quality preconditioner. Again, instead of applying the exact operators A_{uu}^{-1} and A_{vv}^{-1}, which would require a complete LU factorization, we consider approximating the actions of A_{uu}^{-1} and A_{vv}^{-1} using iterative methods. In the context of a GMRES iteration, applying this preconditioner requires solving the sub-problems

$$A_{uu}\hat{\mathbf{u}} = \mathbf{u} \quad \text{and} \quad A_{vv}\hat{\mathbf{v}} = \mathbf{v}. \tag{3.70}$$

To limit the work required to solve the entire system, the solutions to these local problems are approximated quickly with a fixed number of multigrid steps. Since these local problems have less global coupling, the goal is to achieve better scaling results than from applying the AMG method to the global system directly.

3.4.6 Approximating $\widehat{S}$

Recall that $\widehat{S}^{-1}$ is intended to approximate the action of the dense matrix $S^{-1} = (B\mathbf{A}^{-1}B^T - C)^{-1}$. Finding an effective approximation for S arising in Navier–Stokes equations is a difficult problem and has been extensively researched. The two-phase nature of the Navier–Stokes equations further complicates matters. In this section, we consider two methods that are appropriate for approximating the two-phase Schur complement S^{-1}: a two-phase version of the pressure convection-diffusion (PCD) operator [40], and the SIMPLE approximation [88].

$\widehat{S}_{PCD}$: The Pressure Convection Diffusion (PCD) Approximation

In this section, we present a two-phase pressure convection diffusion approximation (PCD) for the Schur complement, $\widehat{S}_{PCD}$. We begin with some notation and introduce a brief derivation of the well known single-phase operator. Following this discussion, we discuss how the single-phase PCD approximation can be extended to a variable density–viscosity setting.

Discrete Operators

Since the PCD operator is an approximation of the Schur complement operator S, it acts on the pressure space of the Navier–Stokes finite element formulation. As such, it is necessary to define several new finite element operators that act on the pressure space. If $\{\psi_i\}_{i=1}^{n_p}$ denotes a basis for the discrete pressure space, then discrete mass, Laplace and advection operators for the pressure can be defined as

$$Q^{(\gamma)}_{p;i,j} = \int_\Omega \gamma\, \psi_i\, \psi_j, \quad A^{(\gamma)}_{p;i,j} = \int_\Omega \gamma\, \nabla\psi_i \cdot \nabla\psi_j, \quad N^{(\gamma)}_{p;i,j} = \int_\Omega \gamma\, (\mathbf{w} \cdot \nabla\psi_j) \cdot \psi_i \tag{3.71}$$

where $\mathbf{w}$ is a velocity field and γ is a scalar function. Combining these terms, we can also define a discrete pressure advection-diffusion-reaction operator

$$F^{(\rho,\mu)}_{p;i,j} = \frac{\alpha}{\Delta t} Q^{(\rho)}_{p;i,j} + A^{(\mu)}_{p;i,j} + N^{(\rho)}_{p;i,j}. \tag{3.72}$$

Forms of operators in (3.71) and (3.72) appear in both the single-phase and variable density-viscosity cases. To distinguish between these two settings, omitted subscripts will denote single-phase flow, while including the superscript γ and ρ will indicate the variable density-viscosity setting.

The Commutator

One way of deriving the PCD operator is to assume that, in some sense, the advection-diffusion-reaction operators defined on the Navier–Stokes pressure and velocity spaces are commutative [40]. While this approach is heuristic, it does illustrates why the PCD operator provides a reasonable approximation to the Schur complement. For a rigorous derivation of the PCD approximation using Fourier analysis and Green's tensors see [57].

To motivate the connection between commuting differential operators and the Schur complement, recall that the Schur complement (stated here without stabilization) matrix has the form

$$S = B\mathbf{A}^{-1}B^T \tag{3.73}$$

where $B^T : M^h \to \mathbf{V}^h$, $\mathbf{A}^{-1} : \mathbf{V}^h \to \mathbf{V}^h$, and $B : \mathbf{V}^h \to M^h$. This highlights that the operator S maps pressure functions into the velocity space, applies an inverse discrete advection-diffusion-reaction operator and maps the result back to the discrete pressure space. Thus, one interpretation of S is an inverse advection-diffusion-reaction operator applied to pressure functions.

Therefore, consider the continuous convection-diffusion operator $\mathcal{L}$ defined on the velocity space and assume an analogous operator $\mathcal{L}_p$ exists on the pressure space. That is, for a viscosity μ and a velocity field $\mathbf{w}$,

$$\mathcal{L} := -\mu\nabla^2 + \mathbf{w}\cdot\nabla + \frac{\alpha}{\Delta t} \quad \text{and} \quad \mathcal{L}_p := (-\mu\nabla^2 + \mathbf{w}\cdot\nabla + \frac{\alpha}{\Delta t})_p \tag{3.74}$$

where $\alpha = 0$ for steady-state problems and $\alpha = 1$ for time dependent problems. Note that since the pressure space is often only $L^2(\Omega)$, the operator $\mathcal{L}_p$ may not be well defined, but as the argument is heuristic in nature, the point is safely ignored.

Using $\mathcal{L}$, $\mathcal{L}_p$ and the divergence operator $\mathcal{B}$, we define a commutator $\mathcal{E}$ acting on the velocity space

$$\mathcal{E} = \mathcal{B}\mathcal{L} - \mathcal{L}_p\mathcal{B}. \tag{3.75}$$

Assuming that $\mathcal{E}$ is *small* in some sense, (e.g. $\mathcal{B}\mathcal{L} - \mathcal{L}_p\mathcal{B} \approx 0$, implying that the action of advection-diffusion operators on the velocity and pressure space are similar) it follows that $\mathcal{B}\mathcal{L} \approx \mathcal{L}_p\mathcal{B}$.

To use this approximate relationship between the continuous pressure and velocity spaces, we must identify discrete analogues for the continuous operators $\mathcal{B}$, $\mathcal{L}$ and $\mathcal{L}_p$. As discussed in [40], the matrix representations of the these operators are

$$\mathcal{L} \sim \mathbf{Q}^{-1}\mathbf{A}, \quad \mathcal{B} \sim Q_p^{-1}B \text{ and } \mathcal{L}_p \sim Q_p^{-1}F_p \tag{3.76}$$

where $\mathbf{Q}$ represents the velocity mass matrix, $\mathbf{A}$ and B come from (3.64) and Q_p and F_p are defined in (3.71)-(3.72). For further details, see Appendix B.

Therefore, provided our commuting assumption holds, $\mathcal{B}\mathcal{L} \approx \mathcal{L}_p\mathcal{B}$ can be represented in matrix form as

$$(Q_p^{-1}B)(\mathbf{Q}^{-1}\mathbf{A}) \approx (Q_p^{-1}F_p)(Q_p^{-1}B). \tag{3.77}$$

Right multiplying (3.77) by $\mathbf{A}^{-1}B^T$ and left multiplying by $Q_p(Q_p^{-1}F_p)^{-1}$ yields

$$Q_pF_p^{-1}B\mathbf{Q}^{-1}B^T \approx B\mathbf{A}^{-1}B^T. \tag{3.78}$$

Rearranging suggests that

$$S^{-1} = (B\mathbf{A}^{-1}B^T)^{-1} \approx (Q_p\, F_p^{-1}B\mathbf{Q}^{-1}B^T)^{-1} = (B\mathbf{Q}^{-1}B^T)^{-1}F_p\, Q_p^{-1}. \tag{3.79}$$

While S takes the form $B\mathbf{A}^{-1}B^T$ in (3.79), this approximation for the Schur complement is also valid for stabilized finite element pairs in which $C \neq 0$ [40].

Single-Phase PCD Operator

In the single-phase setting, where the density ρ (for ease of exposition, let $\rho = 1$) and viscosity μ are constant functions in space, (3.79) becomes

$$\widehat{S}_{PCD}^{-1} = (B\mathbf{Q}^{-1}B^T)^{-1}F_p\, Q_p^{-1} \tag{3.80}$$

where A_p, F_p and Q_p are described in (3.71) and (3.72) (see [40]). From a practical perspective, (3.80) can be applied in a straight forward manner except for the $(B\mathbf{Q}^{-1}B^T)^{-1}$ term. To construct $(B\mathbf{Q}^{-1}B^T)^{-1}$, one needs to compute $\mathbf{Q}^{-1}$ (a dense matrix) and perform an expensive matrix-matrix product. To avoid this, $(B\mathbf{Q}^{-1}B^T)^{-1}$ can be replaced with the spectrally equivalent, sparse pressure Laplace operator A_p [40] so that

$$\widehat{S}_{PCD}^{-1} = A_p^{-1} F_p Q_p^{-1}. \tag{3.81}$$

Two-Phase PCD Operator

In the discrete two-phase setting, the density ρ and viscosity μ are non-constant piecewise continuous functions. As a result, it is not clear that the Laplace and mass operators that appear in the PCD operator (3.81) are correctly scaled for linear systems that arise from the variable density-viscosity problem. Indeed, the Fourier analysis used to derive the single-phase PCD operator in [57] is not valid in the two-phase setting because of the non-constant viscosity and density terms. Therefore, to establish effective scaling parameters for the two-phase PCD operator, it is helpful to view the two-phase NSE as a generalization of the two-phase Stokes problem.

To begin, consider that the inverse viscosity scaled mass matrix $(Q_p^{(1/\mu)})^{-1}$ is an effective Schur complement preconditioner for the steady-state variable-viscosity Stokes problem [74]. Based on this, it seems reasonable to replace Q_p^{-1} with $(Q_p^{(1/\mu)})^{-1}$ in (3.81) to establish a steady-state (e.g. $\alpha = 0$) version of the two-phase PCD operator

$$\begin{aligned} \widehat{\mathcal{S}}_{PCD}^{-1} &= A_p^{-1} F_p^{(\rho,\mu)} (Q^{(1/\mu)})^{-1} = A_p^{-1}(A_p^{(\mu)} + N_p^{(\rho)})(Q^{(1/\mu)})^{-1} \\ &= A_p^{-1} A_p^{(\mu)} (Q^{(1/\mu)})^{-1} + A_p^{-1} N_p^{(\rho)} (Q^{(1/\mu)})^{-1}. \end{aligned} \tag{3.82}$$

However, observe that in the case of a Stokes flow where $N_p^{(\rho)} = 0$, (3.82) becomes $A_p^{-1} A_p^{(\mu)} (Q^{(1/\mu)})^{-1} \neq (Q^{(1/\mu)})^{-1}$ as suggested in [74]. Therefore, it is natural to replace A_p in (3.82) with $A_p^{(\mu)}$ so that

$$\widehat{\mathcal{S}}_{PCD}^{-1} = (A_p^{(\mu)})^{-1} F_p^{(\rho,\mu)} (Q^{(1/\mu)})^{-1}. \tag{3.83}$$

While (3.83) represents a generalization of an established Stokes preconditioner, numerical results reveal that (3.83) does not scale effectively for general Navier–Stokes problems [25].

Therefore, to improve the scaling performance, we instead consider a Schur complement preconditioner designed specifically for the time-dependent Stokes problem (see [73])

$$\widehat{S}^{-1} = \frac{1}{\Delta t}(A_p^{(1/\rho)})^{-1} + (Q_p^{(1/\mu)})^{-1}. \tag{3.84}$$

This operator can be viewed as a generalization of the Cahouet-Chabard preconditioner developed specifically for the single-phase, time-dependent Stokes problem [30, 25]. A notable feature of this operator is the inverse density term that appears in the Laplace operator. To understand where this term comes from, note that in the original Cahouet-Charbard preconditioner, the density appears in the time dependent term as $\dfrac{\rho}{\Delta t}A_p^{-1} = \dfrac{1}{\Delta t}(\frac{1}{\rho}A_p)^{-1}$. Since the density is no longer constant, it is necessary to include the density function with the integral equations that form A_p.
To develop a two-phase PCD approximation that is a NSE generalization of (3.84), one must account for the different scaling terms that appear in the convection-diffusion operator $F_p^{(\rho,\mu)}$ [25]. Specifically, consider decomposing $F^{(\rho,\mu)}$ into terms that are viscosity scaled, $F_1^{(\mu)}$, and terms that are density scaled, $F_2^{(\rho)}$. That is,

$$F_p^{(\rho,\mu)} = A_p^{(\mu)} + (N_p^{(\rho)} + \frac{\alpha}{\Delta t}Q^{(\rho)}) = F_1^{(\mu)} + F_2^{(\rho)}. \tag{3.85}$$

Using this representation, the scaling of the terms $(A_p^{(\gamma)})^{-1}$ and $(Q^{(\gamma)})^{-1}$ can be selected to match either $F_1^{(\mu)}$ or $F_2^{(\rho)}$. Thus,

$$\begin{aligned}\widehat{S}_p^{-1} &= \left(A_p^{(\mu)}\right)^{-1} F_p^{(\mu)} \left(Q^{(\frac{1}{\mu})}\right)^{-1} + \left(A_p^{(\frac{1}{\rho})}\right)^{-1} F_p^{(\rho)} \left(Q^{(\rho)}\right)^{-1} \\ &= \left(Q^{(\frac{1}{\mu})}\right)^{-1} + \left(A_p^{(\frac{1}{\rho})}\right)^{-1} \left(N_p^{(\rho)} + \frac{\alpha}{\Delta t}Q^{(\rho)}\right) \left(Q^{(\rho)}\right)^{-1}\end{aligned} \tag{3.86}$$

since $F_1^{(\mu)} = A_p^{(\mu)}$. Note that when $N_p = 0$, (3.86) assumes the same form as the generalized Cahouet-Chabard preconditioner (3.84). Furthermore, as described in [25], numerical results illustrate that this form of the two-phase PCD approximation performs well in scaling experiments, suggesting that the method accurately captures the variable density and viscosity features of the flow.

In practice, taking $\rho = 1$ for $F_2^{(\rho)}$ and $Q^{(\rho)}$ so that

$$\widehat{S}_p^{-1} = \left(Q^{(\frac{1}{\mu})}\right)^{-1} + \left(A_p^{(\frac{1}{\rho})}\right)^{-1} \left(N_p^{(1)} + \frac{\alpha}{\Delta t} Q^{(1)}\right) \left(Q^{(1)}\right)^{-1} \tag{3.87}$$

produces results that are more stable than (3.86).

SIMPLE

The SIMPLE preconditioner [88] has the form

$$\widehat{S}_p^{-1} = (-C + B\ (\text{diag}(\mathbf{A}))^{-1}\ B^T)^{-1}, \tag{3.88}$$

where C, B, B^T and $\mathbf{A}$ are as given in (3.64).
If we suppose $C = 0$, then $\widehat{S}_p$ acts as a pressure Laplacian operator for small Δt. To see why, recall that $\mathbf{A} = \dfrac{1}{\Delta t}\mathbf{Q} + A + N$ where $\mathbf{Q}$ is the discrete mass operator, A is the discrete velocity Laplace operator and N is the discrete advection operator. Indeed, for sufficiently small Δt, $\mathbf{Q}$ is the dominant term in $\mathbf{A}$. Since B^T and B represent the discrete gradient and divergence operators,

$$B\ (\text{diag}(\mathbf{A}))^{-1}\ B^T \sim \nabla \cdot \left(\frac{I}{\Delta t}\right) \nabla = \frac{1}{\Delta t}\Delta. \tag{3.89}$$

The SIMPLE preconditioner offers several advantages over the PCD operator. First, since it's components are drawn directly from the original linear system, it does not require special treatment between the single and two phase settings. Moreover, stabilization and boundary conditions are automatically incorporated into the preconditioner's construction.
The SIMPLE precondition, however, has several important drawbacks. First, much of the information related to the flow's advective component is lost when everything but the diagonal component is dropped from $\mathbf{A}$. Furthermore, calculating $B(\text{diag}(\mathbf{A}))^{-1}B^T$ explicitly becomes prohibitively expensive for large scale problems.

3.5 Numerical Results

3.5.1 Introduction

In this section, we analyze preconditioner performance for several numeric simulations. In the first set of experiments, we consider the two-dimensional steady-state, lid driven cavity and backward facing step problems [40] modified to replicate a two-phase flow problem. These test problems are used to analyze the performance of the SIMPLE and two-phase PCD preconditioners. Specifically, these experiments allow us to verify our Schur complement approximations are implemented correctly and to assess the effects of stabilization, time-stepping, and boundary condition configurations.

In the second set of experiments, we consider two dynamic free-surface models. The first simulation is a two-dimensional dambreak problem. In this simulation, we compare the efficiency of the two-phase PCD and SIMPLE preconditioners in a dynamic setting while keeping simple boundary conditions and stabilization. The second simulation is the Marin problem, a three-dimensional model that must be run using high-performance computing resources. In this problem, we explore the effectiveness of different preconditioning strategies to scale across thousands of computational cores.

3.5.2 Static Preconditioner Analysis

In this section, we analyze serial preconditioner performance for several static test problems. Specifically, we are interested in studying the GMRES iteration scaling behavior of the two-phase PCD Schur complement approximation (see Section 3.4.6) and the $\widehat{\mathbf{A}}^{-1}$ approximation (see Section 3.4.5). For our analysis, we consider modified versions of two benchmark simulations that appear frequently in the Navier-Stokes literature - the lid driven cavity and the backward facing step problems [40].

Lid Driven Cavity

The lid driven cavity is a standard benchmark problem that appears throughout the Navier-Stokes literature [40]. This simulation allows us to study preconditioner performance independent of the effects of inflow and outflow boundary conditions. Moreover, since the problem has pure Dirichlet conditions, the lid driven cavity problem also allows us to study preconditioner performance when a constant pressure null space exists.

The simulation domain is a closed two-dimensional tank spanning $[-1, 1] \times [-1, 1]$. The bottom and side boundaries of the tank enforce no-flow boundary conditions. Flow within the tank is generated by applying the Dirichlet velocity condition

$$\mathbf{u} = \begin{pmatrix} 1 - x^2 \\ 0 \end{pmatrix}$$

across the top of the tank.

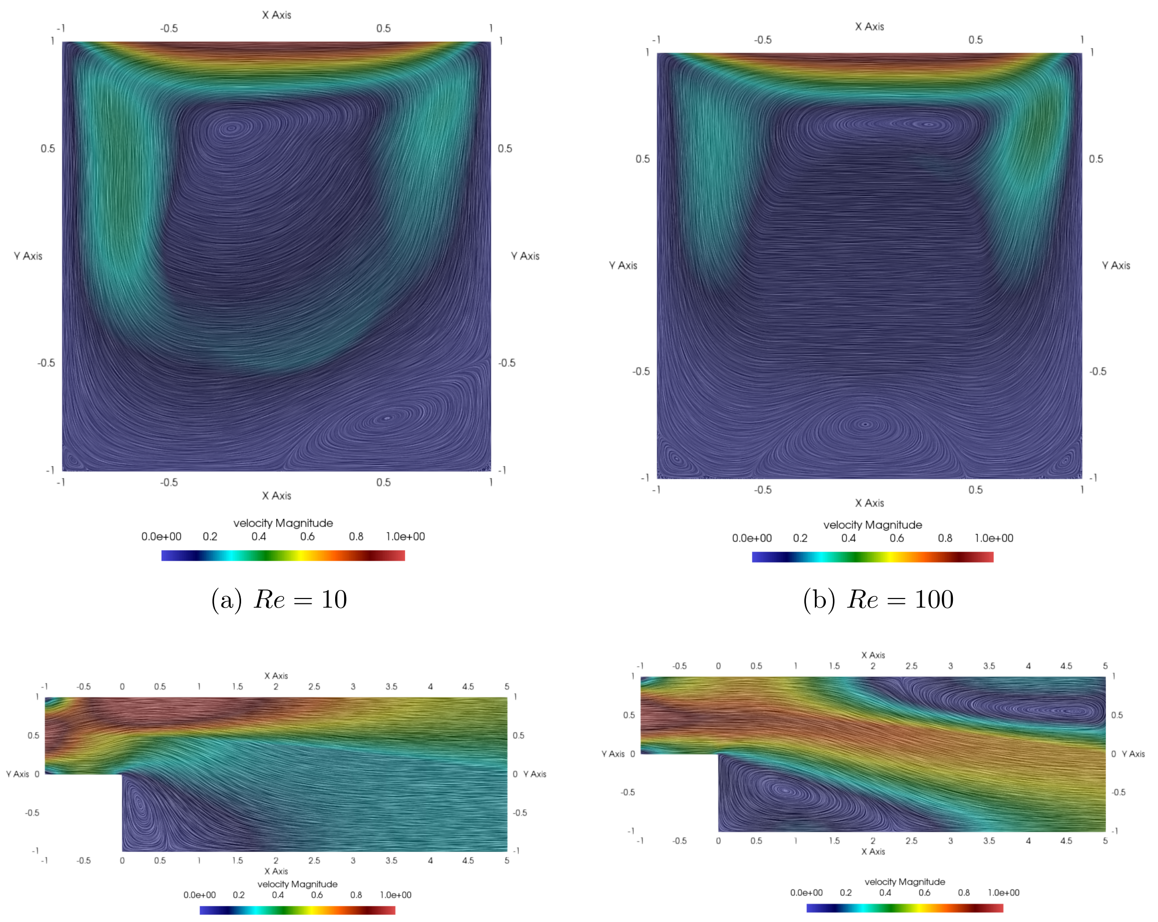
To mimic a two-phase flow, we artificially create a phase change along the interface

$$x^2 + y^2 = \frac{1}{4}. \tag{3.90}$$

The ratio of the density $\frac{\rho_1}{\rho_2}$, is taken to be 1.2×10^{-3} and for the viscosity $\frac{\mu_1}{\mu_2}$ is 1.8×10^{-2}, which reflects the ratios between air and water. Graphics of the lid driven cavity simulation for different Reynolds numbers are shown in Figure 3-2.

Step Problem

Another popular benchmark problem for the Navier-Stokes equation is the step problem which is a modified channel flow in which the domain has a distinct L-shape [40]. As shown in Figure 3-2, the channel spans the x-values $[-1, 5]$, and doubles in width at $x = 0$. This simulation is a useful benchmark since it allows us to study preconditioner performance in the presence of inflow and outflow boundary conditions.



(a) $Re = 10$ (b) $Re = 100$

(c) $Re = 10$ (d) $Re = 100$

Figure 3-2: Two-phase lid driven cavity and step problems

To initiate a left to right flow, the Dirichlet boundary condition

$$\mathbf{u} = \begin{pmatrix} 4y(1-y) \\ 0 \end{pmatrix} \tag{3.91}$$

is specified for the velocity at the inflow boundary $x = -1$.

As with the lid driven cavity problem, we modify the single-phase step problem in a nonphysical way to mimic a two-phase problem. Specifically, the top part of the tank is set to be air and the bottom part of the tank water where the curve

$$y = \frac{1}{2} - \frac{1}{72}(x+1)^2 \tag{3.92}$$

defines the boundary between the two-phases. Examples of the two-phase problem for Reynolds number 10 and 100 are shown in Figure 3-2.

Schur Complement Analysis

In this section we analyze the convergence behavior of the two-phase PCD Schur complement approximation (see Section 3.4.6) for the cavity and step problems. In particular, we assess steady state scaling performance in terms of GMRES iterations for different boundary conditions and pressure stabilizations. In addition, we compare the two-phase PCD Schur complement method to the SIMPLE method (see Section 3.4.6). While the two-phase PCD method should significantly outperform the SIMPLE method in the steady state case, we assess how competitive the SIMPLE method becomes when short time steps are used.

To keep our analysis focused on the Schur complement approximation, we use a direct solver for the **A**-block and apply the action of the two-phase pressure Laplacian operator that appears in the two-phase PCD operator using a single BoomerAMG V-cycle with default settings. Further, the GMRES iteration is performed using a right preconditioner with a relative residual tolerance of 10^{-6}.

Boundary Conditions

The GMRES iteration counts for the two-phase cavity and step problems with different boundary conditions and Reynolds numbers are shown in Table 3.1 and Tables 3.2-3.3, respectively. For a description of the different boundary conditions described in the Tables, the reader is referred to Section 3.3.3.

These results show that for strongly enforced boundary conditions, the two-phase PCD preconditioner scales independently of mesh size. Indeed, Table 3.1 report a constant number of GMRES iterations as the mesh is refined whether using free slip or no slip boundary conditions. This is a promising indication that the boundary modifications outlined in Section 3.4.6 are adequately capturing the boundary condition dynamics.

The results using weakly enforced boundary conditions are not so promising. As high-

	Reynolds Number = 10		Reynolds Number = 100	
h	*No Slip*	*Free Slip*	*No Slip*	*Free Slip*
0.4	25 / 26 (5)	23 / 24 (5)	29 / 31 (10)	28 / 29 (11)
0.2	27 / 29 (4)	24 / 26 (4)	30 / 33 (7)	31 / 33 (11)
0.1	26 / 28 (4)	23 / 25 (4)	29 / 33 (6)	30 / 32 (10)
0.05	26 / 28 (4)	22 / 24 (4)	29 / 33 (6)	30 / 33 (7)
0.025	25 / 27 (4)	22 / 24 (4)	29 / 33 (6)	30 / 33 (6)
0.0125	26 / 28 (4)	22 / 25 (4)	30 / 33 (6)	29 / 34 (5)
0.00625	24 / 27 (3)	22 / 24 (4)	30 / 33 (6)	28 / 33 (4)

Table 3.1: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the steady two-phase lid driven cavity problem with varying mesh size h.

	Strongly Enforced		**Weakly Enforced**	
h	*No Slip*	*Free Slip*	*No Slip*	*Free Slip*
0.4	31 / 31 (4)	25 / 25 (4)	30 / 31 (4)	36 / 37 (4)
0.2	31 / 32 (4)	23 / 23 (4)	31 / 33 (4)	46 / 48 (4)
0.1	32 / 33 (4)	24 / 25 (4)	32 / 34 (4)	60 / 63 (4)
0.05	32 / 33 (4)	24 / 25 (4)	32 / 36 (4)	79 / 84 (4)
0.025	32 / 34 (4)	23 / 25 (4)	32 / 35 (4)	98 / 105 (4)
0.0125	32 / 35 (4)	24 / 26 (4)	32 / 35 (4)	124 / 135 (4)

Table 3.2: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ solution to the backward facing step problem with $Re = 10$ and varying mesh size h.

lighted in Tables 3.2 and 3.3, the two-phase PCD preconditioner does not demonstrate mesh independent scaling when using weakly enforced free slip boundary conditions with inflow and outflow boundary conditions. This suggests that the adjustment for inflow and outflow boundary conditions considered in Section 3.4.6 is failing to account for the boundary integrals outlined in equations (3.39), (3.40) and (3.41).

A few other trends emerge from Tables 3.1-3.3 that merit comment. As expected, more GMRES iterations are required to achieve convergence when the Reynolds number is increased from 10 to 100. It is also worth noting that the number of GMRES iterations are typically lower for free-slip boundary conditions than no-slip. This is not too surprising given that the free-slip boundary conditions enforce less stringent requirements on the solution.

	Strongly Enforced		**Weakly Enforced**	
h	*No Slip*	*Free Slip*	*No Slip*	*Free Slip*
0.4	41 / 44 (6)	38 / 41 (6)	44 / 51 (5)	58 / 66 (6)
0.2	45 / 51 (6)	36 / 42 (6)	49 / 55 (6)	63 / 72 (6)
0.1	44 / 51 (6)	36 / 43 (6)	49 / 56 (6)	76 / 83 (6)
0.05	40 / 45 (6)	34 / 40 (6)	45 / 50 (6)	96 / 105 (6)
0.025	38 / 43 (6)	32 / 38 (6)	42 / 47 (6)	123 / 138 (6)
0.0125	38 / 42 (6)	32 / 38 (6)	42 / 46 (6)	159 / 179 (6)

Table 3.3: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ solution to the backward facing step problem with $Re = 100$ and varying mesh size h.

Stabilization

Next we explore the effects of stabilization on the two-phase PCD preconditioner. For problems of interest, we use the full Proteus subgrid error (SGE) stabilization outlined in Section 3.3.2. To identify potential issues in the application problems, however, it is useful to see how the stabilization method affects preconditioner performance. Ideally, we would like to compare the SGE stabilization with a non-stabilized method. However, since the full SGE approach stabilizes for the advective and pressure terms simultaneously, and we are using $\mathbf{P}_1 - P_1$ finite elements, a pressure stabilization method is needed. Thus, as a basis for comparison, we consider the parameter free pressure projection stabilization method described by Bochev et al. in [22].
GMRES iterations for the two stabilization methods in the lid-cavity and step problem are shown in Tables 3.4 and 3.5, respectively. These results are encouraging and suggest that for both problems, the GMRES iterations are scaling independent of the mesh size for both stabilization techniques. It should be noted, however, that the pressure projection approach does result in fewer iterations than the full SGE approach. Furthermore, it is interesting that this difference in the number of iterations is significantly larger for the step problem than the cavity problem.

Comparison with SIMPLE

To gauge the effectiveness of the two-phase PCD Schur complement approximation, it is helpful to compare its performance with another method. While research has been

	Reynolds Number = 10		Reynolds Number = 100	
h	*Pressure Projection*	*Full SGE*	*Pressure Projection*	*Full SGE*
0.4	25 / 26 (5)	25 / 26 (5)	25 / 27 (5)	29 / 31 (10)
0.2	24 / 26 (3)	27 / 29 (4)	27 / 28 (5)	29 / 33 (7)
0.1	24 / 26 (3)	26 / 28 (4)	28 / 30 (6)	29 / 33 (6)
0.05	24 / 26 (4)	26 / 28 (4)	29 / 31 (6)	29 / 33 (6)
0.025	24 / 26 (4)	25 / 27 (4)	29 / 31 (6)	29 / 33 (6)
0.0125	22 / 25 (3)	26 / 28 (4)	28 / 31 (6)	30 / 33 (6)
0.00625	22 / 25 (3)	26 / 28 (4)	28 / 31 (6)	30 / 33 (6)

Table 3.4: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the steady two-phase lid driven cavity problem with varying mesh size h.

	Reynolds Number = 10		Reynolds Number = 100	
h	*Pressure Projection*	*Full SGE*	*Pressure Projection*	*Full SGE*
0.4	31 / 31 (4)	34 / 35 (5)	41 / 44 (6)	44 / 46 (11)
0.2	31 / 32 (4)	37 / 38 (5)	45 / 51 (6)	52 / 57 (9)
0.1	32 / 33 (4)	40 / 42 (5)	44 / 51 (6)	57 / 68 (8)
0.05	32 / 33 (4)	37 / 39 (4)	40 / 45 (6)	55 / 67 (7)
0.025	32 / 34 (4)	37 / 40 (4)	38 / 43 (6)	56 / 68 (6)
0.0125	32 / 35 (4)	37 / 40 (4)	38 / 42 (6)	55 / 68 (6)

Table 3.5: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the steady two-phase step problem with varying mesh size h.

done to extend Schur complement approximations from single-phase to multi-phase problems, these efforts have focused on the Stokes problem (e.g. see [51, 74, 29, 68]) and are not applicable in the Navier–Stokes setting. Therefore, as a basis of comparison, we consider the SIMPLE Schur complement approximation introduced in Section 3.4.6. Indeed, the algebraic nature of the SIMPLE method captures some limited advective features of the flow and automatically incorporates the variable density and viscosity of the two-phase model.

As shown in Table 3.6, across a range of Reynolds numbers, the SIMPLE approximation shows poor scaling for the steady state two-phase lid-driven cavity problem, while the two-phase PCD approximation demonstrates reliable scaling performance. This result is not surprising because the diagonal block approximation of the $\mathbf{A}$ matrix used in the SIMPLE approximation (recall equation (3.88)) does not capture many of the important advective features of the system.

In applications of interest, however, the Navier–Stokes equations include a time-dependent term. While the cavity and step problems are steady state, we can add a $\frac{1}{\Delta t}$ scaled mass matrix to the velocity-velocity block to mimic the behavior of a time-dependent problem. This modification allows us to examine the performance of the SIMPLE and two-phase PCD preconditioners for different values of $\frac{1}{\Delta t}$.

Results for the step simulation that include the $\frac{1}{\Delta t}$ scaled mass matrix are presented in Table 3.7. For both the PCD and the SIMPLE approximations, the number of GMRES iterations decreases as Δt decreases. This is behavior is to be expected because a smaller Δt increases the magnitude of the added mass matrix. Moreover, since the mass matrix is spectrally equivalent to the identity matrix, increasing the magnitude of the mass matrix in the global matrix will improve the condition number of the system. In turn, an improved condition number typically corresponds to improved convergence performance of Krylov iterative methods.

While the SIMPLE and PCD approximations both exhibit improved performance as Δt decreases, the performance improvement is significantly larger for the SIMPLE method. Indeed, for $\Delta t = 1$, the SIMPLE approximation takes roughly 100 iterations for Reynolds number of 10 and 100, while the PCD approximation takes fewer than

30. However, when $\Delta t = 10^{-5}$, the SIMPLE and two-phase PCD approximations both require 10 or fewer GMRES iterations. This suggests that for time dependent problems in which a small enough step size is used, the SIMPLE approximation may provide a competitive alternative to the PCD approximation.

The results for the driven cavity simulation that include the $\frac{1}{\Delta t}$ scale mass matrix are presented in Table 3.8 and 3.9. These results highlight a different aspect of the SIMPLE and two-phase PCD approximations that need to be considers. As in the step problem, the results for the two-phase PCD method (Table 3.8) suggest that as a shorter time step are taken, the two-phase PCD operator accounts for the effect of the mass operator and solves the system in fewer GMRES iterations. This, again, is expected as the increased presence of the mass operator improves the conditioning of the linear system.

For the cavity problem, the behavior of the SIMPLE method is more complicated than in the step problem. To illustrate, Table 3.9 provides results using two different *hypre* BoomerAMG configurations to apply the SIMPLE approximation. Both configurations use the default *hypre* settings, with different strong threshold parameters (see Section 3.4.3). Specifically, configuration one uses $\alpha = 0.25$ and configuration two uses $\alpha = 0$. Recall that a lower α value implies that fewer entries are discarded in the AMG setup phase. Therefore, a smaller α value results in a more accurate V-cycle but requires a more expensive setup.

As seen in Table 3.9, both configurations exhibit a downward trend in iterations as shorter time steps are taken. However, when configuration one is used, several simulations experience a solver failure. The reason for this instability appears related to the accuracy of the AMG V-cycle. Indeed, when the simulations are run using configuration two, no solver failures are encountered. This suggests that in some contexts, a very accurate *hypre* BoomerAMG configuration must be used to ensure that the SIMPLE preconditioner produces stable results.

h	Re				
	10	$10^{1.5}$	100	$10^{2.5}$	1000
$\frac{1}{16}$	25 / 18	27 / 18	29 / 23	36 / 26	49 / 25
$\frac{1}{32}$	24 / 38	26 / 23	29 / 32	35 / 36	47 / 39
$\frac{1}{64}$	25 / 56	27 / 55	29 / 56	35 / 45	46 / 56
$\frac{1}{128}$	24 / 53	27 / 85	29 / 77	34 / 83	46 / 84
$\frac{1}{256}$	24 / 116	27 / 116	30 / 105	35 / 112	47 / 115

Table 3.6: Preconditioned GMRES iterations using two-phase PCD / SIMPLE for the $\boldsymbol{P}_1 – \boldsymbol{P}_1$ solution to the steady lid driven cavity problem with density ratio $\widehat{\rho} = 1.2 \times 10^{-3}$, viscosity ratio $\widehat{\mu} = 1.8 \times 10^{-2}$ (values for air-water flow), and varying Reynolds number Re and grid size h.

	Reynolds Number = 10		**Reynolds Number = 100**	
Δt	PCD	SIMPLE	PCD	SIMPLE
10^0	24 / 27 (4)	105 / 121 (4)	26 / 30 (5)	96 / 110 (5)
10^{-1}	19 / 21 (4)	101 / 116 (4)	22 / 24 (5)	79 / 88 (5)
10^{-2}	17 / 19 (4)	80 / 89 (4)	19 / 20 (5)	66 / 71 (5)
10^{-3}	15 / 16 (4)	66 / 70 (4)	15 / 16 (5)	26 / 28 (5)
10^{-4}	13 / 14 (4)	25 / 26 (4)	10 / 10 (4)	9 / 10 (4)
10^{-5}	9 / 10 (4)	10 / 10 (4)	6 / 6 (4)	6 / 7 (4)

Table 3.7: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using the two-phase PCD and SIMPLE preconditioners for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the step problem with $h = 0.0125$ and varying Δt.

A-block

Next we turn our attention to approximating the inverse action of the **A**-block. Indeed, recall from Section 3.4.4 that applying the the Schur complement block preconditioner requires approximating the action of both **S** as well as **A**. For the problem sizes considered up to this point, it is appropriate to solve the **A**-block with a direct LU solver. However, for large 3D problems, using a direct method to approximate the action of $\mathbf{A}^{-1}$ will quickly become infeasible, so different approaches to approximating $\mathbf{A}^{-1}$ are necessary.

In Section 3.4.5, we outlined an approach for approximating the **A** block in the saddle-point system that arises in dynamic free-surface models. In this section, we assess

Δt	**Reynolds Number = 10**	**Reynolds Number = 100**
10^0	24 / 28 (2)	25 / 29 (3)
10^{-1}	21 / 23 (2)	20 / 22 (3)
10^{-2}	17 / 19 (2)	18 / 19 (2)
10^{-3}	15 / 16 (2)	14 / 15 (2)
10^{-4}	12 / 13 (2)	9 / 9 (2)
10^{-5}	9 / 9 (2)	7 / 7 (2)

Table 3.8: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two-phase PCD for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the lid driven cavity problem with $h = 0.0125$ and varying Δt.

	Reynolds Number = 10		**Reynolds Number = 100**	
Δt	SIMPLE Config. 1	SIMPLE Config. 2	SIMPLE Config. 1	SIMPLE Config. 2
10^0	78 / 78 (2)	87 / 94 (2)	85 / 88 (3)	88 / 91 (2)
10^{-1}	79 / 80 (2)	81 / 82 (2)	246 / 572 (3)	86 / 87 (2)
10^{-2}	$\times$ / $\times$	77 / 78 (2)	$\times$ / $\times$	72 / 73 (2)
10^{-3}	60 / 62 (2)	63 / 64 (2)	28 / 28 (2)	30 / 30 (2)
10^{-4}	25 / 25 (2)	26 / 27 (2)	10 / 10 (2)	10 / 10 (2)
10^{-5}	9 / 9 (2)	9 / 9 (2)	$\times$ / $\times$	7 / 7 (2)

Table 3.9: Preconditioned GMRES iterations (average / maximum (Newton iterations)) using two different AMG configurations with the SIMPLE method for the $\mathbf{P}_1 - P_1$ strongly enforced boundary condition solution to the lid driven cavity problem with $h = 0.0125$ and varying Δt. Note that $\times$ indicates the simulation stopped due to failure in the solver convergence.

the performance of this method using three different approaches for the cavity and step problems. For the first simulation, we report the results using a direct LU solve for the **A**-block. This exact solve will form the basis for comparison.

For the second and third simulations, we approximate the inverse action of **A** using a GMRES iteration with a relative or absolute stopping tolerance of 0.01. Such a low resolution approach may seem surprising, but since the inverse of **A** is approximated twice per preconditioner application, it is important that the application of $\widehat{\mathbf{A}}^{-1}$ can be performed quickly. It is also important to note that the **A** block solve will require an indeterminate number of GMRES iterations. Therefore, it is necessary to use the FGMRES for the global Krylov method [77].

We consider two preconditioners for this GMRES iteration, both of which are based on the diagonal block preconditioner outlined in Section 3.4.5. First we consider using a full LU factorization to precondition each sub-block of **A**. While this approach does not scale, it gives an indication of the best performance one can expect from using (3.69) as a preconditioner. For the second GMRES iteration, we approximate the inverse action of the diagonal blocks with a single pass of boomerAMG using the default settings. It is worth noting that while the default boomerAMG are often effective for 2D problems, some modifications are necessary for 3D problems.

Results for these three simulations are given in Table 3.10 and provide encouraging feedback about the preconditioning approach to **A**. As expected, the full LU factorization exhibits the lowest GMRES iteration counts, but both FGMRES iteration methods are competitive. Further, while there appears to be a slight deterioration in the number of FGMRES iterations required when the BoomerAMG method is applied to the preconditioner blocks instead of using the full LU factorization, the two methods are very competitive with one another. It should be noted that while the data is not presented here, all three methods demonstrate mesh independence in FGMRES iterations scaling.

In this section, we have presented results which demonstrate that the two-phase PCD approximation to the Schur complement exhibits stable scaling for two-phase variations of the driven cavity and step problems for a variety of boundary conditions,

Re	*Method*		
	LU	FGMRES with LU	FGMRES with AMG
10	23 / 27 (3)	26 / 29 (3)	26 / 30 (3)
100	30 / 33 (6)	33 / 36 (6)	33 / 36 (6)
1000	48 / 57 (6)	54 / 64 (6)	55 / 64 (6)

Table 3.10: Two-phase PCD preconditioned FGMRES iterations using different configurations to solve the **A**-block of the saddle point system for the $\boldsymbol{P}_1 - P_1$ solution to the steady lid driven cavity problem with density ratio $\widehat{\rho} = 1.2 \times 10^{-3}$, viscosity ratio $\widehat{\mu} = 1.8 \times 10^{-2}$ (values for air-water flow), with $h = 0.00625$ and varying Reynolds number Re.

Re	*Method*		
	LU	FGMRES with LU	FGMRES with AMG
10	37 / 40 (4)	38 / 41 (4)	38 / 41 (4)
100	55 / 68 (6)	59 / 71 (6)	60 / 72 (6)

Table 3.11: Two-phase PCD preconditioned FGMRES iterations using different configurations to solve the **A**-block of the saddle point system for the $\boldsymbol{P}_1 - P_1$ solution to the steady step problem with density ratio $\widehat{\rho} = 1.2 \times 10^{-3}$, viscosity ratio $\widehat{\mu} = 1.8 \times 10^{-2}$ (values for air-water flow), with $h = 0.0125$ and varying Reynolds number Re.

stabilization and time stepping regimes. In addition, we have seen evidence that our preconditioning strategy for the **A** block is effective for the steady state problems studied here. In the next section, we examine preconditioner performance in a dynamic free-surface setting.

3.5.3 Dambreak Problem

The first dynamic free-surface simulation we examine is a two dimension dambreak (see [34] and [95]). The linear systems that arise in this simulation are different from the lid-cavity and step problems because the Navier-Stokes equations include a time stepping term. Moreover, the linear system changes during the simulation as the free-surface dynamics evolve.

In this setting, one would expect the SIMPLE operator to become more competitive with the PCD operator. For one, the large temporal term dampens the system's advective features which the SIMPLE operator fails to capture. Furthermore, the

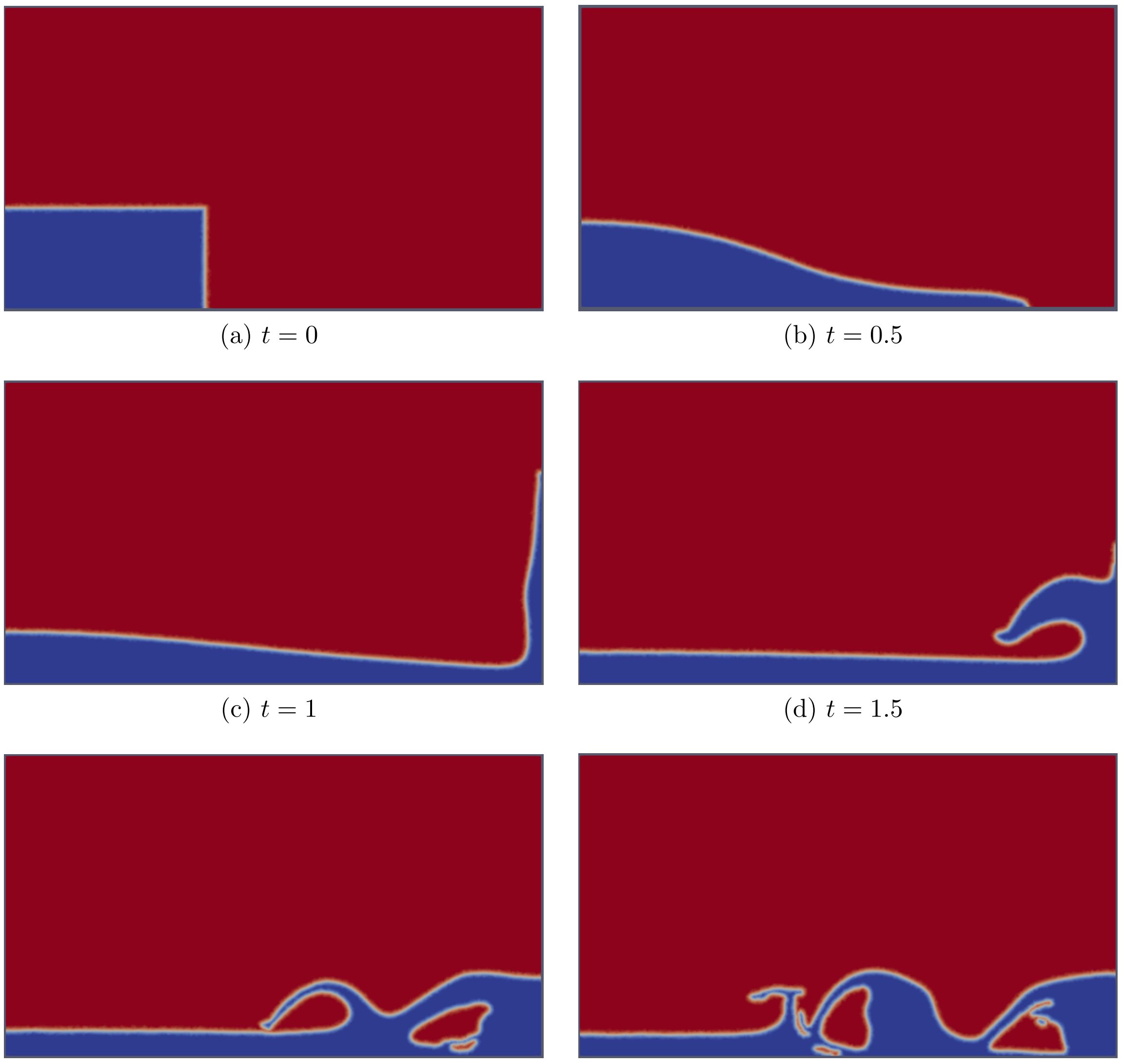
SIMPLE operator is built using components of the linear system. Thus, it is easier to implement and automatically captures the stabilization and boundary conditions of the system.

The PCD preconditioner's superior steady-state performance, does however suggest that the preconditioner can be useful when preconditioning these types of problems. Indeed, the results below suggest that PCD is superior to SIMPLE when larger time steps are permitted. Additional work is needed to optimize Proteus' PCD performance with respect to the stabilization methods and better ways to handle the numerical challenges of dealing with large density / viscosity ratios. However, the results presented herein suggest that the PCD preconditioner could be part of a dynamic preconditioning strategy that varies based on the problems' mesh size, dynamic Reynolds number and time step.

Problem Description

The domain is rectangular, with $\Omega = (0, 3.22) \times (0, 1.8)$, and free-slip conditions (see Section 3.3.3) are applied everywhere on the boundary $\partial\Omega$. Initially, there is a standing column of water in $\Omega_1 = (0, 1.2) \times (0, 0.6)$ with the remaining space being air. The simulation runs for a two second time interval and begins as the column of water collapses under gravity and proceeds to collide with the right-hand wall of the tank. This collision creates a wave and ultimately topological changes in the phases. Figure 3-3 displays several snapshots of the simulation. The dam-break problem provides a good benchmark for testing the two-phase PCD and SIMPLE preconditioners because its features are typical of many dynamic, multi-physics problems of practical interest.

To compare the scaling performance of two-phase PCD and SIMPLE, we consider two simulations. In the first, time steps are selected to ensure that the CFL number is less than or equal to 0.9. Such restrictions are often necessary for nonlinear solver convergence and solution accuracy. However, in some important cases this restriction


(a) $t = 0$ (b) $t = 0.5$

(c) $t = 1$ (d) $t = 1.5$

(e) $t = 1.75$ (f) $t = 2$

Figure 3-3: Evolution of the dam-break simulation in Proteus at selected points in times. The VOF (volume-of-fluid) is plotted with blue representing the water phase and red being the air.

on the CFL number is not strictly necessary.[1] Thus, in the second simulation, we use a fixed time step of $\Delta t = 0.01$. In this case, the CFL number is larger than one for much of the simulation, reaching a maximum of 20.5 and typically being above

[1]Accurate computation of relevant quantities of interest, such as drag force, for fixed hydraulic structures or vessels, frequently results in quasi-steady flows. In particular, the free surface may tend towards a steady wake structure or standing wave pattern, and this structure dominates the force on the given structure. In these cases, it is frequently desirable to use a fixed time step that results in CFL numbers significantly larger than one. Time stepping is then carried out until the quasi-steady hydrodynamic conditions are reached or the quantity of interest has reached a constant or steady periodic value.

	$h = 0.2$	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.0125$
Two-phase PCD	5 / 8 (0.5)	5 / 9 (0.6)	5 / 10 (2.6)	5 / 11 (14.7)	5 / 10 (126.0)
SIMPLE	4 / 10 (0.4)	4 / 10 (0.6)	4 / 10 (2.4)	5 / 13 (15.5)	5 / 17 (140.1)

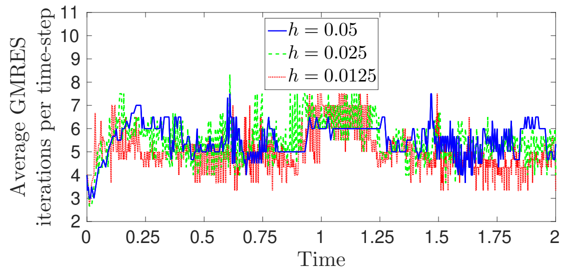
Table 3.12: The average / maximum number of GMRES iterations and simulation run times (in minutes) required across different meshes when running the dam-break problem with the CFL number less than or equal to 0.9.

2.5. Nonetheless, the time step is still small enough to achieve nonlinear solver convergence and solution accuracy. For both simulations, we analyse the average and maximum number of GMRES iterations required at five different levels of mesh refinement. These mesh refinement levels are selected so that, by the final refinement, the physics of the simulation is sufficiently resolved to perform relevant engineering analysis. The dambreak timings were collected using 8 cores of a dedicated 2.3-GHz Intel Xeon Haswell processor with 128 GBytes of DDR4 memory on the Topaz supercomputer in the Department of Defense High Performance Computing Modernization Program.[2]

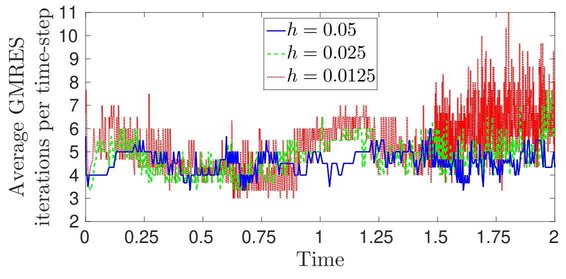
Table 3.12 presents the average and maximum number of GMRES iterations taken during the first simulation with a restricted CFL number. These results suggest that, on average, the SIMPLE and two-phase PCD preconditioners both scale well with the mesh size. However, Table 3.12 also reveals that the maximum number of iterations required by the SIMPLE preconditioner increases as the mesh is refined. As seen in Figure 3-4, the increase in maximum iterations of the SIMPLE preconditioner occurs as the air and water phases begin to undergo topological changes around one and a half seconds into the simulation. Indeed, as the water phase reconnects with itself, it generates a pressure that causes the air phase to accelerate and increase the advective features of the simulation. Since SIMPLE only uses the diagonal elements of the matrix A, it appears unable to fully capture these additional advection dynamics. In contrast, the two-phase PCD preconditioner scales well during this mixing phase of the simulation.

Table 3.12 also reveals that on the most refined meshes, the two-phase PCD pre-

[2]In this example, the number of elements for $h = 0.025$ is 29,328 while for $h = 0.0125$ it is 117,353.



(a) Two-phase PCD



(b) SIMPLE

Figure 3-4: Average preconditioned GMRES iterations per time step for the two-phase PCD and SIMPLE preconditioners with the CFL number less than or equal to 0.9.

conditioner is faster than the SIMPLE approach. One reason for this is that the two-phase PCD preconditioner requires fewer GMRES iterations than the SIMPLE method during the mixing phase of the simulation. A second reason is that the AMG method applied to the SIMPLE preconditioner requires more computational effort than the AMG method used in the two-phase PCD preconditioner. Finally, the two-phase PCD preconditioner tends to exceed the linear solver threshold by a larger margin than the SIMPLE approach, leading to slightly smaller residual norms in the nonlinear solver. Interestingly, this difference slightly reduces the computational effort needed to solve other components in the full RANS2P model.

Results for the second simulation, using a fixed time step $\Delta t = 0.01$, are shown in Table 3.13 and suggest that, on coarse meshes, SIMPLE and the two-phase PCD preconditioner are competitive with one another. In contrast to the first simulation, however, as the mesh is refined, the SIMPLE method requires a rapidly increasing number of GMRES iterations to solve the linear system. Meanwhile, the two-phase PCD preconditioner remains relatively stable, with iteration counts increasing only modestly.

These results are consistent with the steady-state performance observed above in Section 3.5.2. As the mesh is refined for a fixed time step, the advective features of the system become more pronounced and the CFL number increases. As observed for the steady lid driven cavity problem, the SIMPLE approach does not capture the features of an advection dominated flow well enough to provide a robust precondi-

	$h = 0.2$	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.0125$
Two-phase PCD	4 / 8 (0.5)	5 / 9 (0.6)	8 / 14 (1.6)	11 / 25 (5.8)	14 / 34 (26.5)
SIMPLE	4 / 10 (0.4)	4 / 10 (0.6)	5 / 10 (1.5)	10 / 32 (6.6)	$\times$

Table 3.13: The average / maximum number of GMRES iterations and simulation run times (in minutes) required across different meshes when running the dam-break problem with fixed $\Delta t = 0.01$. Note that $\times$ indicates the simulation stopped due to failure in the solver convergence.

tioner. The two-phase PCD preconditioner, however, does account for such features and thus remains capable of producing stable, reliable results in this setting.

Overall, our results for the two-phase PCD preconditioner in a free-surface, multi-physics setting are encouraging. When a restricted CFL number is used, the two-phase PCD preconditioner slightly outperforms the SIMPLE method both in terms of the reducing the number of GMRES iterations required, as well as delivering faster run times. As the CFL number of the flow increases, two-phase PCD demonstrates a significant improvement over the SIMPLE method due to its superior steady-state performance. Together, these results suggest that the two-phase PCD approach can be effectively used as an approximation to the inverse Schur complement in coupled free-surface problems.

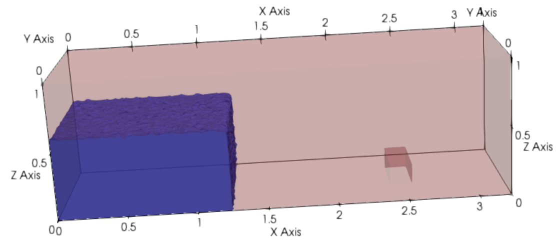
3.5.4 MARIN Problem

The next free-surface benchmark problem we consider is a three-dimensional dam break simulation based on physical experiments run at the Maritime Research Institute Netherlands (MARIN) [59]. These experiments provide measurements of water heights, pressures and forces over time that can be used to validate numerically modeled results. This simulation has been selected because it allows us to study preconditioner performance on a three dimensional, realistic free-surface test problem.

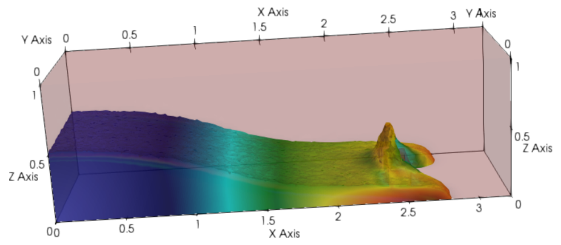
The MARIN simulation takes place in a tank with dimensions 3.22m $\times$ 1m $\times$ 1m. Located around $x = 2.5$, a 0.161m $\times$ 0.161m $\times$ 0.403m obstacle protrudes from the bottom of the tank. The initial setup can be seen in Figure 3-5.

The simulation begins when gravity, acting on a 1m $\times$ 1m $\times$ 0.55m column of water positioned at the front of the tank, causes the column to collapse. As illustrated in the snapshots of the simulation shown in Fig 3-5, the column of water collapses and flows towards the obstacle, making contact around $t = 0.5$ seconds. Following this initial contact, various topological changes occur between the air and water phases as the water over tops and splashes around the obstacle. The next major component of the simulation occurs as the water makes contact with the back wall of the tank. At this time, the water reverses direction and forms a wave resulting in complicated topological changes between the air and the water phases.

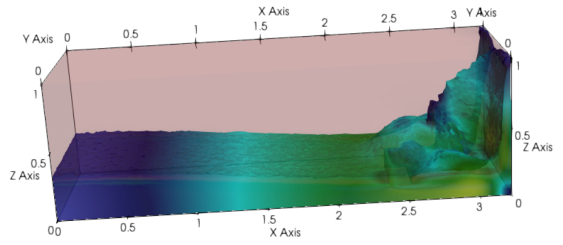
Because of its three-dimensional domain, the MARIN problem is a useful extension



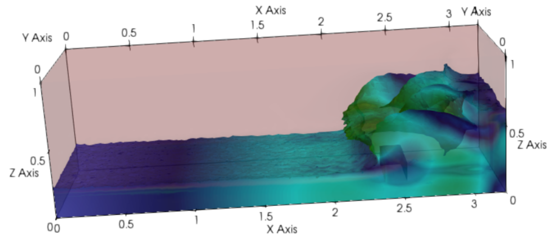
(a) $t = 0$



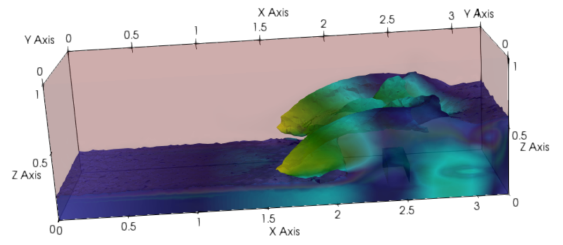
(b) $t = 0.5$



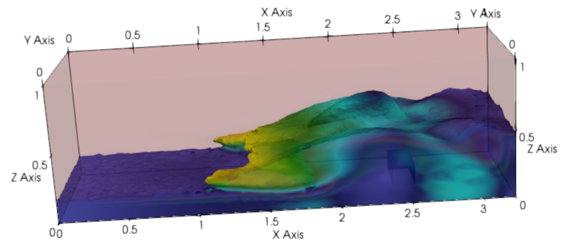
(c) $t = 1$



(d) $t = 1.5$



(e) $t = 1.75$



(f) $t = 2$

Figure 3-5: Evolution of the MARIN simulation in Proteus at selected points in times. The VOF (volume-of-fluid) is plotted with blue representing the water phase and red being the air.

to the dambreak simulation examined in Section 3.5.3. Indeed, the matrices that arise from three-dimensional finite element discretizations present new challenges relative

Refinement (R)	10	15	20	25	30	35	40	45	50
Nodes (000s)	57	186	432	835	1 433	2 264	3 347	4 782	6 544
DoF (000s)	228	743	1 729	3 338	5 733	9 057	13 386	19 128	26 176

Table 3.14: Marin Mesh Statistics

to those that appear in two-dimensional domains. For example, in two-dimensions, it remains feasible to use a direct solver to solve for the **A**-block across a range of computationally relevant refinement levels. The increased matrix bandwidth that arises in a three-dimensional finite-element matrix means that solving the **A**-block with a direct method quickly becomes infeasible and an iterative approach such as preconditioned GMRES must be used. Furthermore, in three-dimensions, the accuracy of multigrid methods deteriorate relative to that in two-dimensions, because of memory and scaling limitations. Specifically, three-dimensional problems typically use complexity reducing algorithms such as aggressive coarsening and multi-level interpolation operators (see Section 3.4.3).

Preconditioner Results

To analyze preconditioner performance for the MARIN problem, we consider mesh refinement levels from $R = 10$ to $R = 50$ where R is the refinement parameter. The relationship between the refinement level and the mesh diameter h is approximately $h \approx \frac{1}{2R}$. At the coarsest level ($R = 10$), a three dimensional tetrahedral mesh with roughly fifty-seven thousand vertices is used. At the finest level ($R = 50$), a three dimensional tetrahedral mesh with roughly 6.5 million vertices is used.
The simulations ran in these experiments use linear finite elements to approximate the three velocity components and the pressure. Therefore, the total number of degrees of freedom is four times the number of vertices. Table 3.14 reports the number of tetrahedral mesh vertices and degrees of freedom for a given refinement level.

Scaling Results

The first set of results presented in this section give a performance overview of the SuperLU_DIST, two-phase PCD, SIMPLE and ASM preconditioning strategies. Simulation run times have been collected across nine refinement levels incrementing by five from $R = 10$ to $R = 50$. Results for the refinement levels $R = 10, 15, 20$ are given in Table 3.15, results for the refinement levels $R = 25, 30, 35$ are given in Table 3.16 and results for the refinement levels $R = 40, 45, 50$ are given in Table 3.17. The tables have been broken into separate blocks to keep the number of degrees of freedom per processor roughly bounded between 50K and 2K.

The second set of results provided in Tables 3.18 – 3.21 highlight the weak scaling performance of the PCD and ASM preconditioners.

Three major observations emerge from these results. First, the data suggests that the SuperLU_DIST approach is the least competitive method considered. Second, the SIMPLE preconditioner is a competitive – and in some cases slightly faster – alternative to the PCD preconditioner. Finally, in nearly all respects, the PCD preconditioner is superior to the ASM preconditioner. In the following sections, we explore the details of these observations in more detail.

SuperLU_DIST

As an initial observation, the data in Table 3.15 suggests that the SuperLU_DIST approach is the least competitive method considered. At the coarsest refinement levels of 10 and 15, SuperLU_DIST recorded significantly slower speeds than the other methods. For instance, when the simulation was run at the $R = 10$ refinement level with 256 processors, it took nearly one and a half hours to complete while all the other approaches finished in under 15 minutes. An even worse relative performance can be observed at the $R = 15$ refinement level.

For the $R = 10$ and $R = 15$ problem sizes, the SuperLU_DIST method does exhibit some limited strong scaling capacity. As highlighted in Figure 3-6 for the $R = 10$ and $R = 15$ refinement levels, initially adding more processors to the SuperLU_DIST method reduces simulation run times. However, Figure 3-6 also makes it clear that simulation run times begin to increase once the number of processors increases past a certain threshold.

The SuperLU_DIST method becomes essentially ineffective for refinement levels of

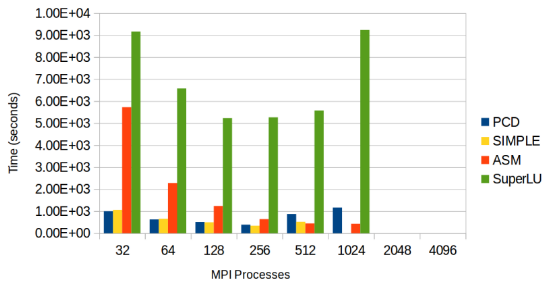
$R = 20$ and above, so no results are reported. Indeed, a trial run of the $R = 20$ case with 512 processors suggested that the simulation would require roughly 72 hours to finish. In contrast, using the same refinement level and number of processors, the PCD method was able to run in under an hour. It is also worth mentioning that for the SuperLU_DIST method, memory usage starts to become an issue at the $R = 20$ level. Indeed, for $R = 20$, at least 256 processors are necessary to ensure that there is sufficient memory for the factorization step to complete.
The poor computational results for the SuperLU_DIST method are not surprising. As discussed in Section 3.4.1, scaling performance is difficult to achieve for direct solvers. The fact that this is a three-dimensional simulation and the linear system is not symmetric further complicates matters.

SIMPLE

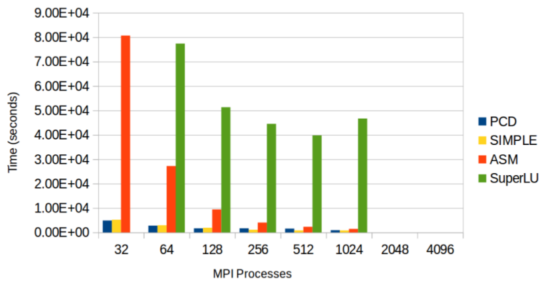
The data reported in Tables 3.15-3.17 suggest that the SIMPLE preconditioner results in similar run times as the PCD preconditioner. In some cases (e.g. $R = 25$), the SIMPLE preconditioner reported times that were faster than the PCD preconditioner, while in other cases, the PCD preconditioner exhibited consistently faster performance (e.g. $R = 45$). Throughout the simulations, however, the timing results were typically within 15 % of each other.
It is not surprising that the SIMPLE and the PCD preconditioners exhibit similar performance. Indeed, both approaches use the same block preconditioner, albeit with different approximations to the Schur complement. As the results highlight, the setup portion of the PCD preconditioner is typically more expensive than the SIMPLE approach. The increased expense in the setup face, however, is offset by faster performance in the linear solver phase.

PCD and ASM

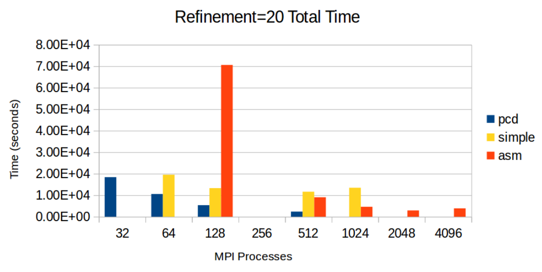
As a final observation, Tables 3.15-3.17 suggest that the two-phase PCD is a superior preconditioner to the ASM preconditioner in nearly all respects. For instance, at refinement level 25, the simulation takes roughly two-hours when 256 processors are used with the PCD preconditioner. In contrast, to run the simulation with the ASM in two-hours, 4096 processors are necessary. This suggests the PCD method was able to achieve the same performance as the ASM using one sixteenth the computational resources. Furthermore, on the most refined computational meshes (i.e., $R \geq 40$), it is not clear how many processors would be necessary for the ASM preconditioner to generate comparable computational run times with the PCD preconditioner. We expand further on the advantages and disadvantages of the two methods below.



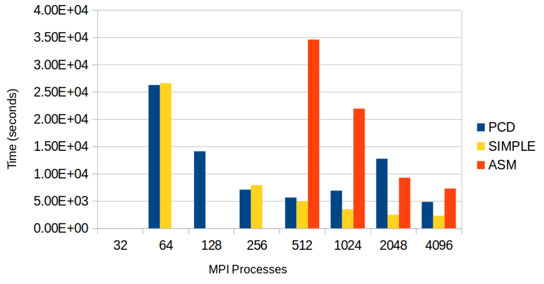
(a) $R = 10$



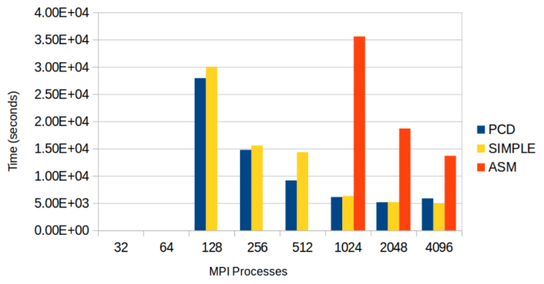
(b) $R = 15$



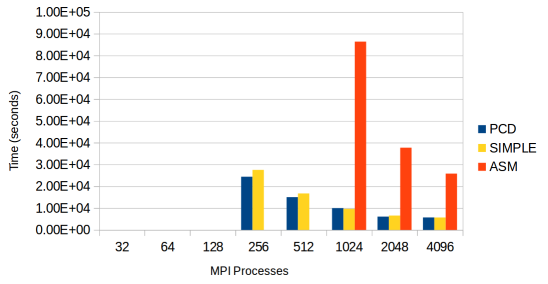
(c) $R = 20$



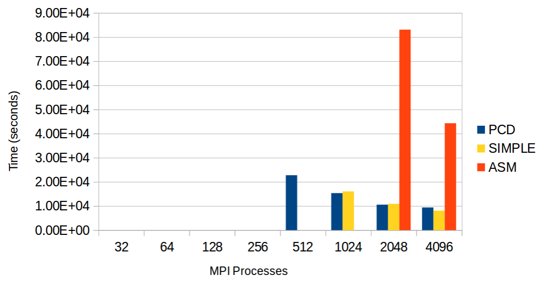
(d) $R = 25$



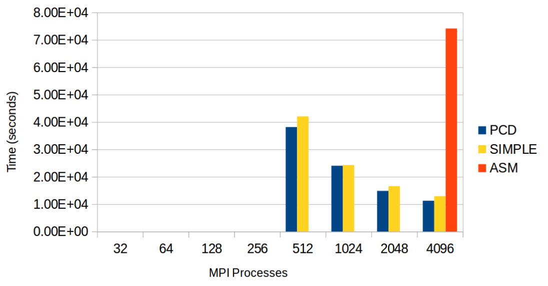
(e) $R = 30$



(f) $R = 35$



(g) $R = 40$



(h) $R = 45$

Figure 3-6: Simulation run times at different refinement levels by preconditioner type

Refinement		32	64	128	256	512	1024
10	DOF / Core	7,121	3,561	1,780	890	445	223
	SuperLU DIST	2.54 / 90 / 1	1.83 / 91 / 1	1.45 / 90 / 1	1.46 / 90 / 1	1.55 / 89 / 1	2.57 / 90 / 1
	PCD	0.28 / 21 / 16	0.17 / 22 / 21	0.14 / 22 / 27	0.11 / 20 / 31	0.24 / 25 / 55	0.32 / 25 / 56
	SIMPLE	0.3 / 9 / 30	0.18 / 10 / 35	0.14 / 10 / 37	0.09 / 9 / 42	0.14 / 9 / 54	**
	ASM	1.59 / 1 / 88	0.63 / 1 / 83	0.34 / 1 / 79	0.18 / 1 / 71	0.12 / 1 / 50	0.12 / 1 / 46
15	DOF / Core	23,205	11,602	5,801	2,901	1,450	725
	SuperLU DIST	> 24 hrs	21.51 / 93 / 1	14.26 / 93 / 1	12.37 / 92 / 1	11.05 / 92 / 1	12.97 / 93 / 0
	PCD	1.36 / 23 / 17	0.77 / 23 / 18	0.46 / 24 / 23	0.47 / 25 / 42	0.43 / 25 / 49	0.26 / 23 / 38
	SIMPLE	1.45 / 9 / 33	0.82 / 10 / 34	0.53 / 11 / 40	0.3 / 11 / 40	0.23 / 9 / 41	0.23 / 9 / 47
	ASM	22.42 / 0 / 96	7.56 / 0 / 94	2.62 / 1 / 90	1.12 / 1 / 86	0.65 / 1 / 82	0.4 / 1 / 73
20	DOF / Core	54,033	27,016	13,508	6,754	3,377	1,689
	SuperLU DIST	*	*	*	†	*	*
	PCD	5.1 / 22 / 19	2.9 / 23 / 24	1.5 / 24 / 21	†	0.7 / 24 / 35	**
	SIMPLE	‡	2.89 / 9 / 36	1.54 / 9 / 37	†	0.56 / 10 / 42	0.49 / 11 / 49
	ASM	> 24 hrs	> 24 hrs	19.61 / 1 / 95	†	2.51 / 1 / 88	1.28 / 1 / 84

Table 3.15: MARIN simulation run times across preconditioning strategies (total run time (hours) / time in NSE preconditioner setup (%) / percent time in NSE linear solve (%)). * indicates that the simulation was not run. ** indicates that a *hypre* error was encountered during simulation. † indicates an error during mesh generation phase. ‡ indicates memory usage limitations.

Refinement		64	128	256	512	1024	2048	4096
25	DOF / Core	52,157	26,079	13,039	6,520	3,260	1,630	815
	PCD	7.3 / 23 / 23	3.9 / 24 / 24	2 / 24 / 22	1.6 / 23 / 36	1.9 / 25 / 54	3.5 / 9 / 83	1.3 / 29 / 49
	SIMPLE	7.39 / 9 / 38	**	2.2 / 11 / 40	1.37 / 11 / 43	0.97 / 11 / 48	0.69 / 10 / 47	0.64 / 8 / 47
	ASM	> 24 hrs	> 24 hrs	> 24 hrs	9.61 / 0 / 93	6.09 / 0 / 0	2.57 / 1 / 87	2.02 / 1 / 84
30	DOF / Core	89,576	44,788	22,394	11,197	5,598	2,799	1,400
	PCD	‡	7.8 / 23 / 23	4.1 / 23 / 23	2.5 / 23 / 27	1.7 / 24 / 35	1.4 / 27 / 41	1.6 / 28 / 47
	SIMPLE	‡	8.34 / 12 / 38	4.33 / 11 / 39	3.99 / 8 / 60	1.76 / 12 / 48	1.44 / 12 / 53	1.37 / 13 / 53
	ASM	‡	> 24 hrs	> 24 hrs	> 24 hrs	9.89 / 0 / 93	5.19 / 1 / 90	3.81 / 1 / 88
35	DOF / Core	141,516	70,758	35,379	17,689	8,845	4,422	2,211
	PCD	‡	‡	6.8 / 24 / 20	4.2 / 23 / 24	2.8 / 24 / 32	1.7 / 25 / 33	1.6 / 26 / 39
	SIMPLE	‡	‡	7.65 / 11 / 39	4.65 / 11 / 43	2.68 / 10 / 44	1.83 / 11 / 50	1.58 / 10 / 54
	ASM	‡	> 24 hrs	> 24 hrs	> 24 hrs	> 24 hrs	10.48 / 0 / 92	7.18 / 0 / 91

Table 3.16: Marin simulation run times across preconditioning strategies (total run time (hours) / time in NSE preconditioner setup (%) / percent time in NSE linear solve (%)). * indicates that the simulation was not run. A single number indicates an approximation of the total run time (based off a four hour simulation) †indicates a segfault occurs with mesh generator. ‡ indicates memory usage limitations.

Refinement		256	512	1024	2048	4096
40	DOF / Core	52,289	26,145	13,072	6,536	3,268
	PCD	*	6.3 / 23 / 21	4.3 / 24 / 29	2.9 / 24 / 34	2.6 / 27 / 41
	SIMPLE	*	**	4.46 / 11 / 46	3.03 / 10 / 50	2.24 / 11 / 56
	ASM	$>$ 24 hrs	$>$ 24 hrs	$>$ 24 hrs	23.07 / 0 / 94	12.3 / 0 / 93
45	DOF / Core	74,719	37,360	18,680	9,340	4,670
	PCD	*	10.6 / 23 / 23	6.7 / 25 / 30	4.1 / 24 / 31	3.1 / 26 / 39
	SIMPLE	*	11.69 / 11 / 41	6.75 / 10 / 45	4.61 / 11 / 50	3.59 / 11 / 58
	ASM	*	*	*	*	20.6 / 0 / 94
50	DOF / Core	102,251	51,125	25,563	12,781	6,391
	PCD	*	16.6 / 23 / 27	9 / 24 / 25	5.7 / 24 / 29	4.2 / 25 / 35
	SIMPLE	*	17.43 / 11 / 41	10.1 / 10 / 43	6.47 / 11 / 47	4.19 / 10 / 49
	ASM	*	*	*	*	*

Table 3.17: Marin simulation run times across preconditioning strategies (total run time (hours) / time in NSE preconditioner setup (%) / percent time in NSE linear solve (%)). * indicates that the simulation was not run. A single number indicates an approximation of the total run time (based off a four hour simulation) †indicates a segfault occurs with mesh generator. ‡ indicates memory usage limitations.

		128	256	512	1024	2048	4096	Factor Increase (34.2)
PCD	Total Run Time	0.57	1.51	1.95	2.14	2.55	3.41	6.0
	Preconditioner Setup Time	0.27	0.36	0.49	0.55	0.67	0.91	3.4
	Linear Solve Time	0.29	0.47	0.79	0.88	0.99	1.41	4.9
ASM	Total Run Time	5.41	7.26	8.01	11.02	14.13	19.48	3.6
	Preconditioner Setup Time	0.04	0.04	0.05	0.06	0.07	0.07	2.0
	Linear Solve Time	4.79	6.56	7.29	10.16	13.14	18.24	3.8

Table 3.18: MARIN weak scaling results for approximately 4K DoF per core. The factor increase represents the ratio of the time taken using the largest number of processors against the time taken using the smallest number of processors. The number listed in the final column of the top row in parentheses denotes the factor by which the problem size increased.

		64	128	256	512	1024	2048	Factor Increase (34.2)
PCD	Total Run Time	2	2.28	2.77	2.82	3.06	3.98	2.0
	Preconditioner Setup Time	0.45	0.52	0.66	0.65	0.71	0.96	2.1
	Linear Solve Time	0.43	0.52	0.91	0.81	0.84	1.33	3.1
ASM	Total Run Time	13.69	18.82	18.11	23.76	29.17	37.75	2.8
	Preconditioner Setup Time	0.06	0.08	0.08	0.09	0.1	0.12	2.0
	Linear Solve Time	12.55	17.49	16.8	22.31	27.56	35.87	2.9

Table 3.19: MARIN weak scaling results for approximately 8K DoF per core. The number listed in the final column of the top row in parentheses denotes the factor by which the problem size increased.

		32	64	128	256	512	1024	Factor Increase (33.6)
PCD	Total Run Time	4.26	4.68	5.05	5.86	5.33	6.95	1.6
	Preconditioner Setup Time	0.94	1.08	1.19	1.35	1.24	1.6	1.7
	Linear Solve Time	0.76	0.91	1.19	1.39	1.15	1.92	2.5

Table 3.20: MARIN weak scaling results for approximately 20K DoF per core. The number listed in the final column of the top row in parentheses denotes the factor by which the problem size increased.

		32	64	128	256	512	Factor Increase (16.1)
PCD	Total Run Time	8.63	8.08	9.08	10.03	11.21	1.3
	Preconditioner Setup Time	1.96	1.88	2.14	2.38	2.67	1.4
	Linear Solve Time	1.59	1.54	1.88	1.99	2.52	1.6

Table 3.21: MARIN weak scaling results for approximately 40K DoF per core. The number listed in the final column of the top row in parentheses denotes the factor by which the problem size increased.

Analysis of the ASM and two-phase PCD preconditioners

In this section, we compare the performance of the two-phase PCD and ASM preconditioners in more detail. As highlighted in Tables 3.15-3.17 and Figure 3-6, the PCD preconditioner generates equivalent or faster simulation run times using less computational resources than the ASM preconditioner. Next, to better understand the relative performance of these methods, we investigate the strong and weak scaling performance of the PCD and ASM preconditioners.

Strong Scaling Capacity

Figures 3-6 illustrate that the ASM exhibits more consistent strong scaling performance than the two-phase PCD method. Simulation run times at refinement level 15 provide a clear illustration of this. As reported in Table 3.15, using 32 processors, simulation runtime using the ASM is roughly 22 hours while the runtime using the PCD method is less than one and a half hours. After increasing the number of processors to 1024, however, the simulation run time using both the ASM and the PCD is less than half an hour. This highlights that, on average, the ASM run time decreased by roughly 50% every time the number of processors doubled while the run time using the two-phase PCD preconditioner decreased by roughly 25% every time the number of processors doubled.

The source of the ASM strong scaling performance is the rapid reduction in computational effort required to perform the direct local solves as the mesh is distributed across more processors. Indeed, the sparse LU factorization algorithm has complexity $\mathcal{O}(n^2)$. This suggests that every time the number of processors doubles, the time required to solve the local problems decreases roughly 75%. One of the metrics reported in Tables 3.22 and 3.23 is the percentage change in time needed to perform the LU factorization component of the ASM preconditioner. While this data suggests that a 75% decrease is the best case scenario, it does highlight every time the number of processors doubles, the factorization time decreases by more than half.

There are two factors, however, that limit the ASM strong scaling capacity as the number of processors increases. First, since the speed up in the factorization step is faster than other parts of the simulation, the percentage of time spent performing the LU factorization decreases. This implies that there is less scope for faster factorizations to reduce overall run times. A second factor affecting the strong scaling performance of the ASM is the deteriorating quality of the ASM preconditioner as the domain is decomposed into smaller and smaller regions.

To analyze the effect of deteriorating preconditioner quality, Tables 3.22 and 3.23 report GMRES iterations for several refinements. As expected, as the number of processors (and hence subdomains) increases, so does the number of GMRES iterations. The computational cost of increased GMRES iterations is manifest in two ways. The first is the increased memory requirements of storing the residual vectors. The second is the increased time spent in the Gram-Schmidt orthogonalization routine that generates the orthogonal Krylov basis.

The additional memory requirements necessary as the number of GMRES iterations increases is not a concern in this context. As the problem is distributed across a larger number of cores, the avaliable memory increases. Therefore, as long as the number of GMRES iterations does not increase too rapidly as the work is distributed, the memory gained from using additional processors is sufficient to absorb the additional storage requirements.

Instead, the limiting factor to strong scaling is the time spent in the Gram-Schmidt orthogonalization step of the GMRES algorithm. As shown in Tables 3.22 and 3.23, while the time spent in the Gram-Schmidt orthogonalization generally decreases as more processors are used, it does so at a slower rate than the rest of the simulation. This causes the percentage of time the simulation spends in that orthogonalization stage to increase.

The strong scaling performance of the two-phase PCD method is not as consistent as it is for the ASM. For some numerical experiments, the two-phase PCD exhibits ideal strong scaling. For example, when $R = 20$, the MARIN simulation runtime decreases by a factor of approximately two as the number of processors increases from 64 to 128. Similar results can be seen at refinement levels $R = 25$ and $R = 30$. In

Refinement		32	64	128	256	512	1024
10	GMRES Iterations	43 / 50	49 / 57	64 / 77	75 / 87	83 / 96	88 / 106
	GSO Function Calls	30422	33804	48543	53082	60874	71547
	Time In GSO (mins)	3	2.5	2.6	2.5	1.7	2.4
	% Time in GSO	3.2	6.6	12.7	23.5	23.4	33.5
	% Δ in LU Fact. Time	–	-69	-59	-69	-68	-58
15	GMRES Iterations	50 / 61	63 / 77	71 / 89	85 / 105	99 / 121	113 / 139
	GSO Function Calls	59075	72689	83116	97931	114749	130921
	Time In GSO (mins)	19.2	27.91	14.17	9.67	10.94	9.93
	% Time in GSO	1.4	6.1	9	14.4	28.2	41.3
	% Δ in LU Fact. Time	–	-72	-72	-69	-59	-75

Table 3.22: Details of the ASM preconditioner applied to the MARIN problem at refinement levels 10 and 15. The first row lists the average / maximum number of GMRES iterations during the simulation. The second, third and fourth rows present diagnostic information about the Gram-Schmidt orthogonalization (GSO) step. The fifth row reports the percentage change in time spent performing the LU factorization component of the ASM preconditioner.

Refinement		128	256	512	1024	2048	4096
20	GMRES Iterations	82 / 103	–	111 / 138	130 / 159	145 / 178	167 / 204
	GSO Function Calls	147158	–	195787	217624	246017	280682
	Time In GSO (mins)	109.62	–	24.06	20.89	20.06	43.49
	% Time in GSO	9.3	–	15.9	27.1	41	67.1
	% Δ in LU Fact. Time	–	–	–	-63	-70	-67
25	GMRES Iterations	*	*	122 / 155	–	161 / 205	184 / 237
	GSO Function Calls	*	*	309854	–	382362	430052
	Time In GSO (mins)	*	*	85.9	–	62.86	74.39
	% Time in GSO	*	*	14.9	–	40.7	61.3
	% Δ in LU Fact. Time	*	*	–	–	–	-75
30	GMRES Iterations	*	*	*	150 / 191	175 / 225	199 / 261
	GSO Function Calls	*	*	*	424468	498007	562476
	Time In GSO (mins)	*	*	*	116.2	95.02	123.29
	% Time in GSO	*	*	*	19.6	30.5	54
	% Δ in LU Fact. Time	*	*	*	–	-61	-69

Table 3.23: Details of the ASM preconditioner applied to the MARIN problem at refinement levels 20, 25 and 30. The first row lists the average / maximum number of GMRES iterations during the simulation. The second, third and fourth rows present diagnostic information about the Gram-Schmidt orthogonalization (GSO) step. The fifth row reports the percentage change in time spent performing the LU factorization component of the ASM preconditioner.

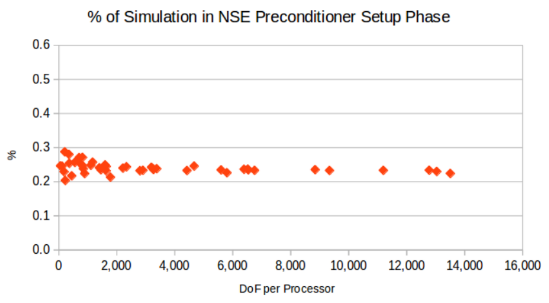
Refinement	32	64	128	256	512	1024	2048	4096
10	11 / 20	11 / 19	12 / 23	12 / 21	12 / 26	13 / 27	*	*
15	12 / 20	12 / 18	12 / 19	12 / 22	13 / 26	13 / 32	*	*
20	12 / 21	13 / 24	12 / 19	†	13 / 23	**	*	*
25	*	13 / 24	13 / 26	13 / 27	13 / 31	13 / 30	14 / 78	14 / 34
30	*	*	13 / 27	13 / 24	13 / 32	13 / 26	13 / 35	14 / 42
35	*	*	*	13 / 28	13 / 31	13 / 33	13 / 28	14 / 42
40	*	*	*	*	13 / 28	13 / 33	13 / 34	14 / 46
45	*	*	*	*	13 / 32	13 / 39	13 / 32	14 / 43
50	*	*	*	*	13 / 32	13 / 32	13 / 39	14 / 40

Table 3.24: GMRES iterations of the PCD preconditioner applied to the MARIN problem. The rows lists the average / maximum number of GMRES iterations during the simulation. ** indicates that a *hypre* error was encountered during simulation. † indicates an error during mesh generation phase. * indicates that the simulation was not run.

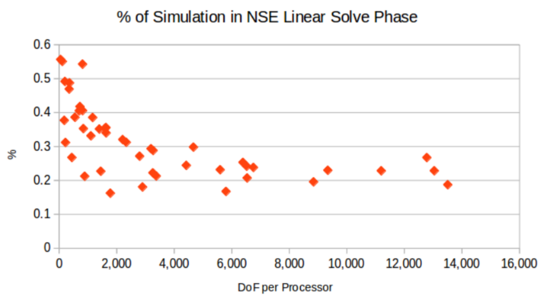
most cases, however, the computational run time does not halve when the number of processors doubles.

The factors that limit the strong scaling potential of the two-phase PCD method are different than that for the ASM. As shown in Table 3.24, the number of GMRES iterations scale reasonably well for two-phase PCD method across processors and mesh refinements. Instead, the limiting factor in strong scaling performance is a result of deteriorating linear solver performance as the number of degrees of freedom per processor decreases. Figure 3-7 plots the percentage of simulation time spent in the NSE preconditioner setup phase and percentage of time spent in the NSE linear solve phase against the number of degrees of freedom per processor for the PCD preconditioner. Thus, as the number of processors is increased, the degrees of freedom per processor becomes smaller. One can observe from this plot that the percentage of time spent in the setup phase remains relative consistent between 20 % and 30 % as the DoF per processor changes. In contrast, the percentage of simulation time spent in the linear solve phase begins to increase as the degrees of freedom per processor decreases below 5K, and then begins to increase rapidly as the degrees of freedom per processor decreases below 2K.

To understand why this is the case, recall that AMG includes a setup phase and a linear solve phase. Moreover, note that different AMG solvers are used for the velocity sub-blocks and the two-phase Laplace operator that appear in the PCD preconditioner. Both AMG solvers use the parallel HMIS coarsening algorithm. After



(a) setup phase



(b) linear solve phase

Figure 3-7: Percent of MARIN simulation time spent in the setup phase as well as in the linear solve phase of the NSE segment of the simulation using the PCD method.

experimenting with different solver options, however, a more aggressive coarsening algorithm was used for the two-phase Laplace operator from the PCD preconditioner (recall (3.87)). Specifically, two levels of aggressive coarsening are used along with a strong coarsening value of $\alpha = 0.5$. For the velocity sub-blocks, the strong coarsening parameter is $\alpha = 0.5$ as well, but no aggressive coarsening is used (see Section 3.4.3 for more details on AMG solver options).

As seen in Figure 3-7, when a problem uses 5K or more DoF per processor, the setup and solve times are roughly the same. However, when the number of DoF per processors begins to fall below 5K, the linear solve phase does not scale well. This is not surprising given the communication costs associated with using AMG across multiple processors.

While the ASM capacity for strong scaling is superior to the two-phase PCD, the two-phase PCD preconditioner still achieves superior timing results using fewer computational resources. This outperformance is clearly illustrated in Figure 3-6. Notably, when enough processors are used, the ASM preconditioner can produce run times that are competitive with the PCD preconditioner. However, the PCD preconditioner is able to achieve these comparable run times for the fastest ASM trial with significantly fewer processors. As the problem size increases, the ASM becomes increasing less competitive with the PCD preconditioner (see Tables 3.15-3.17 and Figure 3-6).

Weak Scaling Analysis

Next we consider the weak scaling performance of the two-phase PCD and ASM preconditioners. Recall that weak scaling refers to the change in run time when the problem size is increased and the work is distributed across a larger number of processors to keep the DoF per processor fixed. The factor increase is one metric that will be considered when analyzing weak scaling performance. For a fixed number of DoF per processor, the factor increase represents the ratio of the time taken using the largest number of processors against the time taken using the smallest number of processors.

Weak scaling results for 4K, 8K, 20K and 40K DoF per processor for the ASM and the two-phase PCD method are given in Tables 3.18-3.21. The rows of the tables report overall run, preconditioner setup, and linear solve time respectively. The final column of each table reports the factor increase in run time between the largest and smallest problem sizes. The factor increase in the problem size is reported in the parentheses in the final column of the top row. Note that the ASM weak scaling results for the 20K and 40K DoF per core have been omitted because they cannot be computed in a reasonable amount of time.

The values of 4K, 8K, 20K and 40K were selected to accommodate the two-phase PCD preconditioner. Recall that in the block Schur complement preconditioner (3.66), the size of the Schur complement is one quarter the total number of degrees of freedom. As a result, the AMG preconditioner used in the Schur complement sub-block has 1K, 2K, 5K and 10K DoF per core respectively.

First, we consider the ASM weak scaling results. As seen in Tables 3.18 and 3.19, in the 4K and 8K DoF per core simulations, the problem size is increased by 34 times. However, the simulation run time using the ASM only increases by a factor of 3.6 and 2.8 respectively. These results are quiet good when one considers that the quality of the ASM preconditioer deteriorates as more processors are used. As shown in Table 3.25, the average number of preconditioned GMRES iterations increased by 2.3 times when the problem size increased 34 times.

Refinement	64	128	256	512	1024	2048	4096
4K	–	69 / 85	94 / 117	115 / 148	143 / 188	178 / 244	234 / 319
8K	60 / 73	83 / 103	99 / 126	124 / 165	161 / 215	200 / 269	–

Table 3.25: GMRES Iterations (average / maximum) running the MARIN problem with ASM preconditioner.

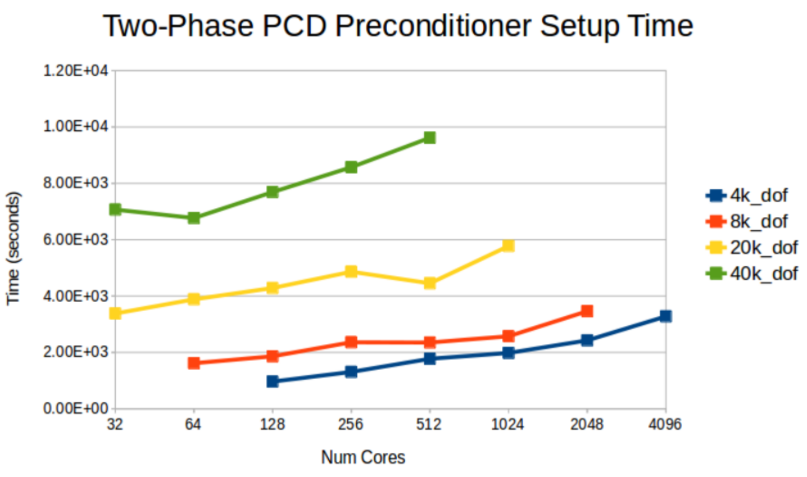
Next we consider the weak scaling results of the PCD method. One important observation from Tables 3.18-3.21 is that the weak scaling performance of the two-phase PCD preconditioner depends on the problem size. In the 4K DoF per core simulation (see Table 3.18), when the problem size is increased by a factor of 34, the total simulation run time increases by a factor of 6. In contrast, when 20K DoF per core are used and the problem size is increased by a factor of 34, the total simulation run time only increases by a factor of 1.6. Notably, this suggests that when sufficient DoF per core are used, the two-phase PCD preconditioner exhibits superior weak scaling performance relative to the ASM. However, when insufficient DoF per core are used, the ASM can actually outperform the weak scaling of the two-phase PCD.
It is also useful to note that Tables 3.18-3.21 and Figure 3-8 illustrate that the setup phase of the two-phase PCD preconditioner has better weak scaling behavior than the linear solve segment. This partial reflects the choice of the AMG settings used for the velocity and Schur complement sub-blocks. Indeed, the weak scaling behavior of the linear solver could be improved using more a expensive AMG setup. Ultimately, the AMG settings were selected to generate reliable performance across all the simulation profiles.
Overall, these results suggest that, provided sufficient DoF per processor are used, the two-phase PCD preconditioner exhibits better weak scaling performance than the ASM preconditioner. Moreoever, it is important to note that even when the number of DoF per processor are small enough for the ASM preconditioner to exhibit better weak scaling performance than the PCD preconditioner, that the PCD preconditioner is still significantly faster than the ASM preconditioner for all simulations considered.

Refinement	32	64	128	256	512	1024	2048	4096
4K	–	–	13 / 28	13 / 31	13 / 28	13 / 30	14 / 37	14 / 37
8K	–	13 / 22	13 / 27	13 / 27	13 / 31	13 / 28	14 / 33	–
10K	13 / 23	13 / 25	13 / 27	13 / 31	13 / 33	13 / 29	–	–
20K	13 / 22	13 / 23	13 / 29	13 / 27	13 / 33	–	–	–

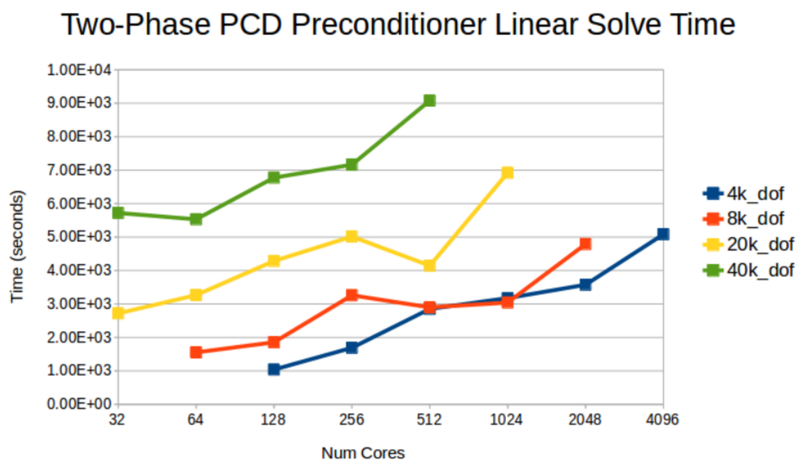
Table 3.26: GMRES Iterations (average / minimum) running the MARIN problem with the two-phase PCD preconditioner.

3.6 Conclusions

In this work we have examined several preconditioning strategies for the two-phase Navier–Stokes equations. Of particular interest has been the introduction of a new Schur complement preconditioner. One novel feature of this preconditioner is a Schur complement approximation designed to handle variable density / viscosity Navier–Stokes equations. In addition, this preconditioner has been constructed in a way to work effectively on high-performance computing clusters. As shown in this work, the design and implementation of this algorithm marks a significant improvement over the current baseline preconditioning strategy used in Proteus.



(a) Preconditioner Setup Time



(b) Linear Solve Time

Figure 3-8: Weak Scaling Results for the MARIN simulation using the two-phase PCD preconditioner.

Chapter 4

Conclusions and Future Work

In this work, we developed a computational framework for an axisymmetric linear elasticity problem and introduced and implemented a new Schur complement approach to preconditioning the variable density/viscosity two-phase Navier–Stokes equations. In the axisymmetric linear elasticity chapter, we showed that the finite element spaces $\Sigma_{h,\underline{\boldsymbol{\sigma}}} = (BDM_1), \Sigma_{h,\sigma} = (P_1), U_h = (P_0)^2, W_h = (P_0)$, and $\Sigma_{h,\underline{\boldsymbol{\sigma}}} = (BDM_2), \Sigma_{h,\sigma} = (P_2), U_h = (P_1)^2, W_h = (P_1)$ form inf-sup stable finite element pairs. For the variable density/viscosity two-phase Naiver–Stokes preconditioner, we demonstrated the methods scaling potential and implemented a global Schur complement preconditioner that exhibits a meaningful improvement over the current preconditioning approach used in Proteus.

As a departure point, this work suggests a number of avenues of future work to pursue. For the axisymmetric problem, new approaches should be explored to see whether inf-sup stability can be established for $(((\mathbf{BDM}_k)^2, P_k) \times (P_{k-1})^2 \times P_{k-1})$. Another path forward is to extend this work on the linear elasticity problem into the poroelasticity problem. As with linear elasticity, this presents another important area which the axisymmetric literature has not explored.

For the iterative linear solver work, perhaps the most important unresolved problem at this point is to better understand the role boundary conditions play in preconditioner scaling performance. The next natural problem to consider would be two and three dimensional versions of the dambreak simulation that include inflow and outflow boundary conditions. As highlighted in section 3.5.2, challenges remain in terms of scaling GMRES iterations when weakly enforced boundary conditions are used. One possible reason for this is the way Neumann conditions are treated in Proteus.

Appendix A

Axisymmetric Linear Elasticity Derivation

In this Appendix, we illustrate how using a change of variable from Cartesian to cylindrical coordinates, the axisymmetric linear elasticity problem can be expressed as the decoupled meridian and azimuthal problems. Recall that cylindrical coordinates form a triple (r, θ, z) where r is the radial distance, θ is the azimuthal coordinate and z is the vertical coordinate. In this section, let $\breve{\Omega}$ denote a three dimensional axisymmetric domain, Ω represent an (r, z) cross section of $\breve{\Omega}$ and Ω_θ denotes the domain of the θ angle.

A.0.1 Cylindrical Coordinate Operators and Function Spaces

First we define the differential forms and inner products that arise in cylindrical coordinates. To begin, the cylindrical coordinate unit vectors are denoted $\mathbf{e}_r, \mathbf{e}_\theta$ and $\mathbf{e}_z$. Expressed in terms of Cartesian unit vectors,

$$\mathbf{e}_r = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}, \quad \mathbf{e}_\theta = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{e}_z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{A.1}$$

One can note from these equations that the cylindrical coordinate unit vectors vary in space. Moreover, unless otherwise specified, we assume tensors and vectors are represented in terms of the cylindrical coordinates unit vectors. That is,

$$\begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} = \phi_1 \mathbf{e_r} + \phi_2 \mathbf{e}_\theta + \phi_3 \mathbf{e_z} \tag{A.2}$$

and

$$\begin{pmatrix} \phi_{rr} & \phi_{r\theta} & \phi_{rz} \\ \phi_{\theta r} & \phi_{\theta\theta} & \phi_{\theta z} \\ \phi_{zr} & \phi_{z\theta} & \phi_{zz} \end{pmatrix} = \phi_{rr}\mathbf{e_{rr}} + \phi_{r\theta}\mathbf{e_{r\theta}} + \phi_{zz}\mathbf{e_{rz}} + \phi_{\theta r}\mathbf{e_{\theta r}} + \phi_{\theta\theta}\mathbf{e}_{\theta\theta} + \phi_{\theta z}\mathbf{e}_{\theta z} \\ + \phi_{zr}\mathbf{e_{zr}} + \phi_{z\theta}\mathbf{e_{z\theta}} + \phi_{zz}\mathbf{e}_{zz} \tag{A.3}$$

where $\mathbf{e}_{ij} = \mathbf{e}_i \otimes \mathbf{e}_j$.

As a result of the spatially varying unit vectors, differential operators in cylindrical coordinates have a different algebraic form than in Cartesian coordinates. These operators are not derived here, but details can be found in many sources including [76].

We use two forms of notation for differential operators in cylindrical coordinates: ∇_{cyl} and ∇_{axi}. The first denotes the complete cylindrical coordinate operator, while the second represents the cylindrical coordinate operator applied to an axisymmetric function (recall that $\dfrac{\partial u}{\partial \theta} = 0$ if u is axisymmetric).

The cylindrical coordinate del operator is

$$\nabla_{\text{cyl}} = \mathbf{e}_r \frac{\partial}{\partial r} + \mathbf{e}_\theta \frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{e}_z \frac{\partial}{\partial z}. \tag{A.4}$$

Applied to the scalar function f, this gives the gradient operators

$$\nabla_{\text{cyl}} f = \frac{\partial f}{\partial r}\mathbf{e}_r + \frac{1}{r}\frac{\partial f}{\partial \theta}\mathbf{e}_\theta + \frac{\partial f}{\partial z}\mathbf{e}_z \quad \text{and} \quad \nabla_{\text{axi}} f = \frac{\partial f}{\partial r}\mathbf{e}_r + \frac{\partial f}{\partial z}\mathbf{e}_z. \tag{A.5}$$

For a vector function $\mathbf{u} = (u_r, u_\theta, u_z)^t$, the gradient tensor is

$$\nabla_{\text{cyl}}\mathbf{u} = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & \dfrac{1}{r}\dfrac{\partial u_r}{\partial \theta} - \dfrac{u_\theta}{r} & \dfrac{\partial u_r}{\partial z} \\ \dfrac{\partial u_\theta}{\partial r} & \dfrac{1}{r}\dfrac{\partial u_\theta}{\partial \theta} + \dfrac{u_r}{r} & \dfrac{\partial u_\theta}{\partial z} \\ \dfrac{\partial u_z}{\partial r} & \dfrac{1}{r}\dfrac{\partial u_z}{\partial \theta} & \dfrac{\partial u_z}{\partial z} \end{pmatrix} \quad \text{and} \quad \nabla_{\text{axi}}\mathbf{u} = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & -\dfrac{u_\theta}{r} & \dfrac{\partial u_r}{\partial z} \\ \dfrac{\partial u_\theta}{\partial r} & \dfrac{u_r}{r} & \dfrac{\partial u_\theta}{\partial z} \\ \dfrac{\partial u_z}{\partial r} & 0 & \dfrac{\partial u_z}{\partial z} \end{pmatrix}. \quad \text{(A.6)}$$

For a vector function $\mathbf{u} = (u_r, u_z)^t$, we also define the gradient operator ∇ such that

$$\nabla\mathbf{u} = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & \dfrac{\partial u_r}{\partial z} \\ \dfrac{\partial u_z}{\partial r} & \dfrac{\partial u_z}{\partial z} \end{pmatrix}. \quad \text{(A.7)}$$

The divergence operator applied to $\mathbf{u} = (u_r, u_\theta, u_z)$ gives

$$\nabla_{\text{cyl}} \cdot \mathbf{u} = \frac{1}{r}\frac{\partial (r\ u_r)}{\partial r} + \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{\partial u_z}{\partial z} \quad \text{and} \quad \nabla_{\text{axi}} \cdot \mathbf{u} = \frac{1}{r}\frac{\partial (r\ u_r)}{\partial r} + \frac{\partial u_z}{\partial z}. \quad \text{(A.8)}$$

The divergence of an $\mathbb{M}^3$ tensor $\underline{\boldsymbol{\sigma}}$ is,

$$\nabla_{\text{cyl}} \cdot \underline{\boldsymbol{\sigma}} = \begin{pmatrix} \dfrac{\partial \sigma_{rr}}{\partial r} + \dfrac{1}{r}\dfrac{\partial \sigma_{r\theta}}{\partial \theta} + \dfrac{\partial \sigma_{rz}}{\partial z} + \dfrac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) \\ \dfrac{\partial \sigma_{\theta r}}{\partial r} + \dfrac{1}{r}\dfrac{\partial \sigma_{\theta\theta}}{\partial \theta} + \dfrac{\partial \sigma_{\theta z}}{\partial z} + \dfrac{1}{r}(\sigma_{\theta r} + \sigma_{r\theta}) \\ \dfrac{\partial \sigma_{zr}}{\partial r} + \dfrac{1}{r}\dfrac{\partial \sigma_{z\theta}}{\partial \theta} + \dfrac{\partial \sigma_{zz}}{\partial z} + \dfrac{1}{r}\sigma_{zr} \end{pmatrix} \quad \text{and}$$

$$\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} = \begin{pmatrix} \dfrac{\partial \sigma_{rr}}{\partial r} + \dfrac{\partial \sigma_{rz}}{\partial z} + \dfrac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) \\ \dfrac{\partial \sigma_{\theta r}}{\partial r} + \dfrac{\partial \sigma_{\theta z}}{\partial z} + \dfrac{1}{r}(\sigma_{\theta r} + \sigma_{r\theta}) \\ \dfrac{\partial \sigma_{zr}}{\partial r} + \dfrac{\partial \sigma_{zz}}{\partial z} + \dfrac{1}{r}\sigma_{zr} \end{pmatrix}. \quad \text{(A.9)}$$

A.0.2 Meridian and Azimuthal Subspaces

Next, we assume all functions are axisymmetric and define the meridian and azimuthal subspaces for tensor and vector functions. In addition, we specify the action of the differential operators introduced in Section A.0.1 on the meridian and azimuthal subspaces.

The meridian and azimuthal subspaces of ${}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$ are

$$
{}_\alpha\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot,\ \breve{\Omega}\ ; \mathbb{M}^3) = \left\{ \underline{\boldsymbol{\sigma}} \in\ {}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)\ : \underline{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_{rr} & 0 & \sigma_{rz} \\ 0 & \sigma_{\theta\theta} & 0 \\ \sigma_{zr} & 0 & \sigma_{zz} \end{pmatrix} \right\}, \tag{A.10}
$$

$$
{}_\alpha\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3) = \left\{ \underline{\boldsymbol{\sigma}} \in\ {}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3) : \underline{\boldsymbol{\sigma}} = \begin{pmatrix} 0 & \sigma_{r\theta} & 0 \\ \sigma_{\theta r} & 0 & \sigma_{\theta z} \\ 0 & \sigma_{z\theta} & 0 \end{pmatrix} \right\}. \tag{A.11}
$$

Note that ${}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3) = {}_\alpha\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3) \oplus {}_\alpha\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$. This decomposition extends naturally to tensors in ${}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{K}^3)$ as well

$$
{}_\alpha\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{K}^3) = \left\{ \underline{\boldsymbol{\sigma}} \in {}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{K}^3) : \underline{\boldsymbol{\sigma}} = \begin{pmatrix} 0 & 0 & \sigma_{rz} \\ 0 & 0 & 0 \\ -\sigma_{rz} & 0 & 0 \end{pmatrix} \right\}, \tag{A.12}
$$

$$
{}_\alpha\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{K}^3) = \left\{ \underline{\boldsymbol{\sigma}} \in {}_\alpha\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{K}^3) : \underline{\boldsymbol{\sigma}} = \begin{pmatrix} 0 & \sigma_{r\theta} & 0 \\ -\sigma_{r\theta} & 0 & \sigma_{\theta z} \\ 0 & -\sigma_{\theta z} & 0 \end{pmatrix} \right\}. \tag{A.13}
$$

For $\underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$,

$$
\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} = \left(\frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) + \frac{\partial \sigma_{rz}}{\partial z} \right) \mathbf{e}_r + \left(\frac{\partial \sigma_{zr}}{\partial r} + \frac{1}{r}\sigma_{zr} + \frac{\partial \sigma_{zz}}{\partial z} \right) \mathbf{e}_z \tag{A.14}
$$

and for $\underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$,

$$
\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} = \left(\frac{\partial \sigma_{r\theta}}{\partial r} + \frac{\partial \sigma_{z\theta}}{\partial z} + \frac{1}{r}(\sigma_{r\theta} + \sigma_{\theta r}) \right) \mathbf{e}_\theta. \tag{A.15}
$$

The meridian and azimuthal subspaces for the displacement space ${}_1\mathbf{L}^2(\breve{\Omega})$ are

$$
{}_1\mathbf{L}^2_M(\breve{\Omega}) = \left\{ \mathbf{u} : \begin{pmatrix} u_r \\ 0 \\ u_z \end{pmatrix} \in {}_1\mathbf{L}^2(\breve{\Omega}) \right\} \quad \text{and} \quad {}_1\mathbf{L}^2_A(\breve{\Omega}) = \left\{ \mathbf{u} : \begin{pmatrix} 0 \\ u_\theta \\ 0 \end{pmatrix} \in {}_1\mathbf{L}^2(\breve{\Omega}) \right\}. \tag{A.16}
$$

For $\mathbf{u}_M \in {}_1\mathbf{L}^2_M(\breve{\Omega})$ and $\mathbf{u}_A \in {}_1\mathbf{L}^2_A(\breve{\Omega})$, the cylindrical gradient operator (A.6) has the form

$$
\nabla_{\text{axi}}\mathbf{u}_M = \begin{pmatrix} \dfrac{\partial u_r}{\partial r} & 0 & \dfrac{\partial u_r}{\partial z} \\ 0 & \dfrac{u_r}{r} & 0 \\ \dfrac{\partial u_z}{\partial r} & 0 & \dfrac{\partial u_z}{\partial z} \end{pmatrix} \quad \text{and} \quad \nabla_{\text{axi}}\mathbf{u}_A = \begin{pmatrix} 0 & \dfrac{-u_\theta}{r} & 0 \\ \dfrac{\partial u_\theta}{\partial r} & 0 & \dfrac{\partial u_\theta}{\partial z} \\ 0 & 0 & 0 \end{pmatrix}. \tag{A.17}
$$

For $\mathbf{u}_M \in {}_1\mathbf{L}^2_M(\breve{\Omega})$ and $\mathbf{u}_A \in {}_1\mathbf{L}^2_A(\breve{\Omega})$, the divergence operator (A.8) has the form

$$
\nabla_{\text{axi}} \cdot \mathbf{u}_M = \frac{1}{r}\frac{\partial_r(r u_r)}{\partial r} + \frac{\partial u_z}{\partial z} \quad \text{and} \quad \nabla_{\text{axi}} \cdot \mathbf{u}_A = 0. \tag{A.18}
$$

Because of axisymmetry, the θ variable does not appear in the meridian or azimuthal subspaces. Therefore, for functions $p, q \in {}_1L^2(\breve{\Omega})$, we define the axisymmetric cylindrical coordinate inner product as

$$
(p, q) = \frac{1}{2\pi} \iint_\Omega \int_{\theta=0}^{2\pi} p\, q\, r\, d\theta\, dr\, dz = \iint_\Omega p\, q\, r\, dr\, dz. \tag{A.19}
$$

When working with the meridian and azimuthal problems, it is helpful to use the following reduced dimensional representations of the meridian and azimuthal subspaces. To begin, elements $\mathbf{u} \in {}_1\mathbf{L}^2_M(\breve{\Omega}; \mathbb{R}^3)$, can be represented as $\mathbb{R}^2$ vectors

$$
\begin{pmatrix} u_r \\ 0 \\ u_z \end{pmatrix} \to \begin{pmatrix} u_r \\ u_z \end{pmatrix} \in {}_1\mathbf{L}^2(\Omega; \mathbb{R}^2). \tag{A.20}
$$

Elements of ${}_1\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}, \mathbb{M}^3)$ can be represented as an $\mathbb{M}^2$ tensor and a scalar function

$$\begin{pmatrix} \sigma_{rr} & 0 & \sigma_{rz} \\ 0 & \sigma_{\theta\theta} & 0 \\ \sigma_{zr} & 0 & \sigma_{zz} \end{pmatrix} \rightarrow \begin{pmatrix} \sigma_{rr} & \sigma_{rz} \\ \sigma_{zr} & \sigma_{zz} \end{pmatrix} \in {}_1\underline{\mathbf{L}}^2(\Omega, \mathbb{M}^2) \quad \text{and} \quad \sigma_{\theta\theta} \in {}_1L^2(\Omega) \tag{A.21}$$

where $\nabla_{\text{axi}} \cdot \left(\begin{pmatrix} \sigma_{rr} & \sigma_{rz} \\ \sigma_{zr} & \sigma_{zz} \end{pmatrix}, \sigma_{\theta\theta} \right) \in {}_1\mathbf{L}^2(\Omega)$.

To specify that the reduced form notation is being used, elements $\mathbf{u} \in {}_1\mathbf{L}^2_M(\breve{\Omega}; \mathbb{R}^3)$ are denoted $\mathbf{u}_M$. Further, the reduced form of $\underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}_M(\nabla\cdot, \breve{\Omega}, \mathbb{M}^3)$ is the pair $(\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta})$ where $\underline{\boldsymbol{\sigma}}_M$ is a tensor component and $\sigma_{\theta\theta}$ is a scalar component of $\underline{\boldsymbol{\sigma}}$. Moreover, $\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}) = \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}$ as defined in (A.14).

Elements of $\mathbf{u} \in {}_1\mathbf{L}^2_A(\breve{\Omega}; \mathbb{R}^3)$ can be identified with scalar functions

$$\begin{pmatrix} 0 \\ u_\theta \\ 0 \end{pmatrix} \rightarrow u_\theta \in {}_1L^2(\Omega) \tag{A.22}$$

and elements of ${}_1\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, can written as a $\mathbb{M}^2$ tensors

$$\begin{pmatrix} 0 & \sigma_{r\theta} & 0 \\ \sigma_{\theta r} & 0 & \sigma_{\theta z} \\ 0 & \sigma_{z\theta} & 0 \end{pmatrix} \rightarrow \begin{pmatrix} \sigma_{r\theta} & \sigma_{\theta r} \\ \sigma_{\theta z} & \sigma_{z\theta} \end{pmatrix} \in {}_1\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \Omega; \mathbb{M}^2). \tag{A.23}$$

To indicate the reduced form is being used, for $\mathbf{u} \in {}_1\mathbf{L}^2_A(\breve{\Omega}; \mathbb{R}^3)$, the reduced form will be expressed simply as the scalar function u_θ. Further, the reduced form of $\underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, is denoted $\underline{\boldsymbol{\sigma}}_A$.

Norms in reduced form are inherited from the norms of the original space. For example, taking $\underline{\boldsymbol{\sigma}} \in {}_1\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}, \mathbb{M}^3)$,

$$\begin{aligned}\|\underline{\boldsymbol{\sigma}}\|^2_{{}_1\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot,\breve{\Omega};\mathbb{M}^3)} &= \|(\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta})\|^2_{{}_1\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot,\Omega)} \\ &= \|\nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta})\|^2_{{}_1\underline{\mathbf{L}}^2(\Omega)} + \|(\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta})\|^2_{{}_1\mathbf{L}^2(\Omega)}.\end{aligned} \tag{A.24}$$

In the following, we take

$$\boldsymbol{\Sigma} = \{(\underline{\boldsymbol{\sigma}}, \sigma) \in {}_1\underline{\mathbf{L}}^2(\Omega, \mathbb{M}^2) \times {}_1L^2(\Omega) : \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\sigma}}, \sigma) \in {}_1L^2(\Omega)\}, \tag{A.25}$$

$$\Sigma = {}_1\underline{\mathbf{H}}(\nabla_{\text{axi}}\cdot, \Omega, \mathbb{M}^2), \tag{A.26}$$

$$U = {}_1\mathbf{L}^2(\Omega), \text{ and } Q = {}_1L^2(\Omega). \tag{A.27}$$

A.0.3 Axisymmetric Weak Form

At this point, we are ready to define the weak form of the meridian and azimuthal problems. First we note that the strong form of the axisymmetric linear elasticity problem (2.12) is

$$\mathcal{A}\underline{\boldsymbol{\sigma}} - \frac{1}{2}(\nabla_{\text{axi}}\mathbf{u} + (\nabla_{\text{axi}}\mathbf{u})^t) = 0 \text{ in } \breve{\Omega} \tag{A.28}$$

$$\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}} = \mathbf{f} \text{ in } \breve{\Omega} \tag{A.29}$$

where we use clamped boundary conditions as described in (2.14). An axisymmetric solution to (A.28) and (A.29) can be expressed in terms of the orthogonal subspaces $\underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}, \mathbb{M}^3)$ and $\underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}, \mathbb{M}^3)$, and ${}_1\mathbf{L}^2_A(\breve{\Omega}; \mathbb{R}^3)$ and ${}_1\mathbf{L}^2_M(\breve{\Omega}; \mathbb{R}^3)$.

Meridian problem

The first step to derive the meridian problem is to multiply (A.28) with a test function $\underline{\boldsymbol{\tau}} \in \underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$ and integrate. For $\underline{\boldsymbol{\sigma}} \in \underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, $\mathcal{A}\underline{\boldsymbol{\sigma}}$ has the form (recall the operator $\mathcal{A}$ (2.13) for $m = 3$)

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \frac{1}{2\mu}\begin{pmatrix} \sigma_{rr} - \frac{\lambda}{2\mu+3\lambda}\operatorname{tr}(\underline{\boldsymbol{\sigma}}) & 0 & \sigma_{rz} \\ 0 & \sigma_{\theta\theta} - \frac{\lambda}{2\mu+3\lambda}\operatorname{tr}(\underline{\boldsymbol{\sigma}}) & 0 \\ \sigma_{zr} & 0 & \sigma_{zz} - \frac{\lambda}{2\mu+3\lambda}\operatorname{tr}(\underline{\boldsymbol{\sigma}}) \end{pmatrix}. \tag{A.30}$$

Therefore, for $\underline{\boldsymbol{\tau}} \in \underline{\mathbf{H}}_M(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$,

$$\mathcal{A}\,\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} = \frac{1}{2\mu}(\sigma_{rr}\tau_{rr} + \sigma_{\theta\theta}\tau_{\theta\theta} + \sigma_{zz}\tau_{zz} + \sigma_{rz}\tau_{rz} + \sigma_{zr}\tau_{zr} - \frac{\lambda}{2\mu+3\lambda}\operatorname{tr}(\underline{\boldsymbol{\sigma}})\operatorname{tr}(\underline{\boldsymbol{\tau}})). \tag{A.31}$$

Using reduced form notation,

$$\begin{aligned} \mathcal{A}\,\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} &= \frac{1}{2\mu}(\underline{\boldsymbol{\sigma}}_M : \underline{\boldsymbol{\tau}}_M - \frac{\lambda}{2\mu+3\lambda}(\operatorname{tr}(\underline{\boldsymbol{\sigma}}_M) + \sigma_{\theta\theta})\operatorname{tr}(\underline{\boldsymbol{\tau}}_M) \\ &\qquad + \sigma_{\theta\theta}\tau_{\theta\theta} - \frac{\lambda}{2\mu+3\lambda}(\operatorname{tr}(\underline{\boldsymbol{\sigma}}_M) + \sigma_{\theta\theta})\tau_{\theta\theta}) \\ &= \mathcal{A}\underline{\boldsymbol{\sigma}}_M : \underline{\boldsymbol{\tau}}_M + \mathcal{A}\sigma_{\theta\theta}\;\tau_{\theta\theta} - \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda}(\sigma_{\theta\theta}\;\operatorname{tr}(\underline{\boldsymbol{\tau}}_M)) + \operatorname{tr}(\underline{\boldsymbol{\sigma}}_M)\tau_{\theta\theta}). \end{aligned} \tag{A.32}$$

Integrating (A.32) over Ω gives the bilinear form $a_M(\cdot,\cdot) : \boldsymbol{\Sigma} \times \boldsymbol{\Sigma} \to \mathbb{R}$,

$$\begin{aligned} a_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), (\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta})) &= (\mathcal{A}\underline{\boldsymbol{\sigma}}_M, \underline{\boldsymbol{\tau}}_M) + (\mathcal{A}\sigma_{\theta\theta}, \tau_{\theta\theta}) \\ &\qquad - \frac{1}{2\mu}\frac{\lambda}{2\mu+3\lambda}[(\sigma_{\theta\theta}, \operatorname{tr}(\underline{\boldsymbol{\tau}}_M)) + (\operatorname{tr}(\underline{\boldsymbol{\sigma}}_M), \tau_{\theta\theta})]. \end{aligned} \tag{A.33}$$

Next, observe that for $\mathbf{u} \in {}_1\mathbf{L}^2_M(\Omega)$ and $(\underline{\boldsymbol{\tau}}, \tau_{\theta\theta}) \in \boldsymbol{\Sigma}$,

$$\begin{aligned} -\int_\Omega \nabla_{\text{axi}}\mathbf{u} : \underline{\boldsymbol{\tau}}\; r\,d\Omega &= -\int_\Omega \begin{pmatrix} \frac{\partial u_r}{\partial r} & \frac{\partial u_r}{\partial z} \\ \frac{\partial u_z}{\partial r} & \frac{\partial u_z}{\partial z} \end{pmatrix} : \begin{pmatrix} \tau_{rr} & \tau_{rz} \\ \tau_{zr} & \tau_{zz} \end{pmatrix}\; r\,d\Omega - \int_\Omega \frac{u_r}{r}\tau_{\theta\theta}\; r\,d\Omega \\ &= -\int_\Omega \nabla\mathbf{u}_M : \underline{\boldsymbol{\tau}}_M\; r\,d\Omega - \int_\Omega \frac{u_r}{r}\tau_{\theta\theta}\; r\,d\Omega. \end{aligned} \tag{A.34}$$

Next, we apply integration by parts to the expression

$$
\begin{aligned}
-\int_{\Omega} \nabla \mathbf{u}_M : \underline{\boldsymbol{\tau}}_M \; r \, d\Omega = & -\int_{\partial\Omega} u_r \; (\underline{\boldsymbol{\tau}}_M)_1 \cdot \mathbf{n} \; r \, d\Omega + \int_{\Omega} u_r \; \nabla \cdot (r \; (\underline{\boldsymbol{\tau}}_M)_1) \; d\Omega \\
& - \int_{\partial\Omega} u_z (\underline{\boldsymbol{\tau}}_M)_2 \cdot \mathbf{n} \; r \, d\Omega + \int_{\Omega} u_z \; \nabla \cdot (r(\underline{\boldsymbol{\tau}}_M)_2) d\Omega.
\end{aligned}
\tag{A.35}
$$

As we are integrating over the domain Ω, the boundary $\partial\Omega$ is comprised of two parts. The first corresponds to the boundary of the entire three dimensional domain $\partial\Omega$ upon which clamped displacement condition $\mathbf{u}_M = \mathbf{0}$ is enforced. The second part of the boundary Γ_0 corresponds to the symmetry axis along which the conditions $u_r = 0$, $(\underline{\boldsymbol{\tau}}_M)_1 \cdot \mathbf{n} = 0$ and $(\underline{\boldsymbol{\tau}}_M)_2 \cdot \mathbf{n} = 0$. Therefore, all of the boundary integrals in (A.35) vanish so that

$$
\begin{aligned}
-\int_{\Omega} \nabla \mathbf{u}_M : \underline{\boldsymbol{\tau}}_M \; r \, d\Omega &= \int_{\Omega} u_r \; \nabla \cdot (r \; (\underline{\boldsymbol{\tau}}_M)_1) \; d\Omega + \int_{\Omega} u_z \; \nabla \cdot (r(\underline{\boldsymbol{\tau}}_M)_2) d\Omega \\
&= \int_{\Omega} \mathbf{u}_M \cdot \nabla_{\text{axi}} \cdot (\underline{\boldsymbol{\tau}}_M) \; r \, d\Omega.
\end{aligned}
\tag{A.36}
$$

Thus from (A.34) and (A.36) from we define the bilinear form $b_M(\cdot,\cdot) : \boldsymbol{\Sigma} \times U \to \mathbb{R}$ as

$$
b_M((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), \mathbf{u}_M) = (\mathbf{u}_M, \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}_M) - (u_r, \frac{\tau_{\theta\theta}}{r}). \tag{A.37}
$$

For $(\underline{\boldsymbol{\sigma}}, \sigma_{\theta\theta} \in \boldsymbol{\Sigma}$, multiplying the left hand side of (A.29) with a test function $\mathbf{v} \in {}_1\mathbf{L}^2_M(\Omega)$ gives

$$
\begin{aligned}
(\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}) \cdot \mathbf{v} &= (\partial_r \sigma_{rr} + \frac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) + \partial_z \sigma_{rz}) v_r + (\partial_r \sigma_{zr} + \frac{1}{r} \sigma_{zr} + \partial_z \sigma_{zz}) v_z \\
&= (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}_M) \cdot \mathbf{v}_M - \frac{1}{r} \sigma_{\theta\theta} v_r.
\end{aligned}
\tag{A.38}
$$

From integrating this expression we define the bilinear form

$$
b_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), \mathbf{v}_M) = ((\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}_M), \mathbf{v}_M) - (v_r, \frac{\sigma_{\theta\theta}}{r}). \tag{A.39}
$$

Finally, multiplying the right hand side of (A.29) with a test function $\mathbf{v} \in {}_1L^2_M(\Omega)$ and integrating, defines the linear functional $(\mathbf{f}_M, \mathbf{v}_M)$.
The meridian problem can now be defined as: Find $((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), \mathbf{u}_M) \in \boldsymbol{\Sigma} \times U$ such that

$$a_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), (\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta})) + b_M((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), \mathbf{u}_M) = 0 \tag{A.40}$$

$$b_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), \mathbf{v}_M) = (\mathbf{f}_M, \mathbf{v}_M)_M \tag{A.41}$$

for all $((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), \mathbf{v}_M) \in \boldsymbol{\Sigma} \times U$.
For the weak symmetry constraint (recall (2.19)), we define the bilinear form $c_M(.,.) : \boldsymbol{\Sigma} \to \mathbb{R}$

$$c_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), p) = (\underline{\boldsymbol{\rho}}_M, p). \tag{A.42}$$

The meridian problem with weak symmetry is: Find $((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), \mathbf{u}_M, p) \in \boldsymbol{\Sigma} \times U \times Q$ such that

$$a_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), (\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta})) + b_M((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), \mathbf{u}_M) + c_M((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), p) = 0 \tag{A.43}$$

$$b_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), \mathbf{v}_M) = (\mathbf{f}, \mathbf{v}_M)_M \tag{A.44}$$

$$c_M((\underline{\boldsymbol{\sigma}}_M, \sigma_{\theta\theta}), q) = 0 \tag{A.45}$$

for all $((\underline{\boldsymbol{\tau}}_M, \tau_{\theta\theta}), \mathbf{v}_M, q) \in \boldsymbol{\Sigma} \times U \times Q$.

Azimuthal Problem

Finally we consider the azimuthal problem. Starting with (A.28) and following the standard variational approach, we multiply with a test function $\underline{\boldsymbol{\tau}} \in \underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$ and integrate over Ω.

Recall that for $\underline{\boldsymbol{\sigma}} \in \underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, $\mathcal{A}\underline{\boldsymbol{\sigma}}$ has the form

$$\mathcal{A}\underline{\boldsymbol{\sigma}} = \frac{1}{2\mu} \begin{pmatrix} 0 & \sigma_{r\theta} & 0 \\ \sigma_{\theta r} & 0 & \sigma_{\theta z} \\ 0 & \sigma_{z\theta} & 0 \end{pmatrix}. \tag{A.46}$$

Thus, for all $\underline{\boldsymbol{\tau}} \in \underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, the first term of (A.28) is

$$\mathcal{A}\ \underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} = \frac{1}{2\mu}(\sigma_{r\theta}\tau_{r\theta} + \sigma_{\theta r}\tau_{\theta r} + \sigma_{\theta z}\tau_{\theta z} + \sigma_{z\theta}\tau_{z\theta}), \tag{A.47}$$

and using reduced form notation, $\mathcal{A}\underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{\tau}} = \frac{1}{2\mu}\underline{\boldsymbol{\sigma}}_A : \underline{\boldsymbol{\tau}}_A$. Integrating (A.47) defines the bilinear form $a_A(\cdot,\cdot) : \Sigma \times \Sigma \to \mathbb{R}$,

$$a_A(\underline{\boldsymbol{\sigma}}_A, \underline{\boldsymbol{\tau}}_A) = \frac{1}{2\mu}(\underline{\boldsymbol{\sigma}}_A, \underline{\boldsymbol{\tau}}_A)_A. \tag{A.48}$$

For the second term in (A.28), taking $\mathbf{u} \in {}_1\mathbf{L}^2_A(\breve{\Omega})$ and $\underline{\boldsymbol{\tau}} \in \underline{\mathbf{H}}_A(\nabla_{\text{axi}}\cdot, \breve{\Omega}; \mathbb{M}^3)$, then integrating by parts gives

$$\begin{aligned} -\int_\Omega \nabla_{\text{axi}}\mathbf{u} : \underline{\boldsymbol{\tau}}\ r\ d\Omega &= -\int_\Omega \begin{pmatrix} 0 & \dfrac{-u_\theta}{r} & 0 \\ \dfrac{\partial u_\theta}{\partial r} & 0 & \dfrac{\partial u_\theta}{\partial z} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & \tau_{r\theta} & 0 \\ \tau_{\theta r} & 0 & \tau_{\theta z} \\ 0 & \tau_{z\theta} & 0 \end{pmatrix} r\ d\Omega \\ &= -\int_\Omega \nabla u_\theta \cdot \begin{pmatrix} \tau_{\theta r} \\ \tau_{\theta z} \end{pmatrix} r\ d\Omega + \int_\Omega \frac{\tau_{r\theta}\ u_\theta}{r}\ r\ d\Omega. \end{aligned} \tag{A.49}$$

Integrating the first term by parts

$$-\int_\Omega \nabla u_\theta \cdot \begin{pmatrix} \tau_{\theta r} \\ \tau_{\theta z} \end{pmatrix} r\ d\Omega = -\int_{\partial\Omega} u_\theta \begin{pmatrix} \tau_{\theta r} \\ \tau_{\theta z} \end{pmatrix} \cdot \mathbf{n}\ r\ \partial\Omega + \int_\Omega u_\theta\ \nabla \cdot \left(r \begin{pmatrix} \tau_{\theta r} \\ \tau_{\theta z} \end{pmatrix} \right) d\Omega. \tag{A.50}$$

Since $\underline{\boldsymbol{\tau}} \cdot \mathbf{n} = 0$ on $\partial\Omega$, it follows that

$$-\int_\Omega \nabla_{\text{axi}}\mathbf{u} : \underline{\boldsymbol{\tau}}\ r\ d\Omega = \int_\Omega \mathbf{u} \cdot (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}})\ r\ d\Omega. \tag{A.51}$$

Using the reduced form notation,

$$\begin{aligned} \int_\Omega \mathbf{u} \cdot (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}})\ r\ d\Omega &= \int_\Omega u_\theta\ (\partial_r \tau_{r\theta} + \partial_z \tau_{z\theta} + \frac{1}{r}(\tau_{r\theta} + \tau_{\theta r}))\ r\ d\Omega \\ &= \int_\Omega u_\theta\ \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}_A\ r\ d\Omega. \end{aligned} \tag{A.52}$$

This defines the bilinear form $b_A(.,.) : Q \times \Sigma \to \mathbb{R}$ as

$$b_A(u_\theta, \underline{\boldsymbol{\tau}}_A) = (u_\theta, \nabla_{\text{axi}} \cdot \underline{\boldsymbol{\tau}}_A). \tag{A.53}$$

For $\underline{\boldsymbol{\sigma}} \in \underline{\mathbf{H}}_A(\nabla\cdot, \breve{\Omega}, \mathbb{M}^3)$, multiplying the left hand side of (A.29) with a test function $\mathbf{v} \in_1 \mathbf{L}^2_A(\breve{\Omega})$ and integrating over Ω gives

$$\begin{aligned} \int_\Omega (\nabla_{\text{axi}} \cdot \underline{\boldsymbol{\sigma}}) \cdot \mathbf{v}\ r\ d\Omega &= \int_\Omega (\partial_r \sigma_{r\theta} + \partial_z \sigma_{z\theta} + \frac{1}{r}(\sigma_{r\theta} + \sigma_{\theta r}))v_\theta\ r\ d\Omega \\ &= b_A(v_\theta, \underline{\boldsymbol{\sigma}}_A). \end{aligned} \tag{A.54}$$

Therefore, the weak form of the azimuthal problem can be defined as: Find $(\underline{\boldsymbol{\sigma}}_A, u_A) \in \Sigma \times Q$ such that

$$a_A(\underline{\boldsymbol{\sigma}}_A, \underline{\boldsymbol{\tau}}_A) + b_A(u, \underline{\boldsymbol{\tau}}_A) = 0 \tag{A.55}$$

$$b_A(v, \underline{\boldsymbol{\sigma}}_A) = (f_\theta, v) \tag{A.56}$$

for all $(\underline{\boldsymbol{\tau}}_A, v) \in \Sigma \times Q$.

Appendix B

Discrete Operator Forms

In this Appendix, we describe how one can derive discrete matrix representations from continuous differential operators. In Section 3.4.4, it is stated that

$$\begin{aligned} \mathcal{B} &= \mathbf{Q}^{-1}B^T, \quad \mathcal{B}^* = Q^{-1}B \\ \mathcal{L} &= \mathbf{Q}^{-1}\mathbf{F}, \quad \mathcal{L}_p = Q^{-1}F_p \end{aligned} \tag{B.1}$$

where $\mathcal{B}$ and $\mathcal{B}^*$ are the gradient and divergence operators respectively and $\mathcal{L}$ and $\mathcal{L}_p$ are advection-diffusion operators for the velocity and pressure space respectively.
To understand where these discrete representations come from, we examine $\mathcal{B}$ and $\mathcal{B}^*$ in more depth. First, let the discrete velocity space be denoted as $\mathbf{V}^h$ and M^h denote the discrete pressure space. Moreover, let $\{\vec{\phi}_i\}_{i=1}^{n_u}$ denote a basis for $\mathbf{V}^h$ and let $\{\psi_i\}_{i=1}^{n_p}$ denote a basis for M^h. Next, consider the discrete negative divergence operator $\mathcal{D}_h : \mathbf{V}^h \to M^h$ which satisfies

$$(\mathcal{D}_h \mathbf{u}_h, q_h) = (-\nabla \cdot \mathbf{u}_h, q_h) \text{ for all } q_h \in M^h. \tag{B.2}$$

Moreover, we can express $\mathbf{u}_h \in \mathbf{V}^h$, and $p_h = \mathcal{D}_h \mathbf{u}_h$ in terms of finite element basis functions

$$\mathbf{u}_h = \sum_{i=1}^{n_u} \mathbf{u}_i \vec{\phi}_i, \text{ and } \quad p_h = \sum_{i=1}^{n_p} \mathbf{p}_i \psi_i. \tag{B.3}$$

If we let B denote the matrix representation of (B.2), then row j of $B\mathbf{u}$ becomes

$$\begin{aligned}
[B\mathbf{u}]_j &= -\sum_{i=1}^{n_u} \mathbf{u}_i(\nabla \cdot \vec{\phi}_i, \psi_j) = (-\nabla \cdot \sum_{i=1}^{n_u} \mathbf{u}_i\vec{\phi}_i, \psi_j) \\
&= (\mathcal{D}_h\mathbf{u}_h, \psi_j) = (p_h, \psi_j) \\
&= (\sum_{i=1}^{n_p} \mathbf{p}_i\psi_i, \psi_j) = \sum_{i=1}^{n_p} \mathbf{p}_i(\psi_i, \psi_j) \\
&= [Q\mathbf{p}]_j.
\end{aligned}$$

Since this holds for an arbitrary row j, it follows that $B\mathbf{u} = Q\mathbf{p}$. In other words, $\mathcal{B}^* := \nabla\cdot = Q^{-1}B$.

Similarly, the discrete gradient operator $\mathcal{G}_h : M_h \to \mathbf{V}^h$ satisfies

$$(\mathcal{G}_h p_h, \mathbf{v}_h) = (\nabla p_h, \mathbf{v}_h) = -(p_h, \nabla \cdot \mathbf{v}_h) \text{ for all } \mathbf{v}_h \in \mathbf{V}^h. \tag{B.4}$$

For a given p_h, let $\mathbf{u}_h = \mathcal{G}_h p_h$, where $\mathbf{u}_h$ and p_h are expressed as in (B.3). Expanding row j of $B^T\mathbf{p}$ gives

$$\begin{aligned}
[B^T\mathbf{p}]_j &= -\sum_{i=1}^{n_p} \mathbf{p}_i(\psi_i, \nabla \cdot \vec{\phi}_j) = (\nabla \sum_{i=1}^{n_p} \mathbf{p}_i\psi_i, \vec{\phi}_j) \\
&= (\nabla p_h, \vec{\phi}_j) = (u_h, \vec{\phi}_j) \\
&= (\sum_{i=1}^{n_u} \mathbf{u}_i\vec{\phi}_i, \vec{\phi}_j) = \sum_{i=1}^{n_u} \mathbf{u}_i(\vec{\phi}_i, \vec{\phi}_j) \\
&= [\mathbf{Q}\mathbf{u}]_j.
\end{aligned}$$

Since this holds for an arbitrary row j, it follows that $B^T\mathbf{p} = \mathbf{Q}\mathbf{u}$, or $\mathcal{B} := \nabla = \mathbf{Q}^{-1}B^T$.

Bibliography

[1] N. Ahmed, C. Bartsch, V. John, and U. Wilbrandt, *An assessment of some solvers for saddle point problems emerging from the incompressible Navier–Stokes equations*, Comp. Meth. Appl. Mech. Engin., 331 (2018), pp. 492–513.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *MUMPS: a general purpose distributed memory sparse solver*, in Applied Parallel Computing. New Paradigms for HPC in Industry and Academia, T. Sorevik, F. Manne, A. Gebremedhin, and R. Moe, eds., no. 1947 in Lecture Notes in Computer Science, Springer,Berlin,Heidelberg, 2001, pp. 986–985.

[3] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and X. S. Li, *Analysis and comparison of two general sparse solvers for distributed memory computers*, ACM Trans. Math. Software, 27 (2001), pp. 388–421.

[4] V. Anaya, D. Mora, C. Reales, and R. Ruiz-Baier, *Stabilized mixed approximation of axisymmetric Brinkman flows*, ESAIM Math. Model. Numer. Anal., 49 (2015), pp. 855–874.

[5] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2001/02), pp. 1749–1779.

[6] D. N. Arnold, F. Brezzi, and J. Douglas, Jr., *PEERS: a new mixed finite element for plane elasticity*, Japan J. Appl. Math., 1 (1984), pp. 347–367.

[7] D. N. Arnold, J. Douglas, Jr., and C. P. Gupta, *A family of higher order mixed finite element methods for plane elasticity*, Numer. Math., 45 (1984), pp. 1–22.

[8] D. N. Arnold and R. Winther, *Mixed finite elements for elasticity*, Numer. Math., 92 (2002), pp. 401–419.

[9] F. Assous, P. Ciarlet, Jr., and S. Labrunie, *Theoretical tools to solve the axisymmetric Maxwell equations*, Math. Methods Appl. Sci., 25 (2002), pp. 49–78.

[10] F. Assous, P. Ciarlet, Jr., S. Labrunie, and J. Segré, *Numerical solution to the time-dependent Maxwell equations in axisymmetric singular domains: the singular complement method*, J. Comput. Phys., 191 (2003), pp. 147–176.

[11] O. Axelsson, X. He, and M. Neytcheva, *Numerical solution of the time-dependent Navier-Stokes equation for variable density–variable viscosity. Part I*, Math. Model. Anal., 20 (2015), pp. 232–260.

[12] A. Baker, R. D. Falgout, T. Kolev, and U. Yang, *Scaling Hypres Multigrid Solvers to 100,000 Cores*, High-Performance Scientific Computing, Springer, London, 10 2012, pp. 261–279.

[13] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, L. McInnes, K. Rupp, B. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc users manual.* http://www.mcs.anl.gov/petsc/petsc-current/docs/manual.pdf, 2016.

[14] A. T. Barker and X.-C. Cai, *Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling*, J. Comput. Phys., 229 (2010), pp. 642–659.

[15] Y. Bazilevs and T. J. R. Hughes, *Weak imposition of Dirichlet boundary conditions in fluid mechanics*, Comput. & Fluids, 36 (2007), pp. 12–26.

[16] Z. Belhachmi, C. Bernardi, and S. Deparis, *Weighted Clément operator and application to the finite element discretization of the axisymmetric Stokes problem*, Numer. Math., 105 (2006), pp. 217–247.

[17] A. Bentley, *Explicit construction of computational bases for RT_k and BDM_k spaces in $\mathbb{R}^3$*, Comput. Math. Appl., 73 (2017), pp. 1421–1432.

[18] M. Benzi and M. A. Olshanskii, *Field-of-values convergence analysis of augmented Lagrangian preconditioners for the linearized Navier-Stokes problem*, SIAM J. Numer. Anal., 49 (2011), pp. 770–788.

[19] M. Benzi, M. A. Olshanskii, and Z. Wang, *Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 66 (2011), pp. 486–508.

[20] A. Bermúdez, C. Reales, R. Rodríguez, and P. Salgado, *Numerical analysis of a finite-element method for the axisymmetric eddy current model of an induction furnace*, IMA J. Numer. Anal., 30 (2010), pp. 654–676.

[21] C. Bernardi, M. Dauge, and Y. Maday, *Spectral methods for axisymmetric domains*, vol. 3 of Series in Applied Mathematics (Paris), Gauthier-Villars, Éditions Scientifiques et Médicales Elsevier, Paris; North-Holland, Amsterdam, 1999. Numerical algorithms and tests due to Mejdi Azaïez.

[22] P. B. Bochev, C. R. Dohrmann, and M. D. Gunzburger, *Stabilization of low-order mixed finite elements for the Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 82–101.

[23] D. Boffi, F. Brezzi, and M. Fortin, *Reduced symmetry elements in linear elasticity*, Commun. Pure Appl. Anal., 8 (2009), pp. 95–121.

[24] D. Boffi, F. Brezzi, and M. Fortin, *Mixed finite element methods and applications*, vol. 44 of Springer Series in Computational Mathematics, Springer, Heidelberg, 2013.

[25] N. Bootland, A. Bentley, C. Kees, and A. Wathen, *Preconditioners for Two-Phase Incompressible Navier–Stokes Flow*, SIAM J. Sci. Comput., 41 (2019), pp. B843–B869.

[26] S. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods*, vol. 15 of Texts in Applied Mathematics, Springer-Verlag New York, 3 ed., 2008.

[27] F. Brezzi, *On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers*, Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge, 8 (1974), pp. 129–151.

[28] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*, vol. 15 of Springer Series in Computational Mathematics, Springer-Verlag, New York, 1991.

[29] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, and L. C. Wilcox, *Parallel scalable adjoint-based adaptive solution of variable-viscosity Stokes flow problems*, Comput. Methods Appl. Mech. Engrg., 198 (2009), pp. 1691–1700.

[30] J. Cahouet and J.-P. Chabard, *Some fast 3D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[31] C. Chen, H. Pouransari, S. Rajamanickam, E. G. Boman, and E. Darve, *A distributed-memory hierarchical solver for general sparse linear systems*, Parallel Comput., 74 (2018), pp. 49–64.

[32] P. Ciarlet, Jr., B. Jung, S. Kaddouri, S. Labrunie, and J. Zou, *The Fourier singular complement method for the Poisson problem. II. Axisymmetric domains*, Numer. Math., 102 (2006), pp. 583–610.

[33] R. Codina, *A stabilized finite element method for generalized stationary incompressible flows*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 2681–2706.

[34] A. Collagrossi and M. Landrini, *Numerical simulation of interfacial flows by smoothed particle hydrodynamics*, J. Comput. Phys., 191 (2003), pp. 448–475.

[35] D. M. Copeland, J. Gopalakrishnan, and J. E. Pasciak, *A mixed method for axisymmetric div-curl systems*, Math. Comp., 77 (2008), pp. 1941–1965.

[36] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, *A survey of direct methods for sparse linear systems*, Acta Numer., 25 (2016), pp. 383–566.

[37] H. De Sterck, R. D. Falgout, J. W. Nolting, and U. M. Yang, *Distance-two interpolation for parallel algebraic multigrid*, Numer. Linear Algebra Appl., 15 (2008), pp. 115–139.

[38] H. De Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039.

[39] S. Deparis, D. Forti, G. Grandperrin, and A. Quarteroni, *FaCSI: A block parallel preconditioner for fluid-structure interaction in hemodynamics*, J. Comput. Phys., 327 (2016), pp. 700–718.

[40] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2 ed., 2014.

[41] H. C. Elman, D. Loghin, and A. J. Wathen, *Preconditioning techniques for Newton's method for the incompressible Navier-Stokes equations*, BIT, 43 (2003), pp. 961–974.

[42] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, vol. 159 of Applied Mathematical Sciences, Springer-Verlag, New York, 2004.

[43] V. Ervin and E. Jenkins, *Stenberg's sufficiency condition for axisymmetric Stokes flow*, tech. report, Clemson University. http://www.clemson.edu/science/departments/mathematical-sciences/about/technical-reports.html.

[44] V. J. Ervin, *Computational bases for RT_k and BDM_k on triangles*, Comput. Math. Appl., 64 (2012), pp. 2765–2774.

[45] V. J. Ervin, *Approximation of axisymmetric Darcy flow using mixed finite element methods*, SIAM J. Numer. Anal., 51 (2013), pp. 1421–1442.

[46] V. J. Ervin, *Approximation of coupled Stokes-Darcy flow in an axisymmetric domain*, Comput. Methods Appl. Mech. Engrg., 258 (2013), pp. 96–108.

[47] R. D. Falgout, J. E. Jones, and U. M. Yang, *The design and implementation of hypre, a library of parallel high performance preconditioners*, in Numerical solution of partial differential equations on parallel computers, vol. 51 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2006, pp. 267–294.

[48] R. S. Falk, *Finite element methods for linear elasticity*, in Mixed Finite Elements, Compatibility Conditions, and Applications. Lecture Notes in Mathematics, D. Boffi and L. Gastaldi, eds., Spring-Verlag Berlin Heidelberg, 2008, pp. 159–194.

[49] M. Farhloul and M. Fortin, *Dual hybrid methods for the elasticity and the Stokes problems: a unified approach*, Numer. Math., 76 (1997), pp. 419–440.

[50] G. P. Galdi, *An introduction to the mathematical theory of the Navier-Stokes equations*, Springer Monographs in Mathematics, Springer, New York, second ed., 2011.

[51] P. P. Grinevich and M. A. Olshanskii, *An iterative method for the Stokes-type problem with variable viscosity*, SIAM J. Sci. Comput., 31 (2009), pp. 3959–3978.

[52] S. Gross, V. Reichelt, and A. Reusken, *A finite element based level set method for two-phase incompressible flows*, Comput. Vis. Sci., 9 (2006), pp. 239–257.

[53] J. Guermond, P. Minev, and J. Shen, *An overview of projection methods for incompressible flow*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 6011–6045.

[54] T. J. R. Hughes, *Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods*, Comput. Methods Appl. Mech. Engrg., 127 (1995), pp. 387–401.

[55] T. J. R. Hughes, M. Mallet, and A. Mizukami, *A new finite element formulation for computational fluid dynamics. II. Beyond SUPG*, Comput. Methods Appl. Mech. Engrg., 54 (1986), pp. 341–355.

[56] C. Johnson and B. Mercier, *Some equilibrium finite element methods for two-dimensional elasticity problems*, Numer. Math., 30 (1978), pp. 103–116.

[57] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier-Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[58] C. E. Kees, I. Akkerman, M. W. Farthing, and Y. Bazilevs, *A conservative level set method suitable for variable-order approximations and unstructured meshes*, J. Comput. Phys., 230 (2011), pp. 4536–4558.

[59] K. M. T. Kleefsman, G. Fekken, A. E. P. Veldman, B. Iwanowski, and B. Buchner, *A volume-of-fluid based simulation method for wave impact problems*, J. Comput. Phys., 206 (2005), pp. 363–393.

[60] F. Kong and X.-C. Cai, *A highly scalable multilevel Schwarz method with boundary geometry preserving coarse spaces for 3D elasticity problems on domains with complex geometry*, SIAM J. Sci. Comput., 38 (2016), pp. C73–C95.

[61] F. Kong and X.-C. Cai, *A scalable nonlinear fluid-structure interaction solver based on a Schwarz preconditioner with isogeometric unstructured coarse spaces in 3D*, J. Comput. Phys., 340 (2017), pp. 498–518.

[62] I. N. Konshin, M. A. Olshanskii, and Y. V. Vassilevski, *ILU preconditioners for nonsymmetric saddle-point matrices with application to the incompressible Navier-Stokes equations*, SIAM J. Sci. Comput., 37 (2015), pp. A2171–A2197.

[63] Y.-J. Lee and H. Li, *On stability, accuracy, and fast solvers for finite element approximations of the axisymmetric Stokes problem by Hood-Taylor elements*, SIAM J. Numer. Anal., 49 (2011), pp. 668–691.

[64] X. S. Li, *An overview of SuperLU: algorithms, implementation, and user interface*, ACM Trans. Math. Software, 31 (2005), pp. 302–325.

[65] X. S. Li and J. W. Demmel, SuperLU_DIST*: a scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29 (2003), pp. 110–140.

[66] D. Logan, *A First Course In The Finite Element Method*, Cengage Learning, 5 ed., 2012.

[67] S. Lungten, W. H. Schilders, and J. M. Maubach, *Threshold incomplete factorization constraint preconditioners for saddle-point matrices*, Linear Algebra and its Applications, 545 (2018), pp. 76–107.

[68] D. A. May, J. Brown, and L. Le Pourhiet, *A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow*, Comput. Methods Appl. Mech. Engrg., 290 (2015), pp. 496–523.

[69] B. Mercier and G. Raugel, *Résolution d'un problème aux limites dans un ouvert axisymétrique par éléments finis en r, z et séries de Fourier en θ*, RAIRO Anal. Numér., 16 (1982), pp. 405–461.

[70] M. E. Morley, *A family of mixed finite elements for linear elasticity*, Numer. Math., 55 (1989), pp. 633–666.

[71] J. Moulin, P. Jolivet, and O. Marquet, *Augmented Lagrangian preconditioner for large-scale hydrodynamic stability analysis*, Comp. Meth. Appl. Mech. Engin., 351 (2019), pp. 718–743.

[72] M. Oh, *A new approach to the analysis of axisymmetric problems*, IMA J. Numer. Anal., 34 (2014), pp. 1686–1700.

[73] M. A. Olshanskii, J. Peters, and A. Reusken, *Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations*, Numer. Math., 105 (2006), pp. 159–191.

[74] M. A. Olshanskii and A. Reusken, *Analysis of a Stokes interface problem*, Numer. Math., 103 (2006), pp. 129–149.

[75] M. A. Olshanskii and Y. V. Vassilevski, *Pressure Schur complement preconditioners for the discrete Oseen problem*, SIAM J. Sci. Comput., 29 (2007), pp. 2686–2704.

[76] L. Quartapelle, *Numerical solution of the incompressible Navier-Stokes equations*, vol. 113 of International Series of Numerical Mathematics, Birkhäuser Verlag, Basel, 1993.

[77] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[78] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.

[79] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[80] F. Shakib, T. J. R. Hughes, and Z. Johan, *A new finite element formulation for computational fluid dynamics. X. The compressible Euler and Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 89 (1991), pp. 141–219. Second World Congress on Computational Mechanics, Part I (Stuttgart, 1990).

[81] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems. II. Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[82] R. Stenberg, *On the construction of optimal mixed finite element methods for the linear elasticity problem*, Numer. Math., 48 (1986), pp. 447–462.

[83] R. Stenberg, *A family of mixed finite elements for the elasticity problem*, Numer. Math., 53 (1988), pp. 513–538.

[84] K. Stüben, *Algebraic multigrid (AMG): an introduction with applications*, in Multigrid, U. Trottenberg, C. Oosterlee, and A. Schüller, eds., Academic Press, 2001.

[85] K. Stüben, *A review of algebraic multigrid*, J. Comput. Appl. Math., 128 (2001), pp. 281–309. Numerical analysis 2000, Vol. VII, Partial differential equations.

[86] T. E. Tezduyar, *Stabilized finite element formulations for incompressible flow computations*, in Advances in applied mechanics, Vol. 28, vol. 28 of Adv. Appl. Mech., Academic Press, Boston, MA, 1992, pp. 1–44.

[87] L. N. Trefethen and D. Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[88] M. ur Rehman, C. Vuik, and G. Segal, *SIMPLE-type preconditioners for the Oseen problem*, Internat. J. Numer. Methods Fluids, 61 (2009), pp. 432–452.

[89] S. P. Vanka, *Block-implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.

[90] F. Veubeke, *A conforming finite element for plate bending*, Int. J. Solids Structures, 4 (1968), pp. 95–108.

[91] F. Veubeke, *Displacement and equilibrium models in the finite element method*, Int. J. for Numer. Meth. Engng, 52 (2001), pp. 287–342.

[92] Y. Wu and X.-C. Cai, *A fully implicit domain decomposition based ALE framework for three-dimensional fluid-structure interaction with application in blood flow computation*, J. Comput. Phys., 258 (2014), pp. 524–537.

[93] J. Xu and L. Zikatanov, *Algebraic multigrid methods*, Acta Numer., 26 (2017), pp. 591–721.

[94] U. M. Yang, *Parallel algebraic multigrid methods—high performance preconditioners*, in Numerical solution of partial differential equations on parallel computers, vol. 51 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2006, pp. 209–236.

[95] Z. Q. Zhou, J. O. De Kat, and B. Buchner, *A nonlinear 3-D approach to simulate green water dynamics on deck*, in Proc. 7th Int. Conf. Num. Ship. Hydrod., J. Piquet, ed., Nantes, 1999, pp. 5.1–1, 15.