Title:
Virtual Node Methods for Incompressible Flow

Author:
Howes, Russell

Acceptance Date:
2014

Series:
UCLA Electronic Theses and Dissertations

Degree:
Ph.D., Mathematics 0540UCLA

Advisor(s):
Teran, Joseph

Committee:
Klug, William S., Vese, Luminita, Anderson, Chris

Permalink:
http://escholarship.org/uc/item/49w3s6fr

Abstract:

UNIVERSITY OF CALIFORNIA
Los Angeles

Virtual Node Methods for Incompressible Flow

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Russell Edward Howes

2014

ABSTRACT OF THE DISSERTATION

Virtual Node Methods for Incompressible Flow

by

Russell Edward Howes

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2014

Professor Joseph Teran, Chair

This thesis details two numerical methods for the solution of incompressible flow problems using the virtual node framework introduced in [1]. The first method is a novel discrete Hodge decomposition for velocity fields defined over irregular domains in two and three dimensions. This new decomposition leads to a sparse, 5-point stencil in 2D (7-point in 3D) at all nodes in the domain, even near the boundary. The corresponding linear system can be factored simply into a weighted product of the standard discrete divergence and gradient operators, is symmetric positive definite, and yields second order accurate pressures and first order velocities (second order in L^1).

The second method is an extension of the work in [2], which simulates the Stokes equations in two dimensions, to a method that models the Navier-Stokes equations in two and three spatial dimensions. The extension to three dimensions is partially accomplished by a new approach to discretizing the multiplier term corresponding to the system jump conditions. This method works either on domains with interfacial discontinuities in material quantities such as density and viscosity, or on irregularly shaped domains with Dirichlet, Neumann, or slip boundary conditions. This method leads to a discrete, KKT system solving for velocities and pressures simultaneously, and yields second order accurate velocities in both time and space, and first order pressures.

The dissertation of Russell Edward Howes is approved.

William S. Klug

Luminita Vese

Chris Anderson

Joseph Teran, Committee Chair

University of California, Los Angeles

2014

To Randall and Marcile, who never shied
away from telling teachers and principals
how to properly educate their children

TABLE OF CONTENTS

List of Figures

List of Tables

ACKNOWLEDGMENTS

My mother and father, Marcile and Randall, deserve primary acknowledgment for their love, effort, and sacrifice in raising, teaching, and seeking out the best for me. I am grateful to to the rest of my family for their love and support, particularly my grandparents and especially my grandfather Raymond Howes, who taught me the importance of hard work.

Numerous members of the math department at BYU, particularly Lynn Garner, Lonette Stoddard, Adrian Stanger, Darrin Doud, and Tyler Jarvis, showed great faith in me and encouraged me to involve myself in mathematics research. Without such encouragement I would not have ended up at UCLA. Sean Warnick and Jeff Humpherys each showed me different, valuable approaches to both scientific research and life; I am grateful to still be in touch with them.

Joseph Teran has been a wonderful advisor, teacher, and mentor. The source paper for Chapter 5 is a joint effort [3] with Craig Schroeder and Joseph Teran and the source for Chapter 6 is joint work [4] with Craig Schroeder, Alexey Stomakhin, and Joseph Teran. It has been a pleasure working with and learning from Joey, Craig, and Alexey, who are all incredibly good at what they do. My work on these papers was partially supported by NSF Grant DGE-1144087 and by the United States Department of Defense (through the National Defense Science & Engineering Graduate Fellowship Program).

Coming to UCLA and learning from some of the world's best mathematicians and scientists is a true blessing in my life. Andrea Bertozzi has been a constant source of wisdom, advice, and support. I would like to thank Tom Chou, Dolores Bozovic, Alan Garfinkel for their time in mentoring me and sharing with me their exciting research and its relevance to my field of inquiry. The members of my doctoral committee, Chris Anderson, Luminita Vese, and William Klug, have given time and effort to reading my thesis and have been great teachers and mentors during my time at UCLA. Maggie Albert, Martha Contreras, and Dimitri Shlyakhtenko supported me through some of the most strenuous points of my graduate school career. Roy Doumani, the UCLA Business of Science Center, and Derek Alderton have each helped me to take a broader perspective of scientific inquiry and its relevance in our world.

VITA

2002–2003	TA, BYU Math Lab, Brigham Young University, Provo, Utah
2003–2005	Missionary, Church of Jesus Christ of Latter-Day Saints, Saint Louis, Missouri
2005–2006	Upper Division Head TA, BYU Math Lab
2006–2008	Research/Teaching Assistant, Information and Decision Algorithms Laboratories (IDeA Labs), BYU
2008	B.S. Mathematics with University Honors and minor in Latin American Studies, Brigham Young University
2008–2011	Teaching Assistant, Department of Mathematics, UCLA. Taught calculus, differential equations, and C++ programming.
2009	M.A. Mathematics, UCLA.
2009–2012	National Defense Science and Engineering (NDSEG) Fellow
2012	Contestant on JEOPARDY!
2012–2013	National Science Foundation (NSF) Graduate Research Fellow
2013	UCLA Certificate in the Business of Science

CHAPTER 1

Introduction

The Navier-Stokes equations describing incompressible fluid flow are among the most studied and most important equations in applied mathematics, and are central to numerous areas of inquiry in physics, chemistry, biology, and engineering. Other systems of equations related to Navier-Stokes, such as inviscid Euler flow and Stokes flow, are also commonly studied with an eye to real-world problems. Modeling and observing the behavior of fluids governed by these equations is typically done through numerical methods instead of by analytic means. Frequently, the systems under observation have curved boundaries that do not easily conform to a normal Cartesian grid, making typical finite-difference methods difficult to implement. Occasionally, two or more materials with different material properties will interact, with a moving interface separating them. Accurately capturing the motion of this interface can also be challenging.

This thesis describes the work of the author with Prof. Joseph Teran and other co-authors on two different methods based on a Virtual Node approach, designed to solve incompressible flow problems in two and three dimensions. The first is meant to solve inviscid Euler flow problems with irregular boundaries, and the other to solve Navier-Stokes problems containing irregular boundaries, and including multiple fluids with different physical parameters such as density and viscosity. Each embeds the domain of interest in a regular Cartesian grid, and each offers improvements over other methods for comparable problems.

We will first review the derivation of the Navier-Stokes equations from conservation laws, as their derivation provides intuition on some of the computational techniques that have been developed for the simulation of incompressible flow (for example, Chorin projection [5]). We will also look at the role of pressure as a Lagrange multiplier in flow problems, which is key

to the design of one of our methods. We will discuss the development, relative advantages and disadvantages of comparable methods for incompressible flow problems. After a brief description of the Virtual Node algorithm for elliptic problems, we will present the first method, which is based on the original Virtual Node algorithm [1]. Numerical examples for the first method method show the accuracy for velocities to be first order in L^∞ and second in L^1. Finally, we will describe the second method, which extends previous work [2] from Stokes flow to the full Navier-Stokes equations, and from two to three dimensions. Numerical examples of the second method indicate second order accurate velocities in L^∞.

CHAPTER 2

Incompressible Flow Problems

2.1 Derivation of Navier-Stokes Equations

The Navier-Stokes equations describe the motion of incompressible Newtonian (viscous) fluids and are derived by applying certain assumptions about fluid properties to physical conservation laws for mass, momentum, and energy. We start by making some very basic assumptions, which constitute the *continuum hypothesis*, that the physical properties of the fluid such as density, pressure, temperature, and velocity exist and are well-defined.

Under these assumptions, we will present a derivation of the equations that will describe Navier-Stokes flow. We first derive an equation describing how mass is conserved, by viewing the fluid from an Eulerian perspective. We start by letting W be a fixed subregion of a fluid-filled domain Ω. The mass of W is given by

$$m(W,t) = \int_W \rho(\boldsymbol{x},t)\, dV$$

and rate of change of mass is given by

$$\frac{d}{dt}m(W,t) = \frac{d}{dt}\int_W \rho(\boldsymbol{x},t)\, dV = \int_W \frac{d\rho}{dt}(\boldsymbol{x},t)\, dV$$

Since mass is conserved, the rate of change of mass of W must be equal to the net rate of mass flow into W, which can be represented by the normal flux at the boundary ∂W: $-\int_{\partial W} \rho\boldsymbol{u}\cdot\boldsymbol{n}\, dA$. Setting these two quantities equal and applying the divergence theorem to the flux term gives us

$$\begin{aligned} \frac{d}{dt}m(W,t) &= -\int_{\partial W} \rho\boldsymbol{u}\cdot\boldsymbol{n}\, dA \\ \Leftrightarrow \int_W \left[\frac{d\rho}{dt}(\boldsymbol{x},t) + \nabla\cdot(\rho\boldsymbol{u})\right] dV &= 0. \end{aligned}$$

Since this is true for any control volume W, we have at each point the relation

$$\frac{d\rho}{dt}(\boldsymbol{x},t)+\nabla\cdot(\rho\boldsymbol{u})=0. \tag{2.1}$$

This equation is known as the *mass continuity equation*.

We here make another material assumption, that of *incompressibility* of the fluid. Physically, this means that the density of a small volume of fluid $dV(t)$ does not change as dV moves according to the velocity of the fluid $\boldsymbol{u}$. The density ρ of dV is a function of t and $\boldsymbol{x}$, and we can write its *total derivative* as

$$\frac{d\rho}{dt}=\frac{\partial\rho}{\partial t}+\frac{\partial\rho}{\partial x}\frac{dx}{dt}+\frac{\partial\rho}{\partial y}\frac{dy}{dt}+\frac{\partial\rho}{\partial z}\frac{dz}{dt}$$

and substitute the velocity $\boldsymbol{u}$ in to obtain

$$\frac{d\rho}{dt}=\frac{\partial\rho}{\partial t}+\boldsymbol{u}\cdot\nabla\rho. \tag{2.2}$$

The quantity $\frac{\partial\rho}{\partial t}+\boldsymbol{u}\cdot\nabla\rho$ is often called the *material derivative* of ρ and is often denoted by $\frac{D\rho}{Dt}$ instead of $\frac{d\rho}{dt}$. Using the mass continuity equation (2.1) to substitute for $\frac{\partial\rho}{\partial t}$ in (2.2) gives us

$$\frac{D\rho}{Dt}=-\rho(\nabla\cdot\boldsymbol{u}) \tag{2.3}$$

and the assumption of incompressibility implies that the material derivative of ρ is zero everywhere. We conclude that

$$\nabla\cdot\boldsymbol{u}=0. \tag{2.4}$$

Conservation of momentum comes from Newton's second law of motion $\boldsymbol{F}=m\boldsymbol{A}$. Specifically, for a control volume W, the rate of change of total inside momentum is equal to the normal flux of momentum through the boundary ∂W, plus forces acting on ∂W, plus any body forces acting inside W. We describe the normal flux of momentum as

$$-\int_{\partial W}\rho\boldsymbol{u}\boldsymbol{u}\cdot\boldsymbol{n}\,dA.$$

The forces acting on ∂W are traction forces equal to the normal component of fluid stress (or *Cauchy stress*) σ at the boundary, modeled by

$$\int_{\partial W}\sigma\boldsymbol{n}\,dA.$$

For Newtonian fluids, we make the assumption that the Cauchy stress takes the form $\sigma(\boldsymbol{x},t) = -p\boldsymbol{I} + 2\mu\ \boldsymbol{E}(\boldsymbol{u})$, where

$$\boldsymbol{E}(\boldsymbol{u}) = \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}^T\right) \tag{2.5}$$

is the *infinitesimal strain tensor* and p is the fluid pressure. The pressure portion of the stress is reactive to the incompressibility constraint (2.4) and the second term represents the effects of viscosity on the fluid. A good derivation of these terms is given in [6]. In our work, the only body force we consider to act on the fluid is gravity, which is represented by

$$\int_W \rho \boldsymbol{g}\ dV.$$

The vector $\boldsymbol{g}$ corresponds to the gravitational constant, which we take in this work to be $-9.8\ \mathrm{m/s^2}$. Our equation for conservation of momentum is then

$$\frac{\partial}{\partial t}\int_W \rho \boldsymbol{u}\ dV = -\int_{\partial W} \rho \boldsymbol{u}\boldsymbol{u}\cdot\boldsymbol{n}\ dA + \int_{\partial W} \sigma \boldsymbol{n}\ dA + \int_W \rho \boldsymbol{g}\ dV, \tag{2.6}$$

and we use the divergence theorem and move the $\rho\boldsymbol{u}\boldsymbol{u}\cdot\boldsymbol{n}$ term to the left side to obtain

$$\int_W \frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla\cdot(\rho \boldsymbol{u}\boldsymbol{u})\ dV = \int_W \nabla\cdot\sigma + \rho\boldsymbol{g}\ dV$$

which gives us

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla\cdot(\rho \boldsymbol{u}\boldsymbol{u}) = \nabla\cdot\sigma + \rho\boldsymbol{g},$$

into which we substitute for σ to yield

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla\cdot(\rho \boldsymbol{u}\boldsymbol{u}) = \nabla\cdot\left(-p\boldsymbol{I} + \mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} + \mu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}^T\right) + \rho\boldsymbol{g}. \tag{2.7}$$

We can simplify terms on both the left- and right-hand sides of (2.7). First we expand both the time derivative and the divergence term involving the second-order tensor (called a *dyad*), and then use the mass continuity equation (2.1) to simplify:

$$\begin{aligned}\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla\cdot(\rho \boldsymbol{u}\boldsymbol{u}) &= \frac{\partial \rho}{\partial t}\boldsymbol{u} + \frac{\partial \boldsymbol{u}}{\partial t}\rho\ +\ \nabla\cdot(\rho \boldsymbol{u})\boldsymbol{u} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} \\ &= \frac{\partial \boldsymbol{u}}{\partial t}\rho\ +\ \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}\end{aligned} \tag{2.8}$$

For the right-hand side, we use properties of the divergence operator to verify that $\nabla \cdot (p\boldsymbol{I}) = \nabla p$, $\nabla \cdot (\nabla \boldsymbol{u}) = \Delta \boldsymbol{u}$, and $\nabla \cdot (\nabla \boldsymbol{u}) = \nabla (\nabla \cdot \boldsymbol{u})$, the latter of which vanishes due to the incompressibility condition (2.4). This allows us to simplify the right-hand side of (2.7) to

$$-\nabla p + \mu \Delta \boldsymbol{u} + \rho \boldsymbol{g}, \tag{2.9}$$

and combine the two halves for our conservation of momentum equation

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \mu \Delta \boldsymbol{u} + \rho \boldsymbol{g}. \tag{2.10}$$

Note that the left-hand side of Equation (2.10) is the material derivative of $\boldsymbol{u}$; we will write it $\frac{D\boldsymbol{u}}{Dt}$. The equations (2.1), (2.4), and (2.10) form a closed system for the variables $\boldsymbol{u}$, ρ, and p. When we assume that the density ρ in a domain is known (we typically take it to be constant, or piecewise constant), we usually omit (2.1) giving us the closed system of Navier-Stokes equations

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \mu \Delta \boldsymbol{u} + \rho \boldsymbol{g}$$

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

for fluid velocity and pressure.

2.2 Energy Minimization Formulation

Here we show another way to derive the equations for incompressible flow, this time as the solution to a constrained minimization problem. We will start by presenting the derivation for the Stokes problem, which makes use of a trick found in [7] to introduce pressure as a Lagrange multiplier.

2.2.1 Stokes Flow

For simplicity, we assume a compact domain Ω with a uniform Dirichlet boundary Γ. The constrained minimization problem describing Stokes flow is

$$\inf_{\boldsymbol{v}} \quad \mu \int_\Omega |\boldsymbol{E}(\boldsymbol{v})|^2 \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \quad \text{s.t.} \quad \nabla \cdot \boldsymbol{v} = 0. \tag{2.11}$$

We can make an equivalent, unconstrained problem by introducing a penalty function:

$$\inf_{\boldsymbol{v}} \quad \mu \int_\Omega |\boldsymbol{E}(\boldsymbol{v})|^2 \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \; + \; \delta(\nabla \cdot \boldsymbol{v} | \{0\}) \tag{2.12}$$

where $\delta(x|\{0\})$ is the characteristic function, defined as 0 where $x = 0$ and ∞ otherwise. The characteristic function still enforces the incompressibility constraint on any solution to the minimization problem. We can rewrite the penalty function as

$$\delta(\nabla \cdot \boldsymbol{v} | \{0\}) = \sup_q \; - \int_\Omega q \, \nabla \cdot \boldsymbol{v} \, dV$$

since unless $\nabla \cdot \boldsymbol{v} = 0$, we can pick q to make the integral arbitrarily large. We have now turned (2.12) into a saddle point problem:

$$\inf_{\boldsymbol{v}} \sup_q \quad \mu \int_\Omega |\boldsymbol{E}(\boldsymbol{v})|^2 \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \; - \; \int_\Omega q \, \nabla \cdot \boldsymbol{v} \, dV \tag{2.13}$$

The solution pair $(\boldsymbol{u}, p)$ satisfying (2.12) can be found with the aid of the calculus of variations.

The Euler-Lagrange equations corresponding to the functional

$$I[q, v] = \mu \int_\Omega |\boldsymbol{E}(\boldsymbol{v})|^2 \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \; - \; \int_\Omega q \, \nabla \cdot \boldsymbol{v} \, dV$$

are the differential equations satisfied by our energy-minimizing $(\boldsymbol{u}, p)$. The functional has first variation functions $i_v(\tau) = I[q, \boldsymbol{v} + \tau \boldsymbol{w}]$ and $i_q(\tau) = I[q + \tau r, \boldsymbol{v}]$ respectively. The constrained minimum will be located where both first variations are zero, so we solve the resulting equations $i_v'(0) = 0$ and $i_q'(0) = 0$ to obtain the two-equation system

$$\begin{aligned} 2\mu \int_\Omega \boldsymbol{E}(\boldsymbol{u}) : \boldsymbol{E}(\boldsymbol{v}) \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \; - \; \int_\Omega p \, \nabla \cdot \boldsymbol{v} \, dV = 0 \qquad & \forall \, \boldsymbol{v} \\ \int_\Omega q \, \nabla \cdot \boldsymbol{u} \, dV = 0 \qquad & \forall \, q. \end{aligned} \tag{2.14}$$

We use the incompressibility constraint $\nabla \cdot \boldsymbol{u}$ to simplify the integral for the bilinear form:

$$2 \int_\Omega \boldsymbol{E}(\boldsymbol{u}) : \boldsymbol{E}(\boldsymbol{v}) \, dV = - \int_\Omega \boldsymbol{v} \cdot \Delta \boldsymbol{u} \, dV$$

Replacing this integral in (2.14) and performing integration by parts on the pressure term leaves us with the variational form of the Stokes equations (with a homogeneous boundary condition):

$$\begin{aligned} -\mu \int_\Omega \boldsymbol{v} \cdot \Delta \boldsymbol{u} \, dV - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, dV \; + \; \int_\Omega \nabla p \; \cdot \boldsymbol{v} \, dV = 0 \qquad & \forall \, \boldsymbol{v} \\ \int_\Omega q \, \nabla \cdot \boldsymbol{u} \, dV = 0 \qquad & \forall \, q. \end{aligned} \tag{2.15}$$

The Laplacian term above comes from the bilinear form by way of the incompressibility constraint. Since the variational form must hold for all test functions $\boldsymbol{v}, q$, it is equivalent to

$$\begin{aligned} 0 = -\nabla p \; + \; \mu \Delta \boldsymbol{u} \; + \; \rho \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} = 0 \\ u|_{\Gamma} = 0. \end{aligned}$$

2.2.2 Euler Flow

The inertial contributions to Euler and Navier-Stokes flow makes the use of an Eulerian approach to deriving the equations very tricky, so we will use a Lagrangian formulation instead. (A description of derivation in an Eulerian framework is given in [8].) We will first assume $\mu = 0$; that is, inviscid Euler flow. Letting $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t)$ and $\rho_0 = \rho(\boldsymbol{X}, 0)$, we will also define two physical quantities: L is the load potential per unit mass, and U is the internal energy per unit mass. We then state the kinetic and potential energy as

$$\begin{aligned} K &= \int_{\Omega_0} \frac{1}{2} \, \rho_0 \, \left| \frac{d\boldsymbol{x}}{dt} \right|^2 \, dV_0 \\ P &= \int_{\Omega_0} \rho_0 \, (U + L) \, dV_0 \end{aligned}$$

where Ω_0 is a unit volume of fluid at time $t = 0$. From the first law of thermodynamics we can state that the change in the internal energy U per unit mass is equal to the work done by the fluid externally, that is:

$$dU = -p \, dV = -p \, d\left(\frac{1}{\rho}\right). \tag{2.16}$$

We further recall that incompressibility means that $\rho_0(\boldsymbol{X}) = \rho(\boldsymbol{x}(\boldsymbol{X}, t), t)$. We will use this definition of incompressibility here instead of $\nabla_{\boldsymbol{x}} \cdot \boldsymbol{u}$, since our variable in the Lagrangian setup is $\boldsymbol{x}$ and not $\boldsymbol{u}$.

Our functional minimizes the difference between kinetic and potential energy, in accordance with Hamilton's Principle [9], subject to the incompressibility constraint:

$$\inf_{\boldsymbol{x}} \quad \int_0^T \int_{\Omega_0} \frac{1}{2} \, \rho_0 \, \left| \frac{d\boldsymbol{x}}{dt} \right|^2 - \rho_0 (U + L) \, dV_0 \, dt \quad \text{s.t.} \quad \rho = \rho_0. \tag{2.17}$$

As with the Stokes formulation, we introduce the Lagrange multiplier q and the constrained minimization problem becomes

$$\inf_{\boldsymbol{u}} \sup_{q} \quad \int_0^T \int_{\Omega_0} \frac{1}{2}\,\rho_0 \left|\frac{d\boldsymbol{x}}{dt}\right|^2 - \rho_0(U+L)\,dV_0\,dt + \int_0^T \int_{\Omega_0} q(\rho-\rho_0)\,dV_0\,dt. \tag{2.18}$$

We vary ρ and $\boldsymbol{x}$ to obtain the Euler-Lagrange equations

$$\begin{cases} \rho_0 \frac{\partial U}{\partial \rho} + q &= 0 \\ \rho_0 \frac{d}{dt}\frac{\partial x_i}{\partial t} + \rho_0 \frac{\partial L}{\partial x_i} + \frac{\partial}{\partial x_i}(q\rho) &= 0 \end{cases} \tag{2.19}$$

by setting the first variation functions equal to zero. To solve the first Euler-Lagrange equation, we differentiate both sides of (2.16) to find that

$$\frac{\partial U}{\partial \rho} = \frac{p}{\rho^2} \Rightarrow q = \frac{p}{\rho}.$$

We can then substitute for q in the remaining Euler-Lagrange equations (the second listed equation of (2.19) for $i = 1,2,3$.) By now we can replace ρ_0 with ρ, and if we assume load potential on the fluid comes from gravitation force, we can also assume that $\rho\boldsymbol{g} = -\nabla L$ and substitute $\boldsymbol{u} = \frac{\partial \boldsymbol{x}}{\partial t}$. This results in

$$\rho \frac{d\boldsymbol{u}}{dt} - \rho\boldsymbol{g} + \nabla p = 0.$$

When we substitute for the material derivative of the velocity, we get the Eulerian formulation of the inviscid Euler equations

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = -\nabla p + \rho\boldsymbol{g}. \tag{2.20}$$

2.2.3 Energy Minimization for Navier-Stokes

For a viscous fluid, we can obtain the Navier-Stokes equations from the Euler equations simply by replacing the pressure term in (2.20) with the total contribution of the Cauchy stress tensor:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = \nabla\cdot\sigma + \rho\boldsymbol{g}. \tag{2.21}$$

Sometimes it is convenient to work directly with the individual components of the stress instead of σ itself. In such cases we can simplify the Cauchy stress term $\nabla\cdot\sigma$ with $-\nabla p + \mu\Delta\boldsymbol{u}$, based on our assumptions used in (2.7) and (2.9). This gives us

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = -\nabla p + \mu\Delta\boldsymbol{u} + \rho\boldsymbol{g}. \tag{2.22}$$

CHAPTER 3

Existing Methods

Exact projection methods for incompressible flow, originally developed by Chorin in [5], are very effective because of their accuracy, stability, and relative ease of implementation. The temporary introduction of artificial compressibility in the advection stage of these algorithms, through the use of an intermediate velocity field $\boldsymbol{u}^*$, simplifies the interaction of the velocity and pressure. The intermediate compressible velocity field must then be projected to its nearest incompressible counterpart via Hodge decomposition. Projection is ultimately done with the solution of a Poisson equation for the pressure, and it is often stabilized with a MAC-style staggering of velocity and pressure variables [10]. This staggering naturally leads to second-order, discrete central difference gradient and divergence operators. The composition of these operators yields the standard 5- (in 2D) or 7-point Laplacian for cell-centered pressures.

A number of extensions to the early projection method of [5] improved on accuracy of the computed velocities and pressure. A review of higher order projection methods can be found in [11], which describes several of these methods and states conditions that must be met for projection methods to attain second order accuracy. The projection procedure can be generalized as follows:

Step 1: The intermediate velocity field $\boldsymbol{u}^*$ is found by solving

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} + \nabla q = -[(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}]^{n+1/2} + \mu \Delta \boldsymbol{u}$$

$$B(\boldsymbol{u}^*) = 0$$

where q represents a partial approximation to the pressure $p^{n+1/2}$ and $B(\boldsymbol{u}^*) = 0$ is some boundary condition on $\boldsymbol{u}^*$ defined as part of the method. In this example from [11], the time derivative is discretized with Backward Euler and is thus first order in time, but other time discretizations

can also be used.

Step 2: Use the Hodge decomposition to project the divergence-free part of the velocity

$$\boldsymbol{u}^* = \boldsymbol{u}^{n+1} + \Delta t \nabla \phi^{n+1}$$
$$\nabla \cdot \boldsymbol{u}^{n+1} = 0$$

where ϕ^{n+1} is the potential uniquely defined (to a constant) by the decomposition.

Step 3: The pressure is then calculated as some function of q and ϕ. It is fairly common for such projection methods to ignore the pressure gradient while computing the advection step (in other words, $q \equiv 0$). Such methods are called *pressure-free projection methods* and yield an update formula for p that is fully in terms of ϕ: for example, the simple

$$p^{n+1/2} = \phi^{n+1}$$

or the more accurate

$$p^{n+1/2} = \phi^{n+1} - \frac{\mu \Delta t}{2} \Delta \phi^{n+1}$$

described in [11]. In these cases, however, the pressure is not required to advance the velocity. In other instances, q is nontrivial and is set to be the previous pressure $p^{n-1/2}$. Here, the pressure update is given by

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu \Delta t}{2} \Delta \phi^{n+1}.$$

When the splitting and pressure updates are done correctly, both these methods have been shown to yield second order spatial accuracy for flow problems on regular domains with Cartesian grids.

3.1 Embedded Methods

Unfortunately, optimal accuracy is difficult to achieve for problems defined over irregular domains because the MAC staggering is designed for Cartesian grids. A natural approach to the numerical approximation of irregular domains or curved interfaces is the finite element method (FEM) with unstructured meshes that conform to the geometry of Γ and $\partial\Omega$. However, mesh-

ing complex interface geometries can prove difficult and time-consuming when the interface frequently changes.

Many researchers have developed approaches that generalize the Cartesian MAC-based projection for regular domains to the irregular case by embedding the irregular domain in a standard Cartesian grid. These methods avoid the problems of remeshing found in FEM implementations. However, it is very difficult to maintain the simplicity of the original Cartesian approach without sacrificing accuracy or efficiency. Most recent developments in embedded methods for incompressible flow have as objectives increased orders of accuracy, faster computation by construction of symmetric, positive definite linear systems, or capabilities to handle such situations as solid-fluid coupling, moving boundaries, or interfacial flow of fluids with different material properties (such as density and viscosity).

Some of the first embedded methods were fictitious domain methods by Hyman [12] and Saul'ev [13]. The fictitious domain approach has been used with incompressible materials in a number of works [14, 15, 16, 17, 18, 19, 20, 21, 22]. These approaches embed the irregular geometry in a simpler domain for which fast solvers exist (e.g. Fast Fourier Transforms). The calculations include fictitious material in the complement of the domain of interest. A forcing term (often from a Lagrange multiplier) is used to maintain boundary conditions at the irregular geometry. Although these techniques naturally allow for efficient solution procedures, they depend on a smooth solution across the embedded domain geometry for optimal accuracy, which is not typically possible.

3.1.1 Immersed Boundary Method

The immersed boundary method (IBM) was introduced in [23], originally developed to model blood flow in the human heart. It has since been applied to a number of other problems in the biosciences and in physics; a good review of applications of the IBM can be found in [24] as well as a detailed exposition of the numerical method. The important detail in the IBM is the use of regularized discrete delta functions to distribute a singular source to grid points near an embedded boundary. This limits the accuracy of the IBM to first order in the case of sharp

discontinuities across an infinitesimally small interface (although in some other cases, such as a thick interface, the IBM is able to attain second order). A recent analysis [25] has suggested the IBM gives $O(1)$ errors in pressure near the interface for interfacial Stokes flow. Furthermore, the IBM suffers from loss of volume near the interface. An improvement to the IBM in [26] modified divergence and gradient stencils to partially fix this problem for Stokes flows. A modification to the IBM gives the Blob Projection Method [27], which yields second order accuracy but requires special conditions on the singular source, which seems to limit the type of geometries it can support. It is also considerably more difficult to implement than traditional IBM.

The IBM is robust and easy to implement; this advantage has led to its continued use in applications over other, more accurate, methods. Some recent applications include modeling fish-sized swimmer dynamics [28], simulating fluid-structure interaction such as platelets in blood [29], insect swarm dynamics [30], and fluid-filled capsules in shear flow [31].

3.1.2 Immersed Interface Methods

The Immersed Interface Method (IIM) is one of the most popular finite difference methods for approximating the Navier-Stokes equations to second-order accuracy. Leveque and Li introduced the IIM in [32] for approximating elliptic interface problems, and the method has since been extended to a number of applications including incompressible flow problems with irregular boundaries or interfaces [33, 34, 35]. Further extensions have taken into account interfacial forces such as surface tension [36]. Finite-difference schemes for the IIM use standard second order central-difference stencils away from an interface, modifying the terms of the standard stencil near an interface. For discontinuous viscosity, the addition of an extra stencil point is generally required at points near the interface to maintain second order accuracy. Jump discontinuities in the pressure are handled by modification of the right-hand side. The IIM achieves second order accuracy by capturing interfacial discontinuities in the pressure, the velocities and their derivatives in a sharp manner. Immersed Interface Methods have also been used in other flow problems including Hele-Shaw flow [37] and also problems in which the viscosity is dis-

continuous across the interfaces [38, 39, 40]. Arbitrarily high orders of accuracy have been achieved for elliptic problems [41]. A good survey of IIM methods and applications can be found in [42].

The Taylor expansion-based discretization of the Immersed Interface Method is considerably more complicated to implement than the IBM, and most applications are only in two space dimensions as a result. However, researchers have applied the IIM to three dimensional flows [43]. An important limitation of the IIM is the lack of symmetry in discretizations arising from problems with discontinuous coefficients. This imposes an obstacle on the overall speed of these methods since fast linear solvers for symmetric systems, such as conjugate gradient or MINRES, cannot be used. One exception to this is continuous viscosity Stokes flow, which only modifies the right-hand side of the linear system and can thus use fast Poisson solvers [33]. The IIM is also complicated from the lack of explicit knowledge of jump conditions on the fluid variables (and their derivatives) along the interface in the case of discontinuous viscosity [38]. Significant efforts have been made to improve the performance of the IIM, particularly through algebraic and geometric multigrid methods [44].

3.1.3 Extrapolation-Based Schemes

A number of schemes have modified a method such as IBM or IIM by introducing fictitious 'ghost' points to enable use of standard second-order stencils near an irregular boundary or immersed interface. One such method is the Ghost Fluid Method (GFM), which guarantees a symmetric, positive definite method by decomposing the flux jump separately in each dimension. [45]. Compared to the IIM, the Ghost Fluid Method is relatively easy to implement. The GFM has been applied to the Poisson problem with interfacial jumps and variable coefficients [45] and also to multiphase incompressible flow [46]. The way in which the GFM decomposes the flux jump condition only takes into account the normal component of the flux, neglecting treatment of the tangential flux terms. This renders the method capable only of achieving first order results for interface problems. Also, in [46] the GFM treats viscous terms explicitly in the discontinuous velocity case, because they cannot be properly decoupled. The ease of im-

plementation of the GFM has led to its use in some applications, including a method [47] for interfacial flows with very high density ratios.

The Matched Interface and Boundary (MIB) method [48, 49] adjusts the approach of the IIM discretization, making modifications to the Poisson stencil on a dimension-by-dimension basis using fictitious points. Enforcement of jump conditions is decoupled from the modified finite difference stencil. The work of [50] applies the framework of the MIB to interfacial flow in two dimensions, with an emphasis on viscous creeping flows in geodynamic processes. The MIB achieves second-order accuracy and, although the resulting matrix is asymmetric, its construction is somewhat more simplified than the IIM. Schemes for the MIB enable very high orders of accuracy for the Poisson equation [49], although these have not yet been applied to flow problems.

3.1.4 Finite Volume Methods

Some methods use finite volume-type approaches in cutting the domain over the MAC grid. One recent and efficient method for flow problems on irregular domains is that of Ng. et al [51]. This work gives convergent results for velocities in L^∞, improving on the results of previous work by Batty [52]. The results in [51] were second order accurate pressures, and second order velocities in two dimensions (first order in three dimensions.) The authors later extended their method [53] to couple incompressible flow with motion of non-stationary rigid bodies. These changes include a second-order accurate time discretization instead of backward Euler, use of a nonzero pressure approximation in determining the intermediate velocity field, and a different treatment of the viscosity term that gives second order accurate velocity gradients, but at the cost of sacrificing symmetry for the intermediate velocity solve.

3.1.5 Extended Finite Element Method

The extended finite element method (XFEM) and related approaches in the finite element literature also make use of geometry embedded in regular elements. Although originally developed for crack-based field discontinuities in elasticity problems, these techniques are also used with

embedded problems in irregular domains. Daux et al. first showed that these techniques can naturally capture embedded Neumann boundary conditions [54, 55]. These approaches are equivalent to the variational cut cell method of Almgren et al. in [56]. Enforcement of Dirichlet constraints is more difficult with variational cut cell approaches [57, 58] and typically involves a Lagrange multiplier or stabilization. Dolbow and Devan recently investigated the convergence of such approaches with incompressible materials and point out that much analysis in this context remains to be completed [59]. Despite the lack of thorough analysis, such XFEM approaches appear to be very accurate and have been used in many applications involving incompressible materials in irregular domains [60, 61, 62, 63, 64].

3.1.6 Other Cartesian Methods

There are also a handful of highly accurate embedded finite difference methods utilizing cut uniform grid cells which have been developed in the context of incompressible flow for irregular domains, although these methods are not applicable to interfacial flows. For example, Marella et al. [65] use collocated grids and define sub cell interface and boundary geometry in cut cells via level sets, and claim second order accuracy in two and three dimensions. Shirokoff et al. [66] use the normal 5-point stencil for material points and use Neumann boundary conditions to compute the stencils at the ghost points. This method also claims second order accuracy in two dimensions. Unfortunately, both these methods produce non-symmetric linear systems.

CHAPTER 4

The Virtual Node Method

The virtual node method was originally developed in [1] as a second order accurate method for elliptic problems in two dimensions with interfaces or irregularly-shaped domains. It has since been extended to elliptic problems in three dimensions [67] and elasticity problems [68].

The virtual node method uses duplicated Cartesian bilinear elements along the interface to introduce additional ghost or "virtual" degrees of freedom that allow for an accurate representation of functions that are defined on irregular domains, or have discontinuities across an interface. Using a variational approach to discretize a system of equations leads to a linear system that is symmetric, allowing for the use of fast solvers. In this chapter we will give a very brief overview of the discretization induced by the virtual node method on a Poisson problem with Neumann boundary conditions; we use a slight variation of this problem in the following chapter. The method can also handle Dirichlet and interface conditions; we refer the reader to [1] for more detail on the implementation in two dimensions, and to [67] for the implementation in three dimensions.

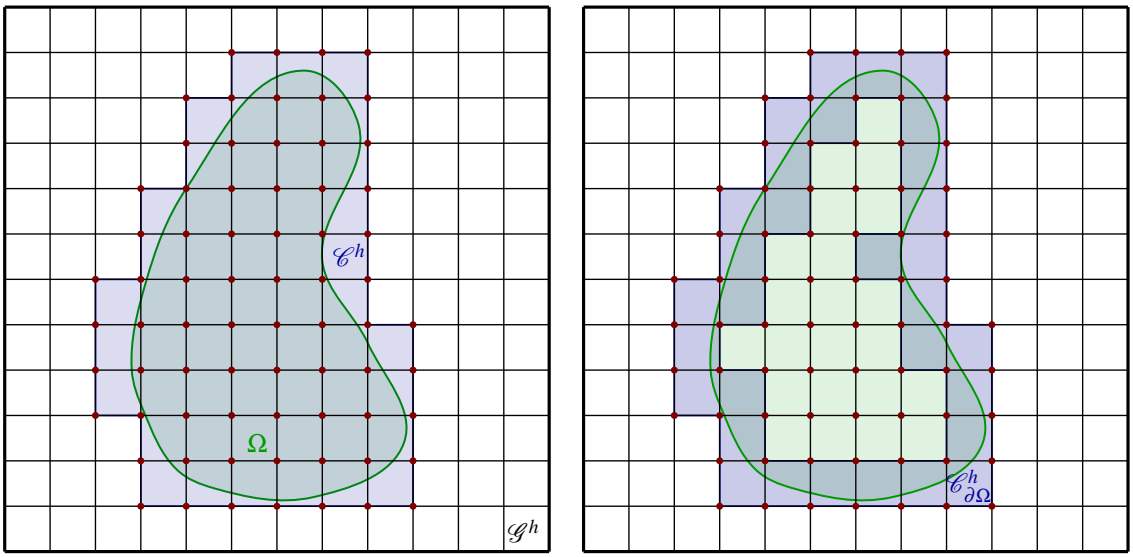
4.1 Discretization

We discretize the embedded Neumann problem

$$-\nabla \cdot (\beta(\boldsymbol{x})\nabla u(\boldsymbol{x})) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega \tag{4.1}$$

$$\beta(\boldsymbol{x})\nabla u(\boldsymbol{x}) \cdot \boldsymbol{n} = q(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega \tag{4.2}$$

over a regular Cartesian grid, which does not have to conform to $\partial\Omega$. We assume the coefficient function β and the forcing term f to be sufficiently smooth. We first embed the domain Ω in a



(a) (b)

Figure 4.1: Grid notation for the virtual node method. Degrees of freedom (red dots) are located at the vertices. The computational domain $\mathscr{C}^h$ [blue cells in (a)] consists of all cells $c_k \subset \mathscr{G}^h_p$ that intersect Ω, and introduces virtual degrees of freedom outside of Ω. The computational boundary $\mathscr{C}^h_{\partial\Omega}$ [blue cells in (b)] contains the cells which only partially intersect Ω.

regular Cartesian grid $\mathscr{G}^h$ with equal grid spacing $\Delta x = \Delta y = h$. We include all Cartesian cells c_k in the grid $\mathscr{G}^h$ that intersect Ω in the discretization, as shown in Figure 4.1, and refer to this subset of $\mathscr{G}^h$ as the computational domain $\mathscr{C}^h = \{c_k \in \mathscr{G}^h,\ c_k \cap \Omega \neq \emptyset\} \subset \mathscr{G}^h$, as shown in Figure 5.1(b). Since Ω and its boundary typically do not align with the Cartesian grid, many of the grid cells c_k only partially intersect the domain Ω, and some nodes of those cells will lie outside Ω. The set of all cells that intersect the boundary is called the computational boundary, $\mathscr{C}^h_{\partial\Omega} = \{c_k \in \mathscr{G}^h,\ c_k \cap \partial\Omega \neq \emptyset\} \subset \mathscr{G}^h$, and the nodes lying outside the domain are what we refer to as the "virtual" degrees of freedom.

We obtain a variational form of this problem by multiplying (4.1) by a test function and integrating:

$$-\int_\Omega \nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x}))\, dx = \int_\Omega f v\, dx$$

We then use integration by parts to obtain

$$\int_\Omega \beta \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} \beta v \nabla u \cdot n \, dS = \int_\Omega f v \, dx$$

$$\int_\Omega \beta \nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx + \int_{\partial\Omega} q v \, dS$$

giving us the variational formulation

$$\begin{aligned} &\text{Find } u \in \mathbf{H}^1(\Omega) \text{ such that} \\ &\int_\Omega \beta \nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx + \int_{\partial\Omega} q v \, dS \\ &\text{for all } v \in \mathbf{H}^1(\Omega). \end{aligned} \tag{4.3}$$

Our finite-dimensional approximation $u^h(\boldsymbol{x})$ to $u(\boldsymbol{x})$ is written $\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{i=1}^N u_i N_i(\boldsymbol{x})$ for $\vec{u} = (u_1, \cdots, u_n) \in \mathbb{R}^n$, where $N_i(\boldsymbol{x})$ are the standard piecewise bilinear interpolation basis functions associated with the grid nodes which compose the cells of $\mathscr{C}^h$. When we replace the continuous functions u and v in with their discrete counterparts, we end up with the linear system

$$\boldsymbol{A}\boldsymbol{u}^h = \boldsymbol{f}^h \tag{4.4}$$

where $\boldsymbol{A} = \{a_{ij}\}$ is defined by

$$a_{ij} = \int_\Omega \beta \nabla N_i \cdot \nabla N_j \, dx \tag{4.5}$$

and $\boldsymbol{f} = (f_1, \cdots, f_n)$ by

$$f_i = \int_\Omega f N_i dx + \int_{\partial\Omega} q N_i \, dS \tag{4.6}$$

For each i, j, the integrals (4.5) and (4.6) are supported over no more than four grid cells in 2D. Since the basis functions N_i in the integral are piecewise smooth over each grid cell, it is practical to numerically evaluate integrals over each grid cell c_k. For details on the numerical integration, we refer the reader to [1]. There, the integrals are approximated by taking cell averages of β, f, and q over $c_k \cap \Omega$, and using an efficient quadrature approach. This produces integral values which are accurate enough to guarantee second order accuracy of the computed solution u^h.

The cellwise integration we use leads to a stiffness matrix $\boldsymbol{A}$ with a 9-point stencil except at virtual nodes. The presentation of the discretization shown in [1] formulates an energy which

has as its Hessian the stiffness matrix $\boldsymbol{A}$ described above. The problem formulation in terms of an energy facilitates a simple trick to reduce the stencil size: a different energy is defined at interior cells, which yields the standard 5-point central difference discretization at nodes which are surrounded by interior cells and not adjacent to the boundary.

Coercivity of the bilinear form in (4.1) guarantees that $\boldsymbol{A}$ is positive semi-definite, and numerical experiments presented in [1] reveal second-order convergence in L^∞.

CHAPTER 5

An Algorithm for Inviscid Euler Flow Over Irregular Domains

Here we present the work of [3], which leverages the Poisson virtual node framework described in the previous chapter to present an efficient discrete Hodge decomposition for velocity fields defined over irregular domains in two and three dimensions. This decomposition method is designed for use in the exact projection discretization of incompressible flow. Our approach uses a signed distance function to represent the irregular domain embedded in a Cartesian grid and follows the variational approach from the previous chapter to create a symmetric positive definite linear system. We present a novel modification to the previous approach that yields a 5-point stencil (7-point in 3D) across the entire computational domain, where the original algorithm required a 9-point stencil (27-point in 3D) near the embedded irregular boundary. We show that this new condensed stencil enables a decomposition of the form $\mathbf{A} = \boldsymbol{G}^T \boldsymbol{M}^{-1} \boldsymbol{G}$, where $\boldsymbol{M}$ is a diagonal weighting matrix and $\boldsymbol{G}$ and $\boldsymbol{D} = -\boldsymbol{G}^T$ are diagonal scalings of the standard central-difference gradient and divergence operators. We use this factored form as the basis of our discrete Hodge decomposition and show that this can be readily used for exact projection in incompressible flow. Numerical experiments suggest our method is second order in L^∞ for pressures and first-order in L^∞, second order in L^1 for velocities. This work was performed jointly with Craig Schroeder and Prof. Joseph Teran.

5.1 Background

Exact projection methods for incompressible flow are very effective because of their accuracy, stability, and relative ease of implementation [5]. The temporary introduction of artificial com-

pressibility in the advection stage of these algorithms simplifies the interaction of the velocity and pressure. This intermediate compressible velocity field must then be projected to its nearest incompressible counterpart via Hodge decomposition. Projection is ultimately done with the solution of a Poisson equation for the pressure, and it is often stabilized with a MAC-style staggering of velocity and pressure variables [10]. This staggering naturally leads to second order, discrete central difference gradient and divergence operators. The composition of these operators yields the standard 5-point Laplacian (7-point in 3D) for cell-centered pressures. Unfortunately, optimal accuracy is difficult to achieve for problems defined over irregular domains because the MAC staggering is designed for Cartesian grids. Many researchers have developed approaches that generalize the Cartesian MAC-based projection for regular domains to the irregular case, however it is very difficult to maintain the simplicity of the original approach without sacrificing accuracy or efficiency. For example, the immersed boundary method [69, 24] can be used to enforce boundary conditions on an irregular domain without any modification to the Cartesian case other than a change in the forcing terms. However, the regularized delta function conception of the right hand side terms degrades the convergence to first-order. The immersed interface method [32, 70] can be used to preserve optimal accuracy, however the associated discrete systems are generally no longer symmetric. This deviation from the standard Laplacian discretization prevents the use of fast solvers, leading to considerable computational expense.

Methods that utilize a level set representation of the irregular domain can be used to define embedded Cartesian discretizations that balance efficiency with improved accuracy by leveraging sub-cell geometric detail [71, 72, 73, 1, 67]. In a recent related work, we showed that optimal velocity accuracy can be achieved for Stokes flow with a virtual node approach [2] which uses such a level set representation. However, using a variational approach yields linear systems that are typically of symmetric KKT type, (see e.g. [74]) so fast solvers like those used for the standard Laplacian discretization are not available. Notably, Gibou et al. have recently shown that level set approaches are very effective for exact projection discretization of incompressible flow [51, 53]. In the present work, we take a virtual node approach (developed in [1] and [67]) to factor the Poisson equation in a manner similar to that presented in [51].

Although there are many methods capable of achieving second order accurate velocities,

these methods tend to either require expensive remeshing, such as with finite elements, or yield linear systems which are indefinite (for example [2]) or even asymmetric (such as [70]).

By contrast, accurate and positive definite methods for solving the Poisson equation are relatively easy to construct, but these discretizations generally do not carry forward to the problem of exact projection. We demonstrate a method that is capable of condensing a 9-point stencil (27-point in 3d) into a 5-point stencil (7-point in 3d) in a manner that admits a factorization of the Poisson operator and leads immediately to an exact projection discretization. This alteration to the Poisson stencil retains the second order convergence of the original Poisson operator, but it only leads to first-order velocities.

5.2 Exact projection and Hodge decomposition

Since our focus is the Hodge decomposition aspect of an exact projection discretization, we will ignore viscous and forcing terms and focus on the inviscid Euler equations

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial \mathbf{x}}\boldsymbol{u} &= 0 \\ \rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{u}}{\partial \mathbf{x}}\boldsymbol{u}\right) &= -\nabla p \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{5.1}$$

with Dirichlet normal velocity boundary conditions $\boldsymbol{u}\cdot\mathbf{n} = U_{BC}$ on $\partial\Omega$. A simple splitting of these equations give rise to the following temporal discretization

$$\begin{aligned} \rho^n\left(\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} + \frac{\partial \boldsymbol{u}^n}{\partial \mathbf{x}}\boldsymbol{u}^n\right) &= \mathbf{0} \\ \rho^n\left(\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t}\right) &= -\nabla p^{n+1} \\ \frac{\rho^{n+1} - \rho^n}{\Delta t} + \frac{\partial \rho^n}{\partial \mathbf{x}}\boldsymbol{u}^{n+1} &= 0. \end{aligned} \tag{5.2}$$

If we take the divergence of the second equation and note that $\nabla\cdot\boldsymbol{u}^{n+1} = 0$, we can equivalently define this step as

$$\begin{aligned} \nabla\cdot\left(\frac{\Delta t}{\rho^n}\nabla p^{n+1}\right) &= \nabla\cdot\boldsymbol{u}^* \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^* - \frac{\Delta t}{\rho^n}\nabla p^{n+1}, \end{aligned} \tag{5.3}$$

where the boundary conditions for the Poisson equation are then of Neumann type $\frac{\Delta t}{\rho^n}\nabla p^{n+1} \cdot \mathbf{n} = \boldsymbol{u}^* \cdot \mathbf{n} - U_{BC}$.

In order to create an exact projection discretization we come up with a discrete volume-weighted approximation to ∇, and denote it by $\boldsymbol{G}$. The Poisson equation for the pressure is then

$$\Delta t \boldsymbol{D}\boldsymbol{M}^{-1}\boldsymbol{G}p^{n+1} = \boldsymbol{D}\boldsymbol{u}^*, \tag{5.4}$$

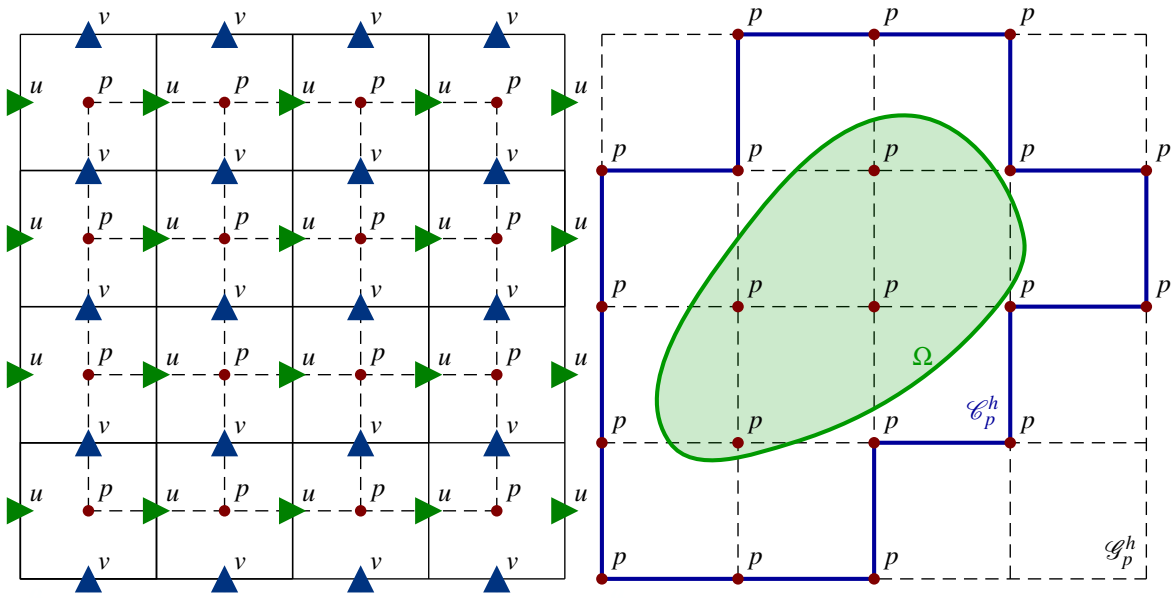
where $\boldsymbol{D} = -\boldsymbol{G}^T$ is the associated discrete approximation to the divergence and $\boldsymbol{M}$ is a diagonal scaling that approximates ρ^n scaled by volume. In the sections that follow, we will show that the virtual node Poisson discretizations developed in [1] and [67] can be rewritten in a form that admits a scaled version of the standard MAC grid-based approximation of ∇ for $\boldsymbol{G}$. This modification to the original algorithm is the key step needed to apply it in Hodge decomposition-based exact projection. That is, with this decomposition we can see algebraically that $\boldsymbol{D}\boldsymbol{u}^{n+1} = 0$ if we define $\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t \boldsymbol{M}^{-1}\boldsymbol{G}p^{n+1}$.

5.3 Discretization

Our extension of [1] to approximate the Poisson problem (5.3) reduces the 9-point stencil of the previous work to a 5-point stencil. This is necessary to allow a decomposition of the Poisson matrix $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{M}^{-1}\boldsymbol{G}$ with a diagonal $\boldsymbol{M}$ and the previously described $\boldsymbol{D}$ and $\boldsymbol{G}$. For simplicity, we present our extension in two dimensions, describing how it differs from [1], and how we treat the right-hand side to enforce the boundary condition on the projected velocity $\boldsymbol{u}^{n+1}$. Finally, we briefly mention the slight modifications to [67] necessary to implement our method in three dimensions.

5.3.1 Condensed Stencil Approach

The energy-based discretization used in [1, 67] results in a 9-point stencil near the boundary in two dimensions. Away from the boundary, a novel quadrature rule was used to condense the stencil to the standard 5-point discretization. We present a novel modification of the stencil



(a) MAC grid (solid) layout superimposed with $\mathscr{G}$ (dashed)

(b) Degrees of freedom for $\mathscr{G}$ cut by Ω

Figure 5.1: Grid notation for our method. Our pressure grid has pressure nodes at the vertices, in contrast with the standard MAC grid (a) with pressures at cell centers. The computational domain (b) consists of all cells $c_k \subset \mathscr{G}_p^h$ that intersect Ω, and introduces virtual degrees of freedom for p, which lie outside of Ω.

coefficients in [1, 67] that admits a 5-point stencil over the entire domain without sacrificing the second order accuracy in L^∞ achieved in the original work.

As in [1, 67], and as described in Chapter 4, we embed the domain Ω in a regular Cartesian grid. In our case, this grid is the subset of a standard MAC grid that has pressure degrees of freedom at its vertices, as shown in Figure 5.1(a). We include in the discretization all cells in $\mathscr{G}_p^h$ that intersect Ω, and refer to this subset of $\mathscr{G}_p^h$ as $\mathscr{C}_p^h = \{c_k \in \mathscr{G}_p^h,\ c_k \cap \Omega \neq \emptyset\} \subset \mathscr{G}_p^h$, as shown in Figure 5.1(b). For convenience let $\Omega_k = c_k \cap \Omega$, and let Ω_k^0 be the same region transformed into coordinates $[0,1] \times [0,1]$. Since Ω and its boundary $\partial\Omega$ typically will not align with elements of the Cartesian grid, our discretization will include many pressure cells that only partially intersect with the domain Ω. Some nodes of those cells will lie outside the domain. We refer to pressure nodes lying outside the domain as "virtual" nodes and their corresponding degrees of

freedom as virtual degrees of freedom.

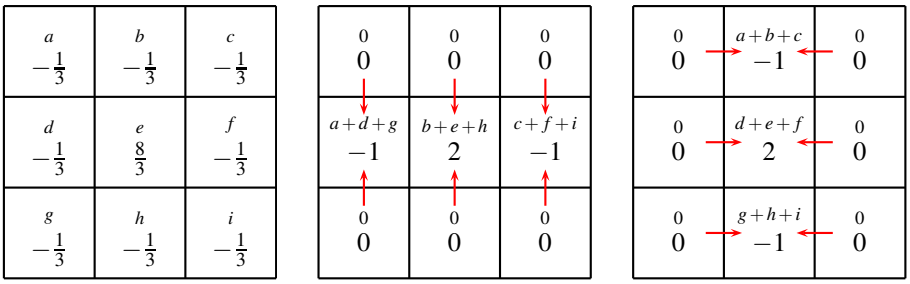
Also as in [1, 67], our discretization is designed by first assuming that our pressure field is piecewise bilinear over the cells in $\mathscr{C}_p^h$. $p(\mathbf{x}) = \sum_{i=1}^{n_p} p_i N_i(\mathbf{x})$ for $\vec{p} = (p_1, \ldots, p_{n_p})^t \in \mathbf{R}^{n_p}$. Here $N_i(\mathbf{x})$ is the standard piecewise bilinear interpolation basis function associated with pressure grid vertex i; and n_p denotes the number of degrees of freedom in the discretization, equal to the number of grid vertices that compose the cells of $\mathscr{C}^h$. Occasionally we will refer to the basis functions as $N_{l,m}$, where l and m represent the position of vertex i on the Cartesian grid. With this assumption, we start our approximation from the quadratic terms in the variational form of the Poisson equation

$$\psi(p) := \sum_{c_k \in \mathscr{C}_p^h} \frac{1}{2\rho} \sum_{r,s,r',s' \in \{0,1\}} \left(\int_{\Omega_k} \nabla N_{l+r,m+s} \cdot \nabla N_{l+r',m+s'} \, d\boldsymbol{x} \right) p_{l+r,m+s} p_{l+r',m+s'} \tag{5.5}$$

where l and m are the two-dimensional indices of the lower left node in pressure cell c_k. As in [1, 67], we perform the integration over cut cells using the divergence theorem to express each entry as a boundary integral. The cut cell boundary geometry is discretized from the level set representation of Ω. The Hessian of this energy gives rise to the matrix in our variational approximation to the Poisson equation: $E_{ij} := \frac{\partial^2}{\partial p_i \partial p_j} \psi(p)$. We will now detail our approach for condensing the $\boldsymbol{E} \in \mathbf{R}^{n_p \times n_p}$ stencil from up to 9 non-zero entries per row to at most 5 non-zero entries per row. We will refer to the condensed stencil matrix as $\boldsymbol{A}$.

The condensing procedure for a generic 9-point stencil, ignoring whether cells are cut or interior, is shown in Figure 5.2. The change of stencil coefficients can be thought of in terms of 'pushing' coefficients from the top and bottom of the stencil into the middle approximating the $-p_{xx}$ portion of the Laplacian (Figure 5.2(b)), and from the left and right sides of the stencil into the center to approximate the $-p_{yy}$ portion of the Laplacian (Figure 5.2(c)). The 'condensing' of the coefficients in this case leads to a 5-point stencil—the standard 5-point stencil if the center point is not incident on any cut cells.

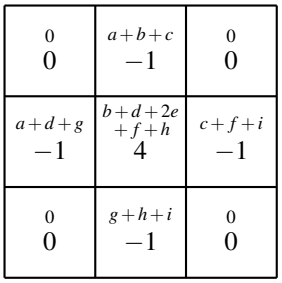
At each node, the corresponding contributions to the stencil come from the four pressure cells incident to that node. For each pressure cell c_k in the discretization, we define $\boldsymbol{E}_{c_k}$ to be the 4×4 matrix containing the contributions of c_k to the matrix $\boldsymbol{E}$. We call $\boldsymbol{E}_{c_k}$ the *element stiffness matrix* corresponding to c_k: $(\boldsymbol{E}_{c_k})_{ij} = \int_{\Omega_k^0} a_{ij} \, d\boldsymbol{x}^0 = \int_{\Omega_k^0} \nabla N_i^0 \cdot \nabla N_j^0 \, d\boldsymbol{x}^0$, where Ω_k^0 is



(a) 2^{nd} order $-\Delta p = -p_{xx} - p_{yy}$

(b) 2^{nd} order $-p_{xx}$

(c) 2^{nd} order $-p_{yy}$



(d) 2^{nd} order $-\Delta p = -p_{xx} - p_{yy}$

Figure 5.2: An illustration of our 'condensed stencil' modification at a generic point in the domain. For a cell away from the boundary $\partial\Omega$, the 9-point second order accurate stencil above (derived from bilinear finite elements) is condensed by our modification to the standard 5-point stencil.

the corresponding scaling of $c_k \cap \Omega$ to the unit square and we denote the four basis functions supported over c_k as in Figure 5.4(b), written in a scaled coordinate frame local to the element:

$$N_0^0 = (1-x)(1-y) \qquad N_1^0 = x(1-y) \qquad N_2^0 = (1-x)y \qquad N_3^0 = xy. \tag{5.6}$$

The element stiffness matrix can be written as

$$\boldsymbol{E}_{c_k} = \int_{\Omega_k^0} \begin{pmatrix} \nabla N_0^0 \\ \nabla N_1^0 \\ \nabla N_2^0 \\ \nabla N_3^0 \end{pmatrix} \begin{pmatrix} \nabla N_0^0 \\ \nabla N_1^0 \\ \nabla N_2^0 \\ \nabla N_3^0 \end{pmatrix}^T d\boldsymbol{x}^0.$$

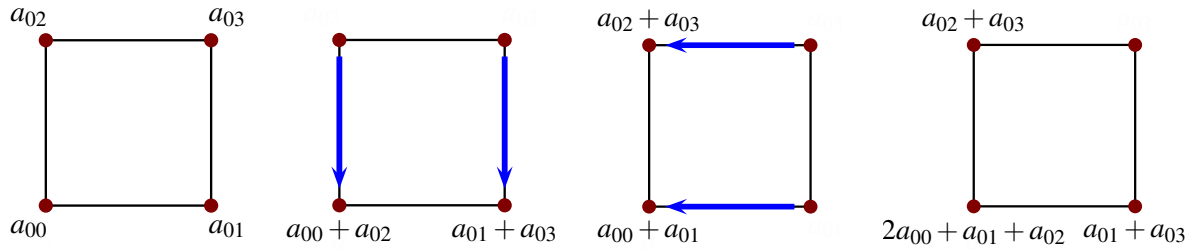
Substituting in (5.6) yields $\boldsymbol{E}_{c_k} =$

$$\int_{\Omega_k^0} \begin{pmatrix} (x-1)^2+(y-1)^2 & -(y-1)^2-x(x-1) & -(x-1)^2-y(y-1) & x(x-1)+y(y-1) \\ -(y-1)^2-x(x-1) & (y-1)^2+x^2 & x(x-1)+y(y-1) & -x^2-y(y-1) \\ -(x-1)^2-y(y-1) & x(x-1)+y(y-1) & (x-1)^2+y^2 & -y^2-x(x-1) \\ x(x-1)+y(y-1) & -x^2-y(y-1) & -y^2-x(x-1) & x^2+y^2 \end{pmatrix} d\boldsymbol{x}^0$$

$$= \int_{\Omega_k^0} \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} d\boldsymbol{x}^0 \tag{5.7}$$

where we give the quadratic polynomials labels in (5.7) to make the condensation operation easier to follow.

We rearrange the elements of $\boldsymbol{E}_{c_k}$ in the manner illustrated in Figure 5.3 for a single row to create a modified element stiffness matrix $\boldsymbol{A}_{c_k}$. This process maintains a stencil consistent with our Poisson problem while moving all nonzero terms of the cellwise stiffness matrix to entries of $\boldsymbol{A}$ where the row and column correspond to equal or adjacent nodes. This condensing process for the four cells incident to a node produces a 5-point stencil for that node, similar to Figure 5.2, and results in the modified matrix



(a) original stencil (b) condensing in y direction $-p_{xx}$ (c) condensing in x direction $-p_{yy}$ (d) Cellwise constribution to condensed stencil

Figure 5.3: We examine the changes made to the first row (corresponding to node 0) of the element stiffness matrix $\boldsymbol{E}_{c_k}$ from [1] by the condensing procedure illustrated in Figure 5.2 to yield our element stiffness matrix (5.8). Condensing the coefficients as shown ensures that the nonzero terms in each row of $\boldsymbol{A}$ correspond only to adjacent nodes on the same row or column, and allows for a 5-point stencil.

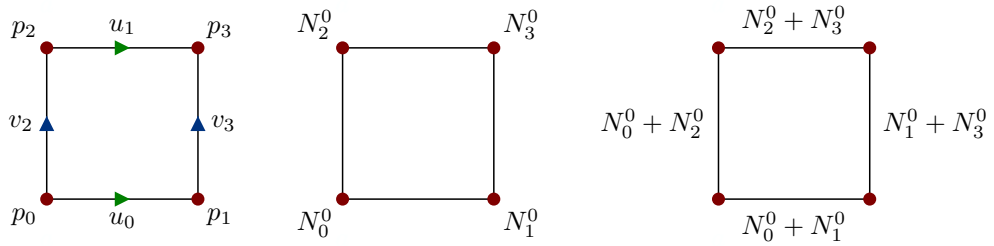
$$\boldsymbol{A}_{c_k} = \int_{\Omega_k^0} \begin{pmatrix} 2a_{00}+a_{01}+a_{02} & a_{01}+a_{03} & a_{02}+a_{03} & 0 \\ a_{10}+a_{12} & 2a_{11}+a_{10}+a_{13} & 0 & a_{12}+a_{13} \\ a_{20}+a_{21} & 0 & 2a_{22}+a_{20}+a_{23} & a_{21}+a_{23} \\ 0 & a_{31}+a_{30} & a_{32}+a_{30} & 2a_{33}+a_{31}+a_{32} \end{pmatrix} d\boldsymbol{x}^0, \tag{5.8}$$

which, after substituting for the a_{ij} terms previously defined in (5.7), simplifies to

$$\boldsymbol{A}_{c_k} = \int_{\Omega_k^0} \begin{pmatrix} 2-y-x & -1+y & -1+x & 0 \\ -1+y & 1-y+x & 0 & -x \\ -1+x & 0 & 1-x+y & -y \\ 0 & -x & -y & x+y \end{pmatrix} d\boldsymbol{x}^0. \tag{5.9}$$

5.3.2 Factorization

Let $\boldsymbol{P}_{c_k}$ be the matrix that relates the pressure indices for pressure cell c_k to indices in the full grid. That is, the matrix $\boldsymbol{P}_{c_k}$ has four columns and as many rows as pressure nodes. It will have a single 1 per column with 0 everywhere else. Similarly, let $\boldsymbol{Q}_{c_k}$ be the matrix that relates



(a) Indexing (b) Basis (c) $\boldsymbol{S}_{c_k}$ diagonals

Figure 5.4: The naming conventions used in referring to pressures and velocities (a) and pressure degrees of freedom on a grid cell (b). Additionally, (c) shows, at each velocity node, the basis functions contributing to that term of the cellwise weight matrix $\boldsymbol{S}$.

the MAC velocity indices incident on pressure cell c_k to indices in the full grid. $\boldsymbol{Q}_{c_k}$ has four columns and as many rows as MAC faces, again with a single 1 per column. These operators will allow us to build the individual pressure cell contributions onto the full system in a rather convenient form later. Figure 5.4 shows the indexing convention we use in this work for the degrees of freedom associated with a pressure cell, and by extension, the columns of $\boldsymbol{P}_{c_k}$ and $\boldsymbol{Q}_{c_k}$.

If $\hat{\boldsymbol{G}}$ is the standard central difference operator defined over the entire domain, then we can define a pressure cellwise $\hat{\boldsymbol{G}}_{c_k}$ by restricting $\hat{\boldsymbol{G}}$ to the pressure cell as

$$\hat{\boldsymbol{G}}_{c_k} = \boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} \boldsymbol{P}_{c_k} = \frac{1}{\Delta x} \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \tag{5.10}$$

With this, we can factor $\boldsymbol{A}_{c_k}$ as

$$\begin{aligned}
\boldsymbol{A}_{c_k} &= \int_{\Omega_k^0} \begin{pmatrix} 2-y-x & -1+y & -1+x & 0 \\ -1+y & 1-y+x & 0 & -x \\ -1+x & 0 & 1-x+y & -y \\ 0 & -x & -y & x+y \end{pmatrix} d\boldsymbol{x}^0 \\
&= \int_{\Omega_k^0} \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}^T \begin{pmatrix} 1-y & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & 1-x & 0 \\ 0 & 0 & 0 & x \end{pmatrix} \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} d\boldsymbol{x}^0 \\
&= \hat{\boldsymbol{G}}_{c_k}^T \boldsymbol{S}_{c_k} \hat{\boldsymbol{G}}_{c_k},
\end{aligned}$$

where we have made the definition

$$\begin{aligned}
\boldsymbol{S}_{c_k} &= \Delta x^2 \int_{\Omega_k^0} \begin{pmatrix} 1-y & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & 1-x & 0 \\ 0 & 0 & 0 & x \end{pmatrix} d\boldsymbol{x}^0 \\
&= \int_{\Omega_k} \begin{pmatrix} N_{l,m}+N_{l+1,m} & 0 & 0 & 0 \\ 0 & N_{l,m+1}+N_{l+1,m+1} & 0 & 0 \\ 0 & 0 & N_{l,m}+N_{l,m+1} & 0 \\ 0 & 0 & 0 & N_{l+1,m}+N_{l+1,m+1} \end{pmatrix} d\boldsymbol{x}. \quad (5.11)
\end{aligned}$$

The matrix $\boldsymbol{S}_{c_k}$ is just a diagonal matrix formed by integrating basis functions over the pressure cell.

Observe that pressure gradients are naturally computed at the MAC faces, which lie on edges of the pressure cells. Mathematically, we can say that $\boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} = \boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} \boldsymbol{P}_{c_k} \boldsymbol{P}_{c_k}^T$. Put another way, we can look solely at the MAC faces associated with a particular pressure cell, then ignore all pressures not associated with the cell. With this observation, it is possible to construct the

full Poisson matrix from the individual pressure cell contributions $\boldsymbol{A}_{c_k}$.

$$\begin{aligned}
\boldsymbol{A} &= \sum_k \boldsymbol{P}_{c_k} \boldsymbol{A}_{c_k} \boldsymbol{P}_{c_k}^T \\
&= \sum_k \boldsymbol{P}_{c_k} \hat{\boldsymbol{G}}_{c_k}^T \boldsymbol{S}_{c_k} \hat{\boldsymbol{G}}_{c_k} \boldsymbol{P}_{c_k}^T \\
&= \sum_k \boldsymbol{P}_{c_k} (\boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} \boldsymbol{P}_{c_k})^T \boldsymbol{S}_{c_k} (\boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} \boldsymbol{P}_{c_k}) \boldsymbol{P}_{c_k}^T \\
&= \sum_k \hat{\boldsymbol{G}}^T \boldsymbol{Q}_{c_k} \boldsymbol{S}_{c_k} \boldsymbol{Q}_{c_k}^T \hat{\boldsymbol{G}} \\
&= \hat{\boldsymbol{G}}^T \left(\sum_k \boldsymbol{Q}_{c_k} \boldsymbol{S}_{c_k} \boldsymbol{Q}_{c_k}^T \right) \hat{\boldsymbol{G}} \\
&= \hat{\boldsymbol{G}}^T \boldsymbol{S} \hat{\boldsymbol{G}},
\end{aligned}$$

where

$$\boldsymbol{S} := \sum_k \boldsymbol{Q}_{c_k} \boldsymbol{S}_{c_k} \boldsymbol{Q}_{c_k}^T.$$

This is a diagonal matrix since it is the sum of diagonal matrices. Each diagonal entry lives at an edge of the pressure grid and is formed by adding the pressure basis functions at the endpoints and integrating the result over the two pressure cells incident to the edge. The matrix $\hat{\boldsymbol{G}}^T \boldsymbol{S} \hat{\boldsymbol{G}}$ is very easy to construct and apply since it is composed of standard central difference gradient and divergence operators, with a simple diagonal scaling in between.

The discussion so far has assumed constant density, omitting it from the discretization. If $\boldsymbol{R}$ is a diagonal matrix defined over MAC faces whose diagonal entries are the density at MAC faces, then the appropriate system is $\boldsymbol{A} = \hat{\boldsymbol{G}}^T \boldsymbol{R}^{-1} \ \boldsymbol{S} \ \hat{\boldsymbol{G}}$, noting that $\boldsymbol{R}$ and $\boldsymbol{S}$ are diagonal and commute. With this factorization complete, we are ready to define $\boldsymbol{G}$ and $\boldsymbol{M}$, which we do as

$$\boldsymbol{G} = \boldsymbol{S}\hat{\boldsymbol{G}} \qquad \boldsymbol{M} = \boldsymbol{S}\boldsymbol{R}.$$

Now, $\boldsymbol{A} = \hat{\boldsymbol{G}}^T \boldsymbol{R}^{-1} \boldsymbol{S} \hat{\boldsymbol{G}} = \boldsymbol{G}^T \boldsymbol{M}^{-1} \boldsymbol{G}$, which is the desired form.

5.3.3 Right-Hand Side

As noted in Section 5.2, the Poisson equation in (5.3) has Neumann boundary conditions

$$\frac{\Delta t}{\rho^n} \nabla p^{n+1} \cdot \mathbf{n} = \boldsymbol{u}^* \cdot \mathbf{n} - U_{BC} \tag{5.12}$$

so we want the right-hand side of our discretized system to approximate the right-hand side of the corresponding weak form for p:

$$\int_\Omega \nabla q \cdot \nabla p \, d\vec{x} - \int_\Omega (\nabla \cdot \boldsymbol{u}^*)\, q \, d\vec{x} + \int_{\partial\Omega} (\boldsymbol{u}^* \cdot \mathbf{n} - U_{BC})\, q \, d\mathbf{S}(\vec{x}) \tag{5.13}$$

where $q = \sum_i q_i N_i$ is a test function. Near the embedded boundary, the operator $(\boldsymbol{G}^T \boldsymbol{u}^*)_i$ approximates not $-\int_\Omega (\nabla \cdot \boldsymbol{u}^*)\, N_i \, d\vec{x}$ but $-\int_\Omega (\nabla \cdot \boldsymbol{u}^*)\, N_i \, d\vec{x} + \int_{\partial\Omega} \boldsymbol{u}^* \cdot \mathbf{n}\, N_i \, d\mathbf{S}(\vec{x})$, which we can understand by recalling the weak form used in constructing the original stiffness matrix $\boldsymbol{E}$. Therefore, to satisfy the Neumann boundary condition (5.12) we need to approximate the rest of the boundary condition

$$-\int_{\partial\Omega} U_{BC}\, N_i \, d\mathbf{S}(\mathbf{x})$$

We approximate U_{BC} with a linear interpolant u_{BC} and solve

$$\Delta t \boldsymbol{G}^T \boldsymbol{M}^{-1} \boldsymbol{G} p^{n+1} = \boldsymbol{G}^T \boldsymbol{u}^* - \boldsymbol{u}_{BC}; \qquad (\boldsymbol{u}_{BC})_i = \int_{\partial\Omega} u_{BC} N_i \, d\mathbf{S}(\mathbf{x}). \tag{5.14}$$

Our projection step then entails solving

$$\begin{aligned} \Delta t \boldsymbol{G}^T \boldsymbol{M}^{-1} \boldsymbol{G} p^{n+1} &= \boldsymbol{G}^T \boldsymbol{u}^* - \boldsymbol{u}_{BC}; \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^* - \Delta t \boldsymbol{M}^{-1} \boldsymbol{G} p^{n+1}. \end{aligned} \tag{5.15}$$

5.3.4 Modifications for Three Dimensions

Implementation of our condensed stencil method in three dimensions is also a straightforward extension of the existing virtual node approach for Poisson problems and has the effect of condensing a 27-point stencil into a 7-point stencil which is equivalent to the standard 7-point stencil away from the boundary. As the stencil in Figure 5.2 was created by 'pushing' terms along the x direction to approximate $-p_{yy}$, and similarly along the y direction to approximate $-p_{xx}$, in three dimensions we push terms of the 27-point stencil along the x and y directions to approximate $-p_{zz}$, and a similar approach generates approximations to $-p_{xx}$ and $-p_{yy}$.

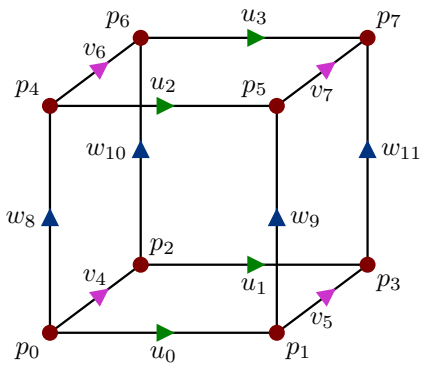
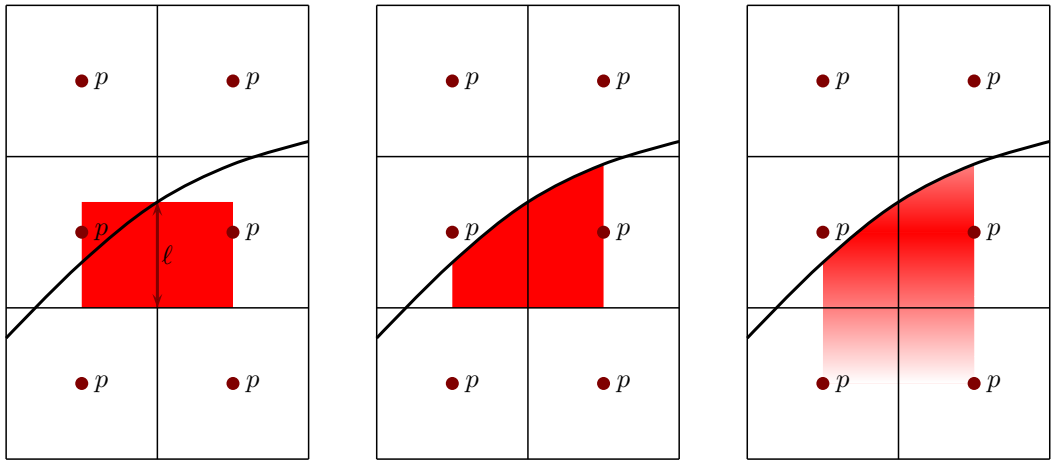


Figure 5.5: Naming conventions used in referring to pressures and velocities on a grid cell in three dimensions. Note that velocities are located at cell edges in our grid, rather than at cell faces as with a MAC grid.

Each pressure grid cell in 3D has eight adjacent pressure nodes and four each of x,y, and z velocities (see Figure 5.5), Therefore, the sizes of the cellwise matrices $\boldsymbol{G}_{c_k}$, $\hat{\boldsymbol{G}}_{c_k}$, $\boldsymbol{S}_{c_k}$, and $\boldsymbol{A}_{c_k}$ differ from the 2D case, as do the matrices $\boldsymbol{P}_{c_k}$ and $\boldsymbol{Q}_{c_k}$ described in Section 5.3.2 relating indices for a pressure cell to indices in the full grid. For example, $\hat{\boldsymbol{G}}$ is a 12-by-8 matrix defined in a straightforward manner like in (5.10).

The virtual node cellwise stiffness matrix $\boldsymbol{E}_{c_k}$ is an 8-by-8 matrix containing contributions of the cell c_k to the stiffness matrix: In three dimensions $(\boldsymbol{E}_{c_k})_{ij} = \int_{\Omega_k^0} a_{ij}\, d\boldsymbol{x}^0 = dx \int_{\Omega_k^0} \nabla N_i^0 \cdot \nabla N_j^0\, d\boldsymbol{x}^0$. The dx scaling appears in the 3D case because the cell volume scales as dx^3 and the gradient functions each scale as $1/dx$. The condensed element stiffness matrix $\boldsymbol{A}_{c_k}$ is derived in a manner analogous to the 2D case, condensing all coefficients to matrix entries where the row node and column node are adjacet. This leads to four nonzero entries per row: the diagonal entry and the columns corresponding to the three adjacent nodes. $\boldsymbol{P}_{c_k}$ has eight rows, $\boldsymbol{Q}_{c_k}$ has twelve rows, and the cellwise scaling matrix $\boldsymbol{S}_{c_k}$ is a 12-by-12 diagonal matrix similar in form to the right-hand side of (5.11). The diagonal entry of $\boldsymbol{S}_{c_k}$ for each velocity node comes from the integrals of the basis functions corresponding to the two adjacent pressure nodes. The diagonal entries of $\boldsymbol{S}$ are created by summing the contributions of the four pressure cells adjacent to each velocity.



(a) $\boldsymbol{S}_{ng}$: rectangle $\ell \times \Delta x$

(b) $\boldsymbol{S}_{pw}$: Inside area of face pressure cell

(c) $\boldsymbol{S}$: Inside area of $N_{l,m} + N_{l+1,m}$

Figure 5.6: Illustration of different constructions of the scaling matrix $\boldsymbol{S}$: $\boldsymbol{S}_{ng}$ from [51], an alternative formulation $\boldsymbol{S}_{pw}$ as described in Section 5.3.5, and the $\boldsymbol{S}$ resulting from our condensed stencil factorization explained in Section 5.3.2.

Two nontrivial differences in the 3D implementation of the virtual node algorithm are detailed in [67]. First, the domain is initialized from the signed distance function ϕ over a grid cell divided into tetrahedra to create a polyhedral approximation to the domain. Second, the method for evaluating integrals over the pressure cells also differs slightly, combining use of the divergence theorem with a quadrature-based approach to evaluating integrals over triangles.

5.3.5 Related Discretizations

The discretization we have proposed is closely related to the discretization from [51]. Their discretization also factors nicely, as was demonstrated in [75]. Expressed in our notation, their discretization is identical except with a different $\boldsymbol{S}$, which we refer to as $\boldsymbol{S}_{ng}$. The value of $\boldsymbol{S}_{ng}$ at a MAC face is proportional to the length of that face that is in the interior. Since $\boldsymbol{S}$ scales as area (in 2D), we find that $\boldsymbol{S}_{ng} = \ell \times \Delta x$ corresponds to the rectangular portion of the MAC face's associated pressure cell intersecting the face at the same point where the interface intersects the MAC face. This is illustrated in Figure 5.6(a).

While our pressure basis $\{N_i\}$ is multilinear, an alternative definition of $\boldsymbol{S}$ with a piecewise constant pressure basis would result in a different scaling matrix which we will call $\boldsymbol{S}_{pw}$. The entries of $\boldsymbol{S}_{pw}$ correspond to the area of the MAC face pressure cell that is in the interior as shown in Figure 5.6(b). If the interface is well-resolved and does not slice out a corner from this pressure cell, the entries in $\boldsymbol{S}_{ng}$ and $\boldsymbol{S}_{pw}$ will be very close. The entries in $\boldsymbol{S}$ are computed similarly to those in $\boldsymbol{S}_{pw}$, except that a weighted area is computed over a wider region as shown in Figure 5.6(c), effectively smoothing out the entries in $\boldsymbol{S}$.

Note that we use $\boldsymbol{G}^T\boldsymbol{u}^{n+1} = \hat{\boldsymbol{G}}^T\boldsymbol{S}\boldsymbol{u}^{n+1} = 0$ as our incompressibility condition rather than the perhaps more intuitive central differenced condition $\hat{\boldsymbol{G}}^T\boldsymbol{u}^{n+1} = 0$. This is in line with the condition used by [51], where the incompressibility condition takes on an intuitive flux-based interpretation.

5.4 Examples

For each example, we take a vector field (u^*, v^*) or (u^*, v^*, w^*) to be the sum of an incompressible, divergence-free component and the gradient of a pressure. We apply our projection method to the vector field and compare the solution with the exact solution of the incompressible component. Each 2D example was discretized on a variety of $N \times N$ grids for resolutions up to 562^2. Our 3D example was discretized on a series of $N \times N \times N$ grids where N ranged up to 320. With each example we provide a graphic depicting the embedded domain, a plot of the computed pressure, and error plots for pressure and for the x-component of the incompressible velocity. In each case we perform linear regression analysis on the error data to obtain estimates of the order of accuracy for the pressure in the L^∞ norm and for velocities in the L^1, L^2, and L^∞ norms. (Although we only present data for the x-component of velocities, the other velocities yield equivalent results.) Generally, our method produces pressures which are second order accurate in L^∞ and velocities which are first-order in L^∞ but second in L^1; however, projection of velocities with zero incompressible component is accurate to second order in L^∞.

5.4.1 Two-Dimensional Projection Example 1

In our first two-dimensional example we project a gradient field velocity used in [51]:

$$\begin{aligned} u^* =& (x^2 - \pi x)(\pi y^2/2 - y^3/3) \\ v^* =& (y^2 - \pi y)(\pi x^2/2 - x^3/3) \end{aligned}$$

The embedded Neumann boundary $\partial\Omega_n$ of the domain is bounded by the curve defined by:

$$\begin{aligned} t_0 =& .00132 \\ r_0 =& .02\sqrt{5} \\ r(t) =& .5 + .2\sin(5t) \\ X(\theta) =& r_0 + r(\theta + t_0)\cos(\theta + t_0) \\ Y(\theta) =& r_0 + r(\theta + t_0)\sin(\theta + t_0). \end{aligned}$$

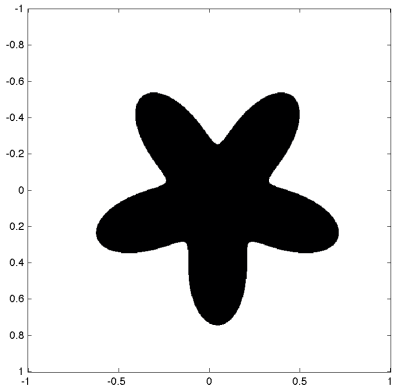
This curve is used in [1]. The domain is shown in Figure 5.7. The computed order of accuracy for pressure in the maximum norm is 1.990 and for velocity in the maximum norm is 1.876, see Figure 5.7. We computed second order accurate velocities in L^∞ whenever projecting a fully irrotational, gradient velocity field.

5.4.2 Two-Dimensional Projection Example 2

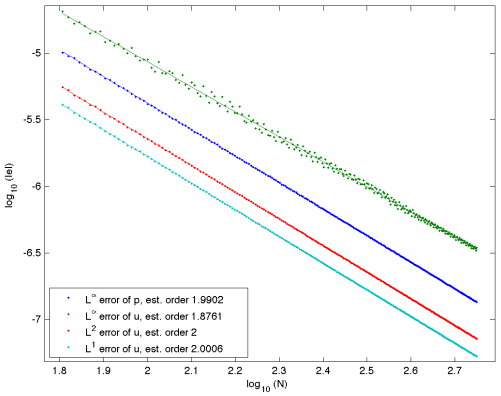
We next project the velocity field given by

$$\begin{aligned} u^* =& x + 2\pi\cos(2\pi x)\sin(2\pi y) \\ v^* =& -y + 2\pi\sin(2\pi x)\cos(2\pi y) \end{aligned}$$

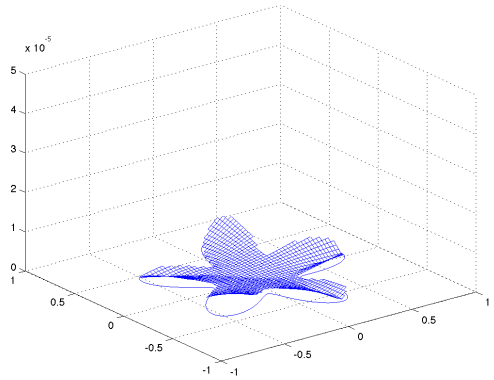
The embedded Neumann boundary $\partial\Omega_n$ of the domain is bounded by the curve defined by:



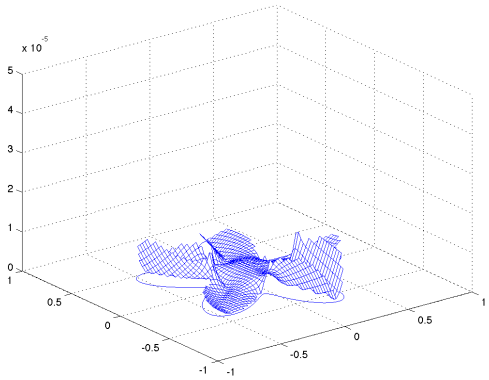
(a)



(b)



(c)



(d)

Figure 5.7: Figures for Example 5.4.1: (a) geometry of $\partial\Omega$ at $N = 80$, (b) convergence plot of the errors, and error plots of the pressure (c) and x-velocity (d) at $N = 80$.

$$
\begin{aligned}
t_0 &= .45234 \\
\theta(t) &= t + \sin(4t) \\
r(t) &= .60125 + .24012\cos(4t + \pi/2) \\
X(\theta) &= r(t+t_0)\cos(\theta(t+t_0)) \\
Y(\theta) &= r(t+t_0)\sin(\theta(t+t_0)),
\end{aligned}
$$

also used in [1], and the domain is shown in Figure 5.8. We computed the orders of accuracy to be 2.000 for pressure in L^∞ and .947 and 1.955 for velocity in L^∞ and L^1 respectively.

5.4.3 Two-Dimensional Projection Example 3

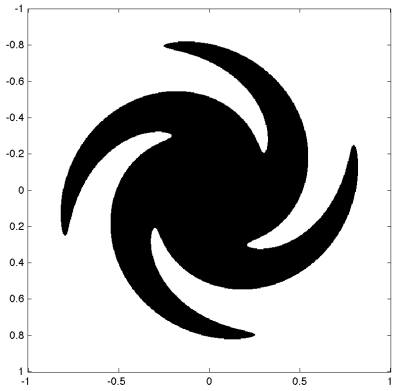
In this example our projected velocity is given as

$$
\begin{aligned}
u^* &= \sin(\pi x)\cos(\pi y) + (x+1)/((x+1)^2 + (y+4)^2) \\
v^* &= -\cos(\pi x)\sin(\pi y) + (y+4)/((x+1)^2 + (y+4)^2)
\end{aligned}
$$

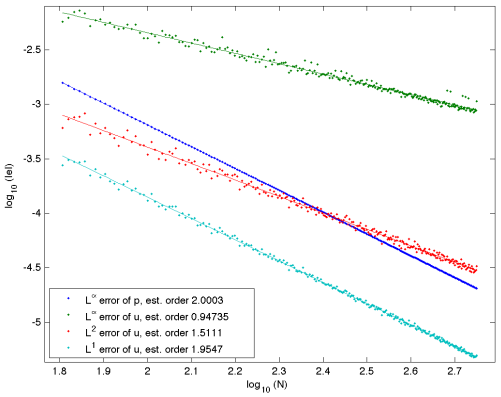
The domain Ω is defined to be the square $[-1,1] \times [-1,1]$ with a circle removed of radius .7 and centered at the origin. Grid-aligned Neumann boundary conditions are defined at the edges of the square, while embedded Neumann boundary conditions are used at the circle boundary. See Figure 5.9. We estimated an order of accuracy of 1.988 for pressure in L^∞, and .819 and 1.972 for velocity in L^∞ and L^1 respectively.

5.4.4 Three-Dimensional Projection Example

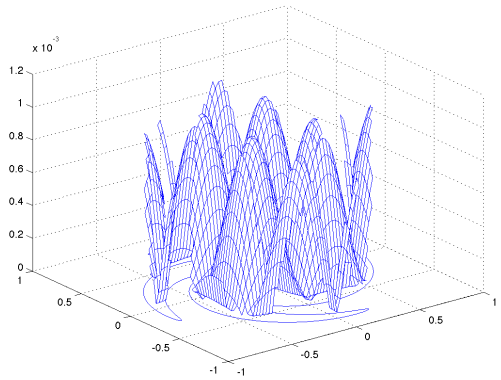
Finally, we present an application of our decomposition method in three dimensions. Here our projected velocity is given by



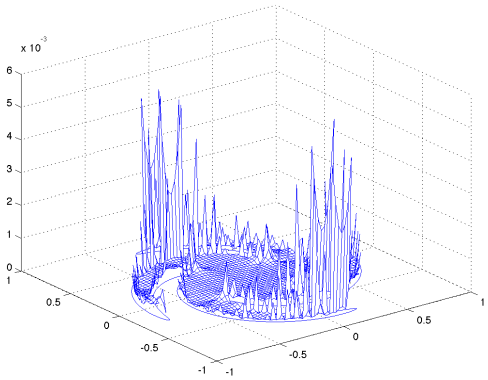
(a)



(b)

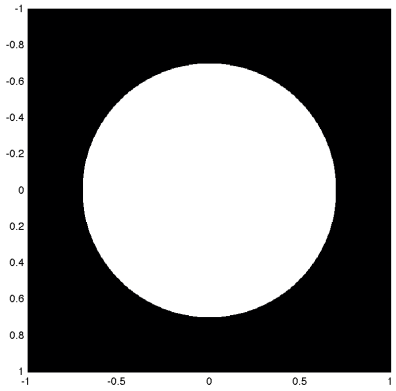


(c)

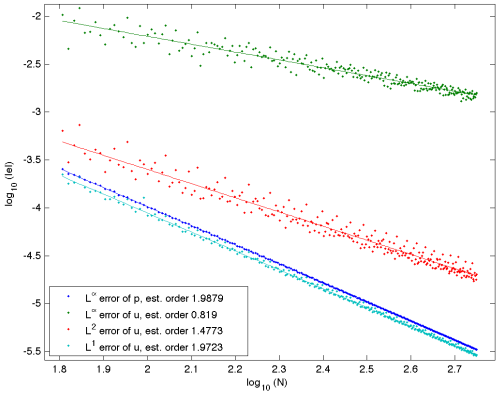


(d)

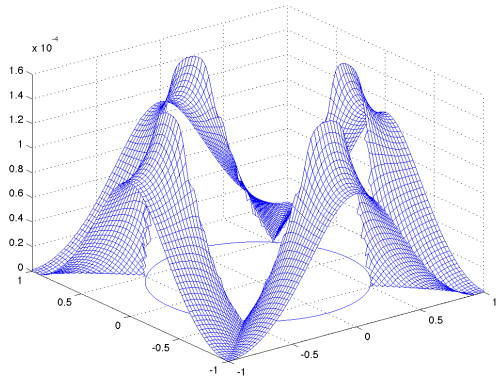
Figure 5.8: Figures for Example 5.4.2: (a) geometry of $\partial\Omega$ at $N = 80$, (b) convergence plot of the errors, and error plots of the pressure (c) and x-velocity (d) at $N = 80$.



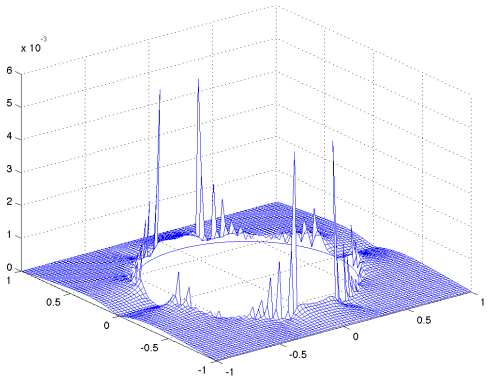
(a)



(b)



(c)

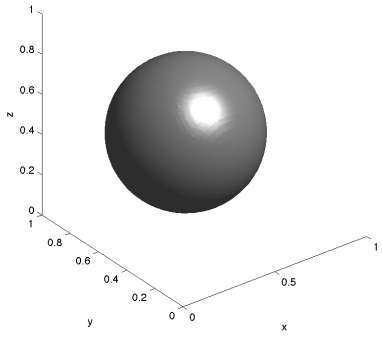


(d)

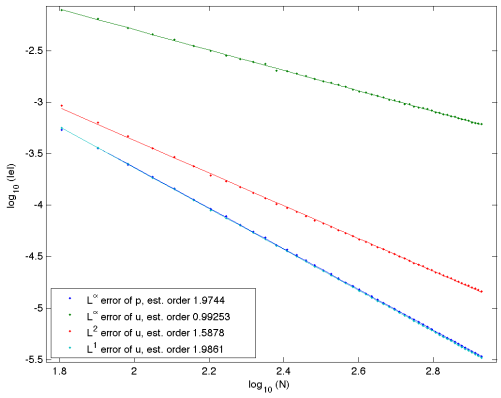
Figure 5.9: Figures for Example 5.4.3: (a) geometry of $\partial\Omega$ at $N = 80$, (b) convergence plot of the errors, and error plots of the pressure (c) and x-velocity (d) at $N = 80$.

$$u^* = .5 - y - \sin(\pi(x+y+z))$$
$$v^* = .5 - z - \sin(\pi(x+y+z))$$
$$w^* = .5 - x - \sin(\pi(x+y+z))$$

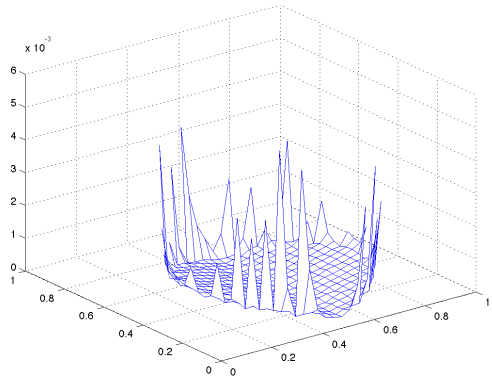
and the domain, shown in Figure 5.10, is a sphere of radius .35 centered at $(x,y,z) = (.4,.5,.5)$. As in the two-dimensional case, we observe second order accuracy (1.974) in L^∞ for the computed pressure, and order .993 in L^∞ and 1.986 in L^1 for velocity.



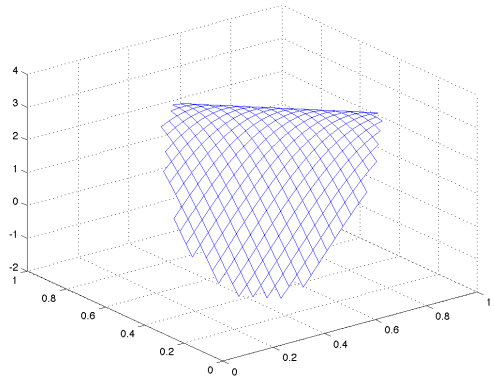
(a)



(b)



(c)



(d)

Figure 5.10: Figures for Example 5.4.4: (a) geometry of $\partial\Omega$ at $N = 32$, (b) convergence plot of the errors, and z-slices of the x-velocity error (c) and x-velocity (d) at $N = 32$.

CHAPTER 6

A Second-Order Algorithm for Navier-Stokes with Interfacial Forces

In this chapter we present the numerical method in [4] for the solution of the Navier-Stokes equations in two and three dimensions that handles interfacial discontinuities due to singular forces and discontinuous fluid properties such as viscosity and density. We show that this also allows for the enforcement of normal stress and velocity boundary conditions on irregular domains. The method improves on the method of [2], which solved the Stokes equations in two dimensions, by using a new discretization of jump and boundary conditions that accurately resolves pressure null modes in both two and three dimensions. Our embedded discretization yields discretely divergence-free velocities that are second order accurate. We implement Dirichlet, Neumann, and slip velocity boundary conditions as special cases of our interface representation. The method leads to a discrete, symmetric KKT system for velocities, pressures, and Lagrange multipliers. We also present a novel simplification to the standard combination of the second order semi-Lagrangian and BDF schemes for discretizing the inertial terms. Numerical results indicate second order spatial accuracy for the velocities (L^∞ and L^2) and first order for the pressure (in L^∞, second order in L^2). Our temporal discretization is also second order accurate. The work in this chapter was done jointly with Craig Schroeder, Alexey Stomakhin, and Prof. Joseph Teran.

6.1 Problem Formulation

We consider the Navier-Stokes equations over an irregular domain $\Omega = \Omega^+ \cup \Omega^-$ with boundary $\partial\Omega = \partial\Omega_d \cup \partial\Omega_n \cup \partial\Omega_s$, where Dirichlet velocity constraints are enforced at $\partial\Omega_d$, Neumann

boundary conditions at $\partial\Omega_n$, and slip conditions at $\partial\Omega_s$.The two subdomains Ω^+ and Ω^- are separated by an interface Γ, which is typically a co-dimension one closed surface. The corresponding equations are

$$\rho \boldsymbol{u}_t + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \nabla\cdot\sigma + \boldsymbol{f}, \qquad \boldsymbol{x}\in\Omega\backslash\Gamma \tag{6.1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \qquad \boldsymbol{x}\in\Omega\backslash\Gamma \tag{6.2}$$

$$[\boldsymbol{u}] = \boldsymbol{a}_i, \qquad \boldsymbol{x}\in\Gamma \tag{6.3}$$

$$[\sigma\cdot\boldsymbol{n}] = \hat{\boldsymbol{f}}, \qquad \boldsymbol{x}\in\Gamma \tag{6.4}$$

$$\boldsymbol{u} = \boldsymbol{b}, \qquad \boldsymbol{x}\in\partial\Omega_d \tag{6.5}$$

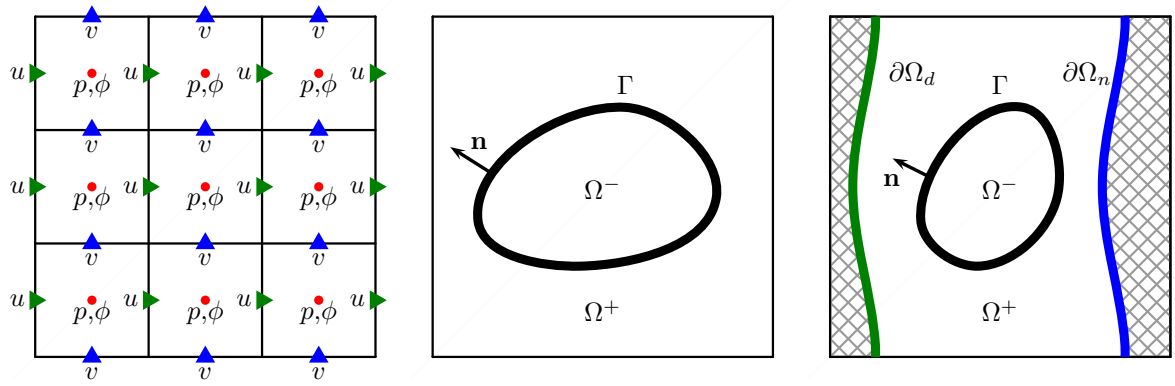
$$\sigma\cdot\boldsymbol{n} = \hat{\boldsymbol{g}}, \qquad \boldsymbol{x}\in\partial\Omega_n \tag{6.6}$$

$$\boldsymbol{n}\cdot\boldsymbol{u} = \boldsymbol{n}\cdot\boldsymbol{c}, \qquad \boldsymbol{x}\in\partial\Omega_s \tag{6.7}$$

$$(\boldsymbol{I}-\boldsymbol{n}\boldsymbol{n}^T)\cdot\sigma\cdot\boldsymbol{n} = (\boldsymbol{I}-\boldsymbol{n}\boldsymbol{n}^T)\hat{\boldsymbol{h}}, \qquad \boldsymbol{x}\in\partial\Omega_s \tag{6.8}$$

where the stress is $\sigma = \mu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T) - p\boldsymbol{I}$, $\boldsymbol{a}_i$ describes velocity jumps at interfaces, $\boldsymbol{b}$ describes velocities at Dirichlet boundaries, $\boldsymbol{c}$ represents slip velocities, $\hat{\boldsymbol{f}}$ describes interface forces, $\hat{\boldsymbol{g}}$ describes Neumann boundary conditions, and $\hat{\boldsymbol{h}}$ describes tangential stresses for slip boundary conditions. Although for physical problems the velocity jump is equal to zero (representing continuity of the velocity), we include a velocity jump which may be nonzero in our formulation of the interface. This is convenient not only for testing our implementation, but for handling the other types of boundary conditions. We take the standard convention of defining $[\boldsymbol{u}] = \boldsymbol{u}^+ - \boldsymbol{u}^-$ for interface jumps and $\boldsymbol{n}$ as the normal pointing outward from Ω^-. The interface Γ and boundary pieces $\partial\Omega_d$, $\partial\Omega_n$, and $\partial\Omega_s$ are assumed to be smooth and not intersect one another. This layout is illustrated in Figure 6.1. We do not consider triple junctions in this paper.

We have previously developed a class of embedded methods that utilize uniform Cartesian grids and sub-cell representations of interface/boundary geometry to achieve optimal accuracy without the need for frequent remeshing [2, 1, 67, 68]. Our use of regular grids simplifies the implementation, permits straightforward numerical linear algebra and naturally allows for higher order accuracy in L^∞. We have used the term *virtual node methods* to describe these techniques since they utilize additional structured degrees that are outside the domain of interest.



(a) (b) (c)

Figure 6.1: (a) MAC layout in two dimensions. The red dots indicate location of pressure and level set variables, the green and blue triangles represent horizontal (u) and vertical (v) variables respectively. (b) Interface Γ separates the fluid domain $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$. In this case all sides of the computational domain have a periodic boundary condition applied. (c) Another layout which our method can handle in practice. Here an embedded Dirichlet boundary condition is applied on the left side of the domain Ω and a Neumann boundary condition on the right.

In the present work, we introduce a new virtual node method for approximating the two-phase Navier-Stokes equations with irregular embedded interfaces and boundaries on a uniform Cartesian MAC grid.

As in [2], we duplicate Cartesian grid cells along the interface Γ to introduce additional virtual nodes that accurately resolve discontinuous quantities. This naturally treats discontinuities in material properties such as viscosity and density. Interface cells are cut and duplicated using a level set that allows for accurate evaluation of integrals needed for the numerical stencils. These stencils (for the viscous stress forces as well as the divergence-free and jump constraints) are constructed from a variational formulation that yields a symmetric linear system. This approach requires the introduction of a Lagrange multiplier variable to maintain continuity of the fluid velocity across the interface. Unfortunately, the introduction of this variable forced the approach in [2] to require that interface domain geometry have a constant normal on each MAC grid cell. Although it is a reasonable restriction in two dimensions, this is not possible in three dimensions and so the method was fundamentally limited to 2D. We present an improved discretization of this Lagrange multiplier term that works naturally in both two and three dimensions without the restriction of a constant normal per MAC grid cell. The necessity of this in [2] was due to the requirement that the discretization resolve null modes in the variational form of the equations exactly. Failure to do this resulted in significant degradation in performance. We show that our new discretization also captures these modes exactly.

We also consider a simplification to the combination of the BDF and second order semi-Lagrangian schemes that is often used in second order Navier-Stokes discretizations to calculate the intermediate velocity field [76]. This simplification reduces the number of semi-Lagrangian interpolation steps required from four to two while retaining the temporal and spatial accuracy of the original method. The interface is evolved using the level set method or, when more appropriate, the particle level set method. Numerical experiments indicate second order accuracy in L^∞ and L^2 for the velocity and first order accuracy in L^∞, second in L^2, for pressure. Numerical experiments indicate a stability restriction on the *minimum* time step size (relative to the grid spacing) that may be taken by our method in the case of a Navier-Stokes discretization. We explore the nature and source of this restriction further.

6.2 Numerical method

Our method extends the framework of [2] from two to three spatial dimensions and from the Stokes equations to the full Navier-Stokes equations. As with [2], our method is second order in L^∞ and L^2 for the velocities and first order in L^∞ for the pressure (second order in L^2) for embedded interfacial discontinuities and irregular boundaries in two-phase flows, though unlike the previous work, we are also second order in time. Our method produces sparse and symmetric indefinite KKT-type linear systems. Furthermore, our approach yields discretely divergence-free velocities. We do not require knowledge of jumps on the fluid variables and their derivatives but rather only of expressions for the interfacial forces.

Our numerical method uses an Eulerian representation of the fluid velocity and pressure. A level set ϕ represents the domain Ω and interface Γ at each time step. We split the Navier-Stokes equation (6.1) into two equations by introducing an intermediate velocity $\boldsymbol{u}^\star$. We use a weak form of the Navier-Stokes interface problem and simultaneously solve for the pressure, velocity, and Lagrange multipliers needed to preserve the specified velocity jumps across the interface. The complete procedure for advancing one time step is:

1. Update level set $\phi^n \rightarrow \phi^{n+1}$
2. Advect time $n-1$ and n velocity
3. Compute intermediate fluid velocity $\boldsymbol{u}^\star$ using BDF
4. Setup symmetric system and right hand side
5. Solve the linear system to obtain p and $\boldsymbol{u}^{n+1}$

We will describe each of these steps in necessary detail, following which we will discuss how to add surface tension forces, and present requirements for the stability of our method.

6.2.1 Spatial discretization

We use a standard MAC layout for our degrees of freedom, with velocity degrees of freedom at faces and pressures at cell centers, as shown in Figure 6.1(a). We maintain our level set at cell centers.

Fluid advection and the implicit update at the end both require valid ghost data in a narrow band around the region where each fluid phase is defined. We accommodate this by storing a separate velocity array for each fluid phase for $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$, which over the course of a time step we update to $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n+1}$. At each MAC face, only one of the velocity arrays is considered to hold a real velocity degree of freedom, based on the sign of the level set interpolated to that face. The other array is treated as ghost data and is populated as needed using the extrapolation of [77]. Note that across interfaces the velocity field, though physically continuous, may have a jump in its derivatives. In our case, we also included support for velocity discontinuities, since this makes analytic tests far easier to construct and thus the method itself easier to test and debug.

In practice, we found using the level set directly to distinguish real and ghost velocities to be too unreliable and on a few occasions led to the use of invalid velocity data. We avoid these problems by explicitly storing which, if any, velocity degree of freedom is valid at each MAC face, both for $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$. Since we do this, there is no need to maintain a level set other than the one at the current time.

6.2.2 Update level set

We discretize our momentum equation at time t^{n+1}. Setting up the intermediate $\boldsymbol{u}^\star$ requires that we know which fluid region is valid at each face, which is determined by ϕ^{n+1}. Thus, we begin our time step by updating our level set $\phi^n \to \phi^{n+1}$. We use the level set method for this task in most of our examples. For examples where the level set method loses volume too quickly, we use the more expensive but more accurate particle level set method [78] instead.

In the case of the level set method, there are two steps: advection and reinitialization. We advect our level set using third order Runge-Kutta [79] in time and fifth order HJ-WENO [80,

81] in space. This update requires an estimate for $\boldsymbol{u}^n$, $\boldsymbol{u}^{n+1}$, and then $\boldsymbol{u}^{n+\frac{1}{2}}$. To obtain these, we merge our per-phase velocity fields into a single velocity field by selecting the non-virtual degree of freedom at each MAC face (as determined by the sign of ϕ). We will call these merged velocities $\boldsymbol{u}_m^{n-1}$ and $\boldsymbol{u}_m^n$. Our velocity estimates for the level set advection are then $\boldsymbol{u}_m^n$, $2\boldsymbol{u}_m^n - \boldsymbol{u}_m^{n-1}$, and $\frac{3}{2}\boldsymbol{u}_m^n - \frac{1}{2}\boldsymbol{u}_m^{n-1}$. Note that our velocities do not live at the same locations as our level set, so interpolation is required to co-locate them prior to advection. For reinitialization we use also use third order Runge-Kutta and fifth order HJ-WENO.

When we use the particle level set method, we perform advection and reinitialization the same way we do for the level set method. In addition to this, the particle level set method also does particle evolution, for which we use third order Runge-Kutta and the same velocity estimates as for the level set advection step.

6.2.3 Discretization of inertial terms

Following [76], we use a variant on semi-Lagrangian for advection and a BDF discretization for the time derivative. This transforms (6.1) into

$$\rho \frac{3\boldsymbol{u}^{n+1} - 4\boldsymbol{u}_d^n + \boldsymbol{u}_d^{n-1}}{2\Delta t} = \nabla \cdot \sigma + \boldsymbol{f}, \tag{6.9}$$

where $\boldsymbol{u}_d^n = \boldsymbol{u}^n(\boldsymbol{x}^n)$ and $\boldsymbol{u}_d^{n-1} = \boldsymbol{u}^{n-1}(\boldsymbol{x}^{n-1})$ indicate that the $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n-1}$ velocities are evaluated at the departure locations obtained by tracing the position back along the characteristics of the fluid flow from the face located at $\boldsymbol{x}^{n+1}$. By introducing an intermediate velocity $\boldsymbol{u}^\star$ we split the Navier-Stokes equations into the two separate equations:

$$\rho \frac{3\boldsymbol{u}^\star - 4\boldsymbol{u}_d^n + \boldsymbol{u}_d^{n-1}}{2\Delta t} = \boldsymbol{0}, \tag{6.10}$$

in which we apply the inertial terms to an intermediate velocity, and

$$\alpha(\boldsymbol{u}^{n+1} - \boldsymbol{u}^\star) = \nabla \cdot \sigma + \boldsymbol{f}, \tag{6.11}$$

where we have introduced $\alpha = \frac{3\rho}{2\Delta t}$ for convenience accommodating the first time step. We discuss the first time step later in this section.

Standard discretization To obtain second order temporal and spatial accuracy, [76] computes the advected $\boldsymbol{u}_d^n$ and $\boldsymbol{u}_d^{n-1}$ using

$$\boldsymbol{u}_d^n = \boldsymbol{u}^n(\boldsymbol{x}^{n+1} - \Delta t \boldsymbol{u}^{n+\frac{1}{2}}(\boldsymbol{x}^{n+1} - \tfrac{1}{2}\Delta t \boldsymbol{u}^n(\boldsymbol{x}^{n+1}))), \tag{6.12}$$

where $\boldsymbol{u}^{n+\frac{1}{2}} = \frac{3}{2}\boldsymbol{u}^n - \frac{1}{2}\boldsymbol{u}^{n-1}$, and

$$\boldsymbol{u}_d^{n-1} = \boldsymbol{u}^{n-1}(\boldsymbol{x}^{n+1} - 2\Delta t \boldsymbol{u}^n(\boldsymbol{x}^{n+1} - \Delta t \boldsymbol{u}^n(\boldsymbol{x}^{n+1}))). \tag{6.13}$$

These formulas require three velocity evaluations each. The innermost is evaluated at the face and is a simple lookup. The middle evaluation requires interpolation, which can be linear interpolation since it will be multiplied by an extra factor of Δt. The outermost velocity evaluation also requires interpolation, but this time quadratic interpolation is required to obtain second order spatial accuracy.

Simplified discretization Taylor series analysis of the advection and BDF process reveals that the standard approach to computing $\boldsymbol{u}_d^n$ and $\boldsymbol{u}_d^{n-1}$ is more expensive than necessary. In particular, it suffices to use

$$\boldsymbol{u}_d^n = \boldsymbol{u}^n(\boldsymbol{x}^{n+1} - \Delta t \boldsymbol{u}^n(\boldsymbol{x}^{n+1})) \tag{6.14}$$

$$\boldsymbol{u}_d^{n-1} = \boldsymbol{u}^{n-1}(\boldsymbol{x}^{n+1} - 2\Delta t \boldsymbol{u}^{n-1}(\boldsymbol{x}^{n+1})). \tag{6.15}$$

That is, the application of inertial terms simplifies down to doing a step of semi-Lagrangian advection on $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n-1}$ and then computing $\boldsymbol{u}^\star$ as a linear combination of the results using (6.10). Normally, using semi-Lagrangian advection would only be expected to produce first order temporal accuracy, but using it in combination with BDF in this way yields second order temporal accuracy. Note that the interaction of semi-Lagrangian advection and BDF does not improve spatial accuracy, so quadratic interpolation is still required for the semi-Lagrangian advection steps. We use this simplified discretization in all of our numerical examples.

First step Since BDF is a multistep method, we need a way to take the first step. We can afford one time step with one order lower, so we simply use a backward Euler discretization,

which leads to the alternate formula

$$\rho \frac{\boldsymbol{u}^\star - \boldsymbol{u}_d^n}{\Delta t} = \boldsymbol{0} \tag{6.16}$$

for computing $\boldsymbol{u}^\star$, where $\alpha = \frac{\rho}{\Delta t}$ is used in (6.11) and $\boldsymbol{u}_d^n$ is computed as in (6.14).

6.2.4 Discretization of implicit terms

We follow the derivation put forth in [2] to discretize the implicit portion of our splitting. From (6.11), the continuity equation, and the boundary conditions associated with the problem, we derive the weak form, which yields our discrete stencils for the velocity and fluid variables. We continue by detailing aspects of our discretization necessary to account for possible null modes of the linear system, and to admit Dirichlet, Neumann, and slip boundary conditions. We then discuss our implementation of computing the integrals necessary to obtain the discrete stencils in our discretization.

6.2.4.1 Continuous weak form

We begin by taking a dot product of both sides of (6.11) by a test function $\boldsymbol{w}$, then integrating both sides over $\Omega \backslash \Gamma$ to get

$$\int_{\Omega\backslash\Gamma} \alpha \boldsymbol{w} \cdot (\boldsymbol{u} - \boldsymbol{u}^\star)\, dV = \int_{\Omega\backslash\Gamma} \boldsymbol{w} \cdot (\nabla \cdot \sigma + \boldsymbol{f})\, dV, \tag{6.17}$$

where we have used $\boldsymbol{u} = \boldsymbol{u}^{n+1}$ for conciseness. We assume for now that we have an interface at Γ but no other non-periodic boundaries. We will discuss other boundary conditions in Section 6.2.4.5. Integration by parts yields

$$\int_{\Omega\backslash\Gamma} \boldsymbol{w} \cdot (\nabla \cdot \sigma)\, dV = \int_{\Omega\backslash\Gamma} \nabla \cdot (\boldsymbol{w} \cdot \sigma) - \nabla \boldsymbol{w} : \sigma\, dV = -\int_\Gamma [\boldsymbol{w} \cdot \sigma] \cdot \boldsymbol{n}\, dA - \int_{\Omega\backslash\Gamma} \nabla \boldsymbol{w} : \sigma\, dV, \tag{6.18}$$

where $\boldsymbol{n}$ is the outward normal from Ω^- and $[\boldsymbol{w}] = \boldsymbol{w}^+ - \boldsymbol{w}^-$ denotes the jump in $\boldsymbol{w}$ across the interface. Then,

$$\int_{\Omega\backslash\Gamma} \alpha \boldsymbol{w} \cdot \boldsymbol{u}\, dV + \int_{\Omega\backslash\Gamma} \nabla \boldsymbol{w} : \sigma\, dV + \int_\Gamma [\boldsymbol{w} \cdot \sigma] \cdot \boldsymbol{n}\, dA = \int_{\Omega\backslash\Gamma} \alpha \boldsymbol{w} \cdot \boldsymbol{u}^\star\, dV + \int_{\Omega\backslash\Gamma} \boldsymbol{w} \cdot \boldsymbol{f}\, dV. \tag{6.19}$$

Utilizing symmetry,

$$\begin{aligned}\int_{\Omega\setminus\Gamma} \nabla \boldsymbol{w} : \boldsymbol{\sigma}\, dV &= \int_{\Omega\setminus\Gamma} \nabla \boldsymbol{w} : (\mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) - p\boldsymbol{I})\, dV \\ &= \int_{\Omega\setminus\Gamma} \mu \nabla \boldsymbol{w} : (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)\, dV - \int_{\Omega\setminus\Gamma} \nabla \cdot \boldsymbol{w} p\, dV \\ &= \int_{\Omega\setminus\Gamma} \frac{\mu}{2}(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T) : (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)\, dV - \int_{\Omega\setminus\Gamma} \nabla \cdot \boldsymbol{w} p\, dV\end{aligned}$$

Using the identity $[\boldsymbol{w}\cdot\boldsymbol{\sigma}] = [\boldsymbol{w}]\cdot\overline{\boldsymbol{\sigma}} + \overline{\boldsymbol{w}}\cdot[\boldsymbol{\sigma}]$, where $\overline{\boldsymbol{\sigma}} = \frac{1}{2}(\boldsymbol{\sigma}^+ + \boldsymbol{\sigma}^-)$ and $\overline{\boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w}^+ + \boldsymbol{w}^-)$,

$$\int_\Gamma [\boldsymbol{w}\cdot\boldsymbol{\sigma}]\cdot\boldsymbol{n}\, dA = \int_\Gamma [\boldsymbol{w}]\cdot\overline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\, dA + \int_\Gamma \overline{\boldsymbol{w}}\cdot[\boldsymbol{\sigma}]\cdot\boldsymbol{n}\, dA = \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA + \int_\Gamma \overline{\boldsymbol{w}}\cdot\hat{\boldsymbol{f}}\, dA, \tag{6.20}$$

where $\hat{\boldsymbol{f}} = [\boldsymbol{\sigma}]\cdot\boldsymbol{n}$ is known but $\boldsymbol{q} = \overline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}$ must be treated as a degree of freedom since its value will not in general be known. Combining these with (6.19) yields

$$\begin{aligned}\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}\, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T) : (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)\, dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w}\, dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA = \\ \int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}^\star\, dV + \int_{\Omega\setminus\Gamma} \boldsymbol{w}\cdot\boldsymbol{f}\, dV + \int_\Gamma \overline{\boldsymbol{w}}\cdot\hat{\boldsymbol{f}}\, dA.\end{aligned} \tag{6.21}$$

Introducing test functions λ and $\boldsymbol{v}$, the weak forms for (6.2) and (6.3) are

$$\int_{\Omega\setminus\Gamma} \lambda\nabla\cdot\boldsymbol{u}\, dV = 0 \tag{6.22}$$

$$\int_\Gamma \boldsymbol{v}\cdot[\boldsymbol{u}]\, dA = \int_\Gamma \boldsymbol{v}\cdot\boldsymbol{a}_i\, dA. \tag{6.23}$$

The equations (6.21-6.23) form our weak form of the Navier-Stokes problem. Note that the test functions $\boldsymbol{w}$, λ, and $\boldsymbol{v}$ complement the unknowns $\boldsymbol{u}$, p, and $\boldsymbol{q}$ respectively. The pressure p and other Lagrange multiplier $\boldsymbol{q}$ enforce the continuity equation and the velocity jump condition, respectively.

6.2.4.2 Discretization

We begin our discretization procedure by cutting each grid cell of the computational domain into portions belonging to Ω^+ and Ω^- with the aid of the level set function ϕ^{n+1}. Each cell that is cut will have one or more triangles (segments in 2D) which are used to calculate integrals

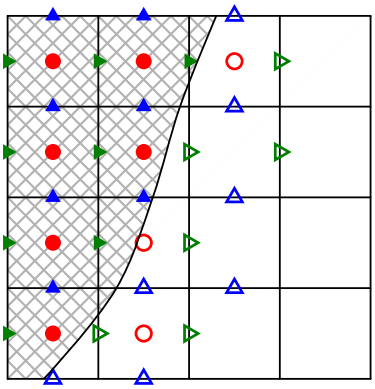


Figure 6.2: A portion of a fluid grid is cut by an interface. The degrees of freedom for the hashed region are shown. Solid markers indicate real degrees of freedom, and hollow markers indicate virtual degrees of freedom. Note that only degrees of freedom with interpolating functions whose support intersects the the hashed region participate in the discretization.

over each cell that is cut. We refer to an individual triangle as a *surface element.* Different approaches exist for performing cutting from a level set; we explain the method we use in Section 6.2.4.6.

Following [2], we place $\boldsymbol{u}$ and p in space according to a standard MAC layout, with p at cell centers and $\boldsymbol{u}$ components at face centers. We create copies of these degrees of freedom near interfaces so that each phase will have virtual values available as shown in Figure 6.2. Although these degrees of freedom can be assigned directly, we create a copy of each degree of freedom for Ω^- and Ω^+, so that each fluid region has its own set of variables, and then discard the variables that are never referenced (those whose interpolating function support does not intersect Ω^- or Ω^+ respectively, see Figure 6.2). The result is the same, but we found the resulting algorithm to be easier to implement. We follow a finite element discretization, letting

$$\boldsymbol{u}^x(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^x N_i^x(\boldsymbol{x}), \quad \boldsymbol{u}^y(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^y N_i^y(\boldsymbol{x}), \quad \boldsymbol{u}^z(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^z N_i^z(\boldsymbol{x}),$$
$$p(\boldsymbol{x}) = \sum_i p_i P_i(\boldsymbol{x}), \quad \boldsymbol{q}(\boldsymbol{x}) = \sum_i q_i \boldsymbol{Q}_i(\boldsymbol{x}), \quad (6.24)$$

as in [2], where $N_i^x(\boldsymbol{x})$, $N_i^y(\boldsymbol{x})$, and $N_i^z(\boldsymbol{x})$ define the standard piecewise trilinear basis functions associated with the velocity nodes for the respective dimension, and $P_i(\boldsymbol{x})$ is 1 in MAC cell i

and 0 otherwise. We have also introduced the (vector-valued) basis $\boldsymbol{Q}_i$ and (scalar) degrees of freedom q_i for $\boldsymbol{q}(\boldsymbol{x})$. The way these are defined is critical to capturing the null mode properly, and we delay the definition of these until Section 6.2.4.4. We discretize the test variables the same way as their corresponding degrees of freedom as

$$\boldsymbol{w}^x(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^x N_i^x(\boldsymbol{x}) \quad \boldsymbol{w}^y(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^y N_i^y(\boldsymbol{x}) \quad \boldsymbol{w}^z(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^z N_i^z(\boldsymbol{x}) \quad \lambda(\boldsymbol{x}) = \sum_i \lambda_i P_i(\boldsymbol{x})$$
$$\boldsymbol{v}(\boldsymbol{x}) = \sum_i v_i \boldsymbol{Q}_i(\boldsymbol{x}). \quad (6.25)$$

This leaves the discretization of the forcing terms. The body force $\boldsymbol{f}$ is discretized as a vector quantity $(\boldsymbol{f}_i^x, \boldsymbol{f}_i^y, \boldsymbol{f}_i^z)$ that is constant per MAC cell as

$$\boldsymbol{f}^x(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^x P_i(\boldsymbol{x}) \quad \boldsymbol{f}^y(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^y P_i(\boldsymbol{x}) \quad \boldsymbol{f}^z(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^z P_i(\boldsymbol{x}). \quad (6.26)$$

The interface force $\hat{\boldsymbol{f}}$ is discretized with a vectoral force $(\hat{\boldsymbol{f}}_i^x, \hat{\boldsymbol{f}}_i^y, \hat{\boldsymbol{f}}_i^z)$ that is constant over each surface element i. Letting E_i be 1 on surface element i and 0 elsewhere,

$$\hat{\boldsymbol{f}}^x(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^x E_i(\boldsymbol{x}) \quad \hat{\boldsymbol{f}}^y(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^y E_i(\boldsymbol{x}) \quad \hat{\boldsymbol{f}}^z(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^z E_i(\boldsymbol{x}). \quad (6.27)$$

The velocity jump is discretized in the same way as $\hat{\boldsymbol{f}}$, so that

$$\boldsymbol{a}^x(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^x E_i(\boldsymbol{x}) \quad \boldsymbol{a}^y(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^y E_i(\boldsymbol{x}) \quad \boldsymbol{a}^z(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^z E_i(\boldsymbol{x}). \quad (6.28)$$

The discretized equations can now be written in matrix form as

$$\begin{pmatrix} \boldsymbol{M}^x + \boldsymbol{A}^{xx} + \boldsymbol{B}^x & \boldsymbol{A}^{xy} & \boldsymbol{A}^{xz} & -\boldsymbol{G}^x & \boldsymbol{H}^x \\ \boldsymbol{A}^{yx} & \boldsymbol{M}^y + \boldsymbol{A}^{yy} + \boldsymbol{B}^y & \boldsymbol{A}^{yz} & -\boldsymbol{G}^y & \boldsymbol{H}^y \\ \boldsymbol{A}^{zx} & \boldsymbol{A}^{zy} & \boldsymbol{M}^z + \boldsymbol{A}^{zz} + \boldsymbol{B}^z & -\boldsymbol{G}^z & \boldsymbol{H}^z \\ -(\boldsymbol{G}^x)^T & -(\boldsymbol{G}^y)^T & -(\boldsymbol{G}^z)^T & \boldsymbol{0} & \boldsymbol{0} \\ (\boldsymbol{H}^x)^T & (\boldsymbol{H}^y)^T & (\boldsymbol{H}^z)^T & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^x \\ \boldsymbol{u}^y \\ \boldsymbol{u}^z \\ \boldsymbol{p} \\ \boldsymbol{q} \end{pmatrix} =$$
$$\begin{pmatrix} \boldsymbol{M}^x(\boldsymbol{u}^\star)^x + \boldsymbol{J}^x\boldsymbol{f}^x + \boldsymbol{K}^x\hat{\boldsymbol{f}}^x \\ \boldsymbol{M}^y(\boldsymbol{u}^\star)^y + \boldsymbol{J}^y\boldsymbol{f}^y + \boldsymbol{K}^y\hat{\boldsymbol{f}}^y \\ \boldsymbol{M}^z(\boldsymbol{u}^\star)^z + \boldsymbol{J}^z\boldsymbol{f}^z + \boldsymbol{K}^z\hat{\boldsymbol{f}}^z \\ \boldsymbol{0} \\ \boldsymbol{L}^x\boldsymbol{a}^x + \boldsymbol{L}^y\boldsymbol{a}^y + \boldsymbol{L}^z\boldsymbol{a}^z \end{pmatrix}, \quad (6.29)$$

where we have defined the matrix blocks ($r,s \in \{x,y,z\}$)

$$(\boldsymbol{M}^r)_{ij} = \alpha \int_{\Omega\setminus\Gamma} N_i^r N_j^r \, dV \qquad (\boldsymbol{A}^{rs})_{ij} = \int_{\Omega\setminus\Gamma} \mu N_{i,s}^r N_{j,r}^s \, dV \tag{6.30}$$

$$(\boldsymbol{B}^r)_{ij} = \sum_{s\in\{x,y,z\}} \int_{\Omega\setminus\Gamma} \mu N_{i,s}^r N_{j,s}^r \, dV \qquad (\boldsymbol{G}^r)_{ij} = \int_{\Omega\setminus\Gamma} N_{i,r}^r P_j \, dV \tag{6.31}$$

$$(\boldsymbol{H}^r)_{ij} = \int_{\Gamma} \Theta_i N_i^r \boldsymbol{Q}_j^r \, dA \qquad (\boldsymbol{J}^r)_{ij} = \int_{\Omega\setminus\Gamma} N_i^r P_j \, dV \tag{6.32}$$

$$(\boldsymbol{K}^r)_{ij} = \int_{\Gamma} \Phi_i N_i^r E_j \, dA \qquad (\boldsymbol{L}^r)_{ij} = \int_{\Gamma} \boldsymbol{Q}_i^r E_j \, dA. \tag{6.33}$$

In the $\boldsymbol{H}^r$ matrices, the value $\Theta_i = 1$ if degree of freedom i corresponds to the Ω^+ phase and $\Theta_i = -1$ if degree of freedom i corresponds to the Ω^- phase. For an interface, $\Phi_i = \frac{1}{2}$ in the $\boldsymbol{K}^r$ matrices; this will change for other types of boundary conditions that we mention in Section 6.2.4.5.

6.2.4.3 Null Modes

The discretization of the Stokes equations in [2] allowed for nullspace modes corresponding to the null modes of the corresponding continuous weak formulation. In the periodic Stokes case, there is a constant velocity mode per dimension and a constant pressure mode. In problems with an interface, the pressure mode will also have nonzero $\boldsymbol{q}$ entries. The primary limitation restricting the discretization in [2] to two dimensions is the inability to capture the pressure mode discretely in 3D. We present a modification to the discretization of $\boldsymbol{q}$ that resolves this limitation and captures null modes discretely in either two or three dimensions.

First, we must identify the null modes for our weak formulation of Navier-Stokes with an interface and a periodic boundary, but no other boundary conditions. (We will discuss the effect of other boundary conditions on null modes in Section 6.2.4.5.) A null mode $(\boldsymbol{u}, p, \boldsymbol{q})$ must

satisfy homogenous versions of (6.21), (6.22), and (6.23) for any $(\boldsymbol{w}, \lambda, \boldsymbol{v})$:

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}\, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}(\nabla\boldsymbol{w} + \nabla\boldsymbol{w}^T) : (\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\, dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w}\, dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA = 0 \tag{6.34}$$

$$\int_{\Omega\setminus\Gamma} \lambda\nabla\cdot\boldsymbol{u}\, dV = 0 \tag{6.35}$$

$$\int_\Gamma \boldsymbol{v}\cdot[\boldsymbol{u}]\, dA = 0. \tag{6.36}$$

Letting $\boldsymbol{w} = \boldsymbol{u}$, $\lambda = p$, and $\boldsymbol{v} = \boldsymbol{q}$,

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{u}\cdot\boldsymbol{u}\, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) : (\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\, dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{u}\, dV + \int_\Gamma [\boldsymbol{u}]\cdot\boldsymbol{q}\, dA = 0 \tag{6.37}$$

$$\int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{u}\, dV = 0 \tag{6.38}$$

$$\int_\Gamma \boldsymbol{q}\cdot[\boldsymbol{u}]\, dA = 0. \tag{6.39}$$

Combining these yields

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{u}\cdot\boldsymbol{u}\, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) : (\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\, dV = 0 \tag{6.40}$$

Both terms are clearly nonnegative. Since $\alpha > 0$, the first term will be positive unless $\boldsymbol{u} = \boldsymbol{0}$. Thus, any null mode necessarily has $\boldsymbol{u} = \boldsymbol{0}$. Note that our weak Navier-Stokes formulation has no translational null modes, unlike the periodic Stokes problem. This reduces the homogeneous system to

$$\boldsymbol{0} = -\int_{\Omega\setminus\Gamma} \nabla\cdot\boldsymbol{w}\, p\, dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA \tag{6.41}$$

$$= -\int_{\Omega\setminus\Gamma} \nabla\cdot(p\boldsymbol{w})\, dV + \int_{\Omega\setminus\Gamma} \boldsymbol{w}\cdot\nabla p\, dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA \tag{6.42}$$

$$= \int_{\Omega\setminus\Gamma} \boldsymbol{w}\cdot\nabla p\, dV + \int_\Gamma [p\boldsymbol{w}]\cdot\boldsymbol{n}\, dA + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA \tag{6.43}$$

It remains to determine conditions on p and $\boldsymbol{q}$ for a null mode. If we choose $f(\boldsymbol{x})$ to be a smooth scalar function that is positive over some set $U \subset \Omega \backslash \Gamma$ and zero elsewhere, then $\boldsymbol{w} = f\nabla p$ would produce

$$\boldsymbol{0} = \int_{\Omega\backslash\Gamma} f\|\nabla p\|^2 \, dV, \tag{6.44}$$

from which $\nabla p = \boldsymbol{0}$ in U, and necessarily, $\nabla p = \boldsymbol{0}$ in $\Omega\backslash\Gamma$. Thus, p is piecewise constant, and

$$\boldsymbol{0} = \int_{\Gamma} [p\boldsymbol{w}] \cdot \boldsymbol{n} \, dA + \int_{\Gamma} [\boldsymbol{w}] \cdot \boldsymbol{q} \, dA \tag{6.45}$$

If f is positive over some $U \subset \Omega$ but zero elsewhere, where $U \cap \Gamma \neq \emptyset$, then $\boldsymbol{w} = f\nabla\phi$, where ϕ is the level set, produces

$$\boldsymbol{0} = \int_{\Gamma} [p] f \nabla\phi \cdot \boldsymbol{n} \, dA = \int_{\Gamma} [p] f \, dA \tag{6.46}$$

From this it follows that $[p] = 0$ in U and thus $[p] = 0$ over Γ. Finally,

$$\boldsymbol{0} = \int_{\Gamma} [\boldsymbol{w}] \cdot (p\boldsymbol{n} + \boldsymbol{q}) \, dA \tag{6.47}$$

By defining $\boldsymbol{w}$ to be f times an arbitrary piecewise constant vector, we are forced to conclude that $\boldsymbol{q} = -p\boldsymbol{n}$, where p is the constant pressure. Thus, the only possible nullspace is $\boldsymbol{u} = \boldsymbol{0}$, $p = c$, and $\boldsymbol{q} = -c\boldsymbol{n}$, where c is a constant.

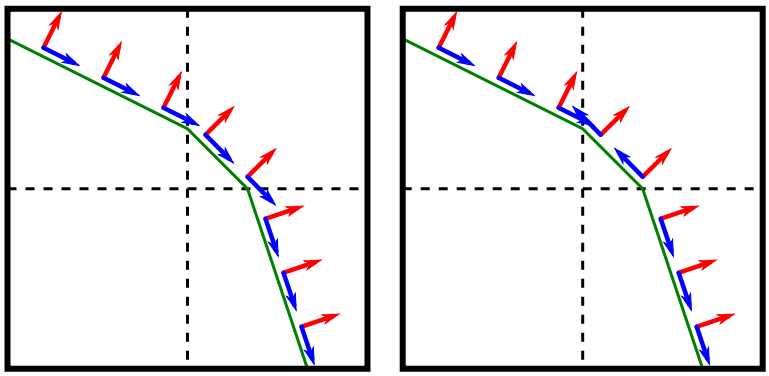
6.2.4.4 Discretizing interface stress jump

In Section 6.2.4.2, we introduced our discretization for all quantities except $\boldsymbol{q}$, whose description we left at (6.24). We will take up this topic here.

If we return to the equation for the constant pressure null mode $\boldsymbol{u} = \boldsymbol{0}$, $p = c$, $\boldsymbol{q} = -c\boldsymbol{n}$, we quickly run into a problem in 3D. In 2D, we can cut the MAC grid cells in a manner yielding one surface element per cut cell (ignoring under-resolved cases where there may be a second element). Such a procedure was employed in [2], and for each cut cell the normal of that element was chosen as $\boldsymbol{n}$. In 3D, it is generally impossible to maintain one surface element per cut cell, so for many cells we do not have an obvious candidate $\boldsymbol{n}$.

If we substitute the null mode into (6.34), we get

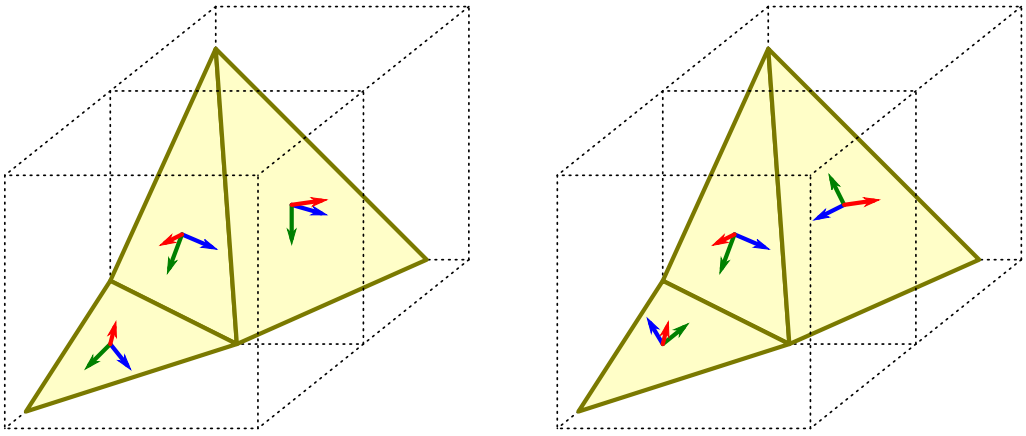
$$-\int_{\Omega\backslash\Gamma} p\nabla \cdot \boldsymbol{w} \, dV + \int_{\Gamma} [\boldsymbol{w}] \cdot \boldsymbol{q} \, dA = 0 \tag{6.48}$$



(a) $\boldsymbol{Q}_n(\boldsymbol{x})$ and $\boldsymbol{Q}_t(\boldsymbol{x})$ for one MAC cell.

(b) $\boldsymbol{Q}_t(\boldsymbol{x})$ assigned inconsistently.

Figure 6.3: MAC cell with doubly-fine subcells and interface elements. $\boldsymbol{Q}_n(\boldsymbol{x})$ and $\boldsymbol{Q}_t(\boldsymbol{x})$ are constant per interface element (and doubly-fine subcell) but not per MAC cell. In 2D, cutting on a doubly-fine grid may produce multiple segments, which must be oriented consistently.



(a) $\boldsymbol{Q}_n(\boldsymbol{x})$, $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ for part of a MAC cell.

(b) $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ oriented inconsistently.

Figure 6.4: $\boldsymbol{Q}_n(\boldsymbol{x})$, $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ are constant per interface element and should be consistent within a MAC cell (left). Consistency in the normal direction is automatic, but if care is not taken, the tangential directions will be inconsistent.

Applying the divergence theorem and using $p = c$ yields

$$\int_\Gamma [\boldsymbol{w}] \cdot (c\boldsymbol{n} + \boldsymbol{q})\, dA = 0 \tag{6.49}$$

In [2], $\boldsymbol{n}$ was constant per cut MAC cell, so discretizing $\boldsymbol{q}$ as constant per surface element (of which they had one per cut MAC cell) allowed them to discretely capture the pressure nullspace. If $\boldsymbol{w}$ were defined as piecewise constant per MAC cell, then discretizing $\boldsymbol{q}$ as constant per cell would lead to $\boldsymbol{q}$ being defined over each MAC cell C_i as

$$\boldsymbol{q}|_{C_i} = -c\left(\int_{\Gamma \cap C_i} dA\right)^{-1} \int_{\Gamma \cap C_i} \boldsymbol{n}\, dA \tag{6.50}$$

as the discrete nullspace. This is not the case, however, since $\boldsymbol{w}$ is discretized with the non-constant bases $N^x(\boldsymbol{x})$, $N^y(\boldsymbol{x})$, and $N^z(\boldsymbol{x})$.

Our solution to this problem is to define a normal component q_n and two tangential components q_{t0} and q_{t1} for each cut MAC cell. Then, the basis $\boldsymbol{Q}_n(\boldsymbol{x})$ for the normal component q_n is the local normal direction $\boldsymbol{Q}_n(\boldsymbol{x}) = \boldsymbol{n}(\boldsymbol{x})$, which will be different for every surface element in the MAC cell. Now, the nullspace will be captured discretely as $q_n = -c$ and $q_{t0} = q_{t1} = 0$. The tangential bases $\boldsymbol{Q}_{t0}(\boldsymbol{x})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ should be orthogonal tangential directions local to each element.

Unlike $\boldsymbol{Q}_n(\boldsymbol{x})$, which will automatically be consistent across elements in a MAC cell, the directions $\boldsymbol{Q}_{t0}(\boldsymbol{x})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$, if not chosen carefully, could vary wildly in 3D. (In 2D, the tangential component can also be chosen consistently, though in 2D only one element is required anyway.) To see why such inconsistency may be problematic, consider a MAC cell cut by two coplanar surface elements e_0, e_1 of equal area. Let $\boldsymbol{Q}_{t0}(\boldsymbol{x}) = -\boldsymbol{Q}_{t0}(\boldsymbol{y})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x}) = -\boldsymbol{Q}_{t1}(\boldsymbol{y})$, where $\boldsymbol{x} \in e_0$ and $\boldsymbol{y} \in e_1$. Such a configuration would be incapable of applying a tangential tractive component in the MAC cell, since the tangential contribution from q_{t0} and q_{t1} to one element would cancel out their contributions to the other element.

To prevent tangential inconsistencies, we define a reference orientation for the MAC cell. The normal direction for this orientation is the weighted normal,

$$\overline{\boldsymbol{n}}' = \int_\Gamma \boldsymbol{n}\, dA \qquad \overline{\boldsymbol{n}} = \frac{\overline{\boldsymbol{n}}'}{\|\overline{\boldsymbol{n}}'\|}, \tag{6.51}$$

where $A > 0$ is the area and $\overline{\boldsymbol{n}}$ is the unit direction. The first reference tangential direction $\bar{\boldsymbol{t}}_0$ is chosen as an arbitrary vector orthogonal to $\overline{\boldsymbol{n}}$, and $\bar{\boldsymbol{t}}_1 = \bar{\boldsymbol{t}}_0 \times \overline{\boldsymbol{n}}$, which we write as $\overline{\boldsymbol{R}} = \begin{pmatrix} \bar{\boldsymbol{t}}_1 & \bar{\boldsymbol{t}}_0 & \overline{\boldsymbol{n}} \end{pmatrix}$. To construct the local orientation for an element e, we begin by mapping the element's normal $\boldsymbol{n}_e$ into tho reference frame as $\hat{\boldsymbol{n}}_e = \overline{\boldsymbol{R}}^T \boldsymbol{n}_e$. Note that if the adjacent elements are similar in orientation, then $\boldsymbol{n}_e \approx \overline{\boldsymbol{n}}$ and $\hat{\boldsymbol{n}}_e \approx \boldsymbol{k}$, where $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ are the axial unit vectors. Similarly, we should have $\hat{\boldsymbol{t}}_{0e} \approx \boldsymbol{j}$ and $\hat{\boldsymbol{t}}_{1e} \approx \boldsymbol{i}$. This suggests choosing

$$\hat{\boldsymbol{t}}'_{0e} = (\boldsymbol{I} - \hat{\boldsymbol{n}}_e \hat{\boldsymbol{n}}_e^T)\boldsymbol{j} \qquad \hat{\boldsymbol{t}}_{0e} = \frac{\hat{\boldsymbol{t}}'_{0e}}{\|\hat{\boldsymbol{t}}'_{0e}\|} \qquad \hat{\boldsymbol{t}}_{1e} = \hat{\boldsymbol{t}}_{0e} \times \hat{\boldsymbol{n}}_e \qquad \boldsymbol{t}_{0e} = \overline{\boldsymbol{R}} \hat{\boldsymbol{t}}_{1e} \qquad \boldsymbol{t}_{1e} = \overline{\boldsymbol{R}} \hat{\boldsymbol{t}}_{0e}. \tag{6.52}$$

We found this local definition of the tangential directions to work well in practice. We can now define the bases for $\boldsymbol{q}$ locally as

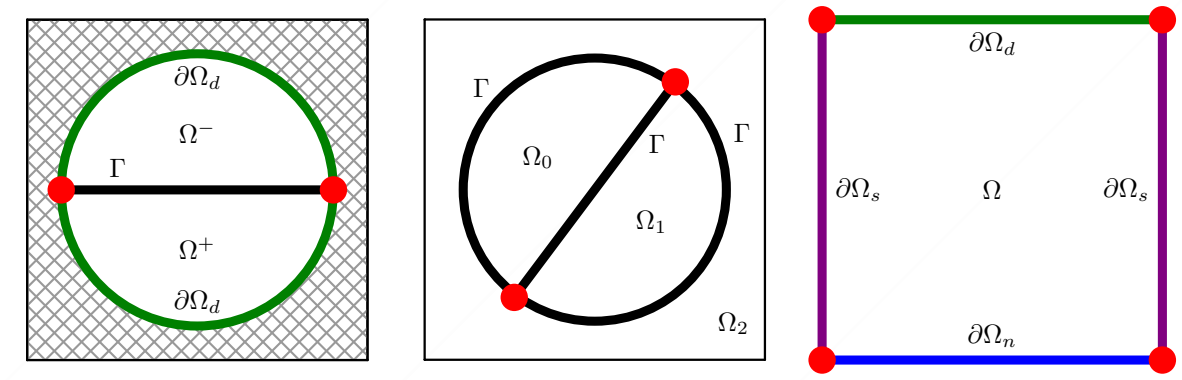
$$\boldsymbol{Q}_n(\boldsymbol{x}) = \boldsymbol{n}_e \qquad \boldsymbol{Q}_{t0}(\boldsymbol{x}) = \boldsymbol{t}_{0e} \qquad \boldsymbol{Q}_{t1}(\boldsymbol{x}) = \boldsymbol{t}_{1e}, \tag{6.53}$$

where e is the element at location $\boldsymbol{x}$. In 2D, the tangential direction is simply chosen by rotating the normal direction clockwise one-quarter turn. Note that unlike the bases for $\boldsymbol{u}$ or $\boldsymbol{p}$, the bases for $\boldsymbol{q}$ are vector quantities. For simplicity of exposition, we index the q_i degrees of freedom uniformly, ignoring the distinction between normal and tangential degrees of freedom. These consistency concerns are illustrated in Figures 6.3 and 6.4.

6.2.4.5 Boundary Conditions

Up to this point, we have described how to discretize the interface Γ splitting the domain Ω, but have not treated boundary conditions on Ω, excepting periodic boundary conditions which can be handled in the obvious way. An advantage of our discretization of the embedded interface is the relative ease with which we can modify it to implement Dirichlet velocity boundary conditions (6.5), Neumann boundary conditions (6.6), and slipping boundary conditions (6.7) and (6.8).

We represent our boundary conditions by treating the regions beyond Dirichlet, Neumann, and slip boundaries as special fluid phases. These boundary condition phases have level sets associated with them representing the location of the boundary. The boundary condition phases are not real fluid. There are no velocity or pressure degrees of freedom associated with these



(a) Interface intersects boundary

(b) Three interfaces meet

(c) Different boundary conditions meet

Figure 6.5: Triple junctions are formed when interfaces or boundary conditions of different types meet at a point. The filled circles represent the location of triple junction in these examples.

phases. This representation allows us to reuse our interface routines with little extra work. The problem of handling the different boundary conditions becomes a problem of handling an interface between a fluid region and boundary condition region. Note that if two boundary condition regions are adjacent, there will be an interface between them, which we can ignore. There will also be a triple junction at the point where the fluid region meets the two boundary condition regions. Some common triple junction configurations are shown in Figure 6.5. We leave the problem of handling triple junctions for future work.

Dirichlet boundary conditions are implemented by treating the region beyond the boundary as having an identically zero velocity. Nonzero Dirichlet velocity boundary conditions ($\boldsymbol{u} = \boldsymbol{b}$ at $\boldsymbol{x} \in \partial\Omega_d$) are treated as velocity jumps at the interface ($[\boldsymbol{u}] = \boldsymbol{b}$ at $\boldsymbol{x} \in \Gamma_d$). The stress in the region beyond the boundary condition can be taken to be continuous with the stress in the fluid, so ($[\sigma \cdot \boldsymbol{n}] = \boldsymbol{0}$ at $\boldsymbol{x} \in \Gamma_d$). There will be $\boldsymbol{q}$ degrees of freedom for Dirichlet boundary conditions just as there are for regular interfaces. Practically speaking this amounts to omitting the velocity degrees of freedom (as well as the associated rows and columns of the system) corresponding to the dirichlet fluid phase.

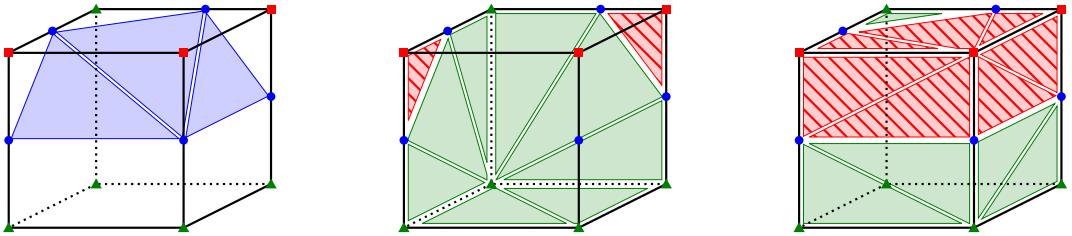
In the Neumann case, we wish to enforce a desired normal stress. We treat the region beyond the boundary as having identically zero stress. In (6.33), we divide the interface stress jump evenly to both sides of the interface ($\Phi_i = \frac{1}{2}$). In the Neumann case, we must put the entire contribution on the side corresponding to the fluid ($\Phi_i = 1$) since the other rows will be discarded. The interface stress now corresponds to the region beyond the boundary, so $\boldsymbol{q} = \overline{\sigma} \cdot \boldsymbol{n} = \boldsymbol{0}$. Eliminating $\boldsymbol{q}$ in this way corresponds to not having any particular velocity jump to enforce ($\boldsymbol{a}$ will never be used). Practically speaking, Neumann boundary conditions are implemented by omitting $\boldsymbol{q}$ degrees of freedom corresponding to the Neumann boundary condition and omitting the corresponding entries of the system. Note that the Dirichlet and Neumann treatments above are equivalent to the standard finite element treatments of these boundary conditions.

Our treatment of the slip boundary condition takes advantage of our division of $\boldsymbol{q}$ degrees of freedom into normal and tangential components. Slip is treated like Dirichlet in the normal direction and Neumann in the tangential directions. The tangential $\boldsymbol{q}$ degrees of freedom, as well as corresponding matrix entries, are omitted. Note that the equation corresponding to the normal component of $\boldsymbol{q}$ enforces the velocity jump condition in the normal direction ($[\boldsymbol{u}] = \boldsymbol{c}$), so omitting degrees of freedom in this way suffices to encode Dirichlet in the normal direction. The tangential portion requires a slight modification, since $(\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n}^T)\hat{\boldsymbol{h}}$ must be used as the interface stress. As in the Neumann case, $\Phi_i = 1$ is used in (6.33). We demonstrate all three types of boundary conditions in our numerical examples.

Since our implementations of Dirichlet and slip boundary conditions do not eliminate the normal components of $\boldsymbol{q}$ degrees of freedom, a Dirichlet or slip boundary condition will not preclude the presence of a null mode. However, a Neumann boundary condition will prevent the existence of a null mode since its $\boldsymbol{q}$ degrees of freedom are removed.

6.2.4.6 Practical implementation

The primary difficulty in implementing the proposed method is computing the necessary integrals. Our pressure basis functions are piecewise constant over MAC cells, but our velocity



(a) Interface elements (b) Boundary elements (back) (c) Boundary elements (front)

Figure 6.6: We use a modified marching cubes table that emits both the usual interface elements (left) as well as a triangulation of the portion of boundary of the cell in each region (center and right). These extra triangles greatly simplify our integration process.

basis functions are piecewise trilinear over cells whose corners contain the respective velocity degrees of freedom. Since we will be integrating products of these bases and their derivatives, we perform our integration over the cells of a doubly-fine grid. Over each doubly-fine cell, these products are all polynomials. The polynomials being integrated may be different, even discontinuous, across the boundaries between adjacent doubly-fine cells (even across those contained in the same MAC cell). This is a consequence of the staggering of the variables.

We represent our regions using level sets (both for the interface and boundary conditions) stored at MAC cell centers. Since we wish to integrate over doubly-fine cells, we interpolate our level set to populate a doubly-fine node-centered level set. This representation allows us to compute our interface geometry using marching cubes in 3D (marching squares in 2D) over the doubly-fine grid. The boundary integrals amount to integrating a polynomial over these triangles. Note that all of the bases, restricted to one interface element, are polynomials.

The volumetric integrals at first seem rather difficult, particularly in light of the rather complicated regions that occur with marching cubes. If approached in the right way, however, they are quite manageable. We begin by converting the volume integral into a surface integral using the divergence theorem as in [2]. This reduces the problem into one of integrating polynomials

of one degree higher over the triangles on the boundary of the cut marching cubes volumes. We augment our marching cubes table (marching squares table for 2D) to emit these triangles (segments in 2D) on the surface of the cube in addition to the triangles on the interface itself. See Figure 6.6 for an illustration. This enhancement of marching cubes is straightforward in practice, as most of the work involved is required to implement marching cubes in the first place. It also greatly simplifies the integration.

The highest degree polynomials we must integrate over cut volumes are of degree six in 3D (degree four in 2D), which occur for $\boldsymbol{M}^x$, $\boldsymbol{M}^y$, and $\boldsymbol{M}^z$. These become degree seven polynomials in 3D (degree five in 2D) once the divergence theorem is applied. The highest degree polynomials we integrate for boundary integrals is three in 3D (two in 2D). We perform all of these integrations using quadrature rules of high enough order (listed in [67]) to get the integral exactly.

Although solving the linear system 6.29 is by far the slowest step of our method, we still perform a few simple optimizations when computing the integrals. The first is to precompute the stencils for uncut doubly-fine cells (there will be eight such integrals required, since each octant of a MAC cell may contribute differently to the final stencil). Additional integrations are only required for integrating in cut cells or computing boundary integrals. Most cells are not cut and can simply use a copy of one of these precomputed stencils. For the cells that are cut, we will be computing many integrals over the same geometry, so we begin by integrating the monomials individually using quadrature rules. With these, integrating the actual basis polynomials reduces to a simple dot product.

We can also take advantage of the way in which the volume integral was converted into a surface integral using divergence theorem to save even more work. That is we can convert a volume integral over $f(x,y,z)$ into a surface integral using

$$\int_\Omega f\,dV = \int_{\partial\Omega} g\,\boldsymbol{n}_x\,dA = \int_{\partial\Omega} h\,\boldsymbol{n}_y\,dA = \int_{\partial\Omega} k\,\boldsymbol{n}_z\,dA$$

where we have integrated the polynomial $f(x,y,z)$ to obtain

$$g = \int f\,dx \qquad h = \int f\,dy \qquad k = \int f\,dz.$$

If we choose the x direction as our preferred direction, then $\boldsymbol{n}_x = 1$ along two faces of the cube, and $\boldsymbol{n}_x = 0$ along the other four. This means we can discard the boundary elements along four of the faces of the cube.

Finally, we only need to compute volume integrals on one side of an interface, since the integrals for the other can be obtained by subtracting from the integral over the whole cube, which we have precomputed.

6.2.5 Solving the system

We solve our system using preconditioned MINRES using the same Jacobi-style preconditioner as in [2]. We project out our nullspace (when we have one) inside the MINRES solver in addition to projecting the right hand side for compatibility, since we have found this to improve the convergence behavior of the solver. This simple preconditioner we employ significantly improves the conditioning of our systems, but in practice the systems remain very slow to solve. We leave the problem of finding a more effective preconditioner for future work.

6.2.6 Surface tension

There are many popular options for introducing surface tension into a fluid discretization that are available to us. Since our discretization has provisions built in for incorporating an interface force $\hat{\boldsymbol{f}}$, we take this approach. We begin by computing normals and curvature at MAC cell centers

$$\boldsymbol{n}_i = \frac{\nabla \phi^{n+1}}{\|\nabla \phi^{n+1}\|} \qquad \boldsymbol{H}_i = \text{Hessian}(\phi^{n+1}) \qquad \kappa_i = \frac{\boldsymbol{n}_i^T \boldsymbol{H}_i \boldsymbol{n}_i - \text{tr}(\boldsymbol{H}_i)}{\|\nabla \phi^{n+1}\|},$$

where all derivatives are computed using central differencing. Using these, we can compute $\hat{\boldsymbol{n}}$ and $\hat{\kappa}$ estimates wherever we need them by interpolating $\boldsymbol{n}_i$ and κ_i using cubic interpolation. Finally, we approximate our surface tension as the interface force

$$\hat{\boldsymbol{f}} = -\beta \hat{\kappa} \hat{\boldsymbol{n}}.$$

Note that $\boldsymbol{n}_i$ and κ_i are only required near the interface, and reinitialization must be performed in a wide enough band for the combined central differencing stencil and cubic interpolation

stencil.

6.2.7 Stability

6.2.7.1 System stability

The final step of our scheme ($\boldsymbol{u}^\star \to \boldsymbol{u}^{n+1}$) applies viscosity and enforces incompressibility. This operation is linear (or affine if there are forcing terms such as inhomogeneous boundary conditions or surface tension). This system can be expressed as

$$\begin{pmatrix} \boldsymbol{M}+\boldsymbol{S} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix},$$

where $\boldsymbol{M}$ contains the inertial blocks, $\boldsymbol{S}$ contains the viscous blocks, and $\boldsymbol{G}$ contains the pressure and interface stress blocks. The vector λ contains the $\boldsymbol{p}$ and $\boldsymbol{q}$ degrees of freedom. Non-homogeneous terms on the right hand side are omitted. Note that the λ degrees of freedom are not state, in that these values can be discarded at the end of the time step. The only state variables present in this system are velocities.

The matrix $\boldsymbol{S}$ made up of the viscous blocks is symmetric positive semi-definite, since from our discretization $\boldsymbol{w}'\boldsymbol{S}\boldsymbol{u}$ is equal to the inner product $\int_\Omega \frac{\mu}{2}\nabla(\boldsymbol{w}+\boldsymbol{w}^T)\cdot\nabla(\boldsymbol{u}+\boldsymbol{u}^T)\,dV$ of the piecewise trilinear functions corresponding to the vectors $\boldsymbol{u},\boldsymbol{w}$. We will substitute $\boldsymbol{S}=\boldsymbol{C}^T\boldsymbol{C}$ for the purposes of this analysis, and rewrite our matrix equation as

$$\begin{pmatrix} \boldsymbol{M}+\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix}.$$

Following [82] and letting $\boldsymbol{w} = \boldsymbol{C}\boldsymbol{v}^{n+1}$, we can transform this system to

$$\begin{pmatrix} \boldsymbol{C}\boldsymbol{M}^{-1} & \boldsymbol{0} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1} & -\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}+\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{M}^{-1} & \boldsymbol{0} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1} & -\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{C}+\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{G} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{u}^\star \\ \boldsymbol{G}^T\boldsymbol{u}^\star \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{I}+\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T & \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{G} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{C}^T & \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{u}^\star \\ \boldsymbol{G}^T\boldsymbol{u}^\star \end{pmatrix}$$

$$\left(\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{G}^T \end{pmatrix} \boldsymbol{M}^{-1} \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{G}^T \end{pmatrix}^T \right) \begin{pmatrix} \boldsymbol{w} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{G}^T \end{pmatrix} \boldsymbol{u}^\star$$

$$(\boldsymbol{P}+\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T)\boldsymbol{z} = \boldsymbol{K}\boldsymbol{u}^\star,$$

where

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \qquad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{G}^T \end{pmatrix} \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{w} \\ \lambda \end{pmatrix}.$$

Since both $\boldsymbol{P}$ and $\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T$ are symmetric positive semi-definite, $\boldsymbol{P}+\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T$ may have the nullspace component $\boldsymbol{z}$ if and only if both PP and $\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T$ do individually. $\boldsymbol{z}^T\boldsymbol{P}\boldsymbol{z} = \boldsymbol{0}$ implies $\boldsymbol{u}^{n+1} = \boldsymbol{0}$, which reduces $\boldsymbol{z}^T\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T\boldsymbol{z} = \lambda^T\boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{G}\lambda$. Since $\boldsymbol{M}$ is symmetric positive definite, we must have $\boldsymbol{G}\lambda = \boldsymbol{0}$. That is, $\boldsymbol{G}$ has a nullspace. We often do have such a nullspace. Though this complicates the analysis, we note that we can ignore this nullspace since we will never get a component in it on the right hand side. In this way, we can solve this system for $\boldsymbol{z}$.

The next step is to recover $\boldsymbol{u}^{n+1}$ from $\boldsymbol{z}$, which we do using the momentum equation

$$(\boldsymbol{M}+\boldsymbol{C}^T\boldsymbol{C})\boldsymbol{u}^{n+1} + \boldsymbol{G}\lambda = \boldsymbol{M}\boldsymbol{u}^\star$$

$$\boldsymbol{u}^{n+1} + \boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{w} + \boldsymbol{M}^{-1}\boldsymbol{G}\lambda = \boldsymbol{u}^\star$$

$$\boldsymbol{u}^{n+1} + \boldsymbol{M}^{-1}\boldsymbol{K}^T\boldsymbol{z} = \boldsymbol{u}^\star$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^\star - \boldsymbol{M}^{-1}\boldsymbol{K}^T\boldsymbol{z}.$$

Finally, the change in kinetic energy due to the update $\boldsymbol{u}^\star \to \boldsymbol{u}^{n+1}$ is

$$\begin{aligned}
\Delta KE &= \frac{1}{2}(\boldsymbol{u}^{n+1})^T \boldsymbol{M}\boldsymbol{u}^{n+1} - \frac{1}{2}\boldsymbol{u}^{\star T}\boldsymbol{M}\boldsymbol{u}^\star \\
&= \frac{1}{2}(\boldsymbol{u}^\star - \boldsymbol{M}^{-1}\boldsymbol{K}^T z)^T \boldsymbol{M}(\boldsymbol{u}^\star - \boldsymbol{M}^{-1}\boldsymbol{K}^T z) - \frac{1}{2}\boldsymbol{u}^{\star T}\boldsymbol{M}\boldsymbol{u}^\star \\
&= \frac{1}{2} z^T \boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T z - z^T \boldsymbol{K}\boldsymbol{u}^\star \\
&= \frac{1}{2} z^T \boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T z - z^T (\boldsymbol{P} + \boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T) z \\
&= -\frac{1}{2} z^T \boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{K}^T z - z^T \boldsymbol{P} z \\
&\leq 0.
\end{aligned}$$

Thus we see that this step does not introduce energy into the system.

6.2.7.2 Time step restriction

Empirically, our method appears to have a stability restriction on the value of the dimensionless quantity $\frac{\Delta t \mu}{\rho \Delta x^2}$ (see Sections 6.3.12.1 and 6.3.12.2). This limits the minimum choice for Δt. Since the criterion depends on refinement as $\frac{\Delta t}{\Delta x^2}$, convergence is possible as long as Δt is refined no faster than Δx^2. In particular, $\Delta t = k\Delta x$ and $\Delta t = k\Delta x^2$ are both suitable refinement strategies.

It is worth discussing the apparent source of this instability in more detail. In the absence of an interface, no instability is observed. When an interface (or boundary condition) is present and instability is observed, it starts near the interface. This in particular suggests that the modified velocity advection scheme proposed is stable. Indeed, the instability is also observed with the original advection scheme or no advection at all. Similarly, instability is observed with BDF or backward Euler.

An unusual characteristic of this stability restriction is that Δt must not be chosen *too small*. To see what may be causing this, consider a time step in the limit $\Delta t \to 0$. In this case, advection has no effect, and the viscosity terms vanish. Using backward Euler eliminates the complications of BDF. The only part of the time integration remaining that has an appreciable effect is setting up the right hand side and solving the system, which we showed will not increase energy. The source of the energy increase is the velocity extrapolation used to fill the ghost cells of $\boldsymbol{u}^\star$ needed for the right hand side. If sufficient viscosity is present, this added energy is dissipated

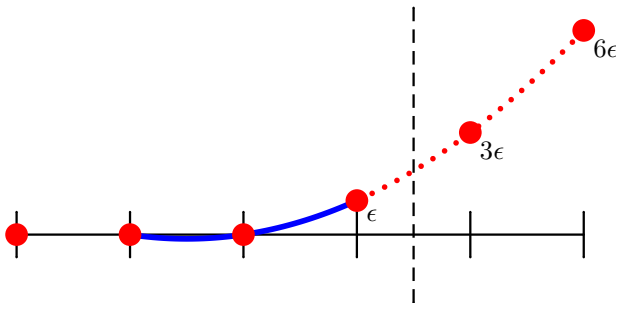


Figure 6.7: Extrapolation amplifies errors. In this case, the ideal solution (black horizontal line) is perturbed by ε at the interface, which is then amplified by quadratic extrapolation to 3ε and 6ε in the ghost region.

as it is introduced, and the scheme remains stable. Examining the role of Δt, μ, and ρ in this system, we can rescale the system so that the only reference to these quantities is through the expression $\frac{\Delta t \mu}{\rho}$. This is consistent with the empirical stability criterion suggested by our numerical examples. Noting that the viscosity blocks are scaled by $\frac{1}{dx^2}$ relative to the inertial blocks leads to the full criterion $\frac{\Delta t \mu}{\rho \Delta x^2}$. This value describes the efficiency with which viscosity is able to damp out energy in our scheme.

To see why extrapolation is able to lead to instability, consider a set of uniformly spaced sample points $u_1, u_2, \ldots$. The value u_0 is to be computed by extrapolation. If $u_k = 1$ for $k \geq 2$ are set to a constant value but $u_1 = 1 + \varepsilon$, so that a small error has been made near the interface, then we will compute $u_0 = 1 + \varepsilon$ with constant extrapolation, $u_0 = 1 + 2\varepsilon$ with linear extrapolation, and $u_0 = 1 + 3\varepsilon$ with quadratic extrapolation (see Figure 6.7). Thus, we see that extrapolation has magnified the error by a factor greater than one. Solving the system pulls some of this energy from the ghost region inside, where it is magnified further by extrapolation in the next time step. The above example has a growth factor of 3, though in practice a value near 1.25 is observed for unstable simulations. Instabilities always exhibit this slow and steady exponential march to infinity. Using lower order extrapolation decreases the growth rate, but even for constant extrapolation the factor is still slightly larger than one. This supports the idea that extrapolation is providing the amplification required for instability.

6.3 Numerical examples

Our method supports a range of boundary conditions and forces. Through a mixture of analytic and more practical tests, we demonstrate second order accuracy for $\boldsymbol{u}$ in L^∞ and L^2, second order accuracy for p in L^2, and first order accuracy for p in L^∞. We also investigate the stability characteristics of our method.

6.3.1 Taylor-Green vortex

The Taylor-Green vortex is a popular analytic accuracy test for single-phase Navier-Stokes. We use a (dimensionless) domain $[0,\pi] \times [0,\pi]$ in which we confine fluid to the region $sin(x)sin(y) \geq k$, where $k = 0.2$. The fluid has $\rho = 1$, $\mu = 1$, and the final time is $T = 0.2$. The analytic solution is

$$u = \sin(x)\cos(y) \qquad v = -\cos(x)\sin(y) \qquad p = \frac{1}{4}\rho(\cos(2x)+\cos(2y))e^{-4\nu t},$$

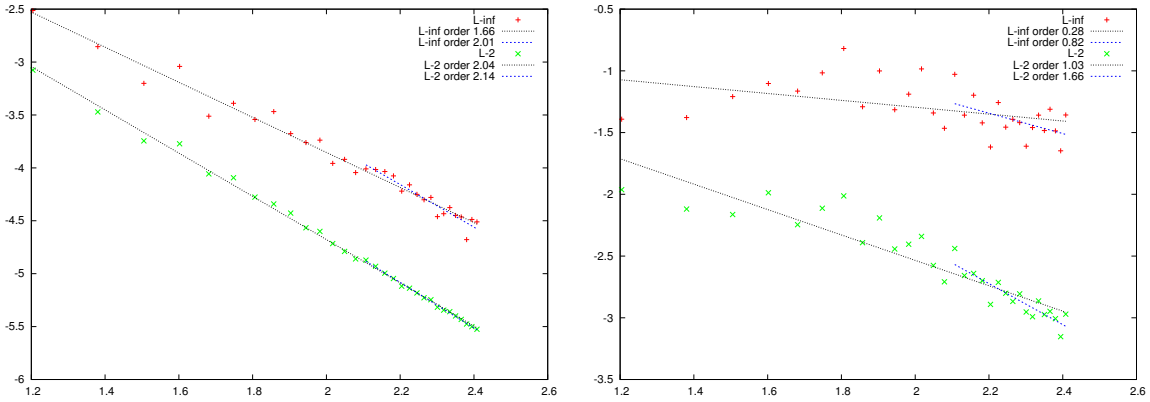
where $\nu = \frac{\mu}{\rho}$. The velocity field is initialized with the analytic velocity. Velocity and pressure errors along with convergence order estimates are shown in Figure 6.8.

6.3.2 Translating Taylor-Green vortex

We test our method on a problem where two fluids are separated by an interface in the periodic domain $[0,2\pi] \times [0,2\pi]$. The interface is initially set to be the circle centered at $(\frac{11\pi}{10},0)$ with radius $\frac{3\pi}{5}$. Each fluid has $\rho = 1$ and $\mu = 2$, and the analytic solution for both fluids is given by a translating Taylor-Green vortex:

$$u = \sin(x-.2t)\cos(y-.5t) \qquad v = -\cos(x-.2t)\sin(y-.5t)$$
$$p = \frac{1}{4}\rho(\cos(2x-.2t)+\cos(2y-.5t))e^{-4\nu t},$$

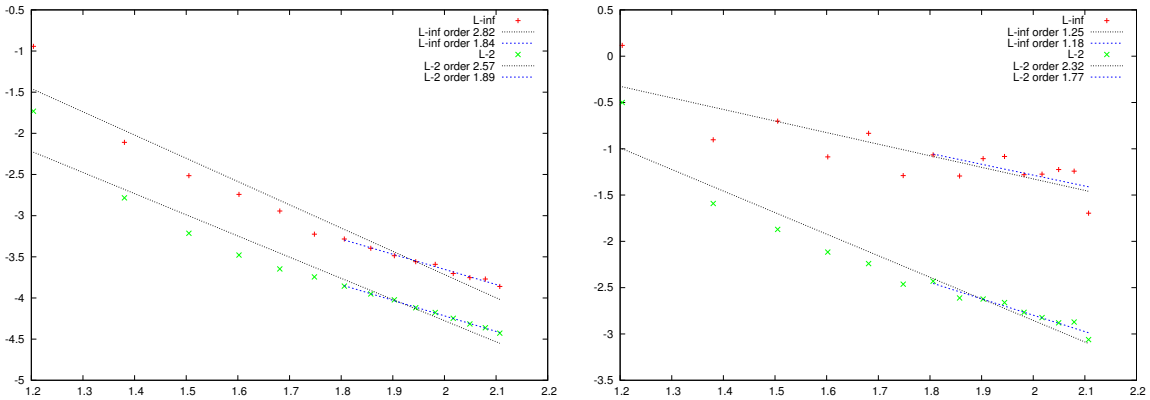
where $\nu = \frac{\mu}{\rho}$. As before, the velocity field is initialized with the analytic velocity. Velocity and pressure errors, and estimates of the convergence order, are shown in Figure 6.9.



(a) Velocity error

(b) Pressure error

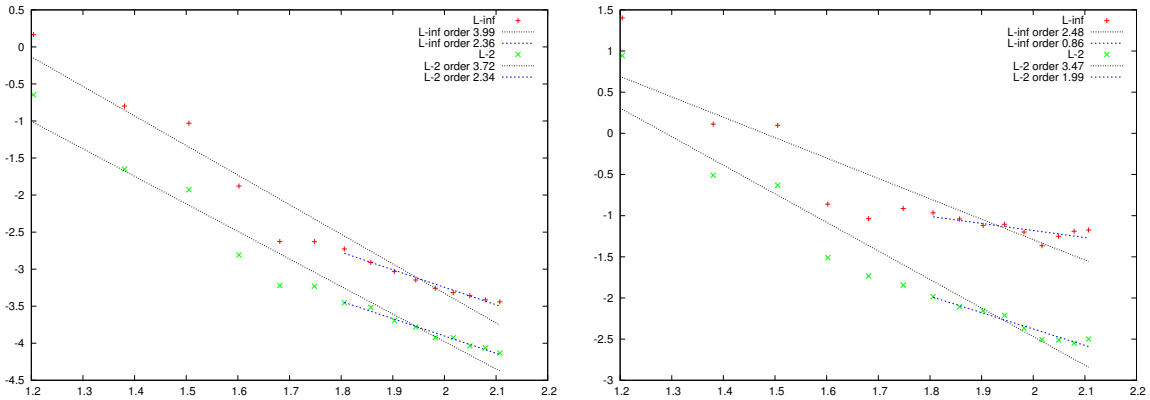
Figure 6.8: Errors in the Taylor-Green vortex, Example 6.3.1, for velocity and pressure (shown log base 10) in L^∞ and L^2, plotted against resolutions from 16 to 256 by increments of 8 (shown log base 10). The pressure does not start to display convergence until the resolution is high, so separate regressions are provided for the highest resolutions 128 to 256 to eliminate bias from resolutions below the convergence regime. The estimated orders for velocity when throwing out the lowest resolutions are 2.01 in L^∞ and 2.14 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.82 in L^∞, 1.66 in L^2.



(a) Velocity error

(b) Pressure error

Figure 6.9: Translating Taylor-Green vortex errors, Example 6.3.2, for velocity and pressure (shown log base 10) plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2, with separate regressions provided for the highest resolutions 64 to 128. The estimated orders for velocity when throwing out the lowest resolutions are 1.84 in L^∞ and 1.89 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 1.18 in L^∞, 1.77 in L^2.



(a) Velocity error

(b) Pressure error

Figure 6.10: Analytic test I errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 64 to 128 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 2.34 in L^∞ and 2.36 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.86 in L^∞, 1.99 in L^2.

6.3.3 Analytic test I

In this analytic test we evolve two fluids, separated by an interface, in the periodic domain $[-\pi,\pi]\times[-\pi,\pi]$. The interface in this example is the circle $x^2+y^2=(.8\pi)^2$. We set an inner boundary at the circle $(x-.2\pi)^2+y^2=(.2\pi)^2$, on which we apply a Neumann boundary condition. The fluid bounded by the inner and outer circles has $\rho^-=1$, $\mu^-=1$, and the outer fluid has $\rho^+=2$, $\mu^+=3$. The fluids are initialized with the analytic velocity and evolved to final time $T=.1$. The analytic solution is given by

$$u=\begin{cases} .2-x & \boldsymbol{x}\in\Omega^- \\ \sin(x)\cos(y) & \text{otherwise}\end{cases} \qquad v=\begin{cases} y & \boldsymbol{x}\in\Omega^- \\ -\cos(x)\sin(y) & \text{otherwise}\end{cases}$$

$$p=\begin{cases} 0 & \boldsymbol{x}\in\Omega^- \\ \frac{1}{4}\rho^+(\cos(2x)+\cos(2y))e^{-4\nu t} & \text{otherwise}\end{cases}.$$

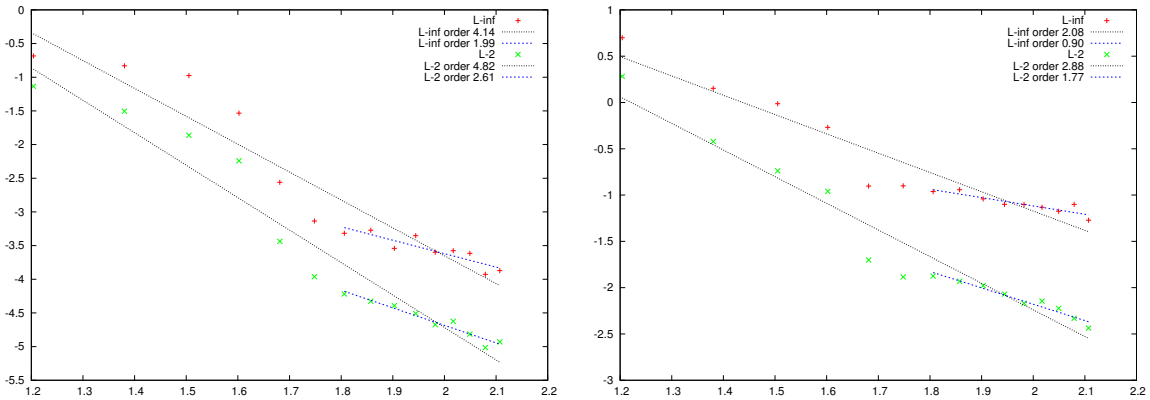
where $\nu=\frac{\mu}{\rho}$. Velocity and pressure errors along with convergence order estimates are shown in Figure 6.10.

6.3.4 Analytic test II

We embed the circle $x^2+y^2=(.8\pi)^2$ into the domain $[-\pi,\pi]\times[-\pi,\pi]$. A circle $(x-.2\pi)^2+y^2=(.2\pi)^2$ separates the larger circle into an inner domain Ω^- and an outer domain Ω^+, and a slip boundary condition is enforced along the boundary of the outer circle. The inner fluid has $\rho^-=1$, $\mu^-=1$ and the outer fluid has $\rho^+=2$, $\mu^+=3$. The velocity field is initialized with the analytic velocity and evolved to the final time $T=.1$. The analytic solution is given by

$$u=\begin{cases} .2-x & \boldsymbol{x}\in\Omega^- \\ -y & \text{otherwise}\end{cases} \qquad v=\begin{cases} y & \boldsymbol{x}\in\Omega^- \\ x & \text{otherwise}\end{cases} \qquad p=\begin{cases} 0 & \boldsymbol{x}\in\Omega^- \\ .5\rho^+(x^2+y^2) & \text{otherwise}\end{cases}.$$

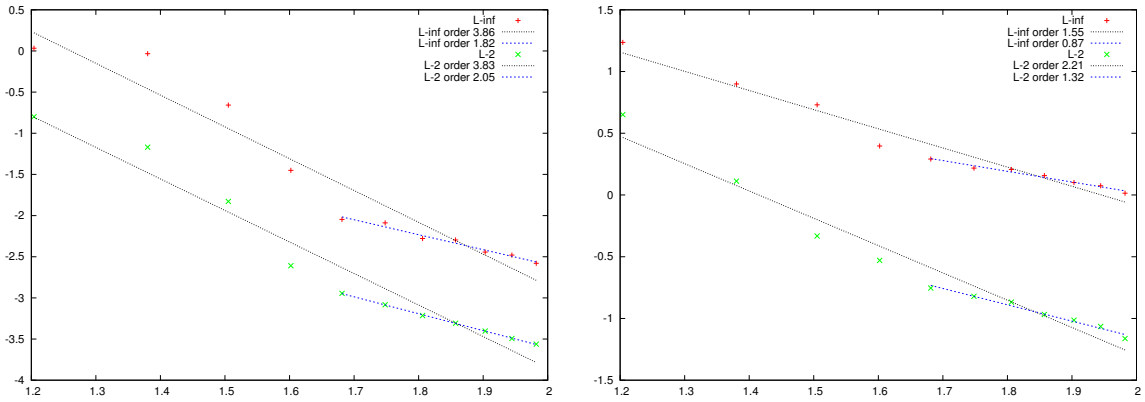
Velocity and pressure errors along with convergence order estimates are shown in Figure 6.11.



(a) Velocity error

(b) Pressure error

Figure 6.11: Analytic test II errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 64 to 128 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 1.99 in L^∞ and 2.61 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.90 in L^∞, 1.77 in L^2.



(a) Velocity error

(b) Pressure error

Figure 6.12: Analytic test II-3D errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 96 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 48 to 96 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 1.82 in L^∞ and 2.05 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.87 in L^∞, 1.32 in L^2.

6.3.5 Analytic test II-3D

We examine a three-dimensional analogue of our test from the previous section: The sphere $x^2+y^2+z^2 = (.8\pi)^2$ is embedded into the dimensionless domain $[-\pi,\pi]\times[-\pi,\pi]\times[-\pi,\pi]$. A shell $(x-.2\pi)^2+y^2+z^2 = (.2\pi)^2$ separates the larger sphere into an inner domain Ω^- and an outer domain Ω^+. As before, the slip boundary condition is enforced along the boundary of the outer circle. The inner fluid has $\rho^- = 1$, $\mu^- = 1$ and the outer fluid has $\rho^+ = 2$, $\mu^+ = 3$. As in previous examples, the velocity field is initialized with the analytic velocity and evolved to the final time $T = .1$. The analytic solution is given by

$$u = \begin{cases} .2 - x & \boldsymbol{x} \in \Omega^- \\ 2z - 3y & \text{otherwise} \end{cases} \qquad v = \begin{cases} y & \boldsymbol{x} \in \Omega^- \\ 3x - z & \text{otherwise} \end{cases}$$

$$w = \begin{cases} -2z & \boldsymbol{x} \in \Omega^- \\ y - 2x & \text{otherwise} \end{cases} \qquad p = \begin{cases} 0 & \boldsymbol{x} \in \Omega^- \\ .5\rho^+(x^2+y^2) & \text{otherwise} \end{cases}.$$

Velocity and pressure errors along with convergence order estimates are shown in Figure 6.12.

6.3.6 Two-phase Couette flow

We run a two-phase Couette flow test, where two phases are separated by a stationary interface. The phases have different density and viscosity. The domain is $[0,1] \times [0,1]$. The fluid is confined by vertical no-slip walls at $x_0 = 0.2$ (where $\boldsymbol{u}(x_0, y) = (0,1)$) and $x_2 = 0.8$ (where $\boldsymbol{u}(x_2, y) = (0,-1)$). Periodic boundary conditions are enforced at the top and bottom of the domain. The interface is vertical at $x_1 = 0.5$, with phase 0 ($\rho^- = 1$, $\mu^- = 1$) occupying $0.2 < x < 0.5$ and phase 1 ($\rho^+ = 2$, $\mu^+ = 3$) occupying $0.5 < x < 0.8$ The analytic solution is $u = 0$, $p = 0$, and

$$v_1 = \frac{v_0\mu^-(x_2 - x_1) + v_2\mu^+(x_1 - x_0)}{\mu^-(x_2 - x_1) + \mu^+(x_1 - x_0)}, \qquad v = \begin{cases} v_0 + \frac{x - x_0}{x_1 - x_0}(v_1 - v_0) & x \le x_1 \\ v_1 + \frac{x - x_1}{x_2 - x_1}(v_2 - v_1) & x > x_1 \end{cases}.$$

The initial velocity is the analytic solution. This test demonstrates that the method correctly (and sharply) handles discontinuities in viscosity. Convergence results are summarized in Figure 6.13.

6.3.7 Parasitic currents

In this test, we check for convergence of parasitic currents in the case of a stationary circle with surface tension. The fluid domain is $[0\,\text{m}, 0.01\,\text{m}] \times [0\,\text{m}, .01\,\text{m}]$, with periodic boundary conditions and an initially circular interface with radius $0.003\,\text{m}$ centered at $(0.005\,\text{m}, 0.005\,\text{m})$. We simulate glycerin inside the circle ($\rho^- = 1261\,\text{kg m}^{-2}$, $\mu^- = 1.4746\,\text{kg s}^{-1}$) and a generic light

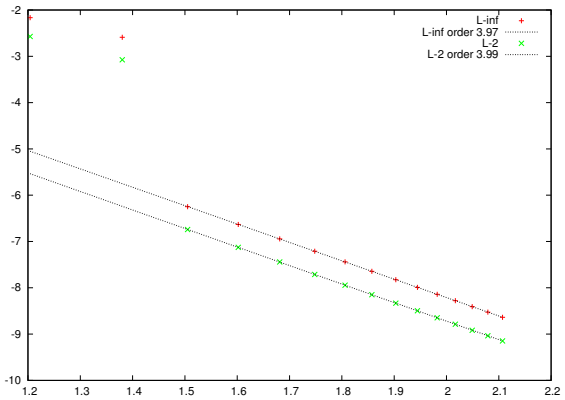
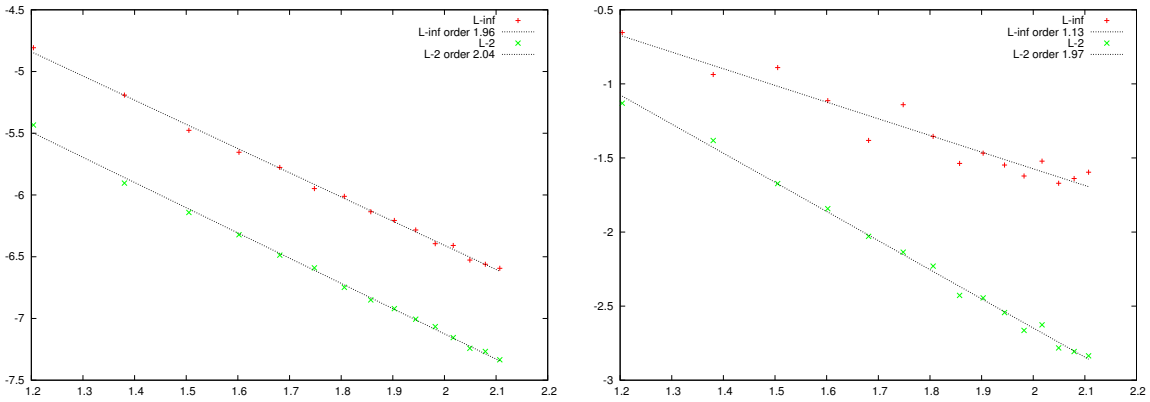


Figure 6.13: Couette flow errors in velocity (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. The analytic solution (piecewise linear velocity, constant zero pressure) would often be resolved exactly by a second order method. In our case, we do observe fourth order velocity convergence on this simple test. Note that the first two resolutions are too small to resolve the setup and have been omitted from the regression. The pressure errors are below 10^{-9} in L^∞ and L^2 for all resolutions and are limited by the convergence tolerance of our MINRES solver.



(a) Velocity error

(b) Pressure error

Figure 6.14: A stationary circle is run with surface tension to test convergence of parasitic currents. Errors in velocity and pressure (y-axis, shown log base 10) are plotted against resolutions from 16 to 256 by increments of 8 (x-axis, shown log base 10). We estimate the velocity to be order 1.96 in L^∞ and 2.04 in L^2. For pressure, we obtained 1.13 in L^∞ and 1.97 in L^2.

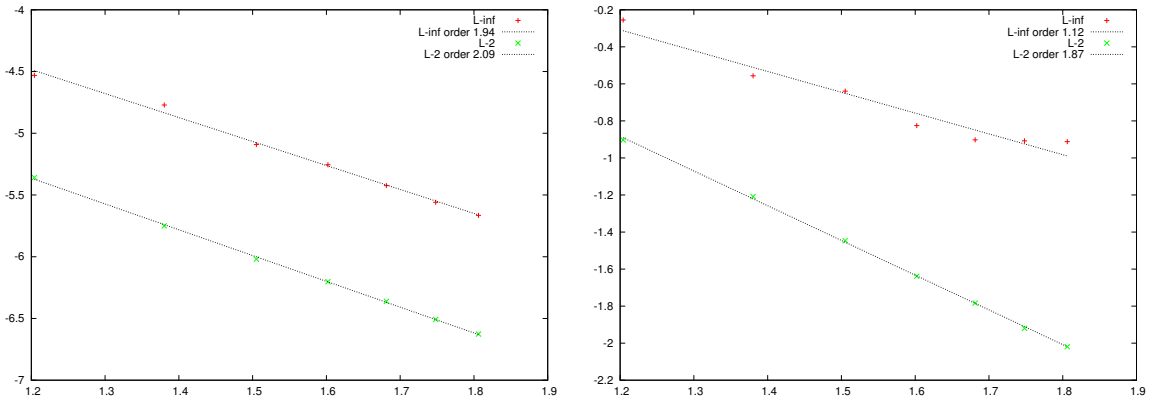
fluid outside ($\rho^+ = 1\,\mathrm{kg\ m^{-2}}$, $\mu^+ = 1\,\mathrm{kg\ s^{-1}}$, similar in density to air but more viscous). The interface is evolved with the level set method. Convergence results are shown in Figure 6.14.

6.3.8 Parasitic currents - 3D

This test is a 3D analogue of Section 6.3.7. The fluid domain is $[0\,\mathrm{m}, 0.01\,\mathrm{m}] \times [0\,\mathrm{m}, .01\,\mathrm{m}] \times [0\,\mathrm{m}, .01\,\mathrm{m}]$, with periodic boundary conditions and an initially spherical interface with radius $0.003\,\mathrm{m}$ centered at $(0.005\,\mathrm{m}, 0.005\,\mathrm{m}, 0.005\,\mathrm{m})$. Glycerin is inside ($\rho^- = 1261\,\mathrm{kg\ m^{-3}}$, $\mu^- = 1.4746\,\mathrm{kg\ m^{-1}\ s^{-1}}$), and a light fluid is outside ($\rho^+ = 1\,\mathrm{kg\ m^{-3}}$, $\mu^+ = 1\,\mathrm{kg\ m^{-1}\ s^{-1}}$). The interface is evolved with the level set method, and the results are shown in Figure 6.15.

6.3.9 Relaxing ellipse

The tests up to this point have been analytic tests. Here we run a relaxing ellipse test similar to the one performed in [2]. Two fluids are separated by an interface in the initial shape of an ellipse. The fluid domain is $[-1\,\mathrm{m}, 1\,\mathrm{m}] \times [-1\,\mathrm{m}, 1\,\mathrm{m}]$, with periodic boundary conditions.



(a) Velocity error

(b) Pressure error

Figure 6.15: A stationary sphere is run with surface tension to test convergence of parasitic currents. Errors in velocity and pressure (y-axis, shown log base 10) are plotted against resolutions from 16 to 64 by increments of 8 (x-axis, shown log base 10). We estimate the velocity to be order 1.94 in L^∞ and 2.09 in L^2. For pressure, we obtained 1.12 in L^∞ and 1.87 in L^2.

The ellipse is centered in the domain with major axis 1.4 m and minor axis 0.8 m. The inside fluid has parameters $\rho^- = 0.01\,\mathrm{kg\ m^{-2}}$ and $\mu^- = 1\,\mathrm{kg\ s^{-1}}$. The outside fluid has parameters $\rho^+ = 0.02\,\mathrm{kg\ m^{-2}}$ and $\mu^+ = 3\ \mathrm{kg\ s^{-1}}$. The surface tension coefficient is $10\,\mathrm{kg\ m\ s^2}$. The simulations were run with time step $\Delta t = (0.01\,\mathrm{m^{-1}s})\Delta x$ until time $T = 0.05\,\mathrm{s}$. Convergence orders are shown in Figure **??**. Snapshots from the simulation are shown in Figure 6.16.

6.3.10 Relaxing ellipsoid - 3D

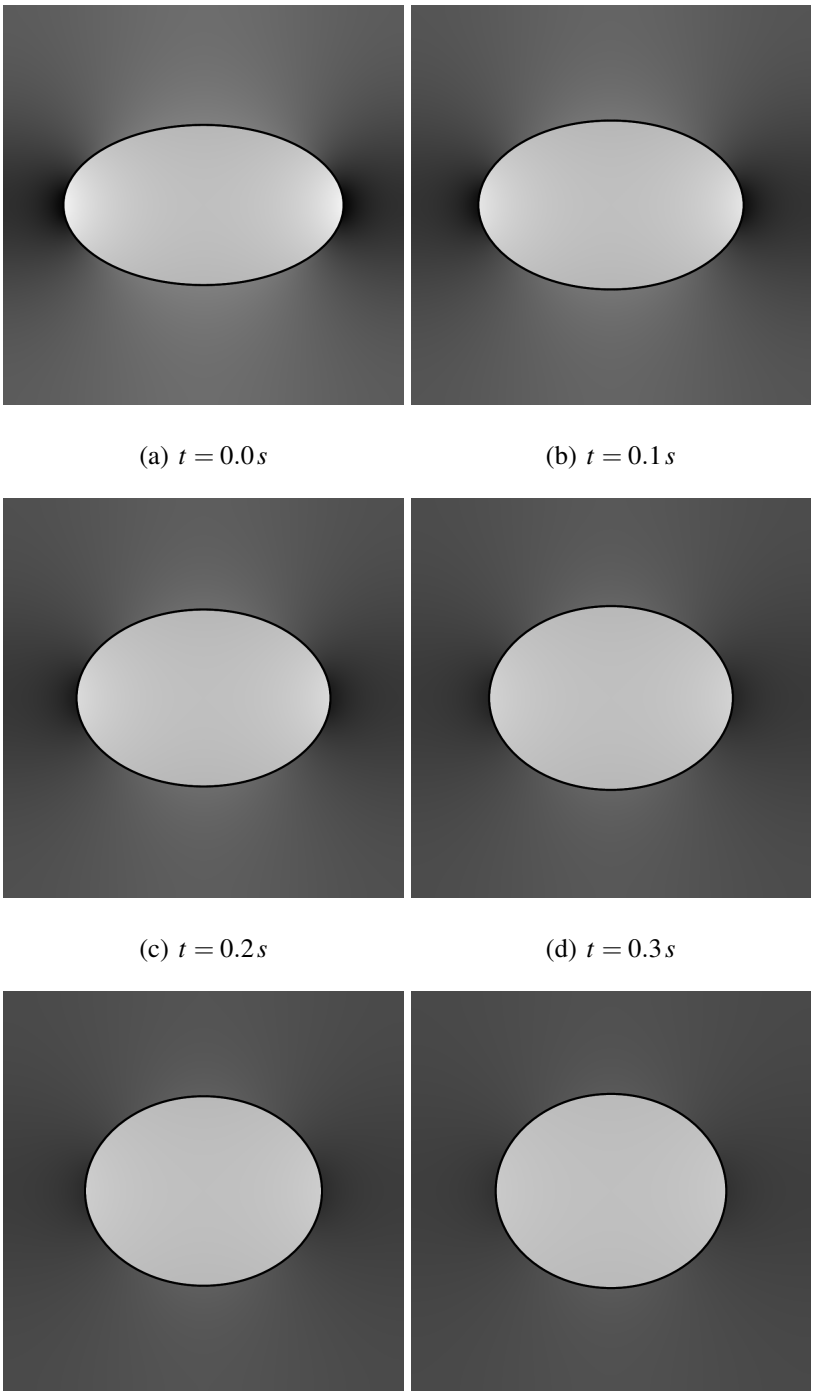
This relaxing ellipsoid test is a 3D analogue of Section 6.3.9. Two fluids are separated by an interface in the initial shape of an ellipsoid. The fluid domain is $[-1\,\mathrm{m}, 1\,\mathrm{m}] \times [-1\,\mathrm{m}, 1\,\mathrm{m}] \times [-1\,\mathrm{m}, 1\,\mathrm{m}]$, with periodic boundary conditions. The ellipsoid is centered in the domain with major and minor axes 1.4 m, 0.8 m, and 0.8 m. The inside fluid has parameters $\rho^- = 0.01\,\mathrm{kg\ m^{-3}}$ and $\mu^- = 1\,\mathrm{kg\ m^{-1}\ s^{-1}}$. The outside fluid has parameters $\rho^+ = 0.02\,\mathrm{kg\ m^{-3}}$ and $\mu^+ = 3\mathrm{kg\ m^{-1}s^{-1}}$. The surface tension coefficient is $10\,\mathrm{kg\,s^2}$. The simulations were run with time step $\Delta t = (0.01\,\mathrm{m^{-1}s})\Delta x$ until time $T = 0.015\,\mathrm{s}$. Convergence orders are shown in Fig-

Resolutions compared			Order ($\boldsymbol{u}$)		Order (p)	
			L^∞	L^2	L^∞	L^2
8	16	32	1.428	1.567	0.990	1.078
16	32	64	2.724	2.948	0.326	1.084
24	48	96	4.178	3.568	0.105	1.317
32	64	128	2.521	2.682	0.216	1.684
48	96	192	2.185	2.074	-0.358	1.638
64	128	256	2.453	2.112	0.457	2.334

Table 6.1: Order of convergence for 2D relaxing ellipse. Note that pressure is too noisy in L^∞ to give a meaningful convergence estimate. On the other hand, the pressure error is transitioning to second in L^2. (The transition to second is not merely noise. Noisy L^∞ and convergence orders consistent with second order in L^2 were also observed when this simulation was run with different parameters.) This suggests that, while the pressure may be noisy, it is converging.

Resolutions compared			Order ($\boldsymbol{u}$)		Order (p)	
			L^∞	L^2	L^∞	L^2
8	16	32	2.779	3.012	1.856	2.188
16	32	64	3.521	3.572	0.773	1.646

Table 6.2: Order of convergence for 3D relaxing ellipsoid.



(a) $t = 0.0s$ (b) $t = 0.1s$

(c) $t = 0.2s$ (d) $t = 0.3s$

(e) $t = 0.4s$ (f) $t = 0.5s$

Figure 6.16: Pressure and interface configuration for the relaxing ellipse of Section 6.3.9. Dark regions have lower pressure and lighter regions have higher pressure.

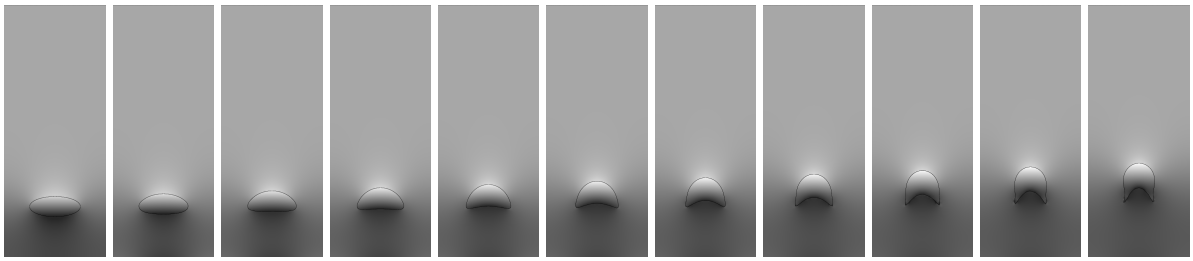
ure 6.2.

6.3.11 Rising Bubbles

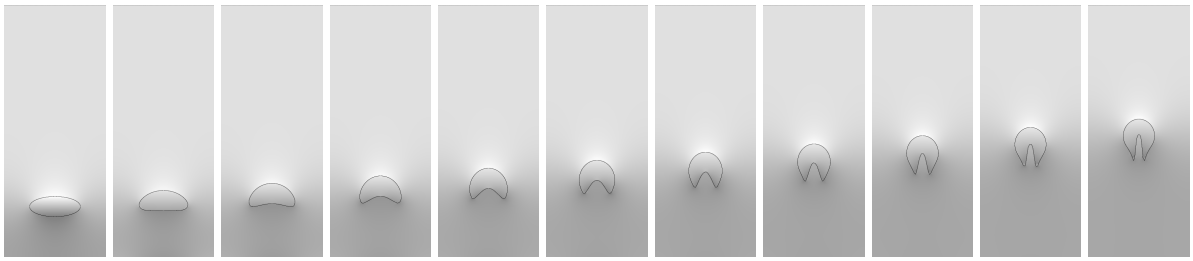
We used our algorithm to simulate a rising fluid bubble surrounded by fluid of differing viscosity and density. We assumed a uniform gravitational acceleration equal to $g = 9.8$, with the fluid densities being $\rho^- = 1$ inside the interface and $\rho^+ = 2$ outside the interface for all simulations. The interface is an ellipse of major radius $a = 0.5$ and minor radius $b = .2$ in a domain $[-1,1] \times [0,5]$, and we center it at $(0,1)$. The top and bottom have zero Dirichlet boundary conditions on the top and bottom, and the sides are periodic. We simulate examples where the inner viscosity $\mu^- = 1$ is less than the outer viscosity $\mu^+ = 3$, and examples where the inner viscosity $\mu^- = 3$ is greater than the outer viscosity $\mu^+ = 1$. These values are similar to those used in the rising bubble example in [2]. For each of these viscosity value pairs, we simulate a rising bubble without (Figures 6.17(a) and 6.17(b)) and with (Figures 6.17(c) and 6.17(d)) surface tension. In the simulations of Figures 6.17(c) and 6.17(d), we include a surface tension force whose coefficient is the same as in the other surface tension examples.

6.3.12 Stability

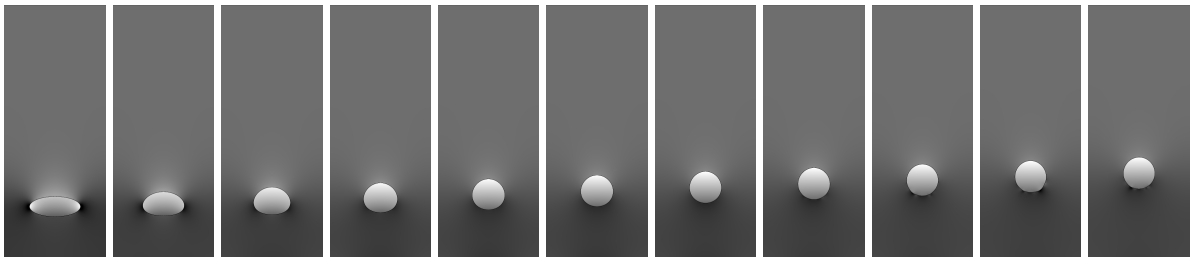
To examine stability, we consider two different examples, and for each run simulations for a variety of parameters that affect our stability. In each case, we choose five values of μ^-, ten values of ρ^-, two values of $\Delta x = 2/N$, and ten values of Δt, each sampled by powers of two. Each of these 1000 simulations is classified as stable or unstable. A simulation is classified as stable if it completes without producing velocities larger than 10 (Section 6.3.12.1) or 2 (Section 6.3.12.2). In practice, the classification was quite unambiguous most of the time (most simulations that are unstable simply explode). The rather low cutoff is much smaller than what would normally be considered 'blowup', and errs on the side of classifying simulations as unstable which do not blow up but still have large uncharacteristic variations in velocity.



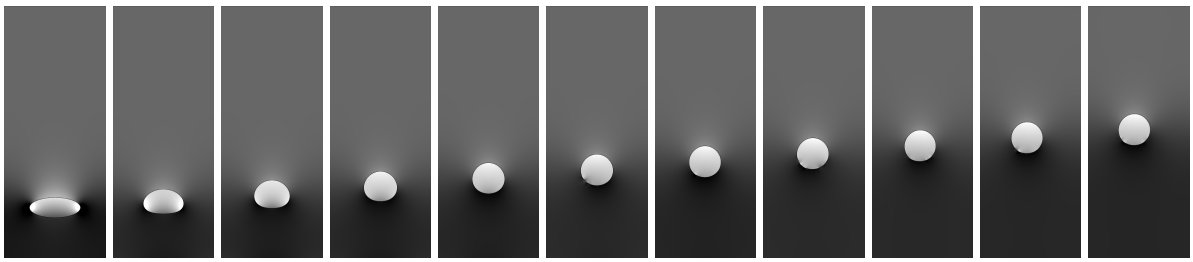
(a) $\mu^- = 1$, $\mu^+ = 3$, no surface tension



(b) $\mu^- = 3$, $\mu^+ = 1$, no surface tension

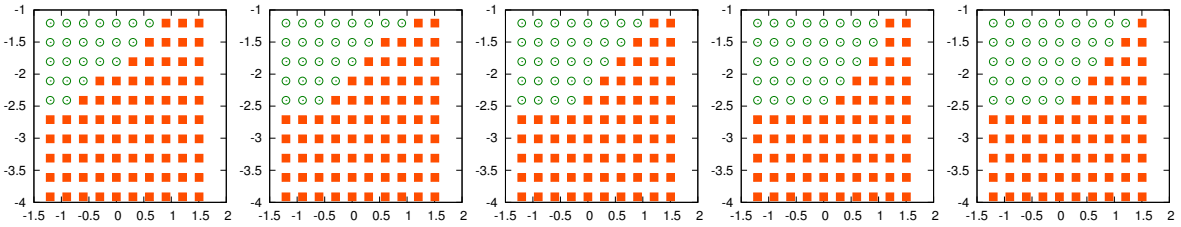


(c) $\mu^- = 1$, $\mu^+ = 3$, surface tension

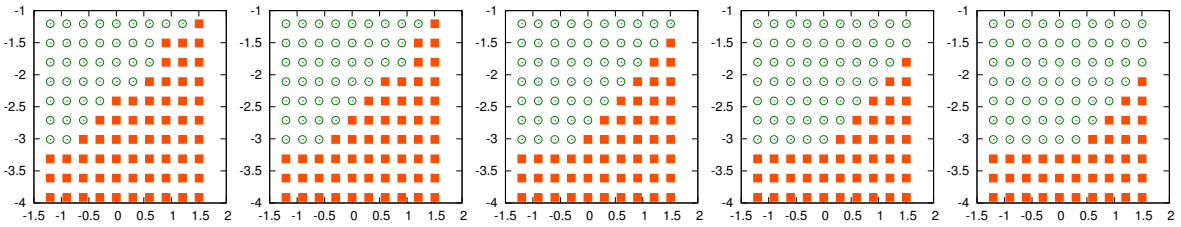


(d) $\mu^- = 3$, $\mu^+ = 1$, surface tension

Figure 6.17: Pressure and interface configurations for the four rising bubble simulations described in Section 6.3.11 at $t = 0.0, 1.0, 2.0, \ldots, 9.0, 10.0\,s$. For each simulation, dark regions correspond to lower pressure and lighter regions have higher pressure.



(a) $\mu^- = 0.25,\ N =$ (b) $\mu^- = 0.5, N = 32$ (c) $\mu^- = 1, N = 32$ (d) $\mu^- = 2, N = 32$ (e) $\mu^- = 4, N = 32$
32

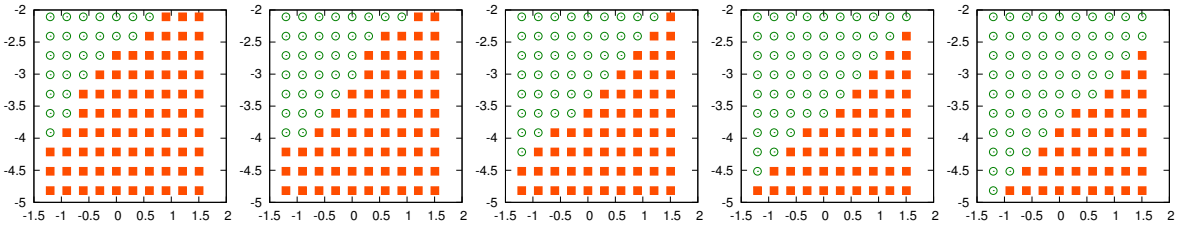


(f) $\mu^- = 0.25,\ N =$ (g) $\mu^- = 0.5, N = 64$ (h) $\mu^- = 1, N = 64$ (i) $\mu^- = 2, N = 64$ (j) $\mu^- = 4, N = 64$
64

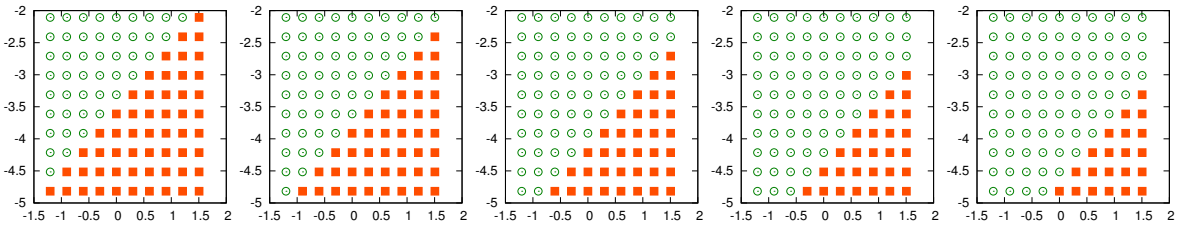
Figure 6.18: These plots show the stability of our method on one of our analytic tests. Each point on the (x, y) grid corresponds to a simulation with $\rho^- = 10^x$, $\rho^+ = 2\rho^-$, $\mu^+ = 2\mu^-$, $\Delta x = 2\pi/N$, and $\Delta t = 10^y$. Circles represent stable simulations, and squares represent simulations that were unstable.

6.3.12.1 Analytic test - stability

We demonstrate the stability characteristics of our method as a function of μ, ρ, Δx, and Δt first for the analytic test in Section 6.3.4 with $\rho^- = 1$, $\mu^- = 1$, $\rho^+ = 2$, and $\mu^+ = 2$. For this simulation, the transition between stable and unstable occurs at $\frac{\Delta t \mu}{\rho \Delta x^2} \approx 0.2$. When this threshold is reached, instabilities begin to develop at the interface between the two phases. In this simulation, there also appears to be a competing stability criterion of the form $\frac{\Delta t L u_\infty}{\Delta x^2}$, where L and u_∞ are a characteristic length and velocity. When this threshold is crossed, instabilities develop at the slip boundary condition. Results are shown in Figure 6.18.



(a) $\mu^- = 0.25,\ N =$ 32 (b) $\mu^- = 0.5, N = 32$ (c) $\mu^- = 1, N = 32$ (d) $\mu^- = 2, N = 32$ (e) $\mu^- = 4, N = 32$



(f) $\mu^- = 0.25,\ N =$ 64 (g) $\mu^- = 0.5, N = 64$ (h) $\mu^- = 1, N = 64$ (i) $\mu^- = 2, N = 64$ (j) $\mu^- = 4, N = 64$

Figure 6.19: These plots show the stability of our method on the relaxing ellipse with varying parameters. Each point on the (x, y) grid corresponds to a simulation with $\rho^- = 10^x$, $\rho^+ = 2\rho^-$, $\mu^+ = 3\mu^-$, $\Delta x = 2/N$, and $\Delta t = 10^y$. Circles represent stable simulations, and squares represent simulations that were unstable.

6.3.12.2 Relaxing ellipse - stability

For this stability test, we use the setup in Section 6.3.9, sampling ρ, μ, Δt, and Δx as before. For this simulation, the transition between stable and unstable occurs at $\frac{\Delta t \mu}{\rho \Delta x^2} \approx 0.1$. Instabilities, when they occur, develop at the interface. Results are shown in Figure 6.19.

CHAPTER 7

Conclusion

7.1 Conclusion

In this work we presented two numerical methods for incompressible flow problems. The first is a method for Hodge decomposition problems for inviscid Euler flow over irregular domains. This method is symmetric positive definite and produces pressures which are second order accurate and velocities which are first order accurate in L^∞ and second in L^1. Future work in this area will involve implementing this decomposition in time-varying Navier-Stokes problems, and in finding a modification which will give second order accurate velocities in L^∞.

The other method is a second order accurate method for the Navier-Stokes equations which can incorporate immersed interfaces, discontinuous fluid properties, and various boundary conditions. We considered examples in which both the fluid viscosities and densities are discontinuous across the interface, examples implementing each type of boundary condition listed in (6.8), and examples showing many combinations of these boundary conditions interacting. We demonstrated the ability of the method to handle interface forces by considering examples with surface tension. This method yields a symmetric indefinite linear system of equations. We also discussed two significant limitations of this method, which may be addressed in future work. The first limitation of our method, its additional stability restriction, effectively restricts its use to problems involving low or moderate Reynolds numbers (Re up to about 20 in practice). The method presented is not suitable for high Reynolds number flows. The second limitation is the KKT system, for which we currently lack an effective preconditioner.

REFERENCES

[1] J. Bedrossian, J. von Brecht, S. Zhu, E. Sifakis, J. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains, J. Comp. Phys. 229 (2010) 6405–6426.

[2] D. Assencio, J. Teran, A second order virtual node algorithm for stokes flow problems with interfacial forces, discontinuous material parameters and irregular domains, To appear, J. Comp. Phys.

[3] R. Howes, C. Schroeder, J. Teran, A virtual node algorithm for hodge decompositions of inviscid flow problems with irregular domains, Methods and Applications of Analysis.

[4] C. Schroeder, A. Stomakhin, R. Howes, J. Teran, A second order virtual node algorithm for navier-stokes flow problems with interfacial forces and discontinuous material properties, Submitted.

[5] A. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comp. Phys. 2 (1967) 12–26.

[6] O. Gonzalez, A. M. Stuart, A First Course in Continuum Mechanics, Cambridge University Press, 2008.

[7] F. Brezzi, M. Fortin, Mixed and hybrid finite elements methods, Springer series in computational mathematics, Springer-Verlag, 1991.

[8] R. L. Seliger, G. B. Whitham, A symmetric positive definite formulation for monolithic fluid structure interaction, Proc. Roy. Soc. A 305 (1968) 1–25.

[9] F. Wan, An Introduction to the Calculus of Variations and its Applications, Chapman and Hall, New York, 1995.

[10] F. Harlow, J. Welch, Numerical calculation of timedependent viscous incompressible flow of fluid with a free surface, Phys. Fl. 8 (1965) 2812–2189.

[11] D. L. Brown, R. Cortez, M. L. Minion, Accurate projection methods for the incompressible navier-stokes equations, J. Comput. Phys. 168 (2) (2001) 464–499.

[12] M. Hyman, Non-iterative numerical solution of boundary-value problems, Appl. Scientific Research, Section B 2 (1) (1952) 325–351.

[13] V. Sau'lev, On solving boundary value problems on high-performance computers by fictitious domain methods, Siberian Mathematical J. 4 (1963) 912–925.

[14] R. Glowinski, T. Pan, T. Hesla, D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow 25 (5) (1999) 755–794.

[15] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg. 112 (1-4) (1994) 133–148.

[16] R. Glowinski, T. Pan, T. Hesla, D. Joseph, J. Periaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flows, J. Comput. Phys. 169 (2001) 363–426.

[17] L. Parussini, V. Pediroda, Fictitious domain approach with hp-finite element approximation for incompressible fluid flow, J. Comput. Phys. 228 (2009) 3891–3910.

[18] L. Parussini, Fictitious domain approach via Lagrange multipliers with least squares spectral element method, J. Sci. Comput. 37 (2008) 316–335.

[19] F. Bertrand, P. Tanguy, F. Thibault, A three-dimensional fictitious domain method for incompressible fluid flow problems, Internat. J. Numer. Methods Fluids 25 (6) (1997) 719–736.

[20] J. M. Teran, C. S. Peskin, Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain, SIAM J. Sci. Comput. 31 (5) (2009) 3404–3416.

[21] G. Biros, L. Ying, D. Zorin, A fast solver for the Stokes equations with distributed forces in complex geometries, J. Comput. Phys. 193 (1) (2004) 317–348.

[22] V. Rutka, A staggered grid-based explicit jump immersed interface method for two-dimensional Stokes flows, Internat. J. Numer. Methods Fluids 57 (10) (2008) 1527–1543.

[23] C. S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.

[24] C. Peskin, The immersed boundary method, Act. Num. 11 (2002) 479–517.

[25] K.-Y. Chen, K.-A. Feng, Y. Kim, M.-C. Lai, A note on pressure accuracy in immersed boundary method for stokes flow, Journal of Computational Physics 230 (12) (2011) 4377 – 4383.

[26] C. S. Peskin, B. F. Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, J. Comput. Phys. 105 (1993) 33–46.

[27] R. Cortez, M. Minion, The blob projection method for immersed boundary problems, Journal of Computational Physics 161 (2) (2000) 428 – 453.

[28] I. Borazjani, F. Sotiropoulos, Numerical investigation of the hydrodynamics of carangiform swimming in the transitional and inertial flow regimes, J. Exp. Biol. 211 (2008) 1541–1558.

[29] V. Shankar, G. B. Wright, A. L. Fogelson, R. M. Kirby, A study of different modeling choices for simulating platelets within the immersed boundary method, Applied Numerical Mathematics 63 (0) (2013) 58 – 77.

[30] P. J. S. A. Ferreira de Sousa, J. C. F. Pereira, J. J. Allen, Two-dimensional compact finite difference immersed boundary method, International Journal for Numerical Methods in Fluids 65 (6) (2011) 609–624.

[31] P. Bagchi, R. M. Kalluri, Dynamics of nonspherical capsules in shear flow, Phys. Rev. E 80 (2009) 016307.

[32] R. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal 31 (1994) 1019–1044.

[33] R. J. LeVeque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709–735.

[34] Z. Li, M.-C. Lai, The immersed interface method for the navierstokes equations with singular forces, Journal of Computational Physics 171 (2) (2001) 822 – 842.

[35] L. Lee, R. J. LeVeque, An immersed interface method for incompressible Navier-Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.

[36] D. V. Le, B. C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comp. Phys (2006) 109–138.

[37] T. Y. Hou, Z. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele-Shaw flow, J. Comput. Phys. 134 (1997) 236–252.

[38] Z. Li, K. Ito, M.-C. Lai, An augmented approach for Stokes equations with a discontinuous viscosity and singular forces, Comput. & Fluids 36 (3) (2007) 622 – 635.

[39] Z. Tan, D. V. Le, K. M. Lim, B. C. Khoo, An immersed interface method for the incompressible Navier-Stokes equations with discontinuous viscosity across the interface, SIAM J. Sci. Comput. 31 (2009) 1798–1819.

[40] Z. Tan, D. V. Le, Z. Li, K. M. Lim, B. C. Khoo, An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane, J. Comput. Phys. 227 (2008) 9955–9983.

[41] X. Zhong, A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity, J. Comput. Phys. 225 (1) (2007) 1066 – 1099.

[42] Z. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, SIAM Press, 2006.

[43] S. Xu, Z. J. Wang, A 3d immersed interface method for fluid-solid interaction, Comput. Methods Appl. Mech. Engrg. 197 (25-28) (2008) 2068 – 2086.

[44] L. Adams, T. Chartier, New geometric immersed interface multigrid solvers, SIAM Journal on Scientific Computing 25 (5) (2004) 1516–1533.

[45] X.-D. Liu, R. P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151–178.

[46] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, J. Sci. Comput. 15 (2000) 323–360.

[47] M. Raessi, H. Pitsch, Consistent mass and momentum transport for simulating incompressible interfacial flows with large density ratios using the level set method, Computers & Fluids 63 (0) (2012) 70 – 81.

[48] Y. Zhou, S. Zhao, M. Feig, High order matched interface and boundary (mib) schemes for elliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (2006) 1–30.

[49] K. Xia, M. Zhan, G. Wei, Mib method for elliptic equations with multi-material interfaces, J. Comput. Phys. 230 (2011) 4588–4615.

[50] Y. Zhou, J. Liu, L. Harry, A matched interface and boundary method for solving multi-flow navierstokes equations with applications to geodynamics, J. Comput. Phys. 231 (2012) 223–242.

[51] Y. Ng, C. Min, F. Gibou, An efficient fluid-solid coupling algorithm for single-phase flows, J. Comp. Phys. 228 (23) (2009) 8807–8829.

[52] C. Batty, F. Bertails, R. Bridson, A fast variational framework for accurate solid-fluid coupling, ACM Transactions on Graphics 26.

[53] F. Gibou, C. Min, Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions, J. Comp. Phys. 231 (8) (2012) 3246–3263.

[54] C. Daux, N. Moës, J. Dolbow, N. Sukumar, T. Belytschko, Arbitrary branched and intersecting cracks with the extended finite element method, Internat. J. Numer. Methods Engrg. 48 (12) (2000) 1741–1760.

[55] N. Sukumar, D. Chopp, N. Moës, T. Belytschko, Modeling holes and inclusions by level sets in the extended finite-element method, Comput. Methods Appl. Mech. Engrg. 190 (46-47) (2001) 6183–6200.

[56] A. Almgren, J. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible euler equations in complex geometries, SIAM J. Sci. Comput. 18 (5) (1997) 1289–1309.

[57] N. Moës, E. Béchet, M. Tourbier, Imposing Dirichlet boundary conditions in the extended finite element method, Internat. J. Numer. Methods Engrg. 67 (12) (2006) 1641–1669.

[58] A. Lew, G. Buscaglia, A discontinuous-Galerkin-based immersed boundary method, Internat. J. Numer. Methods Engrg. 76 (4) (2008) 427–454.

[59] J. Dolbow, A. Devan, Enrichment of enhanced assumed strain approximations for representing strong discontinuities: addressing volumetric incompressibility and the discontinuous patch test, Internat. J. Numer. Methods Engrg. 59 (1) (2004) 47–67.

[60] G. Wagner, N. Moës, W. Liu, T. Belytschko, The extended finite element method for rigid particles in Stokes flow, Internat. J. Numer. Methods Engrg. 51 (3) (2001) 293–313.

[61] R. Becker, E. Burman, P. Hansbo, A Nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity, Comput. Methods Appl. Mech. Engrg. 198 (41-44) (2009) 3352–3360.

[62] A. Coppola-Owen, R. Codina, Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions, Internat. J. Numer. Methods Fluids 49 (12) (2005) 1287–1304.

[63] J. Chessa, T. Belytschko, An extended finite element method for two-phase fluids, J. Appl. Math. 70 (1) (2003) 10–17.

[64] A. Gerstenberger, W. Wall, An extended finite element method/Lagrange multiplier based approach for fluid-structure interaction, Comput. Methods Appl. Mech. Engrg. 197 (19-20) (2008) 1699–1714.

[65] S. Marella, S. Krishnan, H. Liu, H. Udaykumar, Sharp interface Cartesian grid method i: an easily implemented technique for 3d moving boundary computations, J. Comput. Phys. 210 (1) (2005) 1–31.

[66] D. Shirokoff, R. Rosales, An efficient method for the incompressible navierstokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary, Journal of Computational Physics 230 (23) (2011) 8619 – 8646.

[67] J. Hellrung, L. Wang, E. Sifakis, J. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comp. Phys. 231 (4) (2012) 2015–2048.

[68] Y. Zhu, Y. Wang, J. Hellrung, A. Cantarero, E. Sifakis, J. Teran, A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains,, J. Comput. Phys. 231 (21) (2012) 7092–7117.

[69] J. Teran, C. Peskin, Tether force constraints in stokes flow by the immersed boundary method on a periodic domain, SIAM J. Sci. Comp. 31 (5) (2009) 3404–3416.

[70] Z. Li, K. Ito, The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, SIAM Frontiers in Applied mathematics, 2006.

[71] S. Osher, R. Fedkiw, Level set methods and dynamic implicit surfaces, Springer-Verlag, 2002.

[72] F. Gibou, R. Fedkiw, L. Cheng, M. Kang, A second order accurate symmetric discretization of the poisson equation on irregular domains, J. Comput. Phys. 176 (2002) 205–227.

[73] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem, J. Comp. Phys. 202 (2005) 577–601.

[74] J. Dolbow, A. Devan, Enrichment of enhanced assumed strain approximations for representing strong discontinuities: addressing volumetric incompressibility and the discontinuous patch test, Int. J. Num. Meth. Eng. 59 (1) (2004) 47–67.

[75] C. Schroeder, Z. Zheng, R. Fedkiw, Implicit surface tension formulation with a lagrangian surface mesh on an eulerian simulation grid, J. Comput. Phys. 231 (2012) 2092–2115.

[76] D. Xiu, G. E. Karniadakis, A semi-lagrangian high-order method for navier–stokes equations, J. Comput. Phys. 172 (2) (2001) 658–684.

[77] T. D. Aslam, A partial differential equation approach to multidimensional extrapolation, J. Comput. Phys. 193 (1) (2004) 349–355.

[78] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.

[79] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.

[80] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys. 115 (1) (1994) 200–212.

[81] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted eno schemes, J. Comput. Phys. 126 (1996) 202–228.

[82] A. Robinson-Mosher, C. Schroeder, R. Fedkiw, A symmetric positive definite formulation for monolithic fluid structure interaction, J. Comput. Phys. 230 (4) (2011) 1547–1566.