AN ENERGY FORMULATION OF SURFACE TENSION OR WILLMORE FORCE FOR TWO-PHASE FLOW

A Dissertation

by

SPENCER ROBERT PATTY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Andrea Bonito
Committee Members,	Jean-Luc Guermond
	Anastasia Muliana
	Jay Walton
Head of Department,	Emil Straube

August 2017

Major Subject: Mathematics

ABSTRACT

The motion of a biological cell in liquid is a rich subject for modeling. In the early 1970's, it was realized by Canham that biological vesicles with lipid bilayer membranes reach a steady state shape that minimizes bending. Helfrich soon after mathematically quantified the related bending energy and showed that the shapes from minimizing this bending energy match the types of shapes observed in nature. The resulting Canham-Helfrich energy, consisting of bending energy and a constant surface area and volume constraint, is a major component of any model of cellular motility.

To this end, we consider the cellular vesicle to be a closed interface between two fluids and we present a finite element model for a two-phase flow coupling the minimization of some given energy defined on the interface to the incompressible flow of the two fluids, which is then advected according to the resulting velocity field. We provide a general framework for incorporating the energies on the interface and then focus on three applications of energy on the interface: the first is surface tension minimizing the surface area energy, the second minimizes the bending energy without explicit surface area or volume constraints, the third minimizes the Canham-Helfrich energy including the constraints. We present a semi-implicit model for bending energy which uses an implicit levelset formulation for the interface and couples the forces from the interface to the two phase incompressible Navier-Stokes system through the use of an approximate Dirac delta function defined on a band around the interface. By using energies to describe the motion, our model is immediately provided with a sense of energy stability.

We provide various numerical simulations and validations of flow under these three energies in two and three dimensions. Our simulations confirm that enforcing the volume constraint in the incompressible flow is vital to achieve the desired steady state shapes.

DEDICATION

To my Grandma and Grandpa Smutney who taught me the value of an education and to my wife who has stood by me through thick and thin on this journey we call a PhD, I dedicate this work. I love you so much! Thank you all for your sacrifices on my behalf!

ACKNOWLEDGMENTS

I am grateful to my advisor, Andrea Bonito, for his patience and the many insgihtful discussions through the years. I have learned much from you! I am also grateful to my other committee members for their insights and contributions to this work. I have enjoyed many discussions with various other students and professors at Texas A&M that have given me opportunity to see new perspectives on the use of math and scientific computing to describe the world.

I am grateful to my parents for always encouraging me when times were tough and rejoicing with me when I had little victories. I am especially grateful to my wife, Keri, for being my constant companion and keeping my life at least somewhat balanced. You are a light in my life and because of you, I have been able to accomplish and experience more in the last few years than I would ever have been able to do on my own. I am grateful to my two little girls, Camille and Corinne, for helping me see the world through their eyes of wonder.

Finally, I am grateful for the other friends and family that have been such good influences through the last few years. My family and I have been blessed by good friends that have helped us have fun and keep strong!

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Andrea Bonito, Professor Jean-Luc Guermond and Professor Jay Walton of the Department of Mathematics and Professor Anastasia Muliana of the Department of Mechanical Engineering.

All the work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and partially funded through the National Science Foundation under the grant DMS-1254618.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The modelling of biological vesicles like for instance the red blood cell or the various white blood cells or in general any object with a lipid bilayer membrane is important. Both to better understand the biological components but also for solving problems in medical research that could more efficiently use the white blood cells to fight disease. In addition, the mathematics and scientific comupting approaches involved in modelling cellular motility are complex, fascinating and offer much room for improvement.

To this end, we will develop a model for the dynamic flow of a single closed lipid bilayer vesicle in a fluid. The inside and outside fluids of the vesicle can be thought of as two phases of a fluid with an interface each having their respective density and viscosity. The fluids associated with this type of flow move at relatively slow speeds and small pressures so that an appropriate model for the fluid dynamics is the incompressible Navier-Stokes system, described in Chapter 4.

The interface between the two phases is a vesicle wall composed of a lipid bilayer membrane. The steady state shape of normal red blood cells are often described as a pin cushion shape or more technically as a discocyte. Canham in [1] proposed that the minimization of bending energy might be the dominant contributor to the shape of red blood cells and later Helfrich in [2] offered a more general bending energy description. By taking the variation of this energy he showed it was possible to obtain a differential equation for the steady state shape of the vesicle and indeed the solutions did match the shapes that were being observed by cell biologists. This bending energy coupled with the constraints of constant surface area and constant volume of the interior is referred to as the Canham-Helfrich energy.

We will incorporate the forces from minimizing this Canham-Helfrich energy into the

Navier-Stokes system and thus obtain the beginnings of a model for red blood cells or more generally any closed lipid bilayer vesicle. We have not yet added in the internal structure of organelles or the semi-rigid structure of the actin-myosin complex which can generate independent motion, or the complex set of chemical signals that dictate which directions to push or the myriad of other functionalities they accomplish including changing the amount of liquid volume inside the vesicle or the surface area of the vesicle itself. These are all future items to be added once we have the basic physics of the background fluid and the vesicle membrane itself. Note that at this stage this is a decent model of the passive red blood cells and less of a model for the various types of white blood cells who take a more active role in their motility.

We will use an implicit level set method approach to modelling the interface itself where the interface is described as the zero levelset of a smooth function, called the level set function, which will evolve in time so that the interface always stays as the zero levelset. We introduce the levelset method and describe the finite element approximation to transport in Chapter 2 where given a time-varying velocity field, the interface is transported along that velocity field through time. Now, the velocity field is not know a priori but is dependent on the interface itself as well as the liquid properties inside and outside the interface. We describe in Chapter 4, the finite element approximate to the two phase flow incompressible Navier-Stokes system that will be solved to generate the velocity field. We will describe in Chapter 5 a general model for coupling the minimization of an energy defined on the interface with the incompressible Navier-Stokes flow. Since these energies are defined as integrals along the interface we must have a method of integrating along it, so in Chapter 3, we introduce an approximate Dirac delta function that can use the implicit level set function to obtain integrals along the interface. It expands the integral to be over a band around the interface and thereby can use the standard quadrature rules often used by the finite element method on a discretization of the full region on which we are tracking

the vesicle.

Finally, we include three chapters with specific choices of energy building up to the Canham-Helfrich energy. Chapter 6 couples surface tension on the interface to the velocity flow which minimizes the surface area energy. We include some simulations of a rising bubble acting under gravity and various coefficients of surface tension. Chapter 7 gives a formulation of the Willmore energy a simplification of the bending energy described above without the surface area and volume constraints, although coupling these forces to the incompressible Navier-Stokes system preserves the volume due to the incompressibility of the liquids. Finally in Chapter 8 we add to the Willmore energy, the explicit constraints of constant surface area of the interface and constant volume of the interior which constitutes a model for the Canham-Helfrich energy. Several 2D and 3D simulations of flow from various initial shapes and ratios of surface area to volume are included in these chapters. The classic red blood cell shape can be observed in the numerical simulations of Section 8.3.4.

The finite element software behind the simulations presented her have been written using the deal.II version 8.5.0 [3] finite element library in the C++ language and using MPI(Message Passing Interface) [4] for using multiple processors. The discretized mesh is distributed over the MPI processors using the p4est library [5], version 1.1 so that each processor works on a portion of the computational domain. We use Trilinos [6] version 12.10.1 for all of the linear algebra with SuperLU_dist [7] version 5.1.2 accessed through the Amesos interface as a parallel direct solver for the 2D simulations of the Willmore energy of Chapter 7 and the General Minimum Residual (GMRes) iterative solver with the Algebraic Multi-grid (AMG) solver or the Successive Symmetric Overrelaxation (SSOR) algorithm as preconditioners for all the other models. We acknowledge the Texas A&M University Brazos HPC cluster, [8], that contributed to the research reported here as some of the simulations have been performed therein. We are also grateful for access to various other computational resources for scientific computing provided by the Texas A&M

University Mathematics Department on which we have performed the remainder of the simulations.

2. LEVEL SET METHOD

We are interested in a method to track the motion of an initial closed smooth (codimension one) hypersurface Γ_0 as it evolves under the motion of a given velocity field, $\mathbf{v}(t, \mathbf{x})$. We track its motion in some closed region $\Lambda \subset \mathbb{R}^d$ for $d = 2, 3$ that can be treated either as a snapshot of a larger region or as a physical region. The surface or interface at time $t \in [T_0, T_1]$ will be denoted as $\Gamma(t)$ or when it is clear which time we are talking about, just as Γ. We will denote by Ω, the region on the interior of Γ and $\mathbf{n}$, the outer unit normal to Γ. We also denote by $\boldsymbol{\nu}$, the outer unit normal for the boundary of the containing region, $\partial\Lambda$. See Figure 2.1 for an illustration.

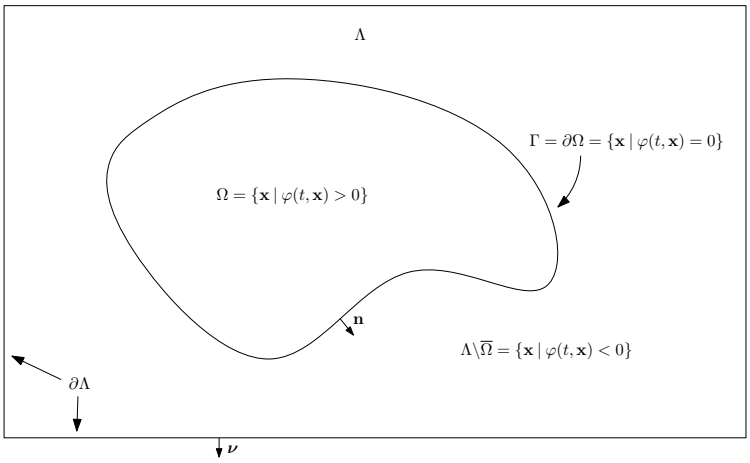


Figure 2.1: Region Λ containing a closed interface Γ as boundary of subregion Ω. These regions are defined implicitly by a level set function, $\varphi(t, \mathbf{x})$. The outer normal, $\mathbf{n}$, to Γ and outer normal, $\boldsymbol{\nu}$, to $\partial\Lambda$ are denoted as well.

To desribe the evolution of Γ, we use a level set method. In this setting, $\Gamma(t)$ is described as the zero level set of some function $\varphi(t, \mathbf{x})$ for $\mathbf{x} \in \Lambda$. Evolving the surface Γ with a velocity $\mathbf{v}(t, \mathbf{x})$ requires us to extend the velocity to Λ and advect φ with it. We choose

to let φ be positive inside of Γ and negative outside, that is $\Omega = \{\mathbf{x} \in \Lambda \mid \varphi(t, \mathbf{x}) > 0\}$ and $\Lambda \backslash \overline{\Omega} = \{\mathbf{x} \in \Lambda \mid \varphi(t, \mathbf{x}) < 0\}$.

In order to obtain the equation that will accomplish this, we consider for a given $t \in [T_0, T_1]$, the set of points $\{\mathbf{x} \in \Lambda \mid \varphi(t, \mathbf{x}) = 0\}$. This set of points then is transported with the velocity field, $\mathbf{v}(t, \mathbf{x})$. That is, we can consider them as particles dependent on time and having the velocity, $\frac{\partial \mathbf{x}}{\partial t} = \mathbf{v}(t, \mathbf{x})$. Then for each $\mathbf{x} \in \Gamma_0 = \{\mathbf{x} \in \Lambda \mid \varphi(T_0, \mathbf{x}) = 0\}$ we require that for $t \in (T_0, T_1]$ we have $\varphi(t, \mathbf{x}(t)) = 0$. Applying the total derivative, we obtain

$$0 = \frac{d}{dt}\varphi(t, \mathbf{x}(t)) = \frac{\partial \varphi}{\partial t} + \frac{\partial \mathbf{x}}{\partial t} \cdot \nabla \varphi = \frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi.$$

Then the level set method is then fully defined by the system

$$\begin{cases} \frac{\partial \varphi}{\partial t}(t, \mathbf{x}) + \mathbf{v}(t, \mathbf{x}) \cdot \nabla \varphi(t, \mathbf{x}) = 0, & \mathbf{x} \in \Lambda, t \in (T_0, T_1] \\ \varphi(T_0, \mathbf{x}) = \varphi_{\text{init}}(\mathbf{x}), & \mathbf{x} \in \Lambda \\ \varphi(t, \mathbf{x}) = \varphi_{\text{inflow}}(t, \mathbf{x}), & \{\mathbf{x} \in \partial\Lambda \mid \boldsymbol{\nu}(\mathbf{x}) \cdot \mathbf{v}(t, \mathbf{x}) < 0\} \end{cases} \tag{2.1}$$

where φ_{init} embeds the initial Γ_0 as it's zero level set and φ_{inflow} characterizes the inflow values of the function φ.

Notice that the above derivation is not particular about which level set we are tracking but it tranports each level set with the velocity field, $\mathbf{v}$. It is common in the literature to choose the level set function to be a signed distance function

$$d(t, \mathbf{x}) := \begin{cases} \min\limits_{\mathbf{y} \in \Gamma(t)} \|\mathbf{x} - \mathbf{y}\|_2, & \mathbf{x} \in \Omega(t) \\ -\min\limits_{\mathbf{y} \in \Gamma(t)} \|\mathbf{x} - \mathbf{y}\|_2, & \text{otherwise.} \end{cases} \tag{2.2}$$

but it is not a requirement. Notice that a property of the signed distance function is that

$|\nabla d| = 1$ for almost every $\mathbf{x} \in \Lambda$. Thus the outer normal to Γ is $\mathbf{n} = -\frac{\nabla d}{|\nabla d|} = -\nabla d$. This implies that it is simple for $\mathbf{x} \in \Lambda$ close to $\Gamma(t)$ to find the closest point $\mathbf{y} \in \Gamma(t)$ on the surface,

$$\mathbf{y} = \mathbf{x} + d(t, \mathbf{x})\nabla d(t, \mathbf{x}).$$

The literature is full of reasons why the signed distance function is a useful choice for our level set function. However, there are some down sides as well. When there are points that are equidistant from multiple places on Γ, the distance function looses regularity, in fact ∇d is no longer well defined at these places. We call this set of points the skeleton as in Figure 2.2.

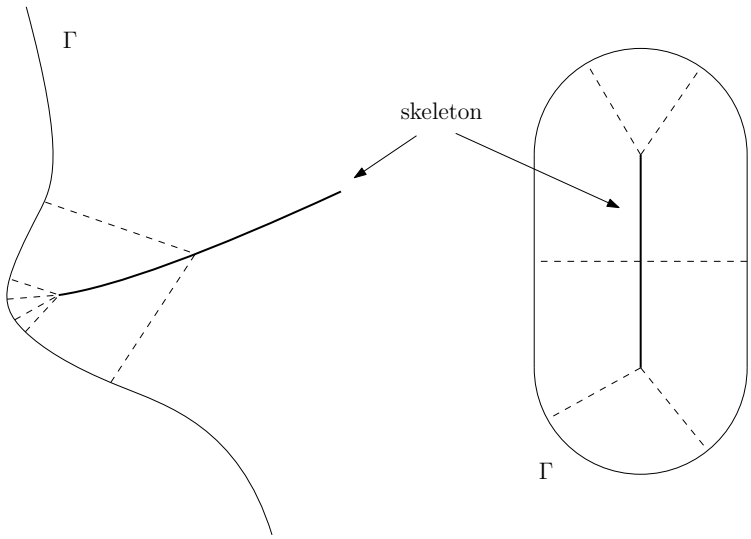


Figure 2.2: The skeleton of Γ is the set of points equidistant from more than one point on Γ, as seen in these two examples.

In addition, the distance function treats and tracks all level sets equally where as we only care about tracking the zero levelset. For these reasons, we choose to use a different shape of level set function. We will initially depend on the distance function, but will then

apply a filter to get a new shape that avoids the above problems. We discuss the choice of filters in Section 2.1. The filter will threshold the distance function to a narrow band around Γ and we will try to make sure that our narrow band does not touch the skeleton although this is not always possible. In doing this, we ensure sure that our levelset gradient is for the most part well defined.

Another problem that arises in the advection of our levelset function is that over time, the levelset shape drifts away from the desired initial profile shape. For instance if we started with a signed distance function, we would desire that our levelset function maintain the signed distance function as it advects. This is however not the case for the above sytem as written and so much work has been dedicated to finding ways to maintain the desired shape over time. We will describe in Section 2.2 some techniques for resolving this problem of maintaining the proper shape over time.

Finally, this system is hyperbolic and so we describe in Section 2.4 an artificial entropy viscosity stabilization scheme to maintain smoothness and stability. We implement an adaptive Lagrange finite element scheme and use an explicit strong stability preserving (SSP) Runge-Kutta algorithm for the time integration. The time integration scheme is discussed in Section 2.5.2. We present the semi-discrete algorithm in Section 2.5.1 and the fully discrete algorithm in Section 2.5.3. Finally we discuss the choice of optimal parameters in Section 2.6 and give some numerical results and validation in Section2.7.

Remark. There are in fact many methods for tracking the interface, Γ. Some of the better known explicit methods are the Volume of Fluid (VOF) method [9] the Marker and Cell (MAC) method [10] and the Immersed Boundary method [11] where the interface is parameterized and tracked explicitly. A recent advance in particle methods is given by [12] whre they discuss handling topological changes. The level set method (starting with [13]) and the phase field method (starting with [14] and [15]) are somewhat related implicit methods which rely on the interface being extracted from a specified level set. Another

approach is the parametric finite element method (see [16]) for modelling the interface directly. See also [17] for a direct comparison between the phase field method and parametric finite element methods. Likewise, it is possible to fit the bulk mesh to have Γ be explicitly described by edges of bulk elements as is common in the Arbitrary Eulerian-Lagrangian method (starting with [18]). However, mesh quality is now more difficult to maintain for large deformations. There are likely many others, but we have chosen to use the level set method as it is simple to implement and does not require any special meshing and can easily handle topological changes. The explicit methods are acceptable when no topological changes occur like pinching or self intersection, however it is not clear how to properly handle the topological changes. In addition, as transport, including stretching occurs, the mesh quality of the explicit methods can deteriorate if something is not done to re-distribute the nodes of the mesh. There are solutions to some of these problems in the literature but we prefer the simplicity of the level set method.

However, the use of the level set method with it's implicit surface introduces a new complication. Mainly, that integration along Γ is no longer a simple thing and so incorporating forces like surface tension or the Willmore force that are described as a differential equation on Γ is more complicated. We will discuss in Chapter 3, the use of an approximate Dirac delta function in conjuntion with this level set function to resolve this issue and then in Chapter 4 the use of the level set method to construct an extended velocity field representing two-phase flow. Finally in Chapters 5 - 8, we use the approximate Dirac delta function and level set function to incorporate forces like surface tension, Willmore and Canham-Helfrich energy flow on Γ into the construction of our velocity field. *end Remark.*

2.1 Level set filters

As we transport the level set function φ through time, we are doing so with the goal of tracking the interface Γ. If we use a distance function, then we not only have Γ, the zero level set, but all of the other level sets. In addition, it is not clear what the inflow values should be at the boundary $\partial\Lambda$ and nor do we really care. We are led to consider that we only care about tracking the zero levelset and maybe a band around it that might accomodate the use of an approximate Dirac delta function. Thus we can think of thresholding the level set function in some smooth way to remove the need to know precisely how far every single point is from Γ as is provided by the signed distance function. We will introduce a filter function $f : \mathbb{R} \to \mathbb{R}$ which has the properties that $f(0) = 0$, f is non decreasing and f is differentiable at 0 with $f'(0) = 1$. We then introduce a filter width $c > 0$ and given a distance function $d(t, x)$, we define our filtered level set function to be

$$\varphi(t, \mathbf{x}) = cf\left(\frac{d(t, \mathbf{x})}{c}\right). \tag{2.3}$$

By a judicious choice of c and f, we can avoid the complications of the inflow values and the ambiguity of $\nabla\varphi$ introduced at points which are equidistant from multiple points on Γ. Given a triangulation, $\mathcal{T}$ of Λ with minimal mesh size, h, we will often choose our filter width $c > 0$ to scale with the mesh size $c = mh^s$ with $0 < s \leq 1$ and $m > 0$. Thus the band around Γ scales with the mesh size. The filter being used and problem being solved will dictate the parameters s and m. Often the choice of approximate Dirac delta function will determine how wide of a band we need to maintain.

There are many possible filters that fit the basic description above, but we will restrict ourselves to filters that can be expressed in the form of an ordinary differential equation as follows. Notice that since the signed distance function grows(decreases) linearly from 0 at Γ, the application of filter is in the normal direction. The distance function has the

property that $|\nabla d(t, \mathbf{x})| = 1$. Then supposing f is differentiable,

$$|\nabla \varphi| = \left| c \nabla f \left(\frac{d(t, \mathbf{x})}{c} \right) \right| = \left| f' \left(\frac{d(t, \mathbf{x})}{c} \right) \right| |\nabla d(t, \mathbf{x})| = \left| f' \left(\frac{d(t, \mathbf{x})}{c} \right) \right| = f' \left(\frac{d(t, \mathbf{x})}{c} \right)$$

since f is non decreasing. In addition, the value of $|\nabla \varphi|$ is constant along each level set and varies only in the normal direction according to the derivative of f. Thus, we choose filters f that can be written in the ode form $f' = L(f)$ with initial conditions $f(0) = 0$. This results in the Eikonal equation that defines the level set equation profile

$$|\nabla \varphi| = S(\varphi) \tag{2.4}$$

with $\varphi = 0$ on Γ for some cut-off function $S : \mathbb{R} \to \mathbb{R}$ resulting from the ode right hand side $L(f)$ function. We will use this cut-off function S when we discuss the reinitialization of the level set function in Section 2.2.

It turns out that the identity function $f(s) = s$ is one such filter since $f'(s) = 1$ and $f(0) = 0$. So the distance function is a (not very interesting) example of a filtered level set. In [19], Coupez introduced some other more interesting examples using the sinusoid and hyperbolic tangent filters for thresholding. These are given respectively by

$$f(s) = \begin{cases} 1 & s > \frac{\pi}{2} \\ \sin(s) & |s| \leq \frac{\pi}{2} \\ -1 & s < -\frac{\pi}{2} \end{cases} \tag{2.5}$$

and

$$f(s) = \tanh(s). \tag{2.6}$$

We will subsequently discuss each of these filters in more details and then finally introduce

a new filter that we have found most useful for cases where an approximate Dirac delta function is being used to integrate along Γ. It preserves the distance function in a band and then thresholds with the hyperbolic tangent filter. This final filter will be the one we implement in most of our simulations.

2.1.1 Sinusoidal level set filter

The sinusoidal filter

$$f(s) = \begin{cases} \frac{2}{\pi} & s > 1 \\ \frac{2}{\pi} \sin\left(\frac{\pi}{2} s\right) & |s| \leq 1 \\ -\frac{2}{\pi} & s < -1 \end{cases} \tag{2.7}$$

is in $C^1(\mathbb{R})$. Since $\frac{d}{ds}\sin(s) = \cos(s) = \sqrt{1 - \sin^2(s)}$ this filter satisfies the ode system

$$f'(s) = \begin{cases} \sqrt{1 - \left(\frac{\pi}{2} f(s)\right)^2}, & |f(s)| \leq \frac{2}{\pi} \\ 0, & \text{otherwise,} \end{cases}$$

with $f(0) = 0$. Combining with the scaling coefficient, $c_f > 0$ and applying this filter to the distance function, the resulting level set function is

$$\varphi(t, \mathbf{x}) = \begin{cases} \frac{2c_f}{\pi} & d(t, \mathbf{x}) > c_f \\ \frac{2c_f}{\pi} \sin\left(\frac{\pi}{2c_f} d(t, \mathbf{x})\right) & |d(t, \mathbf{x})| \leq c_f \\ -\frac{2c_f}{\pi} & d(t, \mathbf{x}) < -c_f \end{cases} \tag{2.8}$$

with the normal profile and it's derivative depicted in Figure 2.3. The corresponding Eikonal equation for the level set profile is

$$|\nabla\varphi| = S(\varphi) := \begin{cases} \sqrt{1 - \left(\frac{\pi}{2c_f}\varphi\right)^2} & |\varphi| < \frac{2c_f}{\pi} \\ 0, & \text{otherwise} \end{cases}$$

with $\varphi = 0$ on Γ.

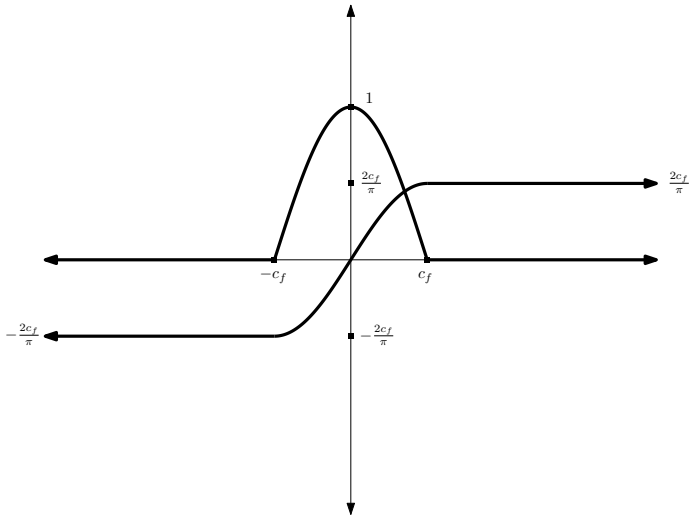


Figure 2.3: The normal to Γ profile of the sinusoidal level set function, equation (2.8), plotted with the normal profile of $|\nabla\varphi|$.

2.1.2 Hyperbolic tangent level set filter

The hyperbolic tangent filter

$$f(s) = \tanh(s) \tag{2.9}$$

is in $C^\infty(\mathbb{R})$ and owing to $\frac{d}{ds}\tanh(s) = \operatorname{sech}^2(s) = 1 - \tanh^2(s)$, satisfies the ode system

$$f'(s) = 1 - f(s)^2 \tag{2.10}$$

with initial conditions $f(0) = 0$. Combining with the scaling coefficient, $c_f > 0$ and applying the filter to the distance function, the resulting level set function is

$$\varphi(t, \mathbf{x}) = c_f \tanh\left(\frac{d(t, \mathbf{x})}{c_f}\right) \tag{2.11}$$

with the normal profile and it's derivative depicted in Figure 2.4. The corresponding Eikonal equation for the level set profile is

$$|\nabla\varphi| = S(\varphi) := 1 - \left(\frac{\varphi}{c_f}\right)^2.$$

with $\varphi = 0$ on Γ.

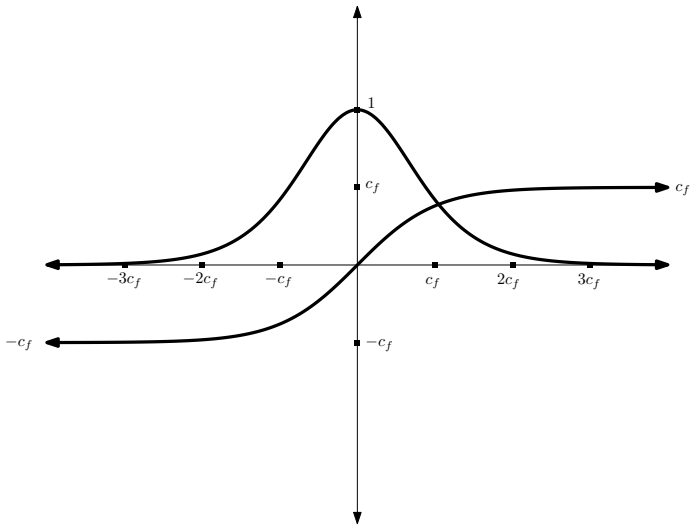


Figure 2.4: The normal to Γ profile of the hyperbolic tangent level set function, equation (2.11), plotted with the normal profile of $|\nabla\varphi|$.

Remark. In addition to being useful for the level set method, the $\tanh$ filter shape is what many phase field models converge to when using the standard double well potential. In actuality, they converge to something more like $\varphi(t, \mathbf{x}) = \tanh\left(\frac{d(t,\mathbf{x})}{c_f\sqrt{2}}\right)$, without the vertical scaling. However, in the phase field models, the width of the filter is either a fixed constant independent of the mesh size if the physical diffuse interface is of a reasonable scale or they choose to scale linearly with the meshsize. In our case the width (and height) always changes with the minimal mesh size. *end Remark.*

2.1.3 Distance function with thresholding for level set filter

This final filter preserves the distance function in a band around Γ and then thresholds with the hyperbolic tangent function outside that band. In studying convergence properties of some approximate Dirac delta functions, we observed that they seem to behave much better when used with a signed distance function. Thus, we choose to keep the distance function in a band that contains the support of the approximate Dirac delta function we are using. We introduce two coefficients, $c_d > 0$ which is the half width of the band where we preserve the distance function and then $c_f > 0$ which will be used for the hyperbolic tangent width. As there are two coefficients now, it is simpler to define the filter with the coefficients already built into the definition. To this end, we define the distance with hyperbolic tangent thresholding filter to be

$$f(s) = \begin{cases} c_d + c_f \tanh\left(\frac{s-c_d}{c_f}\right), & c_d < s \\ s, & |s| < c_d \\ -c_d + c_f \tanh\left(\frac{s+c_d}{c_f}\right), & s < -c_d. \end{cases} \tag{2.12}$$

Notice that this is in $C^2(\mathbb{R})$ and is the solution to the ordinary differential equation

$$f' = \begin{cases} 1 - \left(\frac{f-c_d}{c_f}\right)^2, & f > c_d \\ 1, & |f| < c_d \\ 1 - \left(\frac{f+c_d}{c_f}\right)^2, & f < -c_d \end{cases}$$

with initial conditions $f(0) = 0$. Combining with the scaling coefficient, $c_f > 0$ and applying the filter to the distance function, our resulting level set function is

$$\varphi(t, \mathbf{x}) = \begin{cases} c_d + c_f \tanh\left(\frac{d(t,\mathbf{x})-c_d}{c_f}\right), & c_d < d(t, \mathbf{x}) \\ d(t, \mathbf{x}), & |d(t, \mathbf{x})| < c_d \\ -c_d + c_f \tanh\left(\frac{d(t,\mathbf{x})+c_d}{c_f}\right), & d(t, \mathbf{x}) < -c_d \end{cases} \tag{2.13}$$

with the normal profile and it's derivative depicted in Figure 2.5. The corresponding Eikonal equation for the level set profile is

$$|\nabla\varphi| = S(\varphi) := \begin{cases} 1 - \left(\frac{\varphi-c_d}{c_f}\right)^2, & \varphi > c_d \\ 1, & |\varphi| < c_d \\ 1 - \left(\frac{\varphi+c_d}{c_f}\right)^2, & \varphi < -c_d. \end{cases}$$

with $\varphi = 0$ on Γ.

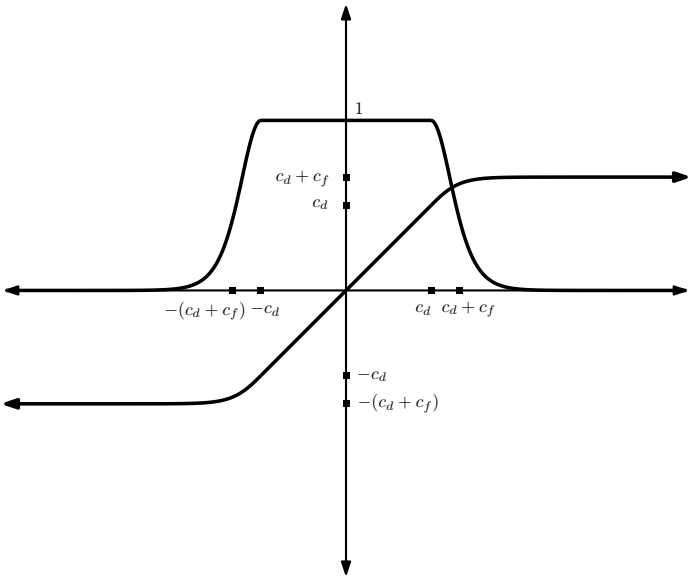


Figure 2.5: The normal to Γ profile of the distance with hyperbolic tangent thresholding level set function, equation (2.13), plotted with the normal profile of $|\nabla\varphi|$.

2.2 Level set reinitialization

As the level set is advected, it tends to drift away from the desired profile shape. There are many suggestions in the literature for how to deal with this problem. The most common and simple suggestion is to periodically pause and reinitialize the level set to the proper shape. This can be done using something like the Fast Marching Method or using an artificial time relaxation of the Eikonal equation

$$|\nabla\varphi| = S(\varphi)$$

with $\varphi = 0$ on Γ. The artificial time relaxation for the level set profile $\varphi(t, \mathbf{x})$ at time $t \in (T_0, T_1]$ is accomplished by introducing an artificial time $\tau \in [0, \infty)$ and auxiliary

function, $\psi^t(\tau, \mathbf{x}) : [0, \infty] \times \Lambda \to \mathbb{R}$ that satisfies the reinitialization equation

$$\frac{d\psi^t}{d\tau}(\tau, \mathbf{x}) + \text{sign}\left(\psi^t(\tau, \mathbf{x})\right)\left(\left|\nabla\psi^t(\tau, \mathbf{x})\right| - S\left(\psi^t(\tau, \mathbf{x})\right)\right) = 0 \tag{2.14}$$

with initial conditions $\psi^t(\tau = 0, \mathbf{x}) = \varphi(t, \mathbf{x})$. The sign function

$$\text{sign}(s) = \begin{cases} 1, & s > 0 \\ 0, & s = 0 \\ -1, & s < 0 \end{cases} \tag{2.15}$$

will be replaced in the approximate system by the discrete sign function

$$\text{sign}(s) := \begin{cases} 1, & s > \varepsilon_S \\ 0, & |s| < \varepsilon_S \\ -1, & s < -\varepsilon_S \end{cases}$$

where $\varepsilon_S = cf(\frac{y_S}{c})$ is the filter applied to y_S and y_S is the distance around the the zero level set we don't want to be reinitialized. Keeping this value small but positive ensures we don't accidentally move the zero level set too much in the discrete reinitialization process. If we use a p-th order Lagrange finite element basis for our approximation, then we will have it dependent on the minimal mesh size per degree of freedom, $y_S = \mathcal{O}\left(\frac{h_{\min}}{p}\right)$. The zero level set will undoubtedly move, but this helps to minimizes the shifting.

The newly reinitialized profile is then the steady state solution, $\psi^t(\tau = \infty, \mathbf{x})$. Notice that we have used the total derivative instead of the partial derivative, but in fact they are the same thing for this system since $\mathbf{x} \in \Gamma$ does not depend at all on τ and the rest are

simply shifted relative to where the initial Γ is located. That is, we have the equality

$$\frac{\partial \psi^t}{\partial \tau}(\tau, \mathbf{x}) = \frac{d\psi^t}{d\tau}(\tau, \mathbf{x}).$$

Then as needed, we pause our level set simulation and solve this system until it is steady and then begin anew with the newly reinitalized solution. We will introduce two integer coefficients, T and R that will govern this usage. They are used as follows: For every T steps of transport, apply R steps of reinitialization. Then the system alternates between transport and reinitialization. It is also convenient to think of the reinitialization step as a post processing operation to the transport step that is triggered only according to T and R.

Remark. If done frequently enough, it is not necessary to always reinitialize to steady state. Sometimes taking $T = 1$ steps of transport and $R = 1$ steps of reinitialization or some small ratio of the two is all that is needed to continuously maintain the solution near steady state. *end Remark.*

Remark. The reinitialization step solves a transport equation and so will need stabilization. As the wave traveling out from Γ can be quite strong and sharp, we will use an explicit linear artificial viscosity to stabilize it. This provides some extra smoothness to the solution but can lead to shifting of Γ, the zero level set, and so we sometimes see a volume or mass loss. In addition, as we will be using a Lagrange finite element scheme to approximate this system, the approximation also carries some inherent variance in volume or mass. We will show in Section 2.7.4 that the relative mass loss is linear in the amount of reinitialization being applied and linear in the mesh/time step size. Thus there will likely be mass loss over time but it can be managed by choosing the amount of reinitialization and mesh size properly. *end Remark.*

2.2.1 A continuous approach to reinitialization

In [20], Coupez proposes a way to integrate the transport and reinitialization step into a single system. We will describe this approach here and discuss it's relative merits. This approach, which we will distinguish from the standard reinitialization scheme as the Coupez reinitialization scheme, seems to work quite well for the sinusoidal and hyperbolic tangent filter cases especially if the filter widths $c_f = \mathcal{O}(h_{\min})$ scale with the mesh size. However, we have not found a way to achieve the right mix of reinitialization and stabilization for the distance with hyperbolic tangent filter which has a significantly larger profile width. In that case, we find that the alternating with specified T and R is sufficient. Nevertheless, we review the derivation of this integrated reinitialization/transport scheme.

Using the two time scales, t and τ we introduce a unitless scalar function, λ, representing the relative scales of time

$$\lambda = \frac{\partial \tau}{\partial t}.$$

We will eventually allow the relative scaling to vary in space but for now consider it to be simply a scalar. Then using the chain rule and thinking of $\tau = \tau(t)$ we have

$$\frac{d\psi^t}{d\tau} = \frac{1}{\lambda}\frac{d\psi^t}{dt}$$

Thus, we can rewrite the reinitialization equation (2.14) with respect to t and then substituting the initial conditions, ie replacing $\psi^t(\tau(t), \mathbf{x})$ with $\varphi(t, \mathbf{x})$ we have

$$\frac{d\varphi}{dt} + \lambda \text{sign}\left(\varphi\right)\left(\left|\nabla\varphi\right| - S\left(\varphi\right)\right) = 0.$$

Finally, we recognize the first term as a total (material) derivative, so using the level set

equation (2.1), we obtain the transport with reinitialization evolution equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi + \lambda \text{sign}\left(\varphi\right)\left(|\nabla \varphi| - S\left(\varphi\right)\right) = 0.$$

Notice that $|\nabla \varphi| = \frac{\nabla \varphi}{|\nabla \varphi|} \cdot \nabla \varphi$ so by introducing the reinitialization velocity,

$$\mathbf{U} := \text{sign}(\varphi)\frac{\nabla \varphi}{|\nabla \varphi|}$$

we obtain the transport equation

$$\frac{\partial \varphi}{\partial t} + (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla \varphi = \lambda \text{sign}(\varphi) S\left(\varphi\right).$$

The true velocity is offset by the artificial reinitialization velocity based on the λ parameter. Notice that the velocity field $\mathbf{v}$ may induce an inflow condition but the $\mathbf{U}$ velocity is pushing toward the boundaries from Γ. Likewise the reinitialization velocity is artificial so that when we check for inflow conditions, we should use only the true velocity, $\mathbf{v}$. However, any CFL condition should use the velocity $\mathbf{v} + \lambda \mathbf{U}$. We thus obtain the level set with reinitialization system

$$\begin{cases} \frac{\partial \varphi}{\partial t} + (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla \varphi = \lambda \text{sign}(\varphi) S\left(\varphi\right), & (t, \mathbf{x}) \in (T_0, T_1] \times \Lambda \\ \varphi(t, \mathbf{x}) = \varphi_{\text{inflow}}(t, \mathbf{x}), & \mathbf{v}(t, \mathbf{x}) \cdot \boldsymbol{\nu}(\mathbf{x}) < 0,\ \mathbf{x} \in \partial\Lambda \\ \varphi(T_0, \mathbf{x}) = \varphi_{\text{initial}}(\mathbf{x}), & \mathbf{x} \in \Lambda. \end{cases} \tag{2.16}$$

Remark. We will discuss how to choose the scalar function, λ, in section 2.3.2.1 after we have discussed the CFL condition as it is best understood in that context. *end Remark.*

2.3 Spatial and temporal discretization notation

Now that we have written down the continuous level set and level set with reinitialization systems, we can discuss our spatial and temporal discretizations. For each $t \in [T_0, T_1]$, we discretize Λ with rectangular (2D) or hexagonal (3D) elements to obtain an adaptive, unfitted, graded mesh, $\mathcal{T}(t)$. We choose to fully refine to some given smallest mesh size in a band around $\Gamma(t)$ so that it is essentially a uniform mesh where the level set filter is varying and then grade the mesh to some given coursest mesh size as quickly as will be allowed by the refinement protocol and mesh data structure in use.

For a specific $t \in (T_0, T_1]$, the size of an element, $K \in \mathcal{T}(t)$, is denoted h_K and calculated as the maximal distance between vertices divided by $\sqrt{d}$ so that the reference element has mesh size $h_{\hat{K}} = 1$. The minimal mesh size, $h = h_{\text{min}}$, is calculated over all cells in the triangulation,

$$h = h_{\text{min}} := \min_{K \in \mathcal{T}(t)} h_K. \tag{2.17}$$

Remark. In our case, we implement our system using the deal.II finite element library ([3]) with p4est ([5]) for the mesh. The data structure being used for the mesh is a forest of octtrees. Each level of a tree represents a refinement (halving in each coordinate direction) of a mesh and the leaves represent the active elements of the mesh. In 2D, an element being refined would have 4 children whereas in 3D it would have 8 children. The deal.II library assumes that for a given active element, K in the mesh, if we consider the set of neighboring cells on the same level as K (active and not) their active children differ by no more than two levels from K's level. That is, there is a limit on how fast you can coursen or refine. Needless to say, we grade the mesh appropriately away from this band of smallest refinement. *end Remark.*

Now, we leave open the possibility that the velocity or any other added component could have an adaptive refinement estimator incorporated into the system. The process

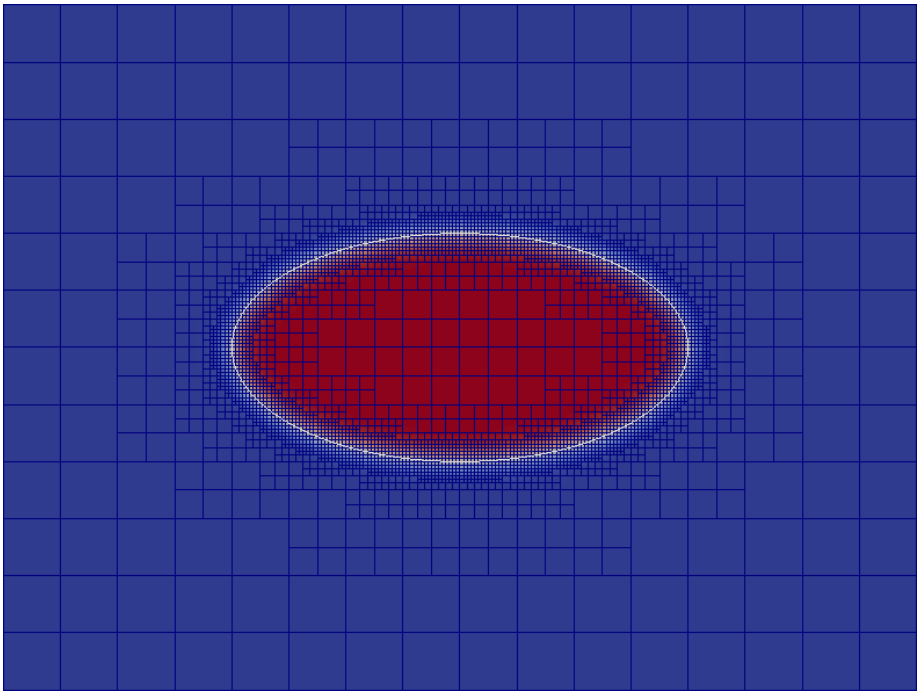


Figure 2.6: The level set function with distance function and hyperbolic tangent thresholding for a 2D ellipse with 2 to 1 ratio. The graded adaptive mesh is depicted.

would be to run through the estimators for those other components and combine them to obtain a refine/coarsen flag for each cell, then we loop through the cells and make sure that all the cells on the band of interest stay or become fully refined. We will finally impose a user specified maximal and minimal level of refinement as well. We will ideologically prefer refinement to coarsening and enforce the level set grading last of all.

In order to avoid unnecessary initial memory usage, the initial mesh will be uniformly refined to a somewhat coarse level somewhere between the maximal and minimal accepted levels and then we will run the above procedure of marking for refinement and coarsening enough times so that the mesh reaches a steady state. As the level set filter parameter may depend on the current minimal mesh size, it may be necessary with the initial refinements to periodically project or interpolate the given initial level set function and if the exact distance function (or filter applied to distance function) is not known for the initial shape,

run the reinitialization scheme to get the new initial level set function starting from the characteristic function of Ω.

The band width will be chosen based on the maximal value the level set will obtain, $\varphi_{\max}$ which is related to the filter width and a user defined parameter $c_a \in [0, 1]$ so that we refine a cell K if the value of the level set at the barycenter, $\mathbf{x}_B \in K$ of that cell is smaller than a given value,

$$|\varphi(t, \mathbf{x}_B)| \leq c_a \varphi_{\max}. \tag{2.18}$$

Otherwise we mark for coursening and let the refinement protocol sort out the details. Typically we will choose $c_a = 0.9$ or there abouts so that we are fully refined on the region where large changes in the level set might be taking place.

Remark. If the level set filter we are using does not have a maximal value, (e.g. distance function) then we must specify the band width ourselves. *end Remark.*

We will use a pth order Lagrange finite element subordinate to our mesh, $\mathbb{Q}^p(\mathcal{T}(t))$ typically with $p = 2$. Since we are not dealing with a uniform mesh, the standard subscript h notation $\varphi_h(t) \in \mathbb{Q}^p(\mathcal{T}(t))$ denoting the discrete solution does not have meaning. We will instead denote the discrete solutions by a capital letter, $\Phi(t) \in \mathbb{Q}^p(\mathcal{T}(t))$ and the continuous solutions by a lower case letter, $\varphi(t)$. On a given cell, $K \in \mathcal{T}(t)$, to scale properly with h-adaptivity and polynomial degree, p, the correct unit of size is mesh size per polynomial degree, $\frac{h_K}{p}$. This measures the distance between degrees of freedom not just between nodes. Likewise the proper scaling minimal measure of the mesh is minimal mesh size per polynomial degree, $\frac{h_{\min}}{p}$ or $\frac{h}{p}$. We will use these terms in the subsequent.

2.3.1 *Temporal discretizations*

We will discretize our time interval, $[T_0, T_1]$ into N timesteps, $t^0 = T_0 < t^1 < t^2 < \cdots < t^{k-1} < t^k < \cdots < t^N = T_1$ with time step

$$\Delta t^k = t^k - t^{k-1}.$$

This can be a uniform time step or an adaptive one based on a Courant-Friedrich-Lewy (CFL) condition. We note that the level set method can stabily support uniform or adaptive time stepping, but we will for most cases use a uniform time step since our velocity model does not provably remain stable and convergent under adaptive time stepping (see [21]). We will discuss the CFL condition for the level set system in Section 2.3.2 which could be used for an adaptive time stepping strategy.

We will denote by superscript

$$\varphi^k(\mathbf{x}) := \varphi(t^k, \mathbf{x}) \tag{2.19}$$

the discrete in time solution at a specific time step, t^k. Likewise we will denote the mesh at our discrete times by a superscript,

$$\mathcal{T}^k := \mathcal{T}(t^k). \tag{2.20}$$

Thus the discrete in space and time level set solution will be denoted

$$\Phi^k \in \mathbb{Q}^p\left(\mathcal{T}^k\right). \tag{2.21}$$

2.3.2 *Adaptive time stepping*

We will apply the standard **Courant-Friedrichs-Lewy (CFL) condition** to our adaptive time stepping system. Given a constant $0 < c_{\text{CFL}}$, and a previous discrete velocity $\mathbf{V}^k$, we choose Δt^{k+1} in preparation for solving discrete velocity, $\mathbf{V}^{k+1}$, and discrete level set, Φ^{k+1}, such that

$$\Delta t^{k+1} \leq c_{\text{CFL}} \frac{\min_{K \in \mathcal{T}} (h_K/p)}{\|\mathbf{V}^k\|_{L^\infty(\Lambda)}} \tag{2.22}$$

where h_K is mesh diameter for cell K and p is the polynomial degree of the level set finite element space. When we use the Navier-Stokes system (Chapter 4) to specify the velocity, we will choose the finite element spaces for velocity and level set to have the same polynomial degree. We discuss the use of the explicit Runge-Kutta 3,3 (RK33) time integration scheme for the level set method in Section 2.5.3. Thus we will typically choose $p = 2$ so that the spatial L^2-error rate will match the RK33 time error rate. For stability purposes, is often needful to choose $c_{\text{CFL}} < 1$. A common value used is $c_{\text{CFL}} = 0.3$ although this needs to be adjusted based on how well the system performs under it. In our case, a value between $c_{\text{CFL}} = 0.2$ and $c_{\text{CFL}} = 0.4$ seem to be most reasonable and stable.

Remark. The SSP RK3 time integrator scheme is stable under this above CFL condition, however the SSP RK2 and first order Forward Euler schemes need a more stringent relationship between mesh size and time step size. For code testing purposes we may use these other schemes and so understanding that for forward Euler, we need $\Delta t = \mathcal{O}\left(\left(\frac{h_{\min}/p}{|\mathbf{v}|}\right)^2\right)$ and RK2 we need $\Delta t = \mathcal{O}\left(\left(\frac{h_{\min}/p}{|\mathbf{v}|}\right)^{4/3}\right)$ is important (here $\mathbf{v}$ is the representative velocity field). *end Remark.*

Remark. As pointed out before, the adaptive time stepping is fine for the level set system however it is not yet well understood for our pressure correction type splitting of the incompressible Navier-Stokes system as described in Section 4. Thus in practice we will use

a uniform time step that obeys the above discussed CFL conditions. In addition, for the various energy minimization models (described in Chapters 6 - 8), we will use a smeared Dirac delta function (described in Chapter 3) which will limit our accuracy in space to $\mathcal{O}(h_{\min}^{3/2})$ and also will introduce a first order splitting in time. It will be advantageous in these cases to make these consistent. That is, let $\Delta t \sim h_{\min}^{3/2}$ which is essentially replacing the CFL condition with a consistency condition. *end Remark.*

2.3.2.1 The CFL number and the Coupez reinitialization parameter, λ

Now that we have introduced the CFL condition, we can specify exactly how to calculate our scalar function, λ, from the reinitialization scheme introduced in Section 2.2. Recall that we defined $\lambda = \frac{\partial \tau}{\partial t}$. We will approximate this with our time discretization as

$$\lambda = \frac{\partial \tau}{\partial t} \approx \frac{\Delta \tau}{\Delta t}$$

where $\Delta\tau$ and Δt are the reinitialization and real time steps respectively. Adapting our CFL condition to both types of time steps,

$$CFL = \frac{\Delta t |\mathbf{v}|}{h/p} \qquad \text{and} \qquad CFL_{\text{reinit}} = \frac{\Delta \tau |\mathbf{U}|}{h/p}$$

we can compute (using $|\mathbf{U}| = \left|\text{sign}(\varphi)\frac{\nabla \varphi}{|\nabla \varphi|}\right| = 1$)

$$\lambda \approx \frac{\Delta \tau}{\Delta t} = \frac{CFL_{\text{reinit}} \cdot h}{|\mathbf{U}|} \frac{|\mathbf{v}|}{CFL \cdot h} = \frac{CFL_{\text{reinit}}}{CFL} \frac{|\mathbf{v}|}{|\mathbf{U}|} = c_\lambda \frac{|\mathbf{v}|}{|\mathbf{U}|} = c_\lambda |\mathbf{v}|$$

which is a unit less quantity because of the hidden reinitialization velocity $|\mathbf{U}| = 1$. We will let $c_\lambda := \frac{CFL_{\text{reinit}}}{CFL} > 0$ be a user given parameter representing the relative rates of reinitialization vs actual time flow. For instance, if $c_\lambda = 0.1$, it essentially says for every ten time steps we take one reinitialization step. While it is happening continuously, we get

$\lvert\mathbf{v}\rvert$ - Norm	Description
$\| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(K)}$	Local Max Norm on cell K (piecewise constant)
$\max\left(c_r, \| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(K)}\right)$	Thresholding of Local Max Norm (bounded away from 0)
$\| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(w(K))}$	Max over patch $w(K)$ around cell K (more intensive to compute)
$\| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(\Lambda)}$	Global Maximum of Velocity
$\frac{h_K}{\Delta t}$	Global Maximum of Allowed Numerical Velocity

Table 2.1: A list of various options for measuring magnitude of velocity in the level set reinitialization scheme. The reinitialization speed depends on the choice of norm of velocity.

roughly the same effect.

Finally, the velocities of our level set can vary quite a bit but since we are using the filters to threshold, we only really care about what is happening near the zero level set. We are free to choose what we mean by $|\mathbf{v}|$ in the above definition and there are a few options available. They range from local to global definitions and it is up to the user to choose the best approach for their specific problem. Ideally a localized solution will be best since we are scaling our reinitialization effort to the behavior of the velocity field, however if the velocity field locally goes to 0, the reinitialization is essentially turned off and in practice, problems can arise. Thus at times, adding in a global aspect can be convenient and lead to better behavior. The simplest solution is to threshold our local velocity size away from 0 with some user given parameter c_r. That is, use $|\mathbf{v}||_K = \max\left(c_r, \| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(K)}\right)$. However, it can also be good to give the reinitialization a global view of what is happening with the maximum global velocity. A list of a few options for the velocity is given in Table 2.1.

For most applications, we find that the Global Maximum

$$\lambda|_K := c_\lambda \| \lvert\mathbf{v}\rvert_{\ell^2} \|_{L^\infty(\Lambda)}.$$

and the Thresholding of Local Max Norm with $c_r = 0.2$,

$$\lambda|_K := c_\lambda \max\left(c_r, \| \, |\mathbf{v}|_{\ell^2} \, \|_{L^\infty(K)}\right)$$

behave roughly the same and seem to do better than the other approaches but again this can be very problem specific.

2.4 Level set stabilization

An efficient method for stabilizing a hyperbolic equation to minimize oscillations is to add a diffusion term with cellwise constant artificial viscosity, ν, to the level set equation. We choose the artificial viscosity term to apply larger diffusion where it is needed to smooth oscillations and smaller where it is not; the goal being to only apply smoothing where it is needed. To this end, we use the explicit entropy residual viscosity method analyzed in [22] of which we will give a brief overview in Section 2.4.1 for the general hyperbolic conservation law system (2.23).

$$\begin{cases} \frac{\partial y}{\partial t}(t, \mathbf{x}) + \nabla \cdot \mathbf{k}(y(t, \mathbf{x})) = g(t, \mathbf{x}), & (t, \mathbf{x}) \in (T_0, T_1] \times \Lambda \\ y\left(T_0, \mathbf{x}\right) = y_0(\mathbf{x}), & \mathbf{x} \in \Lambda, \end{cases} \tag{2.23}$$

with inflow boundary conditions, where $\mathbf{k} \in C^1\left(\mathbb{R}; \mathbb{R}\right)$, and $g \in C^1\left([T_0, T_1] \times \mathbb{R}^d; \mathbb{R}\right)$. Then in Section 2.4.2, we will discuss the application to our specific level set transport problem.

2.4.1 General explicit entropy viscosity method

To begin with, given the system of equations (2.23) we must choose an entropy - entropy flux pair $(E, \mathbf{F})$, where the **entropy**, $E(y) \in C^1\left(\mathbb{R}; \mathbb{R}\right)$, is a strictly convex function

and the **entropy flux**, $\mathbf{F}(y) \in C^1\left(\mathbb{R};\mathbb{R}^d\right)$, is related to E by

$$\mathbf{F}'(y) := E'(y)\mathbf{k}'(y) \qquad \text{or equivalently} \qquad \mathbf{F}(y) := \int E'(y)\mathbf{k}'(y)dy \tag{2.24}$$

For many sytems, there is such an entropy- entropy flux pair that naturally arrises from the physical problem description, however this is not always the case. The use of this entropy pair will give rise to physical solutions but in the case that no entropy pair is known, we can choose one that yields the desired properties for stabilization that we are looking to add.

Remark. To be clear, in the subsequent we are sweeping many of the technical parts of this problem "under the rug"; specifically, dealing with the boundary conditions and applying appropriate restrictions on the flux and entropy to guarantee that a solution, y, to equation (2.23) physically exists and is well behaved. For a more detailed discussion, see [22]. *end Remark.*

Now given our entropy solution, y, this entropy - entropy flux pair satisfies the inequality condition

$$\partial_t E(y) + \nabla \cdot \mathbf{F}(y) - E'(y)g \leq 0$$

with equality if $y \in C^1$. We define our **entropy residual**, $R(y)$ to be

$$R(y) = \partial_t E(y) + \nabla \cdot \mathbf{F}(y) - E'(y)g$$

When we proceed to the discretization, we will overload the definition by using a discrete finite element current solution Y_{new} and previous solution Y_{old} and time step τ, to approximating the partial time derivative, ∂_t, by a first order backward difference,

$$R\left(Y_{\text{new}}, Y_{\text{old}}, \tau\right) = \frac{E\left(Y_{\text{new}}\right) - E\left(Y_{\text{old}}\right)}{\tau} + \nabla \cdot \mathbf{F}\left(Y_{\text{new}}\right) - E'\left(Y_{\text{new}}\right) g. \tag{2.25}$$

Then our **entropy viscosity** is defined locally using the maximal entropy residual on each cell, K,

$$\nu_E(y)|_K = \frac{c_E \left(\frac{h_K}{p}\right)^2 \| \, |R(y)| \, \|_{L^\infty(K)}}{\|E(y) - \text{ave}\,(E(y))\|_{L^\infty(\Lambda)}} \tag{2.26}$$

or using the discrete notation,

$$\nu_E|_K (Y_{\text{new}}, Y_{\text{old}}, \tau) = \frac{c_E \left(\frac{h_K}{p}\right)^2 \| \, |R(Y_{\text{new}}, Y_{\text{old}}, \tau)| \, \|_{L^\infty(K)}}{\|E(Y_{\text{new}}) - \text{ave}\,(E(Y_{\text{new}}))\|_{L^\infty(\Lambda)}} \tag{2.27}$$

where $c_E > 0$ is a user defined coefficient and

$$\text{ave}\,(E(Y_{\text{new}})) = \frac{1}{|\Lambda|} \int_\Lambda E(Y_{\text{new}}) \, d\mathbf{x}$$

is the average entropy value over the region, Λ. Thus

$$\|E(Y_{\text{new}}) - \text{ave}\,(E(Y_{\text{new}}))\|_{L^\infty(\Lambda)}$$

is the maximal entropy deviation over Λ and scales the residual to remove units of entropy so that the resulting quantity has the correct units of viscosity.

We define the **linear viscosity**, ν_L, using the maximum local wave speed, $\beta_K = \| \, |\mathbf{k}'(y)|_{\ell^2} \, \|_{L^\infty(K)}$, as

$$\nu_L(y)|_K = c_L \left(\frac{h_K}{p}\right) \beta_K = c_L \left(\frac{h_K}{p}\right) \| \, |\mathbf{k}'(y)|_{\ell^2} \, \|_{L^\infty(K)} \tag{2.28}$$

where $c_L > 0$ is a user defined linear viscosity coefficient.

Remark. We will describe in Section 2.6 the method of choosing the entropy and linear viscosity coefficients, c_E and c_L. *end Remark.*

Now we can compute our piecewise constant artificial (stabilizing) viscosity, ν using

entropy viscosity, equation (2.26), and linear viscosity, equation (2.28) as the minimum of the two viscosities on each cell K

$$\nu(y)|_K = \min\left(\nu_E(y)|_K , \ \nu_L(y)|_K \right). \tag{2.29}$$

The discrete artificial viscosity is computed similarly using the discrete entropy viscosity, equation (2.27) and linear viscosity, equation (2.28) as

$$\nu\left(Y_{\text{new}}, Y_{\text{old}}, \tau\right)|_K = \min\left(\nu_E\left(Y_{\text{new}}, Y_{\text{old}}, \tau\right)|_K , \ \nu_L(Y_{\text{new}})|_K \right). \tag{2.30}$$

The key is that when the solution is smooth, the entropy residual is small and so entropy viscosity is selected and a higher order smoothing is applied, but when the solution loses smoothness due to shocks or large oscillations, the entropy residual becomes large and so the linear viscosity is the smaller term and we apply a first order smoothing. This attempts to maintain a consistent application of diffusion in relation to the smoothness of the solution.

Now, for $t \in (T_0, T_1]$ and given a Lagrange finite element approximation space $\mathbb{Q}^p\left(\mathcal{T}(t)\right)$, the semidiscrete Galerkin + artifical viscosity approximation for our hyperbolic conservation equation (2.23) is: For a.e. $t \in (T_0, T_1]$, find $Y(t) \in \mathbb{Q}^p\left(\mathcal{T}(t)\right)$ such that for all $V \in \mathbb{Q}^p\left(\mathcal{T}(t)\right)$,

$$\int_\Lambda \left(\partial_t Y + \nabla \cdot \mathbf{k}(Y)\right) V d\mathbf{x} + \sum_{K \in \mathcal{T}} \int_K \nu(Y)|_K \nabla Y \cdot \nabla V d\mathbf{x} = \int_\Lambda g V d\mathbf{x}.$$

Remark. This artificial viscosity can be applied explicitly or implicitly based on the problem at hand. We will apply it explicitly to be consistent with our explicit Runge Kutta time integration routine. A stability analysis of a single forward Euler step with this explicit artificial viscosity stabilization will show that the CFL number should also factor in the

viscosity when entropy viscosity is being applied since it is $\mathcal{O}(h^2)$. Numerically, this can be observed in that if we choose our entropy parameter, c_E, too large we see blowup of the solution instead of smoothing. *end Remark.*

2.4.2 Entropy viscosity stabilization for the level set method

Now that we have introduced the theory and method for a general hyperbolic equation, we will focus in on our level set method with reinitialization. Ideally, the entropy function, $E(y)$, is something with a physical meaning attached to the problem being solved, but in the case of the level set method, there is no recognized physical entropy function so we are free to choose a stand-in entropy function that gives us the behavior we are looking for, that is which keeps the normal direction of the level sets near the zero level set smooth and dampens oscillations far away from the zero levelset. It is useful to consider entropy functions that are convex and differentiable. For instance, we could choose our entropy to be polynomial

$$E(y) = y^s, \qquad E'(y) = sy^{s-1}$$

for s an even integer. It is common to choose a value of $s = 2$ or $s = 4$ but even higher values like $s = 20$ could be appropriate. However, when we use the hyperbolic tangent filters as described in Section 2.1 above, these seem to place not enough smoothing near the zero level set and instead spread out the viscosity a little too much so that instabilities in the solution are not smoothed away. Instead we choose to use a log-type entropy that seems to give a better distribution of viscosity to our system. Supposing that our level set has a horizontal asymptote and max(or min) at $C($ or $-C)$, we define

$$E(y) = \log\left((C-y)(C+y)\right), \qquad E'(y) = -\frac{2y}{(C-y)(C+y)}.$$

This would be perfect if our approximate scheme were maximum-preserving so that we were guaranteed to always stay between $-C$ and C. However, since ours is not a maximum preserving transport scheme, we take the first four terms of the Taylor series expansion around 0 to be our entropy function:

$$E(y) = \log(C^2) - \left(\frac{y}{C}\right)^2 - \frac{1}{2}\left(\frac{y}{C}\right)^4 - \frac{1}{3}\left(\frac{y}{C}\right)^6, \quad E'(y) = -\frac{2y}{C^2} - \frac{2y^3}{C^4} - \frac{2y^5}{C^6} \tag{2.31}$$

This gives a similar behavior to the log definition without the blowup at the max(min) values if they are exceeded.

In the case of our level set method, our flux and rhs sources are in fact linear and given by

$$\nabla \cdot \mathbf{k}(\varphi) = \mathbf{k}'(\varphi) \cdot \nabla\varphi = (\mathbf{v} + \lambda\mathbf{U}) \cdot \nabla\varphi \tag{2.32}$$

and

$$g(t, \mathbf{x}) = \lambda \operatorname{sign}(\varphi) S. \tag{2.33}$$

Thus,

$$\begin{aligned} \nabla \cdot \mathbf{F}(\varphi) &= \mathbf{F}'(\varphi) \cdot \nabla\varphi \\ &= E'(\varphi)\mathbf{k}'(\varphi) \cdot \nabla\varphi \\ &= E'(\varphi)\left(\mathbf{v} + \lambda\mathbf{U}\right) \cdot \nabla\varphi \end{aligned} \tag{2.34}$$

So, the **entropy residual**, $R(\varphi)$, for the level set method with reinitialization is given by

$$\begin{aligned} R(\varphi) &= \partial_t E(\varphi) + \nabla \cdot \mathbf{F}(\varphi) - E'(\varphi) g \\ &= \partial_t E(\varphi) + E'(\varphi)\left(\mathbf{v} + \lambda\mathbf{U}\right) \cdot \nabla\varphi - E'(\varphi)\left(\lambda \operatorname{sign}(\varphi) S(\varphi)\right) \end{aligned}$$

and our disrete entropy residual is

$$R(\varphi_{\text{new}}, \varphi_{\text{old}}, \Delta t) = \frac{E(\varphi_{\text{new}}) - E(\varphi_{\text{old}})}{\Delta t} + E'(\varphi_{\text{new}})(\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla \varphi_{\text{new}} - E'(\varphi_{\text{new}})(f + \lambda \operatorname{sign}(\varphi_{\text{new}}) S(\varphi)) \tag{2.35}$$

The linear viscosity is computed using the speed,

$$\beta_K = \| \, |\mathbf{v} + \lambda \mathbf{U}|_{\ell^2} \, \|_{L^\infty(K)}.$$

Then the level set artificial viscosity is computed from the descrete entropy residual and speed as in equations (2.27), (2.28) and (2.29). We will discuss the specific discrete algorithm with regards to our time integration scheme and our choice of explicit stabilization in Section 2.5.3.

2.5 Semi-discrete and fully-discrete formulations with stabilization

We have now described the continuous model with stabilization and have introduced the notation for space and time discretizations. We will present in Section 2.5.1, the semi-discrete formulation of the level set Galerkin system with stabilization. We will describe the use of the explicit Runge-Kutta time integration schemes in Section 2.5.2 to discretize time and then we will write down the fully discrete formulation in Section 2.5.3.

2.5.1 Semi-discrete formulation

Now that we have defined the discretization spaces (Section 2.3) and the stabilization scheme (Section 2.4) being used, our **semi-discrete Galerkin + stabilization** approximation to the level set method is:

For a.e. $t \in [T_0, T_1]$, find $\Phi(t) \in \mathbb{Q}^p(\mathcal{T}(t))$ such that for all $W \in \mathbb{Q}^p(\mathcal{T}(t))$,

$$\begin{aligned} \int_\Lambda \frac{\partial}{\partial t}\Phi W d\mathbf{x} + \int_\Lambda (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla \Phi W d\mathbf{x} + \sum_K \int_K \nu(\Phi)|_K \nabla \Phi \cdot \nabla W d\mathbf{x} \\ = \int_\Lambda \lambda \operatorname{sign}(\varphi) S(\Phi) W d\mathbf{x} \end{aligned} \tag{2.36}$$

and inflow boundary conditions $\Phi(t, \mathbf{x}) = \mathcal{I}(\varphi_{\text{inflow}}(t, \mathbf{x}))$ where $\mathbf{v}(t, \mathbf{x}) \cdot \boldsymbol{\nu}(\mathbf{x}) < 0$ on $\mathbf{x} \in \partial\Lambda$ and initial conditions $\Phi(T_0) = \mathcal{I}(\varphi_{\text{initial}})$. Here, $\mathcal{I}$ is a projection onto $\mathbb{Q}^p(\mathcal{T}(t))$.

2.5.2 Time integration schemes

We choose to implement the time integration using one of the family of explicit Runge-Kutta schemes. In particular, we choose from the schemes which have the (SSP) Strong Stability Preserving property (see Section 2.5.2.1). We will describe an algorithm for the explicit Runge-Kutta schemes using a Butcher Tableau in Section 2.5.2.2 and then using the so called Alpha-Beta ($\alpha - \beta$) scheme in Section 2.5.2.3. The SSP property reduces the $\alpha - \beta$ scheme to a convex combination of forward Euler steps. Using the Butcher Tableau approach would require us to lag the stabilization whereas the $\alpha - \beta$ scheme lets it be applied in real time on each Euler step. We therefore will primarily use the SSP RK3 in $\alpha - \beta$ form for our time integration scheme for the level set method.

2.5.2.1 Strong stability preserving property

As described in [23] and [24], when we say a time integration scheme has the **strong stability preserving (SSP) property**, we mean that given some norm $\|\cdot\|$ under which the first order explicit Euler scheme is strongly stable,

$$\|u^{k+1}\| \leq \|u^k\|$$

with a condition

$$\Delta t \leq \Delta t_{FE},$$

then this SSP scheme preserves that same stability for the higher order multistep scheme under a possibly more stringent condition

$$\Delta t \leq c\Delta t_{FE}.$$

In [23], it is shown that $c = 1$ for SSP RK2 and SSP RK3, so we can use the same time step in both cases that we would for the forward Euler scheme. The SSP schemes were originally developed for solving hyperbolic conservation laws and also have the total variation diminishing (TVD) property but the SSP property is in fact a stronger property that preserves any convex functional bound, like positivity, that might be satisfied by the forward Euler step.

Remark. Note that while it could be useful, our SSP-RK schemes are not built with the maximum preserving property. We are only exploiting the SSP property to be able to write the system as a sequence of forward Euler steps. There are cases however where having a maximum preserving property would be useful but this requires the use of flux correction techniques. For instance, when adding in more detailed parts of the biological cell's motility, one is led to include various chemical signals in the interior of the cell which need to be advected along with the cell itself. This could be cast as a mixture theory type problem where one would track concentrations of signals and require that the concentration of each component be non negative and sum to 1. Thus positivity preserving and maximum preserving properties would be necessary. See [25] and [26] for more details on how this could be done for a general transport conservation law. *end Remark.*

2.5.2.2 *Butcher tableau representation of explicit Runge-Kutta schemes*

The general explicit Runge-Kutta of order s solving the equation

$$y' = L(t, y)$$

is typically described by a **Butcher tableau**.

$$\begin{array}{c|cccccc} c_1 & & & & & & \\ c_2 & a_{21} & & & & & \\ c_3 & a_{31} & a_{32} & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s(s-1)} & & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s. \end{array}$$

We use the notation of Section 2.3 for our discrete time steps and consider for now continuous in space but discrete in time solutions $y^{k+1} := y(t^{k+1})$. Then the Runge-Kutta update is given by

$$y^{k+1} = y^k + \Delta t \sum_{m=1}^{s} b_m L_m$$

The intermediate steps, denoted by $y^{(1)}, \dots, y^{(s)}$ and intermediate derivatives denoted by $L_1, \dots, L_s$ are computed for $n = 1, \dots, s$ as

$$y^{(n)} = y^k + \Delta t \sum_{m=1}^{n-1} a_{nm} L_m$$

$$L_n = L\left(t^k + c_n \Delta t,\ y^{(n)}\right).$$

The Butcher tableau algorithm for the tableau set (A, b, c), is summarized by Algorithm 1. We have used the functions $apply\_inflow\_bc(t, y)$ and $apply\_intermediate\_inflow\_bc(t, y)$

which are defined in Algorithm 2 and 3 respectively. The intermediate inflow condition is only required if the boundary values are variable in time and provides the full sth order convergence rate in this case (see [27] and [28] for details).

Remark. Notice that for memory usage, we must store y^k and y^{k+1} and then the s intermediate right hand side vectors L_n and a temporary vector y_{tmp} representing the desired $y^{(n)}$ to compute L_n. More practically once we have solved for y^{k+1}, y^k can be deleted, so we really only need memory for $s+3$ vectors, however for convenience in our splittings we keep track of two previous states y^k and y^{k-1}. *end Remark.*

Remark. Technically speaking, our system is a level set transport pde with stabilization of the form

$$y_t + (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla y - \operatorname{div}(\nu \nabla y) = \lambda \operatorname{sign}(y) S(y).$$

If we collect all the non-time derivative terms to the right hand side and let $G(t, y)$ represent everything but the artificial viscosity term, then we have a transport + stabilization decomposition:

$$\begin{aligned} y_t &= -(\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla y + \lambda \operatorname{sign}(y) S(y) + \operatorname{div}(\nu \nabla y) \\ &= G(t, y) + \operatorname{div}(\nu \nabla y). \end{aligned}$$

We discretize in space using a Lagrange finite element basis and Galerkin formulation to obtain an ODE in time. Thus the explicit time integration update $y_{\text{tmp}} = y^k + \sum_{m=1}^{n-1} a_{nm} L_m$ on stage $n \in \{1, \ldots, s\}$ is actually in the form

$$MY^{\text{tmp}} = MY^k + \sum_{m=1}^{n-1} a_{nm}(G_m + S_m)$$

with capital vectors, Y^k, representing the coefficients vector for the basis functions at

time t^k and M, the corresponding mass matrix. Likewise, since we are using an explicit scheme, we obtain G_m and S_m as the right hand side vectors computed from $G(t, y)$ and the stabilization term. An equivalent formulation is obtained by multiplying through by M^{-1}

$$Y^{\text{tmp}} = Y^k + \sum_{m=1}^{n-1} a_{nm} M^{-1} \left(G_m + S_m \right).$$

And so by calling $L_n = M^{-1} \left(G_n + S_n \right)$ we are in the recognizable format for a vector ODE Runge-Kutta problem. Thus, for our s-stage explicit Runge-Kutta scheme, there are s solves of a mass matrix inversion which can be efficiently done using the conjugate gradient iterative method with a Jacobi preconditioner. *end Remark.*

Remark. The addition of the explicit artificial viscosity stabilization $\text{div}(\nu^{(n)} \nabla y^{(n)})$ on each stage $n \in 1, \ldots, s$ requires a little more attention. It turns out that one of the most efficient schemes for the Butcher Tableau algorithm is an "on the fly" approach where we skip applying it to the first stage, $\nu^{(1)} = 0$ and then the subsequent viscosities are lagged one stage of the algorithm. That is, for stage n we compute the viscosity $\nu^{(n)}$ as

$$\nu^{(n)} = \begin{cases} 0, & n = 1 \\ \nu \left(y^{(n-1)}, y^k, c_{n-1} \Delta t \right), & n > 1 \end{cases} \tag{2.37}$$

using the discrete entropy viscosity notation of equation (2.27). Alternatively, we could use $\nu^{(1)} = \nu(y^k, y^{k-1}, \Delta t^k)$, but this requires storing y^{k-1}. These terms are fairly simple to implement but the viscosity for $y^{(n)}$ is lagged (not based on $y^{(n)}$ but on $y^{(n-1)}$) and so if large shocks occur, they are not immediately smoothed out. We will see in Section 2.5.2.3 that using the $\alpha - \beta$ scheme allows us to apply this artificial viscosity in real time without the lag although the benefit is offset by an increase of s-extra mass matrix inversions per time step, effectively doubling the amount of solver time. However, the mass matrix has

constant condition number with respect to n_dofs and so using the conjugate gradient method with a simple Jacobi preconditioner reduces this to a small amount of extra work. *end Remark.*

Algorithm 1: Explicit RK(s) with Butcher tableau algorithm.

Data: y^k, the previous time step
bApplyIntermediateBC, a flag for whether to apply consistent boundary conditions if boundary values are changing in time
Result: y^{k+1}, the next time step

begin
  for $n = 1 \ldots s$ **do**
    $y_{\text{tmp}} = y^k$;
    for $m = 1 \ldots n-1$ **do**
      y_{tmp} += $\Delta t a_{nm} L_m$;
    end
    if *bApplyIntermediateBC* **then**
      apply_intermediate_inflow_bc($t^k + c_n \Delta t$, y_{tmp});
    end
    L_n = $L(t^k + c[n]\Delta t, y_{\text{tmp}})$;
  end
  y^{k+1} = y^k;
  for $n = 1 \ldots s$ **do**
    y^{k+1} += $b_n \Delta t L_n$
  end
  apply_inflow_bc(t^{k+1}, y^{k+1});
end

Remark. We have implemented the explicit strong stability preserving Runge-Kutta 2 and 3 methods for respectively second and third order time integration errors. The Butcher tableaus for the explicit SSP Runge-Kutta 2 (SSP RK2) and explicit SSP Runge-Kutta 3

Algorithm 2: apply_inflow_bc(t,y)

Data: t, the new time
y, the current solution at new time without boundary conditions applied
v, velocity field
$\nu(\mathbf{x})$, the outer normal unit vector for $\mathbf{x} \in \partial\Lambda$
y_{inflow}, The inflow function
Result: y, the solution at new time with boundary conditions applied

begin
  For each degree of freedom $\mathbf{x}_i \in \partial\Lambda$ **if** $\boldsymbol{v}(t, \boldsymbol{x}_i) \cdot \nu(\boldsymbol{x}_i) < 0$ **then**
    $y(\mathbf{x}_i) = y_{\text{inflow}}(t, \mathbf{x}_i)$.
  end
end

Algorithm 3: apply_intermediate_inflow_bc(t,y)

Data: i, the intermediate stage number
$t^i = t + c_i \Delta t$, the intermediate time
y, the current solution at $t + c_i \Delta t$ without boundary conditions applied
v, velocity field
$\nu(\mathbf{x})$, the outer normal unit vector for $\mathbf{x} \in \partial\Lambda$
y_{inflow}, The inflow function
Result: y, the solution at new time with boundary conditions applied

begin
  for *each degree of freedom* $\boldsymbol{x}_j \in \partial\Lambda$ **do**
    if $\boldsymbol{v}(t^i, \boldsymbol{x}_j) \cdot \nu(\boldsymbol{x}_j) < 0$ **then**
      if $i == 1$ **then**
        $y(\mathbf{x}_j) = y_{\text{inflow}}(t^i, \mathbf{x}_j)$;
      else if $i == 2$ **then**
        $y(\mathbf{x}_j) = y_{\text{inflow}}(t^i, \mathbf{x}_j) + c_i \Delta t y'_{\text{inflow}}(t^i, \mathbf{x}_j)$;
      else if $i == 3$ **then**
        $y(\mathbf{x}_j) = y_{\text{inflow}}(t^i, \mathbf{x}_j) + c_i \Delta t y'_{\text{inflow}}(t^i, \mathbf{x}_j) + \Delta t^2 \frac{a_{32}}{a_{21}} y''_{\text{inflow}}(t^i, \mathbf{x}_j)$;
    end
  end
end

(SSP RK3) schemes are

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

42

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ \hline & 1/6 & 1/6 & 2/3 \end{array}$$

end Remark.

2.5.2.3 Alpha-Beta representation of explicit SSP Runge-Kutta schemes

If our explicit s-stage Runge-Kutta scheme has the SSP property (Section 2.5.2.1), then another representation called the $\alpha - \beta$ representation can be more convenient. There may be more than one $\alpha - \beta$ representation for each Butcher tableau but there is at least one set of matrices α and β and vector c where

$$\alpha = \begin{pmatrix} \alpha_{11} & & & \\ \alpha_{21} & \alpha_{22} & & \\ \vdots & \vdots & \ddots & \\ \alpha_{s1} & \alpha_{s2} & \dots & \alpha_{ss} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_{11} & & & \\ \beta_{21} & \beta_{22} & & \\ \vdots & \vdots & \ddots & \\ \beta_{s1} & \beta_{s2} & \dots & \beta_{ss} \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{pmatrix}.$$

Then defining our intermediate steps with $y^{(0)} = y^k$ and for each $n = 1 \dots s$,

$$y^{(n)} = \sum_{m=1}^{n} \alpha_{nm} y^{(m-1)} + \beta_{nm} \Delta t L(t^k + c_m \Delta t, y^{(m-1)})$$

Then our next step is $y^{k+1} = y^{(s)}$. By an algebraic manipulation we can always reduce this scheme back to the standard Butcher tableau, but in the case of the explicit SSP Runge-Kutta schemes, it is possible to write α and β in such a way that β is diagonal (and so better represented as a vector)

$$\beta = \begin{pmatrix} \beta_{11} & & & \\ 0 & \beta_{22} & & \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & \beta_{ss} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_s \end{pmatrix}.$$

and where all of the α_{ij}'s are non negative. In this case the scheme is a sequence of forward Euler steps with the previous state a convex combination of the previous intermediate stage solutions, $y^{(m)}$, as follows: For the n-th ($0 < n \leq s$) stage, if we define

$$y_{\text{tmp}} = \sum_{m=1}^{n} \alpha_{nm} y^{(m-1)},$$

then

$$y^{(n)} = y_{\text{tmp}} + \beta_n \Delta t L(t^k + c_n \Delta t, y^{(n-1)})$$

is a forward Euler step with time-step size $\beta_n \Delta t$. Instead of storing the right hand sides, L_n's, as in the Butcher tableau algorithm, we store the intermediate states $y^{(n)}$ and compute the right hand side vectors L on the fly.

Remark. The level set transport pde with stabilization we are solving is of the form

$$y_t + (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla y - \operatorname{div}(\nu \nabla y) = \lambda \operatorname{sign}(y) S(y).$$

If we collect all the non-time derivative terms to the right hand side and let $G(t, y)$ represent everything there but the artificial viscosity term, then we have a transport + stabilization decomposition:

$$\begin{aligned} y_t &= -(\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla y + \lambda \operatorname{sign}(y) S(y) + \operatorname{div}(\nu \nabla y) \\ &= G(t, y) + \operatorname{div}(\nu \nabla y). \end{aligned}$$

end Remark.

Now, in this $\alpha - \beta$ scheme, we can apply the stabilizing viscosity, ν, in real time instead of lagging as in the Butcher tableau algorithm. We separate each Euler step of the scheme into three steps where we compute the non-stabilized, Galerkin solution, $y_g^{(n)}$,

then use it to construct the artificial viscosity, $\nu^{(n)}$, with equation (2.30) and then apply the explicit stabilization. We summarize the n-th Euler step with real-time stabilization in Algorithm 4.

Algorithm 4: The n-th stage Euler step of $\alpha - \beta$ RK(s) scheme with real-time stabilization.

Data: y_{tmp}, intermediate solution
$y^{(n-1)}$, solution at previous stage
Result: $y^{(n)}$, solution at next stage

1 $y_g^{(n)} = y_{\text{tmp}} + \beta_n \Delta t L\left(t^k + c_n \Delta t, y^{(n-1)}\right)$ `// Galerkin solution`

2 $\nu^{(n)} = \nu\left(y_g^{(n)}, y_{\text{tmp}}, \beta_n \Delta t\right)$ `// construct artificial viscosity`

3 $y^{(n)} = y_g^{(n)} + \beta_n \Delta t \operatorname{div}\left(\nu^{(n)} \nabla y_g^{(n)}\right)$ `// stabilize Galerkin solution`

Remark. Again noting that the spatial discretization of our PDE converts our solution into a vector ODE in time with a mass matrix term. When we use Algorithm 4 to apply stabilization in real time, we must invert the mass matrix twice per stage: once for the Galerkin solution, then again when applying the artificial viscosity stabilization. Thus for an s-stage Runge Kutta algorithm using the $\alpha - \beta$ formulation with real time artifical viscosity, we must invert the mass matrix $2s$ times. Since the mesh will not be refined between stages, it is possible to use a direct method and compute the LU decomposition once and apply forward and backward substitution for all $2s$ times. Alternatively, the mass matrix has constant condition number so that the conjugate gradient method with a Jacobi preconditioner is quick and has minimal complexity. *end Remark.*

Remark. All in all, we require storage room for y^k, the $s+1$ vectors $\{y^{(n)}\}_{n=0}^{s}$, y^{k+1}, y_{tmp} and a right hand side vector L_n. This makes $s+5$ vectors but in reality this can be reduced through removing redundancy (ie $y^{(0)} = y^k$ and $y^{k+1} = y^{(s)}$) to $s+3$ vectors

which is the same amount of memory as the Butcher tableau algorithm. However, the number of mass matrix inversions is doubled when we use the artificial viscosity since we must assemble $y_g^{(n)}$ with a mass matrix inversion and then to apply the viscosity again requires a mass matrix inversion on the explicit div-grad term. One might decide to try using a mass lumping procedure to reduce the effect, but this limits the errors to second order and does not seem to behave well in practice for this problem. However, through the use of the preconditioned conjugate gradient solver and with a Jacobi preconditioner the amount of time it takes to invert can be significantly reduced. Nevertheless, the mass matrix inversion solver still occupies between 10 and 20 percent of the level set module run time. The dominant operation is still assembly, in particular the assembly of the viscosity and div-grad terms. However, when compared to the computation of the velocity system, this increased cost is quite insignificant for the benefit it gives us. *end Remark.*

Remark. An $\alpha - \beta$ representation of the SSP RK2 method is

$$\alpha = \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \end{pmatrix}, \quad \beta = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

An $\alpha - \beta$ representation of the SSP RK3 method is

$$\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 3/4 & 1/4 & 0 \\ 1/3 & 0 & 2/3 \end{pmatrix}, \quad \beta = \begin{pmatrix} 1 \\ 1/4 \\ 2/3 \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ 1 \\ 1/2 \end{pmatrix}$$

end Remark.

2.5.3 Fully-discrete formulation of stabilized level set equation

When solving for the level set at time t^{k+1}, each of the s stages of the Runge-Kutta time integration lives at a certain time $t^{(n)} = t^k + c[n]\Delta t^{k+1}$. We will need an estimate of

Algorithm 5: Explicit RK(s) with $\alpha - \beta$ algorithm.

Data: y^k, the previous time step
Result: y^{k+1}, the next time step

```
begin
    y^(0) = y^k;
    for n = 1 ... s do
        y_tmp = 0;
        for m = 1 ... n do
            y_tmp += α_nm y^(m-1);
        end
        /* ********** Begin Euler Step ********** */
        y^(n) = y_tmp + β_n Δt L(t^k + c[n]Δt, y^(n-1));
        if (use artificial viscosity) then
            y^(n) = add_explicit_viscosity(y^(n), y_tmp, β_n Δt);
        end
        /* ********** End Euler Step ********** */
    end
    y^(k+1) = y^(s);
    apply_inflow_bc(t^(k+1), y^(k+1));
end
```

the velocity at each of these intermediate times. When solving for the velocity and level set components at time t^{k+1}, we solve first for velocity $\mathbf{v}^{k+1} = \mathbf{v}\left(\varphi^k, \varphi^{k-1}\right)$ and then we solve for the level set update $\varphi^{k+1} = \varphi\left(\mathbf{v}^{k+1}, \mathbf{v}^k\right)$. Thus at stage $n \in \{0, \ldots, s\}$ in the level set update, we must extrapolate the needed velocity $\mathbf{v}^{(n)}$ from the known velocities $\mathbf{v}^{k+1}$ and $\mathbf{v}^k$. This is done by setting

$$\mathbf{v}^{(n)} = \mathbf{v}\left(t^k + c[n]\Delta t^{k+1}\right) \approx \mathbf{v}^k + c[n]\left(\mathbf{v}^{k+1} - \mathbf{v}^k\right). \tag{2.38}$$

We likewise set the right hand side term $\mathbf{f}^{(n)} = \mathbf{f}(t^{(n)})$. The rest of the terms are dependent on whether we use the $\alpha - \beta$ or Butcher Tableau approaches.

We start with the initial solution $\Phi^0 \in \mathbb{Q}^p\left(\mathcal{T}^0\right)$, with $\Phi^0 = \mathcal{I}(\varphi_{\text{initial}})$. As we refine

the mesh over time, we must interpolate the previous solutions that will be used onto the current mesh. Because we are using hte explicit Runge-Kutta time integration, we only ever need the previous solution. Thus if we are solving for $\Phi^{k+1} \in \mathbb{Q}^p\left(\mathcal{T}^{k+1}\right)$, we will project $\Phi^k \in \mathbb{Q}^p\left(\mathcal{T}^k\right)$ onto $\mathbb{Q}^p\left(\mathcal{T}^k\right)$. To prevent the use of too much notation, we will not denote this new interpolated solution differently. In particular, we are a uniform fully refined mesh in a band around the zero level set, and since we are using a CFL condition, we our solution will not move too far in a single step. Thus the mesh has not changed on much of the band and the interpolation procedure only affects regions far from the zero level set where we are essentially constant. Thus, we can consider the use of the interpolant a non-issue.

Then given $\Phi^k \in \mathbb{Q}^p\left(\mathcal{T}^{k+1}\right)$, we compute $\Phi^{k+1} \in \mathbb{Q}^p\left(\mathcal{T}^{k+1}\right)$ as the final stage of the explicit Runge-Kutta s time integration of

$$\int_\Lambda \Phi_t W d\mathbf{x} = \int_\Lambda L\left(\Phi\right) W d\mathbf{x} - \sum_{K \in \mathcal{T}^{k+1}} \int_K \nu(\Phi)|_K \nabla\Phi \cdot \nabla W d\mathbf{x} \tag{2.39}$$

for all $W \in \mathbb{Q}^p\left(\mathcal{T}^{k+1}\right)$ and

$$L(\Phi, t) = -(\mathbf{V}(t) + \lambda\mathbf{U}(t)) \cdot \nabla\Phi(t) + \lambda \operatorname{sign}(\Phi(t)) S\left(\Phi(t)\right) \tag{2.40}$$

using the Butcher Tableau Algorithm 1 or the $\alpha - \beta$ Algorithm 5. The computation of the artificial viscosity will be different depending on the algorithm used. We discuss the artificial viscosity for the Butcher Tableau algorithm in Section 2.5.3.1 and for the $\alpha - \beta$ algorithm in Section 2.5.3.2.

2.5.3.1 Artificial viscosity using Butcher tableau algorithm

When we are using the Butcher Tableau format, we include the artificial viscosity term in the right hand side and do so by lagging the dependence on Φ. For the first stage, $\Phi^{(1)}$,

of solving for Φ^{k+1} at time t^{k+1}, there are two possibilities for the artificial viscosity. The first option is the so called "on-the-fly" method mention in [22] and seems to work quite well. It localized the information needed to only use data generated inside a Runge-Kutta step. That is, we don't need to store previous solutions. The second approach is a standard lagged method which uses the previous data to initialize the first stage viscosity. The rest of the residuals on each stage are computed in the standard lagged method. That is, to solve for solutions at time t^{k+1} and for stage $n \in \{1, \ldots, s\}$ we compute the entropy residual as

$$R^{(n)} = \begin{cases} 0, & \text{n=1, "on-the-fly" option} \\ R\left(\Phi^k, \Phi^{k-1}, \Delta t^k\right), & \text{n=1, standard option} \\ R\left(\Phi^{(n-1)}, \Phi^k, c[n-1]\Delta t^{k+1}\right) & \text{n>1.} \end{cases} \tag{2.41}$$

This is a lagged application of the entropy viscosity and so it is can be computed offline and included as a right hand side term so that there is only a single inversion of the mass matrix per stage of the Runge-Kutta step.

2.5.3.2 Artificial viscosity using $\alpha - \beta$ algorithm

We have already discussed the application of the artificial viscosity on an Euler step in Section 2.5.2.3 but add it here to be complete. For each stage, $n \in \{0, \ldots, s\}$, of the SSP Runge-Kutta algorithm with $\alpha - \beta$ representation we have a forward Euler step from Φ_{tmp} to $\Phi^{(n)}$ and time step $c[n]\Delta t^{k+1}$. In this case we follow Algorithm 4 where we compute the Galerkin without stabilization step, $\Phi^{(n)}_{\text{galerkin}}$, then compute the viscosity using the entropy viscosity residual

$$R^{(n)} = R\left(\Phi^{(n)}_{\text{galerkin}}, \Phi^{\text{tmp}}, c[n]\Delta t^{k+1}\right). \tag{2.42}$$

defined in equation (2.35). Then we apply the artificial viscosity stabilization to the Galerkin solution. This approach applies artificial viscosity in real time at the expense of an extra inversion of a mass matrix.

2.6 Choosing parameters for level set method

We briefly discuss techniques for choosing optimal parameters related to the transport, reinitialization, and artificial viscosity components of our level set problem. Finally we give a summary of the optimal parameters for our specific implementation.

2.6.1 Some implementation details

We implement all of these tools using the deal.II library in C++. The deal.II ([3]) finite element library supports adaptive mesh refinement in 2D and 3D with general quadrilateral and hexahedral cells. The corresponding finite elements on the mesh are constructed by tensor products of the standard 1D finite elements. Each triangulation, $\mathcal{T}^k$, is represented by a oct-tree forest data structure using the p4est library([5]) with the active cells as the leaves of the forest and each level of the tree consisting of equal number of refinements of the initial mesh. These libraries together support a completely distributed approach to parallelization of the triangulation with each processor owning a disjoint set of degrees of freedom on the mesh. The adaptive mesh refinement algorithm imposes a maximum of 2 level differences between neighboring cells so that the mesh refinement is well behaved and the rate of grading is also bounded. Finally, we use the Trilinos library([6]) which provides us with the iterative solvers and preconditions for solving our linear systems.

2.6.2 Optimal level set filter parameters

We choose the coefficients distance coefficient c_d to be wide enough for the approximate Dirac measure and Heaviside function as discussed in Chapter 3. In our case this means having it be on the order of $\omega = h^{3/4}$. We find $c_d = 1.4h^{3/4} = 1.4\omega$ is a good size

so that the Heaviside width parameter $(e_H = 1.0\omega)$ and the Dirac delta width parameter $(e_d = 1.0\omega)$ are fully contained in our distance function band which gives a better error approximation than when it overlaps the thresholding filter. In general we recommend to choose $e_d \leq e_H \leq c_d$ but find that setting $e_D = e_H$ works well in most cases. Finally, we let $c_f = 2h$ be the thresholding filter (hyperbolic tangent) width. The parameters are displayed together on the profile shape in Figure 2.7.

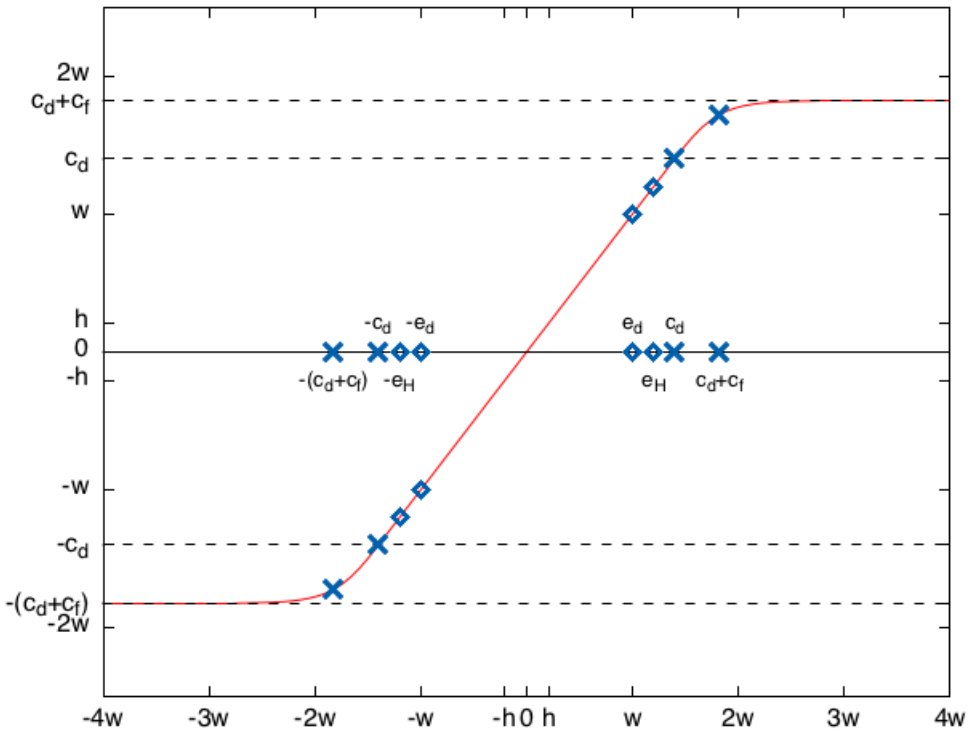


Figure 2.7: Level set parameters c_d and c_f and approximate Dirac delta and Heaviside function widths $e_d \leq e_H$.

2.6.3 Optimal CFL and spatial adaptivity parameters

Even though we are using uniform time steps, we still choose our time step so that our maximal computed CFL number is around 0.3. Sometimes it is necessary to run

the simulation a few times with a coarse grid to determine the maximal velocities and thereby choose the optimal time step. We adaptively refine so that we are fully refined on $|\varphi| \leq c_a(c_d + c_f)$ where $c_a \in (0, 1)$. Typically we will choose c_a to be $c_a = 0.9$ so that we are fully refined where the Dirac measure and Heaviside function are being used, and then a little beyond. Otherwise, we coarsen as quickly as is allowed by our program. We coarsen the mesh away from the band as quickly as is allowed so as to minimize the number of degrees of freedom. The idea is that the important quantities we are tracking live near the zero level set of our band. The rest needs to be resolved enough to capture the physics of the problem at hand.

Remark. It is often convenient to also provide an adaptive estimator for the velocity field to be fully resolved. If this is the case, we choose to prefer refinement over coarsening and always set the level set grading last with a check that we do not exceed the specified minimum and maximum level of refinement of our triangulation as stored by the p4est oct-tree forest. Then we can specify the maximal and minimal levels our mesh is allowed to achieve and thereby control the minimal and maximal mesh sizes. In the end, we always have the band of cells in a fully refined state and then possibly others based on the velocity field in various states of refinement. *end Remark.*

2.6.4 Optimal reinitialization parameters

When we choose the basic tanh thresholding cut-off function, then the Coupez method described above is enough to keep the profile in shape. We find that the reinitialization coefficient, $c_R = 0.01$, works great meaning 1 reinitialization step for every 100 transportation steps. However, when we use the distance function with tanh thresholding the coupez scheme is not in general able to keep up. We have tried c_R ranging from 0.01 all the way up to 1 but the larger c_R is, the more instabilities and distortion of the solution there is. We do use the entropy viscosity stabilization but were not able to find a good

combination of reinitialization and stabilization that preserved the proper advection of our solutions. Thus, when we use the wider distance with tanh thresholding profile shape, we need a different approach.

We can instead use the standard profile reinitialization scheme, equation (2.14) as a post processing step to the transport step. We can choose a certain number of reinitialization steps for each number of transport steps based on the needs of the problem. We can specify the integers R -the number of reinitialization steps- and T -the number of transport steps- before calling reinitialiation. These two user set integers fully specify the reinitialization process. In general, we start with them both set to 1, meaning that for each transport step we apply one reinitialization step. But it is necesary to tune these parameters based on the specific problem being solved. In the case of solving for the Willmore forces, we need the norm of the gradient $|\nabla\varphi|$ to be as smooth as possible since the curvature and Willmore quantities are related to first and second derivatives of $\nabla\varphi$. We find in this case that $R = 1$ reinitialization steps for every $T = 1$ transport step keeps φ smooth enough. On the other side of the spectrum, for the 2D Zalesak disk (as in Section 2.7.2) we choose $R = 1$ reinitialization step for every $T = 200$ transport steps.

As a final note, when we implement the reinitialization scheme, we also need a stabilization scheme for it. In this case, we find that the artificial linear viscosity is perfect and maintains the levelset set as smooth as we need it. Since there is no known physical entropy function for the reinitialization scheme, we must choose an entropy function to use. However, the choices we have come up with do not yet capture the needs of the reinitialization scheme. Thus the entropy viscosity scheme does not seem to work for our reinitialization problem. So in the case of using the distance with tanh thresholding profile, our aproach will be to transport with the entropy viscosity/ linear viscosity stabilization and then reinitialize with linear viscosity stabilization.

Note that the separation of reinitialization and transport does not significantly affect the

total run time since the largest use of time is still in constructing the velocity field. Using the explicit time integration for our transport schemes boil down to solving a mass matrix system multiple times. Using the conjugate gradient iterative method for this results in a constant number of steps to attain convergence. In experiments, we have not noted a significant increase in time using this approach.

2.6.5 Optimal artificial viscosity parameters

To tune the parameters for our artificial viscosity, we start with the linear viscosity and ignore the entropy viscosity term (ie set $c_E = \infty$). We start with linear viscosity high enough (usually around 1 is large enough to start) and work down until it is as small as can be and still dampen the oscillations or instabilities. Use a test case that is not too simple but on a somewhat coarse mesh. The idea is to be able to see if it the current parameter works fairly quickly, but give it enough of a challenge to need the viscosity.

Remark. We find the Zalesak disk rotation example (section 2.7.2) to be a decent problem for testing, but it allows us to set the entropy coefficient too small for two-phase flow problems. *end Remark.*

Once we have set the linear viscosity parameter, we now start with a large, but not too large, value for $c_E \approx 200$. If it is too large, the explicit artificial viscosity is now introducing oscillations instead of dampening them. Then we work it down until we no longer see smooth solutions. Then go back up a little into the smooth regime and you have your c_E value. We ended up with a linear viscosity coefficient of $c_L = 0.2$ and an entropy viscosity coefficient of $c_E = 0.8$ for two-phase flow problems, but c_E could be smaller when the velocity field is nicer, as in the case of Zalesak disk rotations.

2.6.6 Summary of optimal parameters for level set method

For each problem, there may be a slightly better set of parameters, be it smaller linear and/or entropy viscosity coefficient, or more or less reinitialization and smoothing being

applied. However, the parameters in Table 2.2 is a "best" general starting place and seems to work well for our two-phase flow problems discussed in Chapters 6-8. For problems with surface tension or when adding a willmore force along the surface, Γ, between the two-phases, we expect our solutions to have a certain amount of smoothness, so having a higher viscosity is not unreasonable, yet for pure transport problems we expect sharp corners to stay sharp so a smaller amount of smoothing seems to be more appropriate. The problem will dictate the fine tuning of the parameters, T and R.

c_d	c_f	c_{CFL}	c_a	c_L	c_E	T	R
$1.4w$	$2h$	0.35	0.9	0.2	0.8	1	1

Table 2.2: A list of optimal level set parameters for simulation of two-phase flow models. These are to be taken as the starting default values but should be modified on a case by case basis.

2.7 Numerical results and validation for stabilized level set method

We now give a few validation results for our implementation of the level set method. Unless otherwise noted, the $\alpha - \beta$ SSP Runge-Kutta 3 advection algorithm has been used. We test pure advection with the Runge-Kutta Transport test in Section 2.7.1, then add linear viscosity, then entropy viscosity, then the reinitialization scheme for the Zalesak disk rotation test in Section 2.7.2. The periodic single vortex test is conducted with transport + reinitialization + artificial viscosity in Section 2.7.3. Finally, we provide in Section 2.7.4 an analysis of mass/volume loss based on mesh size and reinitialization scheme, T and R.

2.7.1 Runge-Kutta transport tests

We test the Runge-Kutta convergence rates for transport without the artificial viscosity and reinitialization schemes. We let $\Lambda = (0,1)^2$ and $t \in [0, T_{\text{f}}]$ with the exact solution

$\varphi(\mathbf{x}, t) = 2 + \sin(\pi x t)\sin(\pi y t)$ moving under velocity field

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} \sin(t + x)\sin(t + y) \\ \cos(t + x)\cos(t + y) \end{bmatrix}. \tag{2.43}$$

The transports equation $\varphi_t + \mathbf{v} \cdot \nabla\varphi = f$ yields the right hand side to be

$$\begin{aligned} f(\mathbf{x}, t) =& \pi \left[y \sin(\pi x t)\cos(\pi y t) + x \cos(\pi x t)\sin(\pi y t) \right] \\ &+ t\pi \sin(\pi x t)\cos(t + x)\cos(\pi y t)\cos(t + y) \\ &+ t\pi \cos(\pi x t)\sin(t + x)\sin(\pi y t)\sin(t + y). \end{aligned} \tag{2.44}$$

We transport using a uniform spatial refinement of the mesh, $\mathcal{T}^k = \mathcal{T}$, the SSP RK3 time integration with a uniform time step and $\mathbb{Q}^3(\mathcal{T})$ spatial elements. The predicted convergence rates for $L^2(\Lambda)$ in space are thus 4th order and in time, 3rd order. We use an inflow Dirichlet boundary condition and simulate until $T_1 = 0.2$. We uniformly refine in both time and space in order to preserve the CFL number which is estimated to be

$$c_{\text{CFL}} = \left(\frac{\Delta t \max |V|}{(h/p)} \right) = 0.12 \tag{2.45}$$

The observed errors and convergence rates are given in Table 2.3.

2.7.2 Zalesak disk rotation

We let $\Lambda = (-1, 1)^2$ and simulate the rotation of the slotted disk around a circle and back to where it began. The slotted disk is a circle of radius $R = 0.25$ centered at $(c_x, c_y) = (0.5, 0.0)$ with the rectangle of horizontal half width $w = 0.0375$ and vertical half length $L = 0.25$ centered at $(s_x, s_y) = (0.5, 0.25)$. The signed distance function can be described as the set subtraction of the signed distance function of the circle minus the

cycle	Δt	n_steps	min h_K	n_dofs	$L^2(T_{\text{final}})$	rate	$\ell^\infty(L^2)$	rate	$\ell^2(L^2)$	rate
0	5.0000e-03	40	1.2500e-01	625	3.0407e-08	0.0000	3.0407e-08	0.0000	6.0253e-09	0.0000
1	2.5000e-03	80	6.2500e-02	2401	3.7918e-09	3.0035	3.7918e-09	3.0035	7.3992e-10	3.0256
2	1.2500e-03	160	3.1250e-02	9409	4.7366e-10	3.0009	4.7366e-10	3.0009	9.1693e-11	3.0125
3	6.2500e-04	320	1.5625e-02	37249	5.9188e-11	3.0005	5.9188e-11	3.0005	1.1411e-11	3.0064
4	3.1250e-04	640	7.8125e-03	148225	7.3868e-12	3.0023	7.3868e-12	3.0023	1.4210e-12	3.0055

Table 2.3: Transport of exact solution $\varphi(\mathbf{x}, t) = 2 + \sin(\pi x t)\sin(\pi y t)$ under velocity field $\mathbf{v}(\mathbf{x}, t) = [\sin(t+x)\sin(t+y); \cos(t+x)\cos(t+y)]$ using Runge-Kutta 3 time integration and $\mathbb{Q}^3(\mathcal{T})$ elements with $\mathcal{T}$ a uniformly refined mesh of $[0,1]^2$. The estimated CFL number $\left(\frac{\Delta t \max|\mathbf{v}|}{(h/p)}\right)$ is 0.12.

signed distance function of the rectangle in the following manner:

$$\begin{aligned} d(0,\mathbf{x}) &= \min\left\{d_{\text{circle}}(\mathbf{x}), d_{\text{rect}}(\mathbf{x})\right\} \\ &= \min\left\{R - \sqrt{(x-c_x)^2 + (y-c_y)^2}, \max\left\{|x-s_x| - w, |y-s_y| - L\right\}\right\} \end{aligned}$$

Note that this works because we use $d > 0$ to denote the inside of our region. We then apply the $\tanh$ thresholding filter

$$\varphi(0,\mathbf{x}) = \begin{cases} c_d + c_f \tanh\left(\frac{d(x)-c_d}{c_f}\right), & d(x) > c_d \\ d(\mathbf{x}), & |d(x)| < c_d \\ -c_d + c_f \tanh\left(\frac{d(x)+c_d}{c_f}\right), & d(x) < -c_d \end{cases} \tag{2.46}$$

to get our level set function with $c_d = 1.4h^{3/4}$ and $c_f = h$. We fully refine on $|d(\mathbf{x})| < 0.9\left(c_d + c_f\right)$ and adaptively coarsen away from there which yields approximately $4,000$ mesh cells as seen in the multiple time views of Figure 2.8.

The level set is advected using the divergence free velocity

$$\mathbf{v}(t,\mathbf{x}) = [-y; x] \tag{2.47}$$

which rotates counter clockwise completing a full rotation every 2π time units. A visual comparison of the solution after a complete rotation (seen in Figure 2.10) with the initial surface shows us our SSP RK3 transport with reinitialization and artificial viscosity method behaves fairly well.

Remark. Note that we can write down the exact solution at any time, $t \geq 0$ using the rotated coordinates

$$\tilde{x}(t) = (x - c_x)\cos(-t) - (y - c_y)\sin(-t) + c_x$$
$$\tilde{y}(t) = (x - c_x)\sin(-t) + (y - c_y)\cos(-t) + c_y$$

as

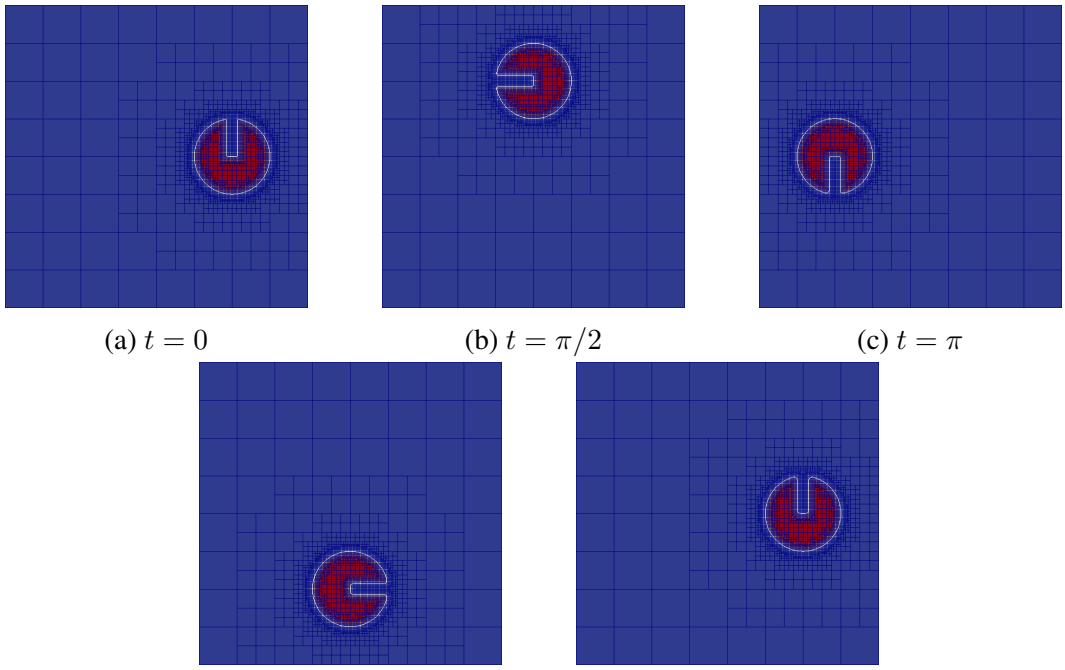
$$\varphi_{\text{exact}}(t, \mathbf{x}) = \varphi(0, \tilde{\mathbf{x}}(t)).$$

end Remark.

For a more complete picture, we show the result in Figure 2.9 of one complete rotation under pure advection, then adding in linear viscosity, then the artificial viscosity using linear and entropy viscosity but no reinitialization then finally the full scheme with reinitialization. Notice in Figure 2.9a that the SSP RK3 algorithm without viscosity or reinitialization maintains the zero level set extremely well, but the rest of the solution is highly distorted and so extra help is needed. We add linear viscosity and after a single rotation, seen in Figure 2.9b, the zero level set is highly distorted but the solution is smooth. We add in the higher order entropy viscosity (seen in Figure 2.9c) which keeps the solution smooth but we still have a little distortion.

Finally, if we add in the profile reinitialization routine with linear viscosity stabilization (seen in Figure 2.10) using $T = 200$ and $R = 1$ meaning one step of reinitialization for every 200 steps of transport, we obtain a smooth solution with minimal distortion.

The linear viscosity does round off the sharp corners, but everywhere else is maintained smooth.



(a) $t = 0$ (b) $t = \pi/2$ (c) $t = \pi$

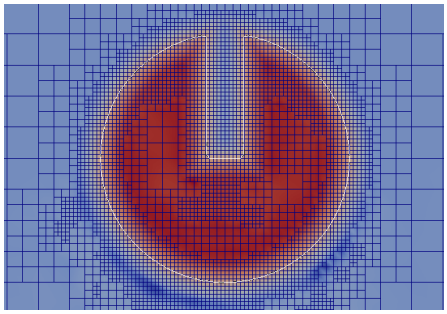
(d) $t = 3\pi/2$ (e) $t = 2\pi$

Figure 2.8: Zalesak disk rotating under divergence free velocity $\mathbf{v}(t, \mathbf{x}) = [-y; x]$ plotted at various times through rotation.

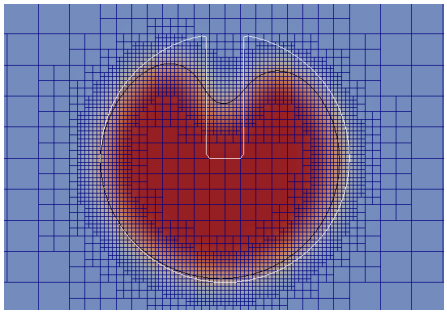
2.7.3 Periodic single vortex

We let $\Lambda = (0, 1)^2$ and run two tests based on a divergence free, $2T_c$-periodic velocity field which distorts and then returns to the original state. Our tests start with a circle of radius $R = 0.15$ centered at $(c_x, c_y) = (0.5, 0.75)$ with distance function

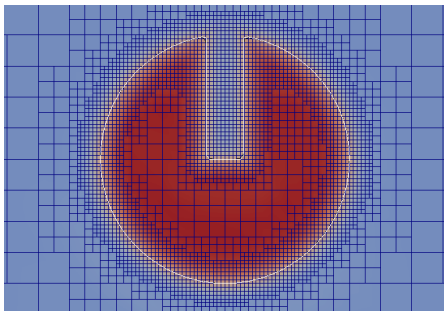
$$d(\mathbf{x}) = R - \sqrt{(x - c_x)^2 + (y - c_y)^2}. \tag{2.48}$$



(a) SSP RK3 only



(b) SSP RK3 + Linear Viscosity ($c_L = 0.1$)



(c) SSP RK3 + Artificial Viscosity ($c_L = 0.1$, $c_E = 0.2$)

Figure 2.9: Zalesak disk comparison after 1 rotation using various add-ons to the basic SSP RK3 algorithm. The white line is the exact solution and the black is the simulated solution.

The velocity field is

$$V(x,y) = \begin{bmatrix} -\sin(\pi x)^2 \sin(2\pi y) \cos(\pi t/T_c) \\ \sin(\pi y)^2 \sin(2\pi x) \cos(\pi t/T_c) \end{bmatrix} \tag{2.49}$$

In the first test, we apply a tanh filter

$$\varphi(\mathbf{x}) = 0.06 \tanh\left(\frac{d(\mathbf{x})}{0.06}\right) \tag{2.50}$$

to equation (2.48) which stays constant through the time and space refinements ensuring that the results are comparable as we do refine. We choose a filter width small enough so

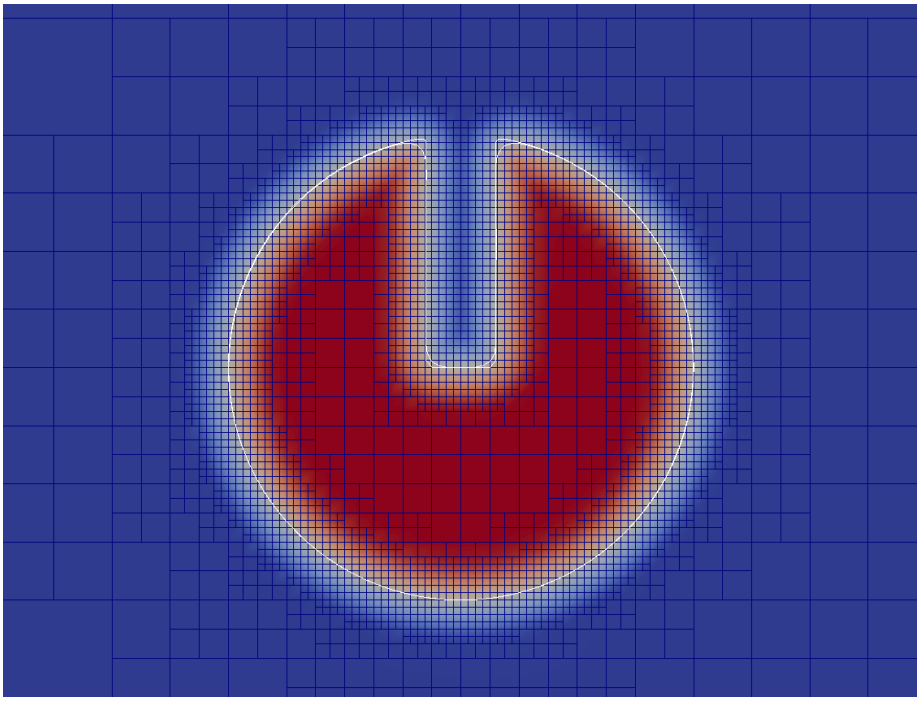


Figure 2.10: A comparison of Zalesak's disk after one rotation using the level set method with SSP RK3 + artificial viscosity and using a single $R = 1$ profile reinitialization time step (with linear viscosity) every $T = 200$ transport time steps.

that the inflow boundary conditions are constant. We let $T_c = 1$ in our velocity field so that a full periodic cycle happens every $2T_c = 2$ time units. We integrate on $t \in [0, 2]$ to obtain a full periodic cycle and compare our final state to the original state. We start with a uniform mesh and refine uniformly in both space and time for convergence analysis. We use the Runge-Kutta 3 time integration with elements in $\mathbb{Q}^3(\mathcal{T}_h)$ but do not use the reinitialization or artificial viscosity for this test. Using an estimated CFL number of 0.24 we obtain the results in Table 2.4

The second test is no longer a convergence analysis, but merely a comparison of initial state and final state when we add back in the reinitialization, the artificial viscosity stabilization and spatial adaptivity, ie the bells and whistles of our level set advection. We observe that under the velocity field, our profile shape loses its sharp shape although it does

cycle	Δt	n_steps	$\min h_K$	n_dofs	$L^2\left(T_{\text{final}}\right)$	rate
0	2.0000e-02	100	2.5000e-01	169	1.8542e-03	0.0000
1	1.0000e-02	200	1.2500e-01	625	1.9824e-04	3.2255
2	5.0000e-03	400	6.2500e-02	2401	1.4190e-05	3.8043
3	2.5000e-03	800	3.1250e-02	9409	1.0445e-06	3.7640

Table 2.4: Transport of a circle (2.50) under single vortex velocity field equation (2.49) with $T_c = 1$ using Runge-Kutta 3 time integration and $\mathbb{Q}^3\left(\mathcal{T}_h\right)$ elements with $\mathcal{T}_h$ a uniformly refined mesh of $[0,1]^2$ and uniform time steps for $t \in [0, T_{\text{final}} = 2]$. The estimated CFL number $\left(\frac{\Delta t \max |V|}{(h/p)}\right)$ is 0.24.

return the zero level set back to it's initial condition. We can observe the final state after a full period in Figure 2.11. The quantity, $\frac{\varphi(\mathbf{x}) - \varphi_{\text{exact}}(\mathbf{x})}{\max |v p_{\text{exact}}|}$ has been plotted to see how well the reinitialization and viscosity system handles the large deformation while maintaining consistent mass. We use the SSP RK3 time integration scheme with artificial entropy viscosity and the reinitialization pattern with linear viscosity of $T = 5$ and $R = 1$; meaning one reinitialization step for every 5 transport steps.

The profile shape is maintained through the simulation and the final state is almost exactly the initial state.

2.7.4 Analysis of volume preserving properties

Given a flow under a divergence free velocity field, the level set method should preserve the volume(mass) of the interior of Γ. However, we are not treating the level set method as a conservation law but merely as an advection system with an artificial viscosity for stabilization. We are not currently building any explicit volume(mass) conservation properties into our system. In addition, the reinitialization step with linear viscosity tends to smooth the solution (this is desirable) but this as well leads to volume(mass) loss. To this end, we present a series of numerical tests which give evidence that the relative volume(mass) loss scales linearly with the amount of reinitialization being applied and also

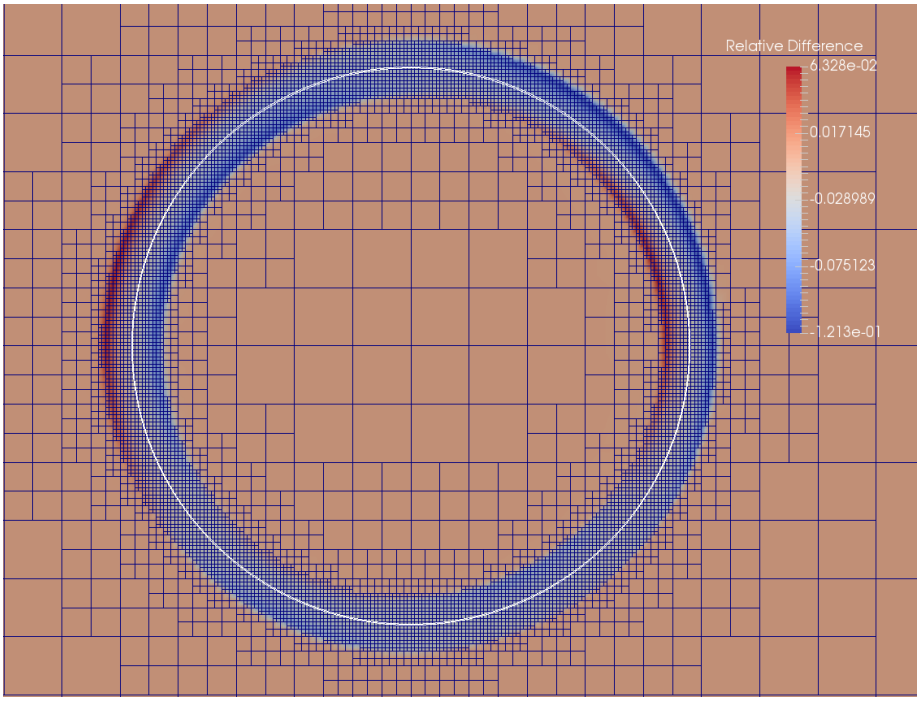
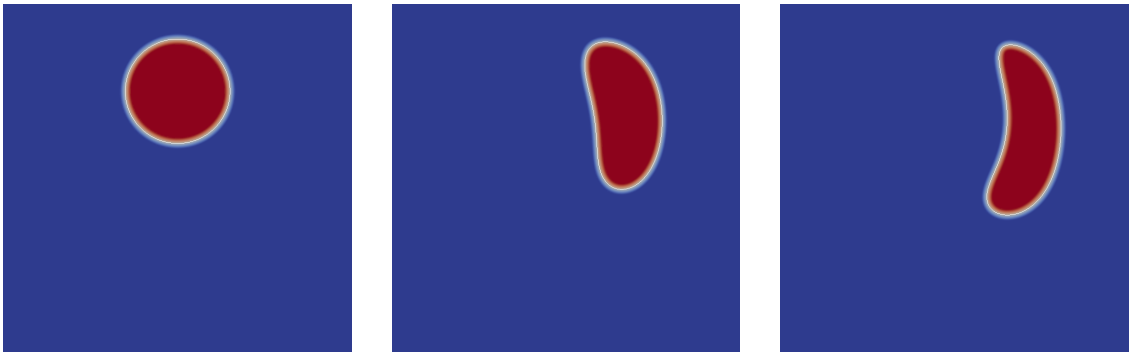


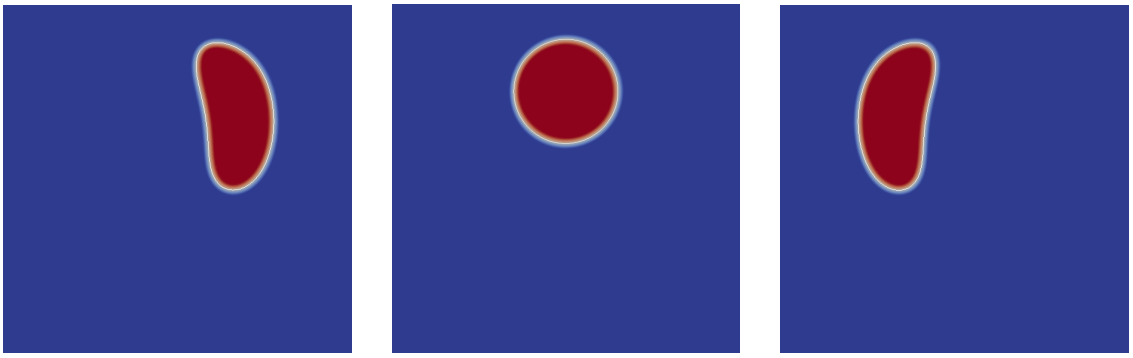
Figure 2.11: The quantity, $\frac{\varphi(\mathbf{x})-\varphi_{\text{exact}}(\mathbf{x})}{\max|\varphi_{\text{exact}}|}$ at time $t = 2$ of the single vortex system after a single period using SSP RK3 + artificial viscosity and $R = 1$ reinitialization steps for every $T = 5$ transport steps.

scales linearly with the mesh size (and time step size under constant CFL condition). Thus we could choose our mesh size and the amount of reinitialization so as to balance the relative volume(mass) loss with the smoothness properties we need.

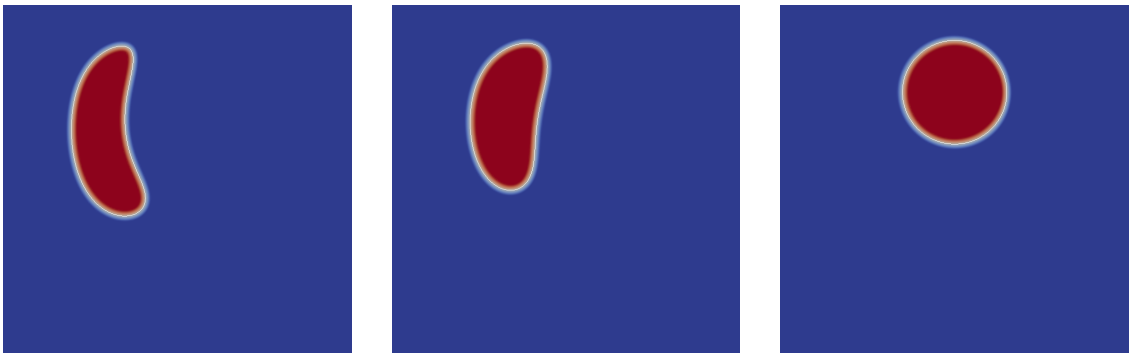
Our first experiment is solving a 2D rising bubble problem with surface tension (see Section 6.3 for full details of experimental setup). We perform the experiment twice, once with a small surface tension coefficient ($\sigma_{st} = 1.96$) and the other time with a medium sized surface tension coefficient ($\sigma_{st} = 24.5$). The small coefficient results in a large deformation of Γ whereas the medium coefficient results in much less deformation but a faster rise velocity. We evolve the system from $t = 0$ to $t = 3$ and compute the mass of the interior of Γ at times $t = 0$, $t = 1.5$ and $t = 3$ using a smoothed Heaviside function (equation (4.11)). We fix the mesh and timestep size and vary the ratio of transport (T)



(a) Time $t = 0.0$ (b) Time $t = 0.25$ (c) Time $t = 0.5$



(d) Time $t = 0.75$ (e) Time $t = 1.0$ (f) Time $t = 1.25$



(g) Time $t = 1.5$ (h) Time $t = 1.75$ (i) Time $t = 2.0$

Figure 2.12: Single vortex solution at various times using SSP RK3 + artificial viscosity with $R = 1$ reinitialization steps for every $T = 5$ transport steps.

steps to reinitialization steps (R). Table 2.5 displays the results of this experiment. In particular, we observe a linear scaling in the amount of relative mass loss with respect to the amount of reinitialization (and artificial linear viscosity) being applied.

T	R	σ_{st}	$V(t=0)$	$V(t=1.5)$	(%)	$V(t=3.0)$	(%)
01	10	24.5	0.196406	0.167626	-14.7%	0.138848	-29.3%
01	01	24.5	0.196406	0.193525	-1.47%	0.190646	-2.93%
02	01	24.5	0.196406	0.194972	-0.73%	0.193543	-1.45%
05	02	24.5	0.196406	0.195259	-0.58%	0.194119	-1.16%
05	01	24.5	0.196406	0.195842	-0.28%	0.195285	-0.57%
01	10	1.96	0.196407	0.167681	-14.6%	0.138889	-29.3%
01	01	1.96	0.196407	0.193575	-1.44%	0.190692	-2.91%
02	01	1.96	0.196407	0.195021	-0.71%	0.193588	-1.44%
05	02	1.96	0.196407	0.195309	-0.56%	0.194163	-1.14%
05	01	1.96	0.196407	0.195892	-0.26%	0.195331	-0.55%

Table 2.5: Estimates of volume(mass) loss for a 2D rising bubble with surface tension with $\Delta t = 0.0005$ and $h_{\min} = 0.00390625$ which corresponds to a CFL condition of $CFL = 0.07$. We choose the Δt_{reinit} to satisfy a reinitialization CFL of $CFL_{reinit} = 0.2$. We vary the number of transport (T) steps to number of reinitialization (R) steps and observe a linear scaling in relative volume loss, computed as $(V(t) - V(0))/V(0)$ where $V(t) = \int_\Lambda H_\varepsilon(\varphi(t, \mathbf{x}))d\mathbf{x}$ and $H_\varepsilon(\varphi)$, defined in equation (4.11) is an approximate indicator function for Ω.

The second experiment now focuses on the medium surface tension coefficient experiment. We choose two common ratios of transport (T) steps to reinitialization (R) steps and do a series of time and space refinements that preserves the CFL condition. We record the volume(mass) at times $t = 0$, $t = -1.5$ and $t = 3.0$. The results are displayed in Table 2.6 where we see evidence of linear scaling in the amount of volume loss with the minimal mesh size.

Δt	$h_{\min}$	T	R	$V(t=0)$	$V(t=1.5)$	(%)	$V(t=3.0)$	(%)
0.0020	0.0078125	01	01	0.196496	0.193371	-1.59%	0.190306	-3.15%
0.0010	0.00390625	01	01	0.1964	0.194776	-0.83%	0.19318	-1.64%
0.0005	0.001953125	01	01	0.196367	0.195522	-0.43%	0.194682	-0.86%
0.0020	0.0078125	02	01	0.196496	0.194803	-0.86%	0.193153	-1.70%
0.0010	0.00390625	02	01	0.1964	0.195498	-0.46%	0.194629	-0.90%
0.0005	0.001953125	02	01	0.196367	0.195884	-0.25%	0.195411	-0.49%

Table 2.6: Estimates of volume(mass) loss for a 2D rising bubble with medium valued ($\sigma_{st} = 24.5$) surface tension coefficient. We fix the number of transport (T) steps to reinitialization steps (R) for two standard usage cases and vary the timestep and minimal mesh size under a $CFL = 0.24$ condition. We observe a linear scaling in relative volume loss, computed as $(V(t) - V(0))/V(0)$ where $V(t) = \int_\Lambda H_\varepsilon(\varphi(t, \mathbf{x}))d\mathbf{x}$ and $H_\varepsilon(\varphi)$, defined in equation (4.11) is an approximate indicator function for Ω.

3. APPROXIMATIONS TO DIRAC DELTA FUNCTION

Given a level set function, $d : \mathbb{R}^n \to \mathbb{R}$ which is a signed distance function implicitly describing a smooth closed hypersurface Γ in $\mathbb{R}^n$, we need to be able to integrate along Γ, the zero level set of $d(\mathbf{x})$. There are many approaches to integrating along Γ. We will discuss some of the approaches in the literature in Section 3.1 and then introduce our approach in Section 3.1.1.

We have two criteria for the choice of which approach to use to integrate along Γ. The first is that it must be provably convergent. The second is that it must be reasonably simple to implement and use a standard quadrature. In fact, our implementation uses the deal.II library ([3]) which only supports quadrilaterals (in 2D) or hexahedra (in 3D). This implies that the quadrature we will allow are quadrature defined on such an element. We have found there are many 2nd order or higher methods for triangular meshes but they rarely if ever generalize to quadrilateral meshes. We will enumerate a few of the approaches that are currently available and then discuss why we chose the method that we have chosen.

There different approaches can loosely be lumped into three groups. The first consists of those approaches that directly extract the surface Γ and then integrate along it. The second group builds a quadrature scheme on each cell that is conforming to Γ in some manner and the third approximates the Dirac delta function, $\delta_\Gamma(\mathbf{x})$. For instance one might use a one dimensional smeared Dirac delta function $\delta_\varepsilon(t) : \mathbb{R} \to \mathbb{R}$ with support on $[-\varepsilon, \varepsilon]$ to approximate the Dirac delta function of the set $\{0\}$ and use the signed distance function to extend this to Γ. This will in theory work since Γ is the set of points such that $d(\mathbf{x}) = 0$. Thus for some smooth function $f : \Gamma \to \mathbb{R}$ with smooth extension to the $\varepsilon-$band around

Γ which is contained in a closed region $\Lambda \subset \mathbb{R}$,

$$\int_\Gamma f(\mathbf{x})d\mathbf{x} = \int_\Lambda f(\mathbf{x})\delta_\Gamma(\mathbf{x})d\mathbf{x} \approx \int_\Lambda f(\mathbf{x})\delta_\varepsilon(d(\mathbf{x}))|\nabla d(\mathbf{x})|d\mathbf{x}.$$

Notice that we have used the same notation for f and its extension. The $|\nabla d(\mathbf{x})|$ term is added to scale properly with the use of $d(\mathbf{x})$, but $|\nabla d(\mathbf{x})| = 1$ so in most cases it can be dropped. We will include it in the subsequent only when using the non-exact distance function or when it is necessary for clarity. Often we will choose

$$\delta_\varepsilon(t) = \begin{cases} \frac{1}{\varepsilon}\psi(\frac{t}{\varepsilon}), & |t| \leq \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

for some kernel function $\psi : [-1, 1] \to \mathbb{R}$. When the domain of integration, Λ, is partitioned, $\mathcal{T}_h$, into quadrilaterals (2D) or hexagonal (3D) elements, the resulting integral can be performed using a standard composite quadrature rule with positive weights, for instance using the Gauss quadrature. This is the approach we will follow but the choice of ψ and ε requires some care as will be seen subsequently. We will prove in Section 3.3 the following theorem:

Theorem 3.3.1. *Let $\ell \geq 1$ and assume $d \in W^{\ell+1,\infty}(\Lambda)$ is the signed distance function implicitly defining a hypersurface, $\Gamma \subset \Lambda$. Let $(\widehat{T}, \mathcal{Q}^\ell(\widehat{T}), \Sigma)$ be the Lagrange finite element triple defined with the reference element, $\widehat{T}$, as the unit square (2D) or unit cube (3D) and $\mathcal{Q}^\ell(\widehat{T}) := \{\Pi_{i=1}^{dim} p_i(\boldsymbol{x}_i) \mid p_i(t) \in \mathcal{P}^\ell([0, 1])\}$, the space of product of polynomials of degree ℓ in each dimension. We let $\{\mathcal{T}_h\}_{h>0}$ be a sequence of partitions of Λ made of quadrilateral (2D) or hexagonal (3D) elements of size h and*

$d_h \in \mathcal{Q}^\ell(\mathcal{T}_h) := \{p(\boldsymbol{x}) \in C(\Lambda) \mid \widehat{p|_T} \in \mathcal{Q}^\ell(\widehat{T})\}$ *satisfying*

$$\|d - d_h\|_{L^\infty(\Lambda)} + h\|\nabla(d - d_h)\|_{L^\infty(\Lambda)} \leq ch^{\ell+1}\|d\|_{W^{\ell+1,\infty}(\Lambda)},$$

for some $c > 0$ *only dependent on* Λ. *Let* $\psi(t) = \frac{693}{512}(1-t^2)^5$, $\varepsilon = ch^{3/4}$ *and* B_ε *and* $B_{h,\varepsilon}$ *to be defined later as the support of* $\delta_\varepsilon(d(\boldsymbol{x})$ *and* $\delta_\varepsilon(d_h(\boldsymbol{x}))$ *repectively. Given* $f \in W^{6,\infty}(B_\varepsilon \cup B_{h,\varepsilon})$ *and a composite quadrature rule,* $Quad_\Lambda(\cdot)$ *with positive weights exact for* $\mathcal{Q}^5(\widehat{T})$, *then the error*

$$E = \left| \int_\Gamma f(\boldsymbol{x})d\boldsymbol{x} - Quad_\Lambda\left(f(x)\delta_\varepsilon(d_h)|\nabla d_h|\right) \right|$$

satisfies

$$E \leq c_1 h^{3/2}\|f\|_{W^{6,\infty}(\Omega_\varepsilon)} + c_2 h^\ell \|f\|_{L^\infty(\Omega_\varepsilon)},$$

where c_1 *and* c_2 *are constants only dependent on* Λ.

In [29], Tornberg provides the convergence rates of E for a family of polynomials, $\psi^{m,k}(t)$ defined in Section 3.1.1 (of which we will use $\psi = \psi^{1,4}$) but the specifics of what requirements were needed on the quadrature rule and what regularity was needed for the function $f(\mathbf{x})$ were not given in her proof. Thus we provide an alternate proof with all the details we need below. This proof takes up most of the rest of this chapter. The notation required is in Section 3.2 and then the proof is broken up into pieces through Section 3.3. Once the theorem is proved, we will give some numerical results in Section 3.4 validating our findings.

3.1 A review of methods for integrating along an implicitly defined curve

There are many approaches to integrating along a curve Γ using an unfitted bulk mesh, $\mathcal{T}_h$. If a parameterization can be achieved, then it is simple enough to integrate along the

parameterization, but in our case Γ is implicitly defined so it is difficult to write down an explicit parameterization. Another approach is to embed a mesh along Γ and then as Γ is transported, we also transport the mesh. This leads to the well studied Immersed Boundary methods as in [11]. Another approach is more direct and takes advantage of the discretization of our distance function on a triangular mesh: Given a triangulation, $\mathcal{T}_h$, composed of triangles in 2D or tetrahedra in 3D and a discrete approximation to the distance function, $d_h(\mathbf{x}) \in P^1(\mathcal{T}_h)$, which is continuous and piecewise linear on this mesh, then the approximate surface $\Gamma_h := \{\mathbf{x} \in \mathbb{R}^n | d_h(\mathbf{x}) = 0\}$ can be simply extracted from the approximate distance function. In fact Γ_h cuts through the cells which change sign of $d_h(\mathbf{x})$ on their vertices and is linear on that cell, since for example, $d_h|_T = a + bx + cy$. So $\Gamma_h|_T = \{\mathbf{x} \mid a + bx + cy = 0\}$ is linear. The zero levelset, Γ_h is found by computing where on each edge the sign changes, of which there are always 2(in 2D) or 3(in 3D) points. Then we connect these points to obtain a line (in 2D) or a plane (in 3D). Thus we can loop through all cells, T, check if Γ_h is in that cell and if so, extract $\Gamma_h|_T$ which can be integrated over directly. This does not however work with quadrilateral or hexahedral elements where $d_h \in \mathbb{Q}^1(\mathcal{T}_h)$ since, for example in 2D, $d_h|_T = (a + bx)(c + dy)$. Thus $\Gamma_h|_T = \{\mathbf{x} \mid (a + bx)(c + dy) = 0\}$ is a rational function. The approach with triangular mesh elements can be generalized to higher order finite elements, $d_h \in P^k(\mathcal{T}_h)$ as in [30] but again does not seem to extend to rectangular meshes.

There are some approaches using a finite difference stencil that give second order convergence as in [31], and [32] with their respecitive proofs in [33] and [34]. These only work with uniform rectangular meshes and do not have clear generalizations to finite element discretizations.

[35] provides a nice moment fitting approach that constructs a bulk quadrature scheme on each cell that intersects Γ using higher order moments for any type of polygonal mesh. It turns out to be a linear system in choosing the weights but is nonlinear in choosing the

quadrature nodes so is only practical if the resulting quadrature rules can be reused many times. As Γ will be evolving in time, this is not the case. In addition, the resulting weights are not guaranteed to be positive so that stability of our finite element matrix systems cannot be guaranteed.

Another scheme which does work for higher order convergence rates on hyper-rectangles is the method of Saye in [36]. It constructs a new quadrature scheme on each cell that captures the integration region desired. The only down side is the complexity of implementation. It may however be justified in the higher order convergence rates achieved and may be an option for future use in this area.

The quadrature schemes of Wen in [37], [38], [39] and [40] achieve higher order convergence rates in 2 and 3 dimensions on rectangular meshes but are more complicated than we want to implement at this time. Again a more detailed comparison between this approach and the method of Saye in [36] would need to be done if a higher order method via localized quadratures were to be implemented.

Now, it has been shown in [41] that naively using a smeared Dirac delta function may not converge. They propose some fixes including to make the width variable based on how Γ cuts through the cell relative to the coordinate axes. This is a simple fix and gives reasonable convergence rates between first and second order but there is no proof of this. another method proposed has a proof of second order convergence in 2D but is not simple to implement and the extension to 3D is even more complicated. Another approach is given in [42] to use an approximate Dirac function that does not have compact support like the Gaussian or hyperbolic secant which is then cut off after a certain width based on the desired accuracy. The resulting support widths are much larger than we want in practice.

3.1.1 A family of Dirac delta functions

We will go back to an early suggestion by Tornberg in [29] where she presents a family of one dimensional polynomials $\psi^{m,k}(t)$ with support on $[-1,1]$ that can be used in the smeared Dirac delta, $\delta_\varepsilon(d(\mathbf{x}))$, per the formulation described above. The parameter $m \in \mathbb{N}$, represents the number of moments that the function satisfies where the n-th moment is defined as

$$M_n(\psi) := \int_{-\infty}^{\infty} t^n \psi(t) dt = \int_{-1}^{1} t^n \psi(t) dt, \tag{3.1}$$

for $n \geq 0$. We say that m moments are satisfied by ψ if

$$M_0(\psi) = 1,\ M_1(\psi) = 0,\ \ldots,\ M_m(\psi) = 0.$$

The parameter $k > 0$ represents the smoothness: $\psi^{m,k} \in C^k([-1,1])$. We will choose our ε-band to scale with the meshsize, h, that is $\varepsilon \sim h^\beta$ with $\beta \in (0,1]$. We would like to always have $\beta = 1$ so that the band is always the same width with regards to the number of mesh elements in the normal direction, but this does not always give the desired convergence rates. In the end, β is chosen to maximize the error convergence rate and will be based on the parameters m and k.

The function, $\psi^{m,k}(t)$, is the minimal degree polynomial that satisfies the desired moments and continuity conditions. Tornberg proves there is a unique minimal poynomial and provides methods for finding them.

The function we will choose for our simulations is

$$\psi^{1,4}(t) = \frac{693}{512}(1-t^2)^5. \tag{3.2}$$

which is the minimal degree polynomial defined on $[-1,1]$ that satisfies $M_0(\psi^{1,4}) = 1$ and $M_1(\psi^{1,4}) = 0$ and is continuous of degree $k = 4$, that is $\psi^{1,4} \in C^4(\mathbb{R})$. To obtain the convergence rate of $h^{3/2}$ with this approximate delta function, we choose $\varepsilon \sim h^{3/4}$. This is close to scaling with h and gives a reasonable convergence rate that is better than first order although not quite the desirable 2nd order.

Remark. The function $\psi^{1,4}$ is also non negative which seems to be a desirable property for these integrals in practice. Those polynomials, $\psi^{m,k}$ which have positive and negative regions do not seem to perform as well numerically as the ones which stay non negative. *end Remark.*

3.2 Notations

Suppose we have a smooth closed hypersurface, $\Gamma = \partial\Omega$ with $\Omega \subset \Lambda \subset \mathbb{R}^d$ as in Figure 3.1 implicitly defined through a distance function, $d(\mathbf{x})$, ie

$$\Gamma = \{\mathbf{x} \in \Lambda \mid d(\mathbf{x}) = 0\}. \tag{3.3}$$

Let $(\widehat{T}, \mathcal{Q}^\ell(\widehat{T}), \Sigma)$ be a Lagrange finite element triple defined with the reference element, $\widehat{T}$, as the unit square (2D) or unit cube (3D) and $\mathcal{Q}^\ell(\widehat{T}) := \{\Pi_{i=1}^{dim} p_i(\mathbf{x}_i) | p_i(t) \in \mathcal{P}^\ell([0,1])\}$, the set of products of polynomials of degree ℓ in each dimension. Let $\{\mathcal{T}_h\}_{h>0}$ be a sequence of quasi-uniform partitions of Λ using quadrilateral (2D) or hexagonal (3D) elements. Let h denote the meshsize of $\mathcal{T}_h$. Given $\ell \geq 1$, define $\mathcal{Q}^\ell(\mathcal{T}_h)$ to be the set of globally continuous polynomials where the pull back to the reference element of the restriction to each element, $T \in \mathcal{T}_h$ is in $\mathcal{Q}^\ell(\widehat{T})$. We let $d_h(\mathbf{x}) \in \mathcal{Q}^\ell(\mathcal{T}_h)$ be a continuous piecewise polynomial finite element approximation to $d(\mathbf{x})$ with

$$\|d - d_h\|_{L^\infty(\Lambda)} + h\|\nabla(d - d_h)\|_{L^\infty(\Lambda)} \leq ch^{\ell+1}\|d\|_{W^{\ell+1,\infty}(\Lambda)},$$

with $c > 0$, a constant dependent only on Λ, and provided that $d \in W^{\ell+1,\infty}(\Lambda)$.

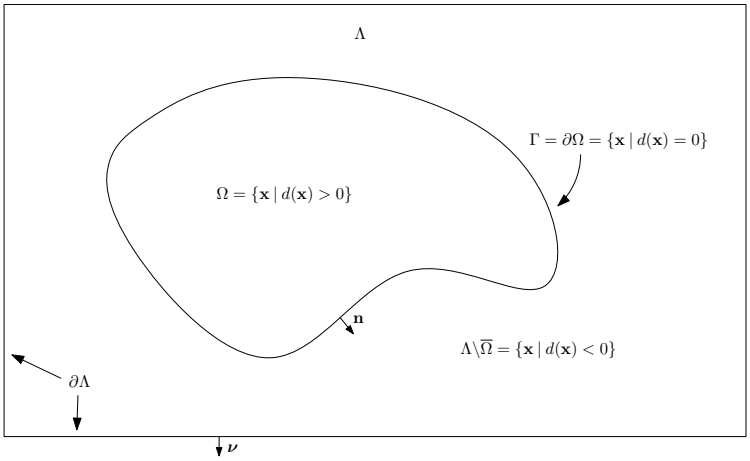


Figure 3.1: Region Λ containing a closed interface Γ as boundary of subregion Ω. These regions are defined implicitly by a signed distance function, $d(\mathbf{x})$. The outer normal, $\mathbf{n}$, to Γ and outer normal, $\boldsymbol{\nu}$, to $\partial\Lambda$ are denoted as well.

Now given $d_h(\mathbf{x})$, we define

$$\Gamma_h = \{\mathbf{x} \in \Lambda \mid d_h(\mathbf{x}) = 0\}.$$

Suppose $f(\mathbf{x}) \in W^{s,\infty}(\Gamma)$, with $s \geq 0$, is a quantity we want to integrate on Γ. We extend it smoothly to a neighborhood of Γ and since our mesh is not fitted to Γ, we convert integrals over Γ to be bulk integrals against an approximate Dirac delta function, δ_ε.

Remark. This extension is one of the most subtle parts of this process. If it can be extended constant in the normal direction then that is best but any smooth extension will work. There will be some regularity requirements (See Theorem 3.3.1) on the normal derivatives of the extension to obtain the desired convergence rates. *end Remark.*

Instead of directly using δ_ε as a function of $\mathbf{x} \in \mathbb{R}^d$, we will use a smooth 1D approxi-

mate Dirac delta function $\delta_\varepsilon(t)$, $t \in \mathbb{R}$ with support on $[-\varepsilon, \varepsilon]$ where

$$\varepsilon = ch^\beta \text{ and } 0 < \beta \leq 1 \tag{3.4}$$

and c is an absolute constant, typically 1, 2 or 3. Then using $d(\mathbf{x})$, our distance function, we extend to multiple dimensions

$$\int_\Gamma f(\mathbf{x})dA = \int_\Lambda f(\mathbf{x})\delta_\Gamma(\mathbf{x})d\mathbf{x} \approx \int_\Lambda f(\mathbf{x})\delta_\varepsilon(d(\mathbf{x}))|\nabla d(\mathbf{x})|d\mathbf{x} = \int_\Lambda f(\mathbf{x})\delta_\varepsilon(d(\mathbf{x}))d\mathbf{x}. \tag{3.5}$$

Notice that the kernel $\delta_\varepsilon(d(\mathbf{x}))$ has support on the ε-band

$$B_\varepsilon := \{\mathbf{x} \in \Lambda \mid |d(\mathbf{x})| \leq \varepsilon\}.$$

around Γ. Likewise,

$$B_{h,\varepsilon} := \{\mathbf{x} \in \Lambda \mid |d_h(\mathbf{x})| \leq \varepsilon\},$$

is the ε-band around Γ_h.

Remark. In reality we only need $d(\mathbf{x})$ to be the distance function in the union of bands $B_\varepsilon \cup B_{h,\varepsilon}$ since any changes outside of that region are not reflected in the integrals. We will choose our level set function as in Chapter 2 to be the distance function on the region containing $B_{h,\varepsilon}$ and then threshold with the hyperbolic tangent outside that region. *end Remark.*

Given a composite quadrature rule $\text{Quad}_\Lambda(\cdot)$ with positive weights and on each element, with support points $\{\mathbf{x}_q\} \subset T$, we also define

$$\mathcal{T}_{h,\varepsilon} := \{T \in \mathcal{T}_h \mid T \cap B_{h,\varepsilon} \text{ contains a quadrature point}\}. \tag{3.6}$$

Thus $\mathcal{T}_{h,\varepsilon}$ is the set of elements containing the band $B_{h,\varepsilon}$ that would be recognized by our quadrature rule, see Figure 3.2. Given a large enough quadrature rule, the set of cells that intersect the band but do not contain a quadrature rule inside the band is very small and since δ_ε smoothly approaches 0 at the edge of the band, the contribution lost is minimal.

We break $\mathcal{T}_{h,\varepsilon}$ into two distinct sets $\mathcal{T}_{h,\varepsilon} = \mathcal{T}_{h,\varepsilon}^{\text{int}} \cup \mathcal{T}_{h,\varepsilon}^{\text{edge}}$ where

$$\mathcal{T}_{h,\varepsilon}^{\text{int}} := \{T \in \mathcal{T}_{h\,\varepsilon} \,|\, T \subset B_{h,\varepsilon}\} \tag{3.7}$$

are those cells completely inside the band $B_{h,\varepsilon}$ and

$$\mathcal{T}_{h,\varepsilon}^{\text{edge}} = \mathcal{T}_{h,\varepsilon} \backslash \mathcal{T}_{h,\varepsilon}^{\text{int}} \tag{3.8}$$

are the set of cells that intersect the boundary of $B_{h,\varepsilon}$ and that have at least one quadrature point inside of $B_{h,\varepsilon}$ as seen in Figure 3.2. We will use these two sets for quadrature error

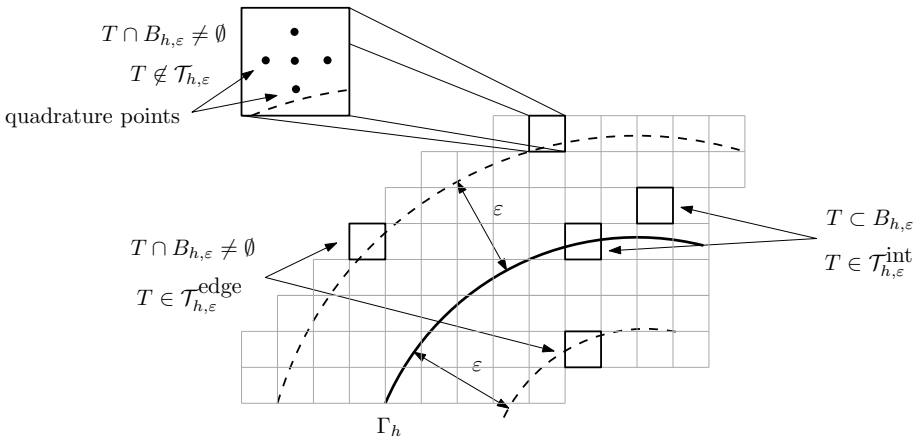


Figure 3.2: The ε band around Γ_h, denoted by $B_{h,\varepsilon}$ intersects with two types of cells, those completely interior, $T \in \mathcal{T}_{h,\varepsilon}^{\text{int}}$ and those not completely in $B_{h,\varepsilon}$ but which the quadrature rule can see, $T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}$. An example of a cell which intersects $B_{h,\varepsilon}$ but is not seen by the quadrature so is not in $\mathcal{T}_{h,\varepsilon}$ is also given.

analysis. In particular they are used in Section 3.3.3. Finally we define the following regions as unions of cells to integrate over

$$D_{h,\varepsilon} := \bigcup_{T \in \mathcal{T}_{h,\varepsilon}} T \tag{3.9}$$

and likewise

$$D_{h,\varepsilon}^{\text{int}} := \bigcup_{T \in \mathcal{T}_{h,\varepsilon}^{\text{int}}} T, \qquad D_{h,\varepsilon}^{\text{edge}} := \bigcup_{T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}} T \tag{3.10}$$

which are used in the analysis of quadrature error of Section 3.3.3.

3.3 Approximating integrals on a surface

The goal of this section is to estimate an upper bound on the error

$$E = \left| \int_{\Lambda} f(\mathbf{x}) \delta_{\Gamma}(d(\mathbf{x})) |\nabla d(\mathbf{x})| d\mathbf{x} - Q_{\Lambda}(f(\mathbf{x}) \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})|) \right| \tag{3.11}$$

where Q_{Λ} is our composite quadrature rule with positive weights, exact for polynomials of degree n on each cell $T \in \mathcal{T}_h$. The complete error bound and main result of this section is found in equation (3.21) of Section 3.3.4. We follow the general ideas of [29], although we had to modify some of the proof to explicitly track the regularity assumptions on f and the requirements for the quadrature rule. We will show below that using $\varepsilon \sim h^{3/4}$ and

$$\delta_{\varepsilon}(t) = \frac{1}{\varepsilon} \psi^{1,4}\left(\frac{t}{\varepsilon}\right) = \begin{cases} \frac{693}{512} \frac{1}{\varepsilon} \left(1 - \left(\frac{t}{\varepsilon}\right)^2\right)^5, & |t| \leq \varepsilon \\ 0, & \text{otherwise} \end{cases},$$

we can obtain a rate of $E \leq ch^{3/2}$ when we use quadrature exact for polynomials of degree 5, and FE space degree $\ell = 2$ for $d_h(\mathbf{x})$ and $f(\mathbf{x}) \in W^{6,\infty}\left(B_{h,\varepsilon} \cup B_{\varepsilon}\right)$ with 2 continuous bounded derivatives normal to Γ. These are the necessary conditions for the rate but we

observe the same rate under much less restrictive conditions including FE space degree $\ell = 1$, quadrature degree exact for polynomials of degree 3 and $f(\mathbf{x})$ with much less smoothness, even piecewise smooth $f \in \mathbb{Q}^{\ell}(\mathcal{T}_h)$. We will offer suggestions of why this might be in Appendix A.

We break the above error term into three parts and estimate each separately:

$$\begin{aligned}
E &= \left| \int_{\Lambda} f(\mathbf{x}) \delta_{\Gamma}(d(\mathbf{x})) |\nabla d(\mathbf{x})| d\mathbf{x} - Q_{\Lambda}(f(\mathbf{x}) \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})|) \right| \\
&\leq \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\Gamma}(d(\mathbf{x})) - \delta_{\varepsilon}(d(\mathbf{x})) \right] |\nabla d(\mathbf{x})| d\mathbf{x} \right| \\
&\qquad + \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\varepsilon}(d(\mathbf{x})) |\nabla d(\mathbf{x})| - \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})| \right] d\mathbf{x} \right| \\
&\qquad\qquad + \left| \int_{\Lambda} f(\mathbf{x}) \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})| d\mathbf{x} - Q_{\Lambda}(f(\mathbf{x}) \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})|) \right| \\
&= E_{\text{analytic}} + E_{\text{fem}} + E_{\text{quad}}
\end{aligned}$$

3.3.1 Analytic error, $E_{analytic}$

Before we begin estimating the analytic error, we need to introduce a little bit of differential geometry and notation. After this is established, we will show that the analytic error only depends on bounded normal derivatives of $f(\mathbf{x})$ and the number of moments, m, that $\delta_{\varepsilon}(t)$ satisfies.

3.3.1.1 Differential geometry

Suppose that we have a parameterization $\chi(\mathbf{u})$ of Γ with $\mathbf{u} = (u, v)$ for $d = 3$ and $\mathbf{u} = u$ for $d = 2$. Note that it is sufficient to suppose we have a single chart for the manifold Γ as otherwise we could introduce a partition of unity and then do all the results locally on each chart. The use of the partition of unity adds no insight and only technical complications so we simplify it away. We focus our efforts on $d = 3$, so that Γ is a 2D

manifold embedded in $\mathbb{R}^3$. Then

$$\Gamma = \{\chi(\mathbf{u}) \mid \mathbf{u} = (u, v) \in U \subset \mathbb{R}^2\}.$$

where $\chi : U \subset \mathbb{R}^2 \to \Gamma \subset \mathbb{R}^3$.

For a function, $g(\mathbf{x})$ defined on Γ, we can introduce the pull-back $g^*(\mathbf{u}) := g(\chi(\mathbf{u}))$ and thus can rewrite integrals on Γ as

$$\int_\Gamma g(s)ds = \int_U g^*(\mathbf{u})q(\mathbf{u})d\mathbf{u}$$

where the area element is $q(\mathbf{u})d\mathbf{u} = \sqrt{|\det G|}d\mathbf{u}$ with the first fundamental tensor $G := \nabla_{\mathbf{u}}\chi\nabla_{\mathbf{u}}\chi^T \in \mathbb{R}^{2\times 2}$ where $\nabla_{\mathbf{u}}\chi \in \mathbb{R}^{2\times 3}$ is the gradient with the i-th row the gradient of χ_i. Another equivalent definition of the area element is $q(\mathbf{u})d\mathbf{u} = |\partial_1\chi(\mathbf{u}) \times \partial_2\chi(\mathbf{u})|d\mathbf{u}$ where $\partial_1\chi(\mathbf{u})$ is the partial derivative with respect to the first variable u and likewise $\partial_2\chi(\mathbf{u})$ is the partial derivative with respect to v.

At each point $\chi(\mathbf{u})$ on Γ, there is a normal-to-Γ unit vector defined as

$$\mathbf{n}(\mathbf{u}) = \frac{\partial_1\chi(\mathbf{u}) \times \partial_2\chi(\mathbf{u})}{|\partial_1\chi(\mathbf{u}) \times \partial_2\chi(\mathbf{u})|}.$$

We assume that Γ is a closed orientable surface so that the normals on Γ all point consistently either outward or inward depending on our parameterization, χ. With $\mathbf{n}(\mathbf{u})$ defined, we can introduce a parameterization for our ε-band around Γ,

$$B_\varepsilon = \{\mathbf{x} \mid |d(\mathbf{x})| \leq \varepsilon\} = \{\mathbf{x} \in \Lambda \mid \mathbf{x} = X(\mathbf{u}, t) := \chi(\mathbf{u}) + t\mathbf{n}(\mathbf{u}),\ \mathbf{u} \in U,\ -\varepsilon \leq t \leq \varepsilon\}.$$

The map X from $(\mathbf{u}, t) \mapsto \mathbf{x}$ has Jacobian matrix

$$J(\mathbf{u}, t) = \begin{bmatrix} \partial_1\chi(\mathbf{u}) + t\partial_1\mathbf{n}(\mathbf{u}) & \partial_2\chi(\mathbf{u}) + t\partial_2\mathbf{n}(\mathbf{u}) & \mathbf{n}(\mathbf{u}) \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

and so to find the volume element, we have $dV = |\det(J)|d\mathbf{u}dt$. We turn to the triple product to evaluate this determinant. Using the cross product definition of $\mathbf{n}$, we have

$$\begin{aligned} \det(J) &= [\partial_1\chi(\mathbf{u}) + t\partial_1\mathbf{n}(\mathbf{u})] \times [\partial_2\chi(\mathbf{u}) + t\partial_2\mathbf{n}(\mathbf{u})] \cdot \mathbf{n}(\mathbf{u}) \\ &= |\partial_1\chi \times \partial_2\chi| + t\left(\partial_1\mathbf{n} \times \partial_2\chi + \partial_1\chi \times \partial_2\mathbf{n}\right) \cdot \mathbf{n} + t^2\left(\partial_1\mathbf{n} \times \partial_2\mathbf{n}\right) \cdot \mathbf{n} \end{aligned}$$

Likewise, using $P_n = I - \mathbf{n} \otimes \mathbf{n}$,

$$\begin{aligned} \partial_i\mathbf{n} &= (I - \mathbf{n} \otimes \mathbf{n}) \frac{\partial_i\left(\partial_1\chi \times \partial_2\chi\right)}{|\partial_1\chi \times \partial_2\chi|} \\ &= \frac{1}{|\partial_1\chi \times \partial_2\chi|}\left[P_n\partial_i\left(\partial_1\chi \times \partial_2\chi\right)\right]. \end{aligned}$$

Recognizing $q(\mathbf{u}) = |\partial_1\chi \times \partial_2\chi|$, we simplify the determinant to

$$\begin{aligned} \det(J) &= |\partial_1\chi \times \partial_2\chi| + t\left[\partial_1\mathbf{n} \times \partial_2\chi + \partial_1\chi \times \partial_2\mathbf{n}\right] \cdot \mathbf{n} + t^2\left(\partial_1\mathbf{n} \times \partial_2\mathbf{n}\right) \cdot \mathbf{n} \\ &= |\partial_1\chi \times \partial_2\chi|\left(1 + p_1(\mathbf{u})t + p_2(\mathbf{u})t^2\right) \\ &= q(\mathbf{u})\left(1 + p_1(\mathbf{u})t + p_2(\mathbf{u})t^2\right) \end{aligned}$$

where

$$p_1(\mathbf{u}) = \left[\frac{\left[P_n\partial_1\left(\partial_1\chi \times \partial_2\chi\right)\right] \times \partial_2\chi \cdot \mathbf{n} - \left[P_n\partial_2\left(\partial_1\chi \times \partial_2\chi\right)\right] \times \partial_1\chi \cdot \mathbf{n}}{|\partial_1\chi \times \partial_2\chi|^2}\right] \tag{3.12}$$

and

$$p_2(\mathbf{u}) = \left[\frac{[P_n \partial_1 (\partial_1 \chi \times \partial_2 \chi)] \times [P_n \partial_2 (\partial_1 \chi \times \partial_2 \chi)] \cdot \mathbf{n}}{|\partial_1 \chi \times \partial_2 \chi|^3} \right]. \tag{3.13}$$

Thus for a function $g(\mathbf{x})$ defined on B_ε, we have

$$\int_{B_\varepsilon} g(\mathbf{x}) d\mathbf{x} = \int_{-\varepsilon}^{\varepsilon} \int_U g^*(\mathbf{u}, t) q(\mathbf{u}) (1 + t p_1(\mathbf{u}) + t^2 p_2(\mathbf{u})) d\mathbf{u} dt.$$

Remark. Tornberg proves in [29] that for the simpler case of $d = 2$ we get $d\mathbf{x} = q(u)(1 - t\kappa(u))dudt$ with $\kappa(u)$ the curvature of the curve, Γ, evaluated at the position, u. Here $q(u) = |\chi'(u)|$ is the 1-dimensional area element for a curve. *end Remark.*

Finally, note that $d(\mathbf{x}) = d(X(\mathbf{u}, t)) = d\left(\chi(\mathbf{u}) + t\mathbf{n}(\mathbf{u})\right) = t$ so that $\delta_\varepsilon(d(\mathbf{x})) = \delta_\varepsilon(t)$. We also note that $|\nabla d(\mathbf{x})| = 1$ since we are using the exact distance function in this computation.

3.3.1.2 Estimate on analytic error

We want a bound on

$$E_{\text{analytic}} = \left| \int_\Lambda f(\mathbf{x}) \left(\delta_\Gamma(d(\mathbf{x})) - \delta_\varepsilon(d(\mathbf{x}))\right) |\nabla d(\mathbf{x})| d\mathbf{x} \right|.$$

To this end, we define

$$I_\Gamma := \int_\Gamma f(\mathbf{x}) d\mathbf{x} = \int_U f^*(\mathbf{u}) q(\mathbf{u}) d\mathbf{u}$$

and

$$I_\varepsilon := \int_{B_\varepsilon} f(\mathbf{x}) \delta_\varepsilon(d(\mathbf{x})) |\nabla d(\mathbf{x})| d\mathbf{x} = \int_U \int_{-\varepsilon}^{\varepsilon} f^*(\mathbf{u}, t) \delta_\varepsilon(t) q(\mathbf{u}) (1 + t p_1(\mathbf{u}) + t^2 p_2(\mathbf{u})) dt d\mathbf{u}.$$

Expand $f^*(\mathbf{u}, t)$ in a Taylor series of t centered at $(\mathbf{u}, 0)$. We assume that $f^*(\mathbf{u}, t)$ has N continuous derivatives in t and bounded $N+1$ derivative in t. Then,

$$f^*(\mathbf{u}, t) = \sum_{i=0}^{N} \frac{t^i}{i!} \frac{\partial^i f^*(\mathbf{u}, 0)}{\partial t^i} + \mathcal{O}(t^{N+1}).$$

Plugging this into our formulation for I_ε, we simplify in terms of moment functions (defined in equation (3.1))

$$\begin{aligned}
I_\varepsilon &= \int_U \int_{-\varepsilon}^{\varepsilon} \left[\sum_{i=0}^{N} \frac{t^i}{i!} \frac{\partial^i f^*(\mathbf{u}, 0)}{\partial t^i} + \mathcal{O}(t^{N+1}) \right] \delta_\varepsilon(t) q(\mathbf{u})(1 + t p_1(\mathbf{u}) + t^2 p_2(\mathbf{u})) dt d\mathbf{u} \\
&= \sum_{i=0}^{N} \frac{1}{i!} \int_U \frac{\partial^i f^*(\mathbf{u}, 0)}{\partial t^i} q(\mathbf{u}) \int_{-\varepsilon}^{\varepsilon} t^i \delta_\varepsilon(t)(1 + t p_1(\mathbf{u}) + t^2 p_2(\mathbf{u})) dt d\mathbf{u} \\
&\quad + \int_U \int_{-\varepsilon}^{\varepsilon} \mathcal{O}\left(t^{N+1}\right) \delta_\varepsilon(t) q(\mathbf{u})(1 + t p_1(\mathbf{u}) + t^2 p_2(\mathbf{u})) dt d\mathbf{u} \\
&= \int_U f^*(\mathbf{u}, 0) q(\mathbf{u}) d\mathbf{u} \int_{-\varepsilon}^{\varepsilon} \delta_\varepsilon(t) dt + \sum_{\alpha=1}^{N} c_{\alpha,f} M_\alpha(\delta_\varepsilon(t)) + \mathcal{O}\left(M_{N+1}(\delta_\varepsilon(t))\right)
\end{aligned}$$

where

$$\begin{aligned}
c_{\alpha,f} = \frac{1}{\alpha!} \int_U \frac{\partial^\alpha f^*(\mathbf{u}, 0)}{\partial t^\alpha} q(\mathbf{u}) d\mathbf{u} \\
+ \frac{1}{(\alpha-1)!} \int_U \frac{\partial^{(\alpha-1)} f^*(\mathbf{u}, 0)}{\partial t^{(\alpha-1)}} q(\mathbf{u}) p_1(\mathbf{u}) d\mathbf{u} \\
+ \frac{1}{(\alpha-2)!} \int_U \frac{\partial^{(\alpha-2)} f^*(\mathbf{u}, 0)}{\partial t^{(\alpha-2)}} q(\mathbf{u}) p_2(\mathbf{u}) d\mathbf{u}
\end{aligned}$$

for $\alpha \geq 2$ and only the first two terms when $\alpha = 1$.

We note that for $\delta_\varepsilon(t) := \frac{1}{\varepsilon}\psi\left(\frac{t}{\varepsilon}\right)$, we have

$$\begin{aligned} M_\alpha(\delta_\varepsilon(t)) &= \int_{-\varepsilon}^{\varepsilon} t^\alpha \delta_\varepsilon(t) dt \\ &= \frac{1}{\varepsilon}\int_{-\varepsilon}^{\varepsilon} t^\alpha \psi\left(\frac{t}{\varepsilon}\right) dt \\ &= \int_{-1}^{1} (\varepsilon\tau)^\alpha \psi(\tau) d\tau \\ &= \varepsilon^\alpha M_\alpha(\psi(t)) \end{aligned}$$

so that

$$I_\varepsilon = I_\Gamma M_0(\psi(t)) + \sum_{\alpha=1}^{N} C_{\alpha,f}\varepsilon^\alpha M_\alpha(\psi) + \mathcal{O}(\varepsilon^{N+1}).$$

Thus the analytic error depends entirely on the continuity of $f(\mathbf{x})$ in the normal direction to Γ and on the moments of kernel ψ that we choose to use for our δ_ε approximation to δ_Γ. If $M_0(\psi) = 1$, then

$$\begin{aligned} E_{\text{analytic}} &= |I_\Gamma - I_\varepsilon| \\ &= \left| -\sum_{\alpha=1}^{N} C_{\alpha,f}\varepsilon^\alpha M_\alpha(\psi) + \mathcal{O}\left(\varepsilon^{N+1}\right) \right| \\ &\leq \sum_{\alpha=1}^{N} |C_{\alpha,f}|\, \varepsilon^\alpha\, |M_\alpha(\psi)| + \mathcal{O}\left(\varepsilon^{N+1}\right) \end{aligned} \tag{3.14}$$

where f has $N+1$ bounded derivatives in the normal to Γ direction.

Remark. It is also interesting to notice that if the normal direction grows like a polynomial of small enough degree compared to the moments of our Dirac kernel, then it is possible to have no contribution whatsoever from the analytic error, ie $E_{\text{analytic}} = 0$. In particular, if $f(\mathbf{x})$ is constant in the normal direction and $M_0(\psi) = 1.$, then there is no analytic error. It is possible in this case to choose a different relationship between ε and meshsize h than we will describe below to get higher convergence rate. *end Remark.*

Example 1

$$\psi^{1,2}(t) := \begin{cases} \frac{35}{32}(1-t^2)^3, & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

has

$$M_0(\psi^{1,2}) = 1$$
$$M_1(\psi^{1,2}) = 0$$
$$M_2(\psi^{1,2}) = 1/9$$

so if we choose f with at least 2 bounded derivatives in the normal direction, then we have

$$|E_{\text{analytic}}| \leq c\varepsilon^2. \tag{3.15}$$

Example 2

$$\psi^{1,4}(t) := \begin{cases} \frac{693}{512}(1-t^2)^5, & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

has

$$M_0(\psi^{1,4}) = 1$$
$$M_1(\psi^{1,4}) = 0$$
$$M_2(\psi^{1,4}) = 1/13$$

so if we choose f with at least 2 bounded derivatives in the normal direction, then we have

$$E_{\text{analytic}} \leq c\varepsilon^2. \tag{3.16}$$

3.3.2 FEM error E_{fem}

We want an estimate on

$$E_{\text{fem}} = \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\varepsilon}(d(\mathbf{x})) |\nabla d(\mathbf{x})| - \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})| \right] d\mathbf{x} \right|.$$

First, notice that by the triangle inequality and using relationship (3.2), $|\nabla d_h(\mathbf{x})| = |\nabla d_h(\mathbf{x}) - \nabla d(\mathbf{x})| + |\nabla d(\mathbf{x})| \leq ch^{\ell} + 1$ where $c > 0$ is only dependent on Λ. Thus we have the bound,

$$\| |\nabla d_h| - 1 \|_{L^{\infty}(\Lambda)} = \| |\nabla d_h| - |\nabla d| \|_{L^{\infty}(\Lambda)} \leq ch^{\ell}. \tag{3.17}$$

Second, since $\delta'_{\varepsilon}(\tau) = \frac{1}{\varepsilon^2} \psi' \left(\frac{\tau}{\varepsilon} \right) \leq \frac{C}{\varepsilon^2}$ with $C > 0$ a constant dependent only on choice of Dirac kernel, ψ, and $|d_h - d\|_{L^{\infty}(\Lambda)} \leq ch^{\ell+1}$, we have

$$\|\delta_{\varepsilon}(d_h) - \delta_{\varepsilon}(d)\|_{L^{\infty}(\Lambda)} \leq \max_{\tau \in \mathbb{R}} |\delta'_{\varepsilon}(\tau)| \, \|d_h - d\|_{L^{\infty}(\Lambda)} \leq \tilde{C} \frac{h^{\ell+1}}{\varepsilon^2},$$

where $\tilde{C} > 0$ is a constant dependent only on Λ and the Dirac kernel, ψ. Thus,

$$\begin{aligned}
E_{\text{fem}} &= \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\varepsilon}(d(\mathbf{x})) |\nabla d(\mathbf{x})| - \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})| \right] d\mathbf{x} \right| \\
&= \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\varepsilon}(d(\mathbf{x})) |\nabla d(\mathbf{x})| \pm \delta_{\varepsilon}(d(\mathbf{x})) |\nabla d_h(\mathbf{x})| - \delta_{\varepsilon}(d_h(\mathbf{x})) |\nabla d_h(\mathbf{x})| \right] d\mathbf{x} \right| \\
&\leq \left| \int_{\Lambda} f(\mathbf{x}) \delta_{\varepsilon}(d(\mathbf{x})) \, \big| |\nabla d(\mathbf{x})| - |\nabla d_h(\mathbf{x})| \big| \, d\mathbf{x} \right| \\
&\qquad + \left| \int_{\Lambda} f(\mathbf{x}) \left[\delta_{\varepsilon}(d(\mathbf{x})) - \delta_{\varepsilon}(d_h(\mathbf{x})) \right] |\nabla d_h(\mathbf{x})| d\mathbf{x} \right| \\
&\leq \left(c \frac{h^{\ell}}{\varepsilon} \|f(\mathbf{x})\|_{L^{\infty}(B_{h,\varepsilon} \cup B_{\varepsilon})} + c \frac{h^{\ell+1}}{\varepsilon^2} \|f\|_{L^{\infty}(B_{h,\varepsilon} \cup B_{\varepsilon})} \right) \int_{B_{h,\varepsilon} \cup B_{\varepsilon}} d\mathbf{x} \\
&\leq \|f\|_{L^{\infty}(B_{h,\varepsilon} \cup B_{\varepsilon})} \left(ch^{\ell} + c \frac{h^{\ell+1}}{\varepsilon} \right) |\Gamma|
\end{aligned} \tag{3.18}$$

since $|B_\varepsilon \cup B_{h,\varepsilon}| \leq c\varepsilon|\Gamma|$.

3.3.3 *Quadrature error, E_{quad}*

We first calculate the error from quadrature on each cell

$$E_T^{\text{quad}} = \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right)|\nabla d_h(\mathbf{x})| d\mathbf{x} - Q_T\left(f\delta_\varepsilon\left(d_h\right)|\nabla d_h|\right) \right|.$$

By adding and subtracting $\int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right)|\nabla d| d\mathbf{x}$ and $Q_T\left(f\delta_\varepsilon\left(d_h\right)|\nabla d|\right)$, and using that

$$\| |\nabla d_h| - |\nabla d| \|_{L^\infty(\Lambda)} \leq ch^\ell,$$

we get (for quadrature with positive weights and thus $\sum_j w_j = |T|$)

$$\begin{aligned}
E_T^{\text{quad}} &\leq \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right)\left(|\nabla d_h| - |\nabla d|\right) d\mathbf{x} \right| \\
&\quad + \left| Q_T\left(f\delta_\varepsilon\left(d_h\right)\left(|\nabla d_h| - |\nabla d|\right)\right) \right| \\
&\quad + \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right) d\mathbf{x} - Q_T\left(f\delta_\varepsilon\left(d_h\right)\right) \right| \\
&\leq \left(ch^\ell\right) \|f(\mathbf{x})\delta_\varepsilon(d_h(\mathbf{x}))\|_{L^\infty(T)} \left(|T| + \sum w_j\right) \\
&\quad + \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right) d\mathbf{x} - Q_T\left(f\delta_\varepsilon\left(d_h\right)\right) \right| \\
&\leq c\frac{h^\ell}{\varepsilon}|T|\|f\|_{L^\infty(T)} + \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right) d\mathbf{x} - Q_T\left(f\delta_\varepsilon\left(d_h\right)\right) \right|
\end{aligned}$$

since $\|\delta_\varepsilon(d_h(\mathbf{x}))\|_{L^\infty(T)} \leq \frac{c}{\varepsilon}$.

We will estimate the quantity

$$E_T := \left| \int_T f(\mathbf{x})\delta_\varepsilon\left(d_h(\mathbf{x})\right) - Q_T\left(f\delta_\varepsilon\left(d_h\right)\right) \right|$$

using the Bramble-Hilbert lemma and then give estimates on the aggregate sum of errors.

It turns out that we have different smoothness of the term δ_ε on the cell T depending on whether $T \in \mathcal{T}_{h,\varepsilon}^{\text{int}}$ is on the interior of the band $B_{h,\varepsilon}$ or $T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}$ intersects the boundary of that region. Notice that we assumed $\psi \in C^k$ on the closed set $[-1,1]$ and the $k+1$ derivative is not continuous but is bounded. In fact, it has a jump at the edges of the band. Thus $\delta_\varepsilon(d_h(\mathbf{x})) \in W^{k+1,\infty}(\Lambda)$. Finally, notice that $\delta_\varepsilon(d_h(\mathbf{x}))$ is polynomial on the cells entirely in $D_{h,\varepsilon}$ (see equation (3.9)) and so is infinitely smooth in that region. We will also need to make assumptions on the smoothness of f which will be built up as needed. In the end we will show that we need $f \in W^{k+2,\infty}(\Lambda)$ to get the desired convergence rate.

3.3.3.1 Quadrature analysis for $T \in \mathcal{T}_{h,\varepsilon}^{edge}$, cells intersecting boundary of $B_{h,\varepsilon}$

We focus first on those cells $T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}$ that intersect the boundary of $B_{h,\varepsilon}$, mainly where $d_h(\mathbf{x}) = \pm\varepsilon$ at some $\mathbf{x} \in T$. On these cells,

$$\delta_\varepsilon(d_h(\mathbf{x})) \in W^{k+1,\infty}(T).$$

Recall that we denote the regularity of f by the parameter $s > 0$, $f(\mathbf{x}) \in W^{s,\infty}(T)$. So by choosing $s \geq k+1$,

$$f(\mathbf{x}) \in W^{s,\infty}(T) \subset W^{k+1,\infty}(T).$$

We will use a quadrature rule that is exact for polynomials of degree k on reference element, $\widehat{T}$, that is for $\mathcal{Q}^k(\widehat{T})$. Then using $\phi = f(\mathbf{x})\delta_\varepsilon(d_h(\mathbf{x}))$ and mapping from the cell, T, to a reference cell, $\hat{T}$, we have

$$E_T(\phi) = c\,(\det B_T)\,\hat{E}_{\hat{T}}\left(\hat{\phi}\right)$$

where the $\hat{}$ notation refers to objects defined on the reference element, $\widehat{T}$, so that

$$\hat{E}_{\hat{T}}\left(\hat{\phi}\right) := \left| \int_{\widehat{T}} \widehat{\phi}(\widehat{\mathbf{x}}) d\widehat{\mathbf{x}} - Q_{\widehat{T}}\left(\widehat{\phi}\right) \right|.$$

Then

$$\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \leq c\|\hat{\phi}\|_{L^\infty(\hat{T})} \leq c\|\hat{\phi}\|_{W^{k+1,\infty}(\hat{T})}.$$

So that the linear map $\hat{\phi} \mapsto \hat{E}_{\hat{T}}\left(\hat{\phi}\right)$ is bounded by $c\|\hat{\phi}\|_{W^{k+1,\infty}(\hat{T})}$. Notice that by our choice of quadrature rule, $\hat{E}_{\hat{T}}\left(\hat{\phi}\right)$ vanishes for $\hat{\phi} \in \mathbb{Q}^k(\hat{T})$, so by the Bramble-Hilbert lemma,

$$\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \leq \hat{c}\left|\hat{\phi}\right|_{W^{k+1,\infty}(\hat{T})}.$$

By using that $\widehat{d_h} \in \mathcal{Q}^\ell(\widehat{T})$ and $\delta_\varepsilon(t) = \frac{1}{\varepsilon}\psi\left(\frac{t}{\varepsilon}\right)$, along with the product and chain rule, we can bound

$$\|\delta_\varepsilon(d_h(\mathbf{x}))\|_{W^{j,\infty}(T)} \leq \frac{C}{\varepsilon^{j+1}}, \tag{3.19}$$

for all $j \geq 0$. Thus using (3.19) and the Cauchy-Schwarz inequality,

$$\begin{aligned}
E_T(\phi) &\leq c\left(\det B_T\right)\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \\
&\leq c\left(\det B_T\right)\left|\hat{\phi}\right|_{W^{k+1,\infty}(\hat{T})} \\
&\leq c\left(\det B_T\right)\sum_{j=0}^{k+1}\left|\hat{f}\right|_{W^{k+1-j,\infty}(\hat{T})}\left|\widehat{\delta_\varepsilon \circ d_h}\right|_{W^{j,\infty}(\hat{T})} \\
&\leq c\left(\det B_T\right)h^{k+1}\sum_{j=0}^{k+1}|f|_{W^{k+1-j,\infty}(T)}\left|\delta_\varepsilon \circ d_h\right|_{W^{j,\infty}(T)} \\
&\leq c\left(\det B_T\right)h^{k+1}\|\delta_\varepsilon(d_h(\mathbf{x}))\|_{W^{k+1,\infty}(T)}\|f\|_{W^{k+1,\infty}(T)} \\
&\leq c\left(\det B_T\right)\frac{h^{k+1}}{\varepsilon^{k+2}}C\|f\|_{W^{k+1,\infty}(T)}.
\end{aligned}$$

Now, since $\det B_T = c|T|$ and noting that

$$\sum_{T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}} |T| = ch\,|\Gamma|\,,$$

we have

$$\begin{aligned}\sum_{T \in \mathcal{T}_{h,\varepsilon}^{\text{edge}}} E_T &\leq c|\Gamma| \frac{h^{k+2}}{\varepsilon^{k+2}} C \|f\|_{W^{k+1,\infty}(D_{h,\varepsilon}^{\text{edge}})} \\ &\leq c|\Gamma| \left(\frac{h}{\varepsilon}\right)^{k+2} \|f\|_{W^{k+1,\infty}(D_{h,\varepsilon}^{\text{edge}})}.\end{aligned}$$

3.3.3.2 Quadrature analysis for $T \in \mathcal{T}_{h,\varepsilon}^{int}$, cells completely interior to $B_{h,\varepsilon}$

We focus next on those cells $T \in \mathcal{T}_{h,\varepsilon}^{\text{int}}$ that are completely on the interior of $D_{h,\varepsilon}$. Then in this case, pulling back to the reference element

$$\widehat{\delta_\varepsilon(d_h(\mathbf{x}))} \in \text{polynomial}(\widehat{T}) \subset W^{k+2,\infty}(\widehat{T})$$

and if we choose $s \geq k+2$, then

$$f(\mathbf{x}) \in W^{s,\infty}(T) \subset W^{k+2,\infty}(T).$$

We now require a quadrature rule that is exact for polynomials of degree $k+1$ on the reference element, in $\mathcal{Q}^{k+1}(\widehat{T})$. Then using $\phi = f(\mathbf{x})\delta_\varepsilon(d_h(\mathbf{x}))$ and mapping from the cell T to a reference cell $\hat{T}$, we have

$$E_T(\phi) = c\,(\det B_T)\,\hat{E}_{\hat{T}}\left(\hat{\phi}\right)$$

and

$$\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \leq c\|\hat{\phi}\|_{L^\infty(\hat{T})} \leq c\|\hat{\phi}\|_{W^{k+2,\infty}(\hat{T})}$$

so that the linear map $\hat{\phi} \mapsto \hat{E}_{\hat{T}}\left(\hat{\phi}\right)$ is bounded by $c\|\hat{\phi}\|_{W^{k+2,\infty}(\hat{T})}$. Notice that by our choice of quadrature rule, $\hat{E}_{\hat{T}}\left(\hat{\phi}\right)$ vanishes for $\hat{\phi} \in \mathcal{Q}^{k+1}(\hat{T})$, so by the Bramble-Hilbert lemma,

$$\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \leq \hat{c}\left|\hat{\phi}\right|_{W^{k+2,\infty}(\hat{T})}.$$

Thus using that $\|\delta_\varepsilon(d_h)\|_{W^{j,\infty}(T)} \leq \frac{c}{\varepsilon^{j+1}}$ and the Cauchy-Schwarz inequality,

$$\begin{aligned}
E_T(\phi) &\leq c\left(\det B_T\right)\hat{E}_{\hat{T}}\left(\hat{\phi}\right) \\
&\leq c\left(\det B_T\right)\left|\hat{\phi}\right|_{W^{k+2,\infty}(\hat{T})} \\
&\leq c\left(\det B_T\right)\sum_{j=0}^{k+2}\left|\hat{f}\right|_{W^{k+2-j,\infty}(\hat{T})}\left|\widehat{\delta_\varepsilon \circ d_h}\right|_{W^{j,\infty}(\hat{T})} \\
&\leq c\left(\det B_T\right)h^{k+2}\sum_{j=0}^{k+2}|f|_{W^{k+2-j,\infty}(T)}\left|\delta_\varepsilon \circ d_h\right|_{W^{j,\infty}(T)} \\
&\leq c\left(\det B_T\right)\frac{h^{k+2}}{\varepsilon^{k+3}}\|f\|_{W^{k+2,\infty}(T)}.
\end{aligned}$$

Now, since $\det B_T = c|T|$ and noting that

$$\sum_{T\in\mathcal{T}_{h,\varepsilon}^{\text{int}}}|T| = c\varepsilon\left|\Gamma\right|,$$

we have

$$\sum_{T\in\mathcal{T}_{h,\varepsilon}^{\text{int}}}E_T \leq c|\Gamma|\frac{h^{k+2}}{\varepsilon^{k+2}}\|f\|_{W^{k+2,\infty}(D_{h,\varepsilon}^{\text{int}})}.$$

3.3.3.3 Full quadrature error

If $f \in W^{k+2,\infty}(D_{h,\varepsilon})$ that is, $s = k+2$ and using a quadrature rule exact on $\mathcal{Q}^{k+1}(\widehat{T})$, then our results for both cases hold and we have the combined error bound

$$\begin{aligned} E_{\text{quad}} &= \sum_{T \in \mathcal{T}_{h,\varepsilon}} E_T^{\text{quad}} \\ &\leq \sum_{T \in \mathcal{T}_{h,\varepsilon}} E_T + ch^{\ell} \|f\|_{L^\infty(T)} \\ &\leq c|\Gamma| \frac{h^{k+2}}{\varepsilon^{k+2}} \|f\|_{W^{k+2,\infty}\left(D_{h,\varepsilon}\right)} + ch^{\ell} \|f\|_{L^\infty\left(D_{h,\varepsilon}\right)}. \end{aligned} \tag{3.20}$$

3.3.4 Full error bound

We can now combine all of the above bounds together into a single bound.

$$\begin{aligned} E &= E_{\text{analytic}} + E_{\text{fem}} + E_{\text{quad}} \\ &\leq c\varepsilon^m + c\left(h^{\ell} + \frac{h^{\ell+1}}{\varepsilon}\right) \|f\|_{L^\infty\left(D_{h,\varepsilon}\right)} \\ &\qquad + c\frac{h^{k+2}}{\varepsilon^{k+2}} \|f\|_{W^{k+2,\infty}\left(D_{h,\varepsilon}\right)} + c|\Gamma| \left(h^{\ell} + \frac{h^{\ell+1}}{\varepsilon}\right) \|f\|_{L^\infty\left(D_{h,\varepsilon}\right)} \\ &\leq C_1\varepsilon^m + C_2\frac{h^{k+2}}{\varepsilon^{k+2}} \|f\|_{W^{k+2,\infty}\left(D_{h,\varepsilon}\right)} + C_3\left(h^{\ell} + \frac{h^{\ell+1}}{\varepsilon}\right) \|f\|_{L^\infty\left(D_{h,\varepsilon}\right)} \end{aligned} \tag{3.21}$$

We are now able to describe how the rate, $\beta \in (0,1]$ in $\varepsilon = ch^{\beta}$ will been chosen. Supposing that ℓ is large enough that the C_3 term is higher order and as described before, $s = k+2$ and quadrature eqxact for polynomial of degree $k+1$, the limiting terms are $C_1\varepsilon^m$ and $C_2\frac{h^{k+2}}{\varepsilon^{k+2}}$ which we enforce to be asymptotically equal. For a specific kernel, ψ,

the parameters m and k can be computed as above so we can solve for ε in terms of h

$$\varepsilon \sim h^{\frac{k+2}{m+k+2}}. \tag{3.22}$$

There are additional simplifications and assumptions that can be made in certain cases that are described fully by Tornberg in [29] to raise the exponential power slightly, but the above analysis is sufficient for our case and in the below two examples, there are no changes to be made.

3.3.4.1 Example Dirac kernels

Example 1

$$\psi^{1,2}(t) := \begin{cases} \frac{35}{32}(1-t^2)^3, & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

has $m = 2$ and $k = 2$. Thus we choose $\varepsilon = ch^{2/3}$ to balance the C_1 and C_2 terms, so that we have the rate

$$\begin{aligned} E &\leq C_1\varepsilon^2 + C_2\frac{h^4}{\varepsilon^4}\|f\|_{W^{k+2,\infty}(\Lambda)} + C_3\left(h^\ell + \frac{h^{\ell+1}}{\varepsilon}\right)\|f\|_{L^\infty(\Lambda)} \\ &\leq ch^{4/3}\|f\|_{W^{k+2,\infty}(\Lambda)} + ch^\ell\|f\|_{L^\infty(\Lambda)} \end{aligned} \tag{3.23}$$

so for FE degree $\ell \geq 2$, $f(\mathbf{x}) \in W^{s,\infty}(\Lambda)$ with $s = k+2 = 4$ and at least 2 bounded derivatives in the normal direction to Γ, quadrature exact for degree $n = k+1 = 3$ give us an expected convergence rate $E \leq ch^{4/3}$.

Example 2

$$\psi^{1,4}(t) := \begin{cases} \frac{693}{512}(1-t^2)^5, & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

has $m = 2$ and $k = 4$. Thus we choose $\varepsilon = ch^{3/4}$ to balance the C_1 and C_2 terms, so that we have the rate

$$\begin{aligned} E &\leq C_1\varepsilon^2 + C_2\frac{h^6}{\varepsilon^6}\|f\|_{W^{k+2,\infty}(\Lambda)} + C_3\left(h^\ell + \frac{h^{\ell+1}}{\varepsilon}\right)\|f\|_{L^\infty(\Lambda)} \\ &\leq ch^{3/2}\|f\|_{W^{k+2,\infty}(\Lambda)} + ch^l\|f\|_{L^\infty(\Lambda)} \end{aligned} \tag{3.24}$$

so for FE degree $\ell \geq 2$, $f(\mathbf{x}) \in W^{s,\infty}(\Lambda)$ with $s = k+2 = 6$ and at least 2 bounded derivatives in the normal direction to Γ, quadrature exact for degree $n = k+1 = 5$ give us an expected convergence rate $E \leq ch^{3/2}$.

In all of our computations of the next few chapters we will use $\delta_\varepsilon(t) = \frac{1}{\varepsilon}\psi^{1,4}\left(\frac{t}{\varepsilon}\right)$ so we will emphasize the above results with the following theorem.

Theorem 3.3.1. *Let $\ell \geq 1$ and $d \in W^{\ell+1,\infty}(\Lambda)$ be the signed distance function implicitly defining a hypersurface, $\Gamma \subset \Lambda$. Let $(\widehat{T}, \mathcal{Q}^\ell(\widehat{T}), \Sigma)$ be the Lagrange finite element triple defined with the reference element, $\widehat{T}$, as the unit square (2D) or unit cube (3D) and $\mathcal{Q}^\ell(\widehat{T}) := \{\Pi_{i=1}^{dim} p_i(\boldsymbol{x}_i) \,|\, p_i(t) \in \mathcal{P}^\ell([0,1])\}$, the space of products of polynomials of degree ℓ in each dimension. We let $\{\mathcal{T}_h\}_{h>0}$ be a sequence of partitions of Λ made of quadrilateral (2D) or hexagonal (3D) elements of size h and $d_h \in \mathcal{Q}^\ell(\mathcal{T}_h) := \{p(\boldsymbol{x}) \in C(\Lambda) \mid \widehat{p|_T} \in \mathcal{Q}^\ell(\widehat{T})\}$ satisfying*

$$\|d - d_h\|_{L^\infty(\Lambda)} + h\|\nabla(d - d_h)\|_{L^\infty(\Lambda)} \leq ch^{\ell+1}\|d\|_{W^{\ell+1,\infty}(\Lambda)},$$

for some $c > 0$ only dependent on Λ. Let $\psi(t) = \frac{693}{512}(1-t^2)^5$, $\varepsilon = ch^{3/4}$ and B_ε and $B_{h,\varepsilon}$ to be defined respectively as the support of $\delta_\varepsilon(d(\boldsymbol{x})$ and $\delta_\varepsilon(d_h(\boldsymbol{x}))$. Given $f \in W^{6,\infty}(B_\varepsilon \cup B_{h,\varepsilon})$ and a composite quadrature rule, $Quad_\Lambda(\cdot)$ with positive weights exact

for $\mathcal{Q}^5(\widehat{T})$, then the error

$$E = \left| \int_\Gamma f(\boldsymbol{x}) d\boldsymbol{x} - \boldsymbol{Quad}_\Lambda \left(f(x)\delta_\varepsilon(d_h)|\nabla d_h| \right) \right|$$

satisfies

$$E \leq c_1 h^{3/2} \|f\|_{W^{6,\infty}(\Omega_\varepsilon)} + c_2 h^\ell \|f\|_{L^\infty(\Omega_\varepsilon)},$$

where c_1 and c_2 are constants only dependent on Λ.

Remark. It is interesting to note that in actual simulations, we often get the full $h^{3/2}$ convergence rate even with simple quadratures like the trapezoidal quadrature. We include in Appendix A a discussion of why that might happen. Although it is not a proof, it gives some intuition as to why we might be seeing the full rates. *end Remark.*

3.4 Numerical results

In [41], it is demonstrated that naively using an approximate Dirac function like the hat function $\psi(t) = 1 - |t|$ and $\varepsilon = mh$ for m some positive integer, leads to non convergence when used in our multidimensional extension, $\delta_\varepsilon(\mathbf{x}) = \frac{1}{\varepsilon}\psi\left(\frac{d_h(\mathbf{x})}{\varepsilon}\right)$.

In their analysis, they point out that the main problem is that on a uniform grid, the numerical integral of δ_ε deteriorates as the normal line across Γ becomes unaligned with the grid points. They note that one of the worst case scenarios for a closed boundary embedded in 2D would be a long skinny capsule shaped boundary angled at 45 degrees to the grid. In 3D, a worst case scenario would be with a 3D object which is as far from orthogonal as it can be from the main axes. We therefore will perform our tests using this scenario to verify the convergence rates. We perform our tests in 2D using a capusle shaped region (see Figure 3.3) which is angled at θ degrees from the x axis with length L and radius a. When L is large compared to a, we do not have as much cancellation of errors as when the domain is more circular. This allows us to truly test the convergence

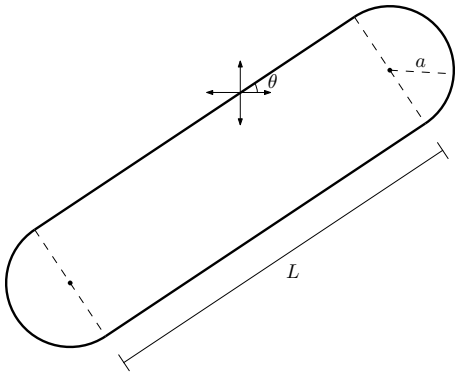


Figure 3.3: Capsule shape for testing the convergence using δ_ε.

properties of the approximation, δ_ε.

Finally, we are working with a mesh which is independent of the interface involved. The interface is defined implicitly as the zero level set of the function $d_h(\mathbf{x})$ and the uniform mesh is constructed independently. Note that this is part of the power of the level set method, that the interface we are tracking can move independent of the mesh. We are also running our simulations with a rectangular mesh as opposed to a triangular mesh.

3.4.1 Test cases

We demonstrate the above predicted rates for the dirac kernel $\psi^{1,4}$. We compute three quantities integrated around Γ for testing convergence. We approximate

$$\int_\Gamma f(\mathbf{x}) d\mathbf{x}$$

with $Q\left(f(\mathbf{x})\delta_\varepsilon(d_h(\mathbf{x}))|\nabla d_h(\mathbf{x})|\right)$ using $d_h(\mathbf{x}) = I_h d(\mathbf{x})$ the interpolant of the exact distance function onto $\mathbb{Q}^l(\mathcal{T}_h)$. The three functions $f(\mathbf{x})$ are

$$f_1(x, y) = 1, \tag{3.25}$$

which gives the perimeter of Γ, $|\Gamma| = 2L + 2\pi a$,

$$f_2(x, y) = xy \sin^2(x) \tag{3.26}$$

and

$$f_3(x, y) = x^2. \tag{3.27}$$

The last two are computed accurately to 16 decimal points using Maple for the various choices of θ, a and L of our capsule. In order to demonstrate convergence of this method regardless of the position of the interface with respect to the grid, we calculate the average error and the maximal error over multiple shifts in the grid. To be specific, for each mesh size h, we subdivide the interval $[0, h]$ into N pieces of size $\frac{h}{N}$ with starting point $x_i = i\frac{h}{N}$ for $i = 0 \dots N-1$. Then we use the product of this 1D subdivision to get N^d coordinates $\mathbf{x_c} = (x^1_{c_1}, x^2_{c_2}, \dots, x^d_{c_d})$ for $\mathbf{c} \in [0, \dots, N-1]^d$ which will be our center points for the grid. We shift Γ and f by $\mathbf{x_c}$ so that the value of the integral stays the same but the location of the gridpoints has been changed. The maximal error and the average error is computed over these N^d shifts and then the process is repeated on the next mesh size h.

3.4.2 Convergence rates

The rates are computed using the minimal mesh size per degree of freedom, (h/p). We adaptively refine our mesh so that we are fully refined in the ε-band around Γ_h and coarsen away from that region as fast as is computationally practical. Thus $E_i \approx ch^{\alpha_i}$ for

$$\alpha_i = \frac{\log(E_i/E_{i-1})}{\log\left(\frac{(h/p)_i}{(h/p)_{i-1}}\right)}.$$

We observe in these Tables 3.1-3.6 that the errors for integrating the three functions, f_1, f_2 and f_3, defined in equations (3.25) - (3.27), are indeed of desired order, $\mathcal{O}(h^{3/2})$, over

the two capsule shapes consistent with Theorem 3.3.1. The first three tables, Table 3.1-3.3 and Figure 3.4 give errors over the long skinny capsule with $\theta = \frac{\pi}{4}$, $L = 1.4$ and $a = 0.1\sqrt{2}$. The last three tables Tables 3.4-3.6 and Figure 3.5 give errors over the shorter and fatter capsule shape with $\frac{\pi}{6}$, $L = 1.0$ and $a = 0.2\sqrt{2}$.

When we compute the perimeter on both domains, where the analytic error does not contribute (see remark in Section 3.3.1.2) to the total error, then the quadrature and finite element errors dominate and are observed to be on a smaller scale. In addition they seem to be a little more erratic, which we hypothesize is due to how Γ cuts through the cells. However, it still give the desired order of convergence over the long run.

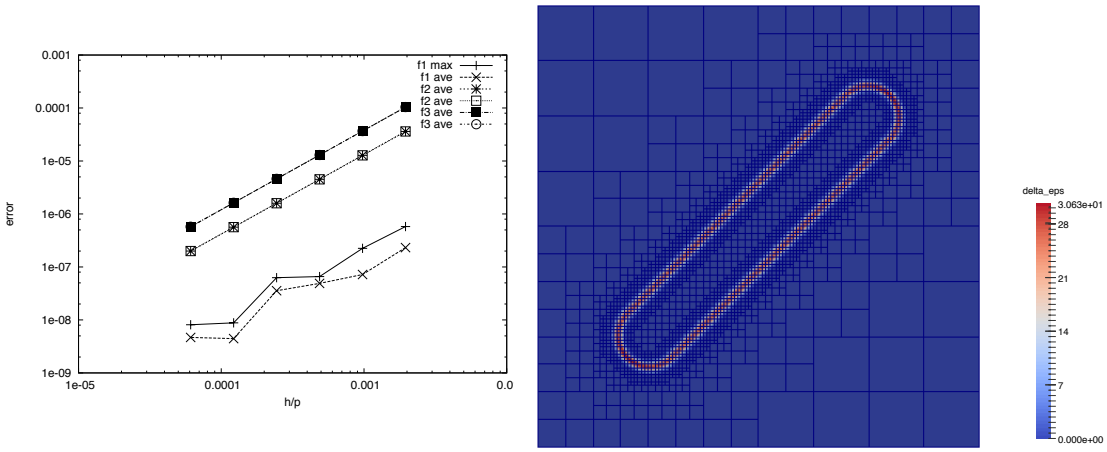


Figure 3.4: Error rates and plot of $\delta_\varepsilon(\mathbf{x})$ with fully refined $B_{h,\varepsilon}$ band around Γ_h. Note that the average and maximum errors for f_2 and f_3 are overlaid.

We observe in these Tables 3.1-3.6 that the errors for integrating the three functions, f_1, f_2 and f_3, defined in equations (3.25) - (3.27), are indeed of desired order, $\mathcal{O}(h^{3/2})$, over the two capsule shapes consistent with Theorem 3.3.1. The first three tables, Table 3.1-3.3 and Figure 3.4 give errors over the long skinny capsule with $\theta = \frac{\pi}{4}$, $L = 1.4$ and

Table 3.1: Capsule with $\theta = \frac{\pi}{4}$, $L = 1.4$ and $a = 0.1\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_1(\mathbf{x}) = 1$ to give perimeter.

cycle	dofs	h/p	f_1 MaxError	f_1 MaxRate	f_1 AveError	f_1 AveRate
0	72937	1.9531e-03	5.7904e-07	-	2.3168e-07	-
1	166405	9.7656e-04	2.2464e-07	1.3660	7.1941e-08	1.6872
2	378149	4.8828e-04	6.5877e-08	1.7698	4.9009e-08	0.5538
3	851457	2.4414e-04	6.2776e-08	0.0696	3.5658e-08	0.4588
4	1869457	1.2207e-04	8.8137e-09	2.8324	4.4432e-09	3.0046
5	4304141	6.1035e-05	8.1088e-09	0.1203	4.6835e-09	-0.0760

Table 3.2: Capsule with $\theta = \frac{\pi}{4}$, $L = 1.4$ and $a = 0.1\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_2(\mathbf{x}) = xy\sin^2(x)$.

cycle	dofs	h/p	f_2 MaxError	f_2 MaxRate	f_2 AveError	f_2 AveRate
0	72937	1.9531e-03	3.6134e-05	-	3.6093e-05	-
1	166405	9.7656e-04	1.2771e-05	1.5005	1.2761e-05	1.5000
2	378149	4.8828e-04	4.5130e-06	1.5007	4.5117e-06	1.5000
3	851457	2.4414e-04	1.5957e-06	1.4999	1.5951e-06	1.5000
4	1869457	1.2207e-04	5.6459e-07	1.4989	5.6396e-07	1.5000
5	4304141	6.1035e-05	1.9947e-07	1.5011	1.9939e-07	1.5000

$a = 0.1\sqrt{2}$. The last three tables Tables 3.4-3.6 and Figure 3.5 give errors over the shorter and fatter capsule shape with $\frac{\pi}{6}$, $L = 1.0$ and $a = 0.2\sqrt{2}$.

When we compute the perimeter on both domains, where the analytic error does not contribute (see remark in Section 3.3.1.2) to the total error, then the quadrature and finite element errors dominate and are observed to be on a smaller scale. In addition they seem to be a little more erratic, which we hypothesize is due to how Γ cuts through the cells. However, it still give the proper order of convergence.

Table 3.3: Capsule with $\theta = \frac{\pi}{4}$, $L = 1.4$ and $a = 0.1\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_3(\mathbf{x}) = x^2$.

cycle	dofs	h/p	f_3 MaxError	f_3 MaxRate	f_3 AveError	f_3 AveRate
0	72937	1.9531e-03	1.0405e-04	-	1.0390e-04	-
1	166405	9.7656e-04	3.6779e-05	1.5003	3.6736e-05	1.5000
2	378149	4.8828e-04	1.2996e-05	1.5008	1.2988e-05	1.5000
3	851457	2.4414e-04	4.5965e-06	1.4995	4.5921e-06	1.5000
4	1869457	1.2207e-04	1.6259e-06	1.4993	1.6236e-06	1.5000
5	4304141	6.1035e-05	5.7459e-07	1.5006	5.7401e-07	1.5000

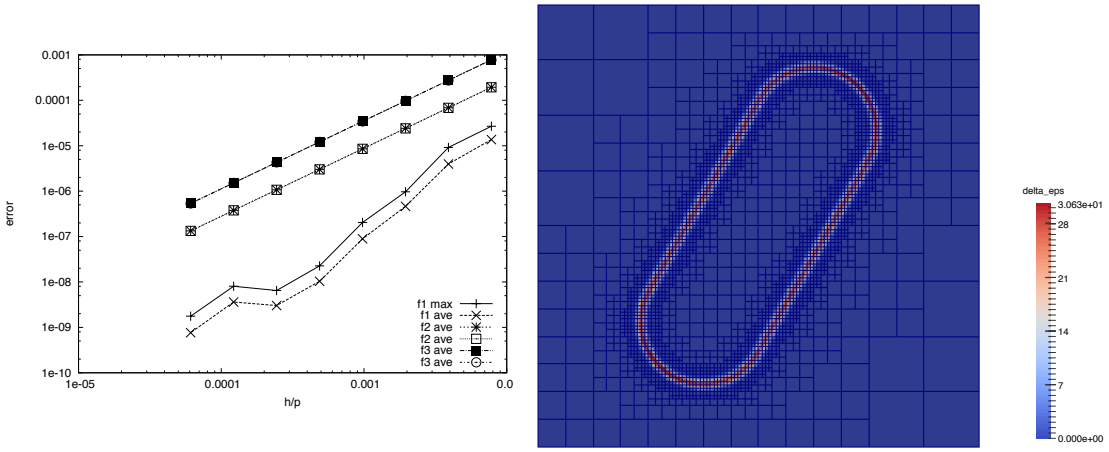


Figure 3.5: Error rates and plot of $\delta_\varepsilon(\mathbf{x})$ with fully refined $B_{h,\varepsilon}$ band around Γ_h. $\theta = \frac{\pi}{6}$, $L = 1.0$, $a = 0.2\sqrt{2}$, $\varepsilon = h^{3/4}$ using kernel $\psi^{1,4}(t)$.

Table 3.4: Capsule with $\theta = \frac{\pi}{6}$, $L = 1.0$ and $a = 0.2\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_1(\mathbf{x}) = 1$ to give perimeter.

cycle	dofs	h/p	f_1 MaxError	f_1 MaxRate	f_1 AveError	f_1 AveRate
0	14725	7.8125e-03	2.6859e-05	-	1.3647e-05	-
1	33622	3.9062e-03	9.1671e-06	1.5509	3.9492e-06	1.7889
2	74749	1.9531e-03	9.6633e-07	3.2471	4.6261e-07	3.0940
3	167081	9.7656e-04	2.0429e-07	2.2419	8.9133e-08	2.3758
4	375325	4.8828e-04	2.2573e-08	3.1780	1.0368e-08	3.1038
5	847725	2.4414e-04	6.4746e-09	1.8017	3.0246e-09	1.7773
6	1925989	1.2207e-04	8.0338e-09	-0.3113	3.6157e-09	-0.2575
7	4409953	6.1035e-05	1.7616e-09	2.1892	7.6421e-10	2.2422

Table 3.5: Capsule with $\theta = \frac{\pi}{6}$, $L = 1.0$ and $a = 0.2\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_2(\mathbf{x}) = xy\sin^2(x)$.

cycle	dofs	h/p	f_2 MaxError	f_2 MaxRate	f_2 AveError	f_2 AveRate
0	14725	7.8125e-03	1.9421e-04	-	1.9308e-04	-
1	33622	3.9062e-03	6.8589e-05	1.5016	6.8266e-05	1.5000
2	74749	1.9531e-03	2.4162e-05	1.5053	2.4136e-05	1.5000
3	167081	9.7656e-04	8.5404e-06	1.5003	8.5336e-06	1.5000
4	375325	4.8828e-04	3.0179e-06	1.5008	3.0171e-06	1.5000
5	847725	2.4414e-04	1.0669e-06	1.5001	1.0667e-06	1.5000
6	1925989	1.2207e-04	3.7742e-07	1.4992	3.7713e-07	1.5000
7	4409953	6.1035e-05	1.3341e-07	1.5003	1.3334e-07	1.5000

Table 3.6: Capsule with $\theta = \frac{\pi}{6}$, $L = 1.0$ and $a = 0.2\sqrt{2}$ using $\varepsilon = h^{3/4}$. Computation performed using 25 shifts of grid. We use $\mathbb{Q}^2(\mathcal{T}_h)$ and $d_h(\mathbf{x}) = I_h d(\mathbf{x})$, the Lagrange interpolant of $d(\mathbf{x})$. Gaussian quadrature is used with two points in each dimension which is exact for polynomials of degree 3. Integrating $f_3(\mathbf{x}) = x^2$.

cycle	dofs	h/p	f_3 MaxError	f_3 MaxRate	f_3 AveError	f_3 AveRate
0	14725	7.8125e-03	7.8148e-04	-	7.7608e-04	-
1	33622	3.9062e-03	2.7597e-04	1.5017	2.7439e-04	1.5000
2	74749	1.9531e-03	9.7131e-05	1.5065	9.7012e-05	1.5000
3	167081	9.7656e-04	3.4332e-05	1.5004	3.4299e-05	1.5000
4	375325	4.8828e-04	1.2130e-05	1.5009	1.2127e-05	1.5000
5	847725	2.4414e-04	4.2884e-06	1.5001	4.2874e-06	1.5000
6	1925989	1.2207e-04	1.5172e-06	1.4990	1.5158e-06	1.5000
7	4409953	6.1035e-05	5.3629e-07	1.5004	5.3592e-07	1.5000

4. TWO PHASE FLOW WITH INCOMPRESSIBLE NAVIER-STOKES

We are studying the effects of various forces that live on a smooth closed interface, modeled by a hyper-surface $\Gamma \subset \mathbb{R}^d$, between two fluids on the dynamic flow of that interface. Thus we will need to construct a global velocity field that must somehow incorporate the forces from that interface into its system. The fluids we are studying are moving at a slow enough speed and at pressures moderate enough to be considered incompressible. In addition the interface will be considered as if it were a solid barrier so that the two fluids are considered immiscible. For instance, if we were studying the motion of a red blood cell, the interface would be the cell vesicle itself and the two fluids would be the interior of the cell and the medium through which it is traveling. For small time scales, it is reasonable to assume no exchange between the cell cytoplasm and extra cellular fluids. In applications, we will typically assume that we are capturing the dynamics in only a portion of a larger region that includes the interface we are modeling and we are free to decide how large or that portion we are observing. Thus it is safe to assume that on the time scales we are considering, the interface, Γ will stay away from the boundary of our stated domain. There are many other applications that allow for Γ to intersect the boundary of the domain, but we will not study them here. The two fluids (or phases) will have their own density and viscosity which could be in general spatially variable, but for most applications, will be considered to be constant within their individual regions and with the jump at the interface.

With all these assumptions, we are left with a two-phase flow where the fluid velocity is described by the incompressible time dependent Navier-Stokes equations. We will obtain a velocity and pressure field that balance all the forces and will then update the interface according to that velocity field. The level set method described in Chapter 2 will be used

to transport the interface using the generated velocity field.

In Section 4.1, we will write down the general two-phase incompressible Navier-Stokes system. Then in Section 4.2, we will describe the discretization of time and space via the Rotational Incremental Pressure Correction model, which is a splitting of the general incompressible Navier-Stokes equations and which consists of determining the velocity field followed by a pressure correction. In Section 4.3, we will focus on the boundary conditions that are applied to the pressure and velocity fields and how their interaction affects the convergence rates. In Section 4.4 we will discuss the addition of an Streamlined Upwind/Petrov Galerkin stabilization scheme to the velocity equation and other techniques to stabilize and improve the discretized system behavior. Finally, in Section 4.5 we will demonstrate some numerical results based on the described algorithm. The numerical results described in Section 4.5 will be for a single phase flow but will show the expected rates for the incompressible Navier-Stokes rotational pressure correction algorithm. To be complete, we have included in Appendix B, a review of theory and expected convergence rates pertaining to the pressure correction splittings of the incompressible Navier-Stokes equations.

The specific application of forces on the interface will be included in their own chapters. We will study the application of surface tension in Chapter 6, of the Willmore force in Chapter 7 and of the Canham-Helfrich force in Chapter 8.

4.1 Incompressible Navier-Stokes

We consider a two-phase flow on a Lipschitz domain $\Lambda \subset \mathbb{R}^d$ with boundary $\partial\Lambda$. The domain, Λ is separated into two regions Ω and $\Lambda\backslash\overline{\Omega}$ where Ω is a smooth domain.

The fluid motion is described by its velocity $\mathbf{u} : [T_0, T_1] \times \Lambda \rightarrow \mathbb{R}^d$ and pressure $p : [T_0, T_1] \times \Lambda \rightarrow \mathbb{R}$. The fluid is assumed to be incompressible so that the velocity and pressure are related via the incompressible time dependent Navier-Stokes equations:

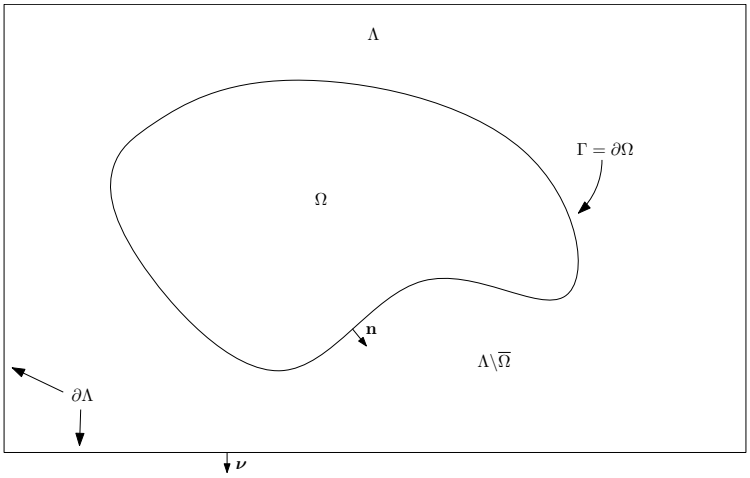


Figure 4.1: Region Λ including a subdomain Ω. The outside pointing normal to Ω is $\mathbf{n}$ while the outside pointing normal to Λ is $\boldsymbol{\nu}$.

$$
\begin{cases}
\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u}\right) - \nabla\cdot(2\mu\nabla^s\mathbf{u} - pI) = \rho\mathbf{f} + \mathbf{g} & \text{in } (T_0,T_1]\times\Lambda \\
\nabla\cdot\mathbf{u} = 0 & \text{in } (T_0,T_1]\times\Lambda \\
[\![2\mu\nabla^s\mathbf{u} - pI]\!]\cdot\mathbf{n} = \mathbf{f}_\Gamma & \text{on } (T_0,T_1]\times\Gamma \\
[\![\mathbf{u}]\!] = 0 & \text{on } (T_0,T_1]\times\Gamma \\
\mathbf{u} = \mathbf{u}_d & \text{on } (T_0,T_1]\times\partial\Lambda_d \\
(2\mu\nabla^s\mathbf{u} - pI)\cdot\boldsymbol{\nu} = \mathbf{f}_{\partial\Omega} & \text{on } (T_0,T_1]\times\partial\Lambda_n \\
\mathbf{u}\cdot\boldsymbol{\nu} = 0,\ \ ((2\mu\nabla^s\mathbf{u} - pI)\cdot\boldsymbol{\nu})\times\boldsymbol{\nu} = 0 & \text{on } (T_0,T_1]\times\partial\Lambda_s \\
\mathbf{u} = \mathbf{u}_0 & \text{on } \{T_0\}\times\Lambda
\end{cases}
\tag{4.1}
$$

Here we have used density, $\rho : [T_0,T_1]\times\Lambda\to\mathbb{R}$, the (dynamic) viscosity, $\mu : [T_0,T_1]\times\Lambda\to\mathbb{R}$, and the right hand side source terms, $\mathbf{f},\mathbf{g} : [T_0,T_1]\times\Lambda\to\mathbb{R}^d$. We denote by ∇^s, the symmetric gradient $\nabla^s\mathbf{u} := \frac{\nabla\mathbf{u}+\nabla\mathbf{u}^T}{2}$. We denote by $\mathbf{n}$, the outward pointing unit

normal vector on Γ and by $\boldsymbol{\nu}$, the outward pointing unit normal vector on $\partial\Lambda$.

We will allow for a wide range of boundary conditions, including Dirichlet, natural, slip ($\mathbf{u} \cdot \boldsymbol{\nu} = 0$ and $((2\mu\nabla^s\mathbf{u} - pI) \cdot \boldsymbol{\nu}) \times \boldsymbol{\nu} = \mathbf{0}$) or no-slip ($\mathbf{u} = 0$) boundary conditions, to be applied to velocity on $\partial\Lambda$. However, at this time we will not distinguish them more than Dirichlet, natural, and slip boundary conditions: $\partial\Lambda = \partial\Lambda_d \cup \partial\Lambda_n \cup \partial\Lambda_s$ where we can lump the no-slip type conditions into the set of Dirichlet-type conditions. These sets must be non overlapping with $\partial\Lambda_d$ a closed set. We will primarily use Dirichlet or slip conditions but leave open the opportunity of using natural boundary conditions as well. We allow for any combination of these three conditions and all three do not need to appear, but when pure Neumann boundary conditions are used, we may need to add an extra constraint to the system to maintain solvability. See Section 4.3.1 and in particular Section 4.3.2 for more details on the application of these boundary conditions. The initial velocity, $\mathbf{u}_0$, must be compatible with the chosen boundary conditions and be divergence free.

Then we denote by $\mathbf{u}_d : [T_0, T_1] \times \partial\Lambda_d \to \mathbb{R}^d$, the Dirichlet data and $\mathbf{f}_{\boldsymbol{\nu}} : [T_0, T_1] \times \partial\Lambda_n \to \mathbb{R}^d$, the natural boundary condition data.

We use the standard jump notation, using $\mathbf{n}$ as the outward unit normal to the surface we are jumping across,

$$[\![\mathbf{r}(\mathbf{x})]\!] := \lim_{\epsilon \to 0^+} \mathbf{r}(\mathbf{x} + \epsilon\mathbf{n}) - \lim_{\epsilon \to 0^+} \mathbf{r}(\mathbf{x} - \epsilon\mathbf{n}).$$

Then we denote by $\mathbf{f}_\Gamma : \Gamma \times [T_0, T_1] \to \mathbb{R}^d$ the forcing data on Γ which could represent surface tension or the Willmore force, or any other force which lives on the boundary, Γ, and which needs to be balanced against the forces of the Navier-Stokes system. The specific application of these forces and derivation of techniques to incorporate them into the discretized model will be the subject of the application chapters and are the main

subject of this thesis.

We will for simplicity assume that density and viscosity are constant within each of the two fluids with a possible jump at the interface, Γ. We then denote by ρ^{in} and μ^{in}, the density and viscosity on Ω, the interior region to Γ, and ρ^{out} and μ^{out} on $\Lambda \backslash \overline{\Omega}$, the region outside of Ω. If Γ is not closed, then we designate the region into which the normal $\mathbf{n}$ points, the outer region. We can combine these terms into a global definition using the characteristic function, $H(t, \mathbf{x})$ for $\Omega = \Omega(t)$ at time t,

$$\rho(t, \mathbf{x}) = H(t, \mathbf{x})\rho^{\text{in}} + (1 - H(t, \mathbf{x}))\, \rho^{\text{out}}$$
$$\mu(t, \mathbf{x}) = H(t, \mathbf{x})\mu^{\text{in}} + (1 - H(t, \mathbf{x}))\, \mu^{\text{out}}$$

where

$$H(t, \mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \Omega(t) \\ 0, & \text{otherwise.} \end{cases}$$

4.1.1 Weak form of continuous Navier-Stokes system

We define the following function spaces for the weak form of the continuous Navier-Stokes equations

$$\mathbb{V}^0 = \{\mathbf{v} \in \left[H^1(\Lambda)\right]^d \mid \mathbf{v} = 0 \text{ on } \partial\Lambda_d,\ \mathbf{v} \cdot \boldsymbol{\nu} = 0 \text{ on } \partial\Lambda_s\}$$
$$\mathbb{V} = \{\mathbf{v} \in \left[H^1(\Lambda)\right]^d \mid \mathbf{v} = \mathbf{u}_d \text{ on } \partial\Lambda_d,\ \mathbf{v} \cdot \boldsymbol{\nu} = 0 \text{ on } \partial\Lambda_s\}$$
$$\mathbb{M} = L^2(\Lambda)$$
$$\mathbb{X} = H^1(\Lambda)$$
$$\mathbb{X}^0 = \{\psi \in H^1(\Lambda) \mid \psi = 0 \text{ on } \partial\Lambda_n \cup \partial\Lambda_s\}.$$

Remark. The space $\mathbb{X}^0$ is defined here to keep the function space definitions located together but will only be used for the pressure correction variable, ψ, in the rotational incremental pressure correction splitting described in Section 4.2. *end Remark.*

Using these function spaces, a weak formulation of (4.1) reads: For a.e. $t \in [T_0, T_1]$, find $(\mathbf{u}(t), p(t)) \in \mathbb{V} \times \mathbb{M}$ such that for all $(\mathbf{v}, q) \in \mathbb{V}^0 \times \mathbb{M}$,

$$\begin{aligned}
&\int_\Lambda \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda 2\mu \nabla^s \mathbf{u} : \nabla^s \mathbf{v} d\mathbf{x} \\
&- \int_\Lambda p \operatorname{div} \mathbf{v} d\mathbf{x} + \int_\Lambda q \operatorname{div} \mathbf{u} d\mathbf{x} - \int_{\partial \Lambda_n} (2\mu \nabla^s \mathbf{u} - pI) \cdot \boldsymbol{\nu} \cdot \mathbf{v} d\mathbf{x} \\
&\qquad - \int_\Gamma ([\![2\mu \nabla^s \mathbf{u} - pI]\!] \cdot \mathbf{n}) \cdot \mathbf{v} d\mathbf{x} = \int_\Lambda (\rho \mathbf{f} + \mathbf{g}) \cdot \mathbf{v} d\mathbf{x}
\end{aligned}$$

which using our boundary conditions simplifies to

$$\begin{aligned}
&\int_\Lambda \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda 2\mu \nabla^s \mathbf{u} : \nabla^s \mathbf{v} d\mathbf{x} - \int_\Lambda p \operatorname{div} \mathbf{v} d\mathbf{x} \\
&+ \int_\Lambda q \operatorname{div} \mathbf{u} d\mathbf{x} = \int_\Lambda (\rho \mathbf{f} + \mathbf{g}) \cdot \mathbf{v} d\mathbf{x} + \int_{\partial \Lambda_n} \mathbf{f}_{\partial \Omega} \cdot \mathbf{v} d\mathbf{x} + \int_\Gamma \mathbf{f}_\Gamma \cdot \mathbf{v} d\mathbf{x}
\end{aligned} \tag{4.2}$$

At this stage, we recognize that when directly discretized in space and time, equation (4.2) leads to a nonlinear saddle point problem. There are many good approaches to solving the resulting problem but we will take a different route which will hopefully be more efficient for our needs. Instead of directly discretizing, we will split the system into a series of linear updates. This limits the convergence rates we can expect to at best second order in space, but that is adequate for our needs We will also use a second order in time backward difference discretization. We choose to work within the well known family of pressure correction models originally introduced by Chorin and Temam in [43] to solve the incompressible Navier-Stokes problem. In particular we implement the rotational, incremental pressure correction method suggested by Timmermans, Minev and Van De Vosse

in [44] with a modification based on a penalty point of view for dealing with variable density as discussed in [45] by Guermond and Salgado. This setup will obtain optimal order of convergence rates for velocity and pressure in the various norms desired. A good review of many of these projection type methods is given by Guermond, Minev and Shen in [46]. A review of the expected orders of convergence and related theorems from these types of time stepping methods is provided in Appendix B. We will discuss the necessary algorithmic details below. Refer to [47] and [21] for modifications to recover optimal rates in the case of natural boundary conditions on velocity and pressure.

4.1.2 A discretization in space

We consider a family of adaptive triangulations $\{\mathcal{T}(t)\}_{t\in[T_0,T_1]}$ of Λ consisting of regular rectangular (2D) or hexagonal elements (3D). For a specific $t \in [T_0, T_1]$, the element size of an element $K \in \mathcal{T}(t)$ is denoted h_K and calculated as the maximal distance between vertices divided by $\sqrt{d}$ so that the reference element has mesh size $h_{\hat{K}} = 1$. The minimal mesh size, $h = h_{\text{min}}$, is calculated over all cells in the triangulation,

$$h = h_{\text{min}} := \min_{K\in\mathcal{T}(t)} h_K.$$

We use the Taylor-Hood finite element for the velocity-pressure unknowns, that is we choose the d dimensional vector Lagrange finite element degree $p > 1$ on the reference element, $\widehat{K}$ for velocity and the scalar Lagrange finite element of degree $p - 1$ for the

pressure space. Thus we define the discrete spaces for $t \in [T_0, T_1]$,

$$\begin{aligned}
\mathbb{V}^0\left(\mathcal{T}(t)\right) = \{\mathbf{V} \in \left[C\left(\overline{\Lambda}\right)\right]^d \mid \forall K \in \mathcal{T}(t), \widehat{\mathbf{V}}\Big|_{\widehat{K}} \in \left[\mathbb{Q}^p(\widehat{K})\right]^d, \\
\mathbf{V}|_{\partial\Lambda_d} = \mathbf{0},\ \mathbf{V}\cdot\boldsymbol{\nu}|_{\partial\Lambda_s} = 0\} \\
\mathbb{V}\left(\mathcal{T}(t)\right) = \{\mathbf{V} \in \left[C\left(\overline{\Lambda}\right)\right]^d \mid \forall K \in \mathcal{T}(t), \widehat{\mathbf{V}}\Big|_{\widehat{K}} \in \left[\mathbb{Q}^p(\widehat{K})\right]^d, \\
\mathbf{V}|_{\partial\Lambda_d} = \mathbf{I}^p_{\mathcal{T}(t)}\left(\mathbf{f}_d\right),\ \mathbf{V}\cdot\boldsymbol{\nu}|_{\partial\Lambda_s} = 0\} \\
\mathbb{M}\left(\mathcal{T}(t)\right) = \{P \in C\left(\overline{\Lambda}\right) \mid \forall K \in \mathcal{T}(t), \widehat{P}\Big|_{\widehat{K}} \in \mathbb{Q}^{p-1}(\widehat{K})\} \\
\mathbb{X}\left(\mathcal{T}(t)\right) = \{P \in C\left(\overline{\Lambda}\right) \mid \forall K \in \mathcal{T}(t), \widehat{P}\Big|_{\widehat{K}} \in \mathbb{Q}^{p-1}(\widehat{K})\} \\
\mathbb{X}^0\left(\mathcal{T}(t)\right) = \{P \in \mathbb{X}\left(\mathcal{T}(t)\right) \mid P|_{\partial\Lambda_n \cup \partial\Lambda_s} = 0\}
\end{aligned}$$

where $\mathbf{I}^p_{\mathcal{T}(t)}$ is the Lagrange interpolant from continuous vector functions on vector piecewise continuous polynomials of degree p subordinate to the triangulation $\mathcal{T}(t)$ and the hat notation, $\widehat{\mathbf{V}}\Big|_{\widehat{K}}$, denotes the standard bi-linear pull back onto the reference element $\hat{K}$ from real element K.

We will denote functions in these discrete spaces by a capital letter notation. For the test functions, there is no explicit time dependence other than through the mesh that the polynomials are subordinate to, so when it is clear enough by context, we will drop the time notation. Thus our finite element solutions will be velocity, $\mathbf{U}(t) \in \mathbb{V}\left(\mathcal{T}(t)\right)$ and pressure, $P(t) \in \mathbb{M}\left(\mathcal{T}(t)\right)$ and the test functions will be respectively $\mathbf{V} \in \mathbb{V}^0\left(\mathcal{T}(t)\right)$ and $Q \in \mathbb{M}\left(\mathcal{T}(t)\right)$.

4.1.3 A discretization in time

We discretize our time $t \in [T_0, T_1]$ into N steps $t^0 = T_0 < t^1 < \cdots < t^k < t^{k+1} < \cdots < t^N = T_1$ with time step

$$\Delta t^{k+1} := t^{k+1} - t^k$$

We leave open the possibility for adaptive time stepping based on a CFL condition described in Section 2.3.2 but note that no stability condition has been discovered as of yet for the pressure correction algorithm with a higher order discretization of the time derivative. Stability has been shown for a BDF1 time discretization in [47], but we will in general use a higher order discretization. Thus in practice, we will choose a uniform time step when the pressure correction algorithm is being utilized.

We denote the functions at a specific time with a superscript,

$$\mathbf{u}^k := \mathbf{u}(t^k)$$
$$p^k := p(t^k).$$

Since the density and viscosity are possibly variable, we also denote

$$\rho^k := \rho\left(t^k\right)$$
$$\mu^k := \mu\left(t^k\right).$$

Additionally, we use the subscript notation $(f)_{\Delta t}$ to mean the sequence $\{f^k\}_{k=0}^N$ of solutions at the time steps t^k. With this notation, we can discuss norms in both time and space of our solutions. For instance, for any Banach space, $(B, \|\cdot\|_B)$, the maximum norm in time of our velocity approximation error $(\mathbf{u} - \mathbf{U})_{\Delta t}$ is

$$\|(\mathbf{u} - \mathbf{U})_{\Delta t}\|_{\ell^\infty(B)} := \max_{k=0}^{N} \left\|\mathbf{u}(t^k) - \mathbf{U}^k\right\|_B \tag{4.3}$$

and the ℓ^2 norm in time is

$$\|(\mathbf{u} - \mathbf{U})_{\Delta t}\|_{\ell^2(B)} := \left(\sum_{k=0}^{N} \Delta t^k \left\|\mathbf{u}\left(t^k\right) - \mathbf{U}^k\right\|_B^2\right)^{1/2}. \tag{4.4}$$

We do not distinguish notationally between norms of scalar functions and vector functions.

Finally, we combine the spatial and temporal discretizations to denote the finite element function spaces at time t^k. Using $\mathcal{T}^k := \mathcal{T}\left(t^k\right)$, we have

$$\mathbb{V}^0\left(\mathcal{T}^k\right), \mathbb{V}\left(\mathcal{T}^k\right) \text{ and } \mathbb{M}\left(\mathcal{T}^k\right).$$

4.1.3.1 Backward difference formulae for time derivatives

The backward difference formula (BDF) class of approximations to the time derivative can be described generally using the following notation: An m-th order (BDF-m) approximation of the first derivative in time uses a linear combination of the previous m time step solutions. It is convenient to use the operator $D_{t,m}$ to represent the BDF-m operation of first derivative in time, mainly

$$\frac{\partial f}{\partial t}(t, \mathbf{x})\bigg|_{t=t^{k+1}} = D_{t,m} f^{k+1} + \mathcal{O}\left((\Delta t^{k+1})^m\right) \tag{4.5}$$

and

$$D_{t,m} f^{k+1} = \frac{\beta_0}{\Delta t^{k+1}} f^{k+1} - \sum_{i=1}^{m} \frac{\beta_i}{\Delta t^{k+1}} f^{k+1-i}. \tag{4.6}$$

Remark. When it is clear from context which order, m, we want to use, we will drop the m from the notation and simply write $D_t f^{k+1}$ for the BDF time derivative approximation. *end Remark.*

The first order (BDF1) approximation has $\beta_0 = \beta_1 = 1$ for uniform or adaptive time steps so that

$$D_{t,1} f^{k+1} = \frac{f^{k+1} - f^k}{\Delta t^{k+1}}. \tag{4.7}$$

The second order (BDF2) approximation with uniform time steps has $\beta_0 = 3/2$, $\beta_1 = 2$

and $\beta_2 = -1/2$, so that

$$D_{t,2} f^{k+1} = \frac{3}{2\Delta t} f^{k+1} - \frac{2}{\Delta t} f^k + \frac{1}{2\Delta t} f^{k-1}. \tag{4.8}$$

For the second order (BDF2) adaptive time stepping, we define the ratio

$$r := \frac{\Delta t^{k+1}}{\Delta t^k} \tag{4.9}$$

then let $\beta_0 = \frac{1+2r}{(1+r)}$, $\beta_1 = 1 + r$, $\beta_2 = -\frac{r^2}{(1+r)}$ giving

$$D_{t,2} f^{k+1} = \frac{1+2r}{1+r} \frac{1}{\Delta t^{k+1}} f^{k+1} - (1+r) \frac{1}{\Delta t^{k+1}} f^k + \frac{r^2}{1+r} \frac{1}{\Delta t^{k+1}} f^{k-1}. \tag{4.10}$$

It is a simple calculation to show that this adaptive algorithm reduces to the uniform version above with uniform time steps, i.e. $r = 1$. There are higher order BDF formulae, but we will not use them here.

The adaptive BDF1 and BDF2 algorithms have truncation errors $\mathcal{O}\left(\Delta t^{k+1}\right)$ and $\mathcal{O}\left(\left(\Delta t^k + \Delta t^{k+1}\right) \Delta t^{k+1}\right)$ respectively, where as the uniform algorithms are $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta t^2)$ respectively.

4.1.4 Tracking the interface, Γ

We use the level set method described in Chapter 2 to track the interface, Γ as it moves in time. That is, at any given time, we have a level set function $\varphi(t, x) : [T_0, T_1] \times \Lambda \to \mathbb{R}$ such that $\Omega(t) = \{\mathbf{x} \in \Lambda \mid \varphi(t, \mathbf{x}) > 0\}$ and $\Gamma(t) := \{\mathbf{x} \in \Lambda \mid \varphi(t, \mathbf{x}) = 0\}$. We will keep the level set function as a signed distance function in a neighborhood of Γ so that the value of $\varphi(t, \mathbf{x})$ describes roughly how far from $\Gamma(t)$ we are. Outside a band, we will let it taper off to flat since we don't need to use distances except near the band. This level set function will be provided or solved for independently of the Navier-Stokes method. It

is discussed in Chapter 2. Using, $\varphi(t,x)$, we can make some simple modifications to our density and viscosity as well. In particular, we smooth out the jump at Γ, with a smoothed Heaviside function effectively creating a small ε-band where mixing of the fluids occurs: Given $\varepsilon > 0$, define the approximate Heaviside function,

$$H_\varepsilon(t,\mathbf{x}) := H_\varepsilon\left(\frac{\varphi(t,\mathbf{x})}{\varepsilon}\right) \tag{4.11}$$

or in short hand, $H_\varepsilon(\varphi(t,\mathbf{x})) = H_\varepsilon(t,\mathbf{x})$ where for $s \in \mathbb{R}$, the Heaviside kernel is defined as

$$H_\varepsilon(s) = \begin{cases} 0, & s \leq -1 \\ \frac{1}{2} + \frac{693}{512}(s - \frac{5}{3}s^3 + 2s^5 - \frac{10}{7}s^7 + \frac{5}{9}s^9 - \frac{1}{11}s^{11}), & |s| < 1 \\ 1, & 1 \leq s. \end{cases} \tag{4.12}$$

so that $H'_\varepsilon(s) = \delta_\varepsilon(s)$. We will let ε scale with the minimal mesh size $\varepsilon = ch_{\min}$, with $c \in \{1, 2, ...\}$ a positive integer. Then our two-phase smoothed density and viscosity are given by

$$\rho^k(\mathbf{x}) = \rho^+ H_\varepsilon(t^k,\mathbf{x}) + \rho^- \left(1 - H_\varepsilon(t^k,\mathbf{x})\right) \tag{4.13}$$

$$\mu^k(\mathbf{x}) = \mu^+ H_\varepsilon(t^k,\mathbf{x}) + \mu^- \left(1 - H_\varepsilon(t^k,\mathbf{x})\right) \tag{4.14}$$

Notice that the density and viscosity are piecewise constant everywhere except for the small smoothed region around the boundary. For most cases, it is can be treated them as if they are completely piecewise constant. If there are other advanced modifications to the model, for instance a nonlinear shear thinning of the viscosity, then the above definitions will need to be modified accordingly and it may become truly variable.

4.2 The rotational incremental pressure-correction model

We now describe a version of the rotational, incremental pressure correction splitting with variable density as discussed in [45]. We will initially describe the splitting algorithm using discrete in time but non discrete in space functions, then at the end, we will write down the fully discrete algorithm with the appropriate discrete Lagrange polynomial spaces. Now, the splitting we use separates the direct dependence of the velocity on the pressure by first solving for an intermediate velocity that might not be divergence free and which is based on the previous pressure. Then we update it to be divergence free by using a Helmholtz-like decomposition into a solenoidal (or more simply a divergence free) term and a gradient. We then use those terms to correct the previous pressure to be compatible with the current divergence free velocity; hence the name pressure correction model.

So supposing that we have $\mathbf{u}^k$ and $\mathbf{u}^{k-1}$, divergence free velocity solutions at times t^k and t^{k-1}, we first seek the incremental velocity $\tilde{\mathbf{u}}^{k+1} \in \mathbb{V}\left(\mathcal{T}^{k+1}\right)$ which is not necessarily divergence free. We use the BDF2 time derivative, $D_{\tau,2}\tilde{\mathbf{u}}^{k+1} = \frac{\beta_0}{\Delta t^{k+1}}\tilde{\mathbf{u}}^{k+1} - \frac{\beta_1}{\Delta t^{k+1}}\mathbf{u}^k - \frac{\beta_2}{\Delta t^{k+1}}\mathbf{u}^{k-1}$ and linearize the convection term to be $\left(\mathbf{u}^* \cdot \nabla\right)\tilde{\mathbf{u}}^{k+1}$, where we have introduced the second order linear extrapolation at time t^{k+1} of the velocity

$$\mathbf{u}^* := \mathbf{u}^k + \frac{\Delta t^{k+1}}{\Delta t^k}\left(\mathbf{u}^k - \mathbf{u}^{k-1}\right)$$

using the previous divergence free velocities. This extrapolated velocity will also be used as a proxy for the current velocity when calculating various scaling coefficients thus preserving linearity or our system.

Thus, the first step is to find $\tilde{\mathbf{u}}^{k+1} \in \mathbb{V}$ such that for all $\mathbf{v} \in \mathbb{V}^0$:

$$\begin{aligned}\int_\Lambda \rho^{k+1} \frac{\beta_0}{\Delta t^{k+1}} \tilde{\mathbf{u}}^{k+1} \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda \rho^{k+1} \left(\mathbf{u}^* \cdot \nabla\right) \tilde{\mathbf{u}}^{k+1} \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda 2\mu^{k+1} \nabla^s \tilde{\mathbf{u}}^{k+1} : \nabla^s \mathbf{v} d\mathbf{x} \\ = \int_\Lambda p^k \operatorname{div} \mathbf{v} d\mathbf{x} + \int_\Lambda \rho^{k+1} \left(\frac{\beta_1}{\Delta t^{k+1}} \mathbf{u}^k d\mathbf{x} + \frac{\beta_2}{\Delta t^{k+1}} \mathbf{u}^{k-1} \right) \cdot \mathbf{v} d\mathbf{x} \\ + \int_\Lambda \left(\rho^{k+1} \mathbf{f}^{k+1} + \mathbf{g}^{k+1} \right) \cdot \mathbf{v} d\mathbf{x} + \int_{\partial\Lambda_n} \mathbf{f}_{\boldsymbol{\nu}}^{k+1} \cdot \mathbf{v} d\mathbf{x} + \int_\Gamma \mathbf{f}_\Gamma^{k+1} \cdot \mathbf{v} d\mathbf{x}.\end{aligned} \tag{4.15}$$

Next we seek the divergence free velocity $\mathbf{u}^{k+1} \in \mathbb{V}$, and the pressure correction, $\psi^{k+1} \in \mathbb{M}$ as solution to

$$\begin{cases} \frac{\beta_0}{\Delta t^{k+1}} \left(\mathbf{u}^{k+1} - \tilde{\mathbf{u}}^{k+1} \right) + \frac{1}{\rho^{k+1}} \nabla \psi^{k+1} = 0 \\ \nabla \cdot \mathbf{u}^{k+1} = 0 \\ \mathbf{u}^{k+1} \cdot \boldsymbol{\nu} \big|_{\partial\Lambda_d} = 0 \end{cases} \tag{4.16}$$

and finally we seek the pressure, $p^{k+1} \in \mathbb{M}$, as solution to the pressure update with rotational term:

$$p^{k+1} = p^k + \psi^{k+1} - \left(\min_\Lambda \mu^{k+1}(\mathbf{x}) \right) \operatorname{div} \tilde{\mathbf{u}}^{k+1}. \tag{4.17}$$

As written, this system is not directly implementable since we need both $\mathbf{u}^{k+1}$ and $\tilde{\mathbf{u}}^{k+1}$ to solve for update ψ^{k+1}. However we can simplify by applying the divergence operator to equation system (4.16) and use the divergence free properties of $\mathbf{u}^{k+1}$ to reduce the system to only use the $\tilde{\mathbf{u}}^{k+1}$ and ψ^{k+1} variables:

$$\begin{cases} -\nabla \cdot \left(\frac{1}{\rho^{k+1}} \nabla \psi^{k+1} \right) = -\frac{\beta_0}{\Delta t^{k+1}} \operatorname{div} \tilde{\mathbf{u}}^{k+1}, & \mathbf{x} \in \Lambda \\ \frac{\partial \psi^{k+1}}{\partial \boldsymbol{\nu}} = 0, & \mathbf{x} \in \partial\Lambda_d \\ \psi^{k+1} = 0, & \mathbf{x} \in \partial\Lambda_n. \end{cases} \tag{4.18}$$

Note that by our assumption of variable density, we cannot simply pull out the ρ^{k+1} term

as is common in the literature. The weak form of equation system (4.18) is then the non-constant coefficient Poisson equation given as: Find $\psi^{k+1} \in \mathbb{X}^0$ which satisfies for all $w \in \mathbb{X}^0$ the weak equality

$$\int_\Lambda \nabla \left(\frac{1}{\rho^{k+1}} \psi^{k+1} \right) \cdot \nabla w d\mathbf{x} = -\frac{\beta_0}{\Delta t^{k+1}} \int_\Lambda \operatorname{div} \tilde{\mathbf{u}}^{k+1} w d\mathbf{x}. \tag{4.19}$$

Remark. Note that we are imposing homogeneous Dirichlet boundary conditions on the pressure correction function in exactly the opposite locations of the Dirichlet boundary conditions of the velocity component; that is, in the Neumann and slip regions. This turns out to be a naturally compatible condition between pressure and velocity (See [46]). However it does in fact add a boundary layer to pressure where it is imposed and we do not recover the full convergence rate in the L^2-norm of pressure and H^1-norm of velocity. When open boundary conditions are used on velocity, we can modify the pressure correction system to also have open boundary conditions to avoid this boundary layer and recover the full convergence rates. See Section 4.3.2 for more details. *end Remark.*

4.2.1 The pressure correction algorithm

In summary, the basic BDF2 splitting algorithm with rotational form is as follows: Given $(\mathbf{u}^k, \mathbf{u}^{k-1}, p^k, p^{k-1}, \rho^{k+1}, \mu^{k+1})$, solve equation (4.15) for $\tilde{\mathbf{u}}^{k+1}$, then solve equation (4.19) for ψ^{k+1} which we use to update p^{k+1} in (4.17). Finally we update $\mathbf{u}^{k+1}$ from equation (4.16) as

$$\mathbf{u}^{k+1} = \tilde{\mathbf{u}}^{k+1} - \frac{\Delta t^{k+1}}{\beta_0} \nabla \left(\frac{1}{\rho^{k+1}} \psi^{k+1} \right). \tag{4.20}$$

After this sequence of steps, we have our desired velocity, $\mathbf{u}^{k+1}$, and pressure, p^{k+1}.

Remark. Note that the literature also calls this a projection method because under the

assumption of constant density, equation (4.20) can be rearranged as

$$\tilde{\mathbf{u}}^{k+1} = \mathbf{u}^{k+1} + \frac{\Delta t^{k+1}}{\rho^{k+1}\beta_0}\nabla\psi^{k+1} \tag{4.21}$$

and we can think of $\mathbf{u}^{k+1}$ as the projection of $\tilde{\mathbf{u}}^{k+1}$ into the divergence-free subset of velocities. *end Remark.*

In the next section, we will discuss some additional improvements and simplifications for our actual algorithm and implementation that take us away from the projection idea.

4.2.2 A penalty method instead of a projection method

In [45] by Guermond, it is pointed out that instead of thinking of the incremental pressure correction scheme as a projection method, we can view it as a linear perturbation method. Rannacher then pointed out in [48] that this can be viewed as a penalty method on the divergence of the velocity. Once we have this point of view, we are not bound in the pressure correction equation (4.16) to use the variable ρ^{k+1} but can introduce instead a penalty scaling factor

$$\chi \in \left(0, \min_{\mathbf{x}\in\Lambda} \rho(0,\mathbf{x})\right]. \tag{4.22}$$

This range is chosen solely based on the stability analysis. In practice we almost always choose

$$\chi = \rho^0_{\min} := \min_{\mathbf{x}\in\Lambda} \rho(0,\mathbf{x}). \tag{4.23}$$

Thus our pressure correction equation (4.19) can be rewritten in the form

$$\int_\Lambda \nabla\psi^{k+1}\cdot\nabla w d\mathbf{x} = -\frac{\beta_0\chi}{\Delta t^{k+1}}\int_\Lambda \operatorname{div}\tilde{\mathbf{u}}^{k+1} w d\mathbf{x}. \tag{4.24}$$

See [45] for the complete details.

4.2.3 Summary of fully-discrete pressure correction algorithm

We must also start to distinguish between β_0 values at different time steps if we are using adaptive time stepping. At time t^k, we denote by β_0^k the β_0 value at that time. When no superscript is given, it is assumed to be at the current time value, t^{k+1}. It might be just as convenient to simply use β_0, β_0^{old} and β_0^{oldold} when sitting at time t^{k+1} to denote β_0's at times t^{k+1}, t^k and t^{k-1} respectively.

It is observed in the "Implementation" section of [46] that there is some ambiguity about which velocity $\mathbf{u}^{k+1}$ or $\tilde{\mathbf{u}}^{k+1}$ is really the right velocity. It is possible to adjust the entire system (except for the nonlinear convection term) to use the incremental velocities, $\tilde{\mathbf{u}}^{k+1}$. We choose to always use the incremental velocity and never solve for the divergence free velocities, $\mathbf{u}^{k+1}$. We will remove the divergence free velocity terms $\mathbf{u}^k$ and $\mathbf{u}^{k-1}$ from equation (4.15) and replace them with the corresponding $\tilde{\mathbf{u}}$ velocity terms using the relations

$$\rho^{k+1}\frac{\beta_i}{\Delta t^{k+1}}\mathbf{u}^{k+1-i} = \rho^{k+1}\frac{\beta_i}{\Delta t^{k+1}}\tilde{\mathbf{u}}^{k+1-i} - \frac{\Delta t^{k+1-i}}{\Delta t^{k+1}}\frac{\beta_i}{\beta_0^{k+1-i}}\nabla\psi^{k+1-i}, \qquad \text{for } i = 1, 2$$

which are scaled versions of equation (4.16) at previous time steps.

The only question that remains is how to handle the nonlinear convection term. We will implement a semi-implicit linearization of the $\mathbf{u}^{k+1}\cdot\nabla\tilde{\mathbf{u}}^{k+1}$ using second order extrapolations from previous incremental velocities. It is shown in [49] that this still leads to a stable convergent solution. To this end, we define the second order extrapolation velocity

$$\tilde{\mathbf{u}}^* := \tilde{\mathbf{u}}^k + \frac{\Delta t^{k+1}}{\Delta t^k}\left(\tilde{\mathbf{u}}^k - \tilde{\mathbf{u}}^{k-1}\right)$$

and use the semi implicit $\tilde{\mathbf{u}}^*\cdot\nabla\tilde{\mathbf{u}}^{k+1}$ for the convection term.

At this point, we have completely removed the $\mathbf{u}$ terms and are left only with $\tilde{\mathbf{u}}$ terms.

We will use the incremental velocity everywhere as if it were our true velocity field, so we will drop the tilde notation from here forward as it will always denote a $\tilde{\mathbf{u}}$ term in the subsequent. We will also switch to the fully discrete in space solutions. Thus we seek a velocity $\mathbf{U}^{k+1} \in \mathbb{V}\left(\mathcal{T}^{k+1}\right)$, a pressure correction $\Psi^{k+1} \in \mathbb{X}^0\left(\mathcal{T}^{k+1}\right)$ and a pressure $P^{k+1} \in \mathbb{M}\left(\mathcal{T}^{k+1}\right)$ which are solved in the three sequential steps:

(i) Velocity Prediction: Given $(\mathbf{U}^k, \mathbf{U}^{k-1}, P^k, P^{k-1}, \Psi^k, \Psi^{k-1})$, find $\mathbf{U}^{k+1} \in \mathbb{V}\left(\mathcal{T}^{k+1}\right)$ that satisfies for all $\mathbf{V} \in \mathbb{V}^0\left(\mathcal{T}^{k+1}\right)$,

$$\begin{aligned}
&\int_\Lambda \rho^{k+1} \frac{\beta_0}{\Delta t^{k+1}} \mathbf{U}^{k+1} \cdot \mathbf{V} d\mathbf{x} + \int_\Lambda \rho^{k+1} \left(\mathbf{U}^* \cdot \nabla\right) \mathbf{U}^{k+1} \cdot \mathbf{V} d\mathbf{x} \\
&\qquad + \int_\Lambda 2\mu^{k+1} \nabla^s \mathbf{U}^{k+1} : \nabla^s \mathbf{V} d\mathbf{x} - \int_\Gamma \mathbf{f}_\Gamma^{k+1} \cdot \mathbf{V} d\mathbf{x} \\
&= \int_\Lambda \left(P^k + \frac{\Delta t^k}{\Delta t^{k+1}} \frac{\beta_1}{\beta_0^k} \Psi^k + \frac{\Delta t^{k-1}}{\Delta t^{k+1}} \frac{\beta_2}{\beta_0^{k-1}} \Psi^{k-1} \right) \operatorname{div} \mathbf{V} d\mathbf{x} \\
&\qquad + \int_\Lambda \rho^{k+1} \left(\frac{\beta_1}{\Delta t^{k+1}} \mathbf{U}^k + \frac{\beta_2}{\Delta t^{k+1}} \mathbf{U}^{k-1} \right) \cdot \mathbf{V} d\mathbf{x} \\
&\qquad + \int_\Lambda \left(\rho^{k+1} \mathbf{f}^{k+1} + \mathbf{g}^{k+1} \right) \cdot \mathbf{V} d\mathbf{x} + \int_{\partial\Lambda_n} \mathbf{f}_{\boldsymbol{\nu}}^{k+1} \cdot \mathbf{V} d\mathbf{x}.
\end{aligned} \tag{4.25}$$

(ii) Pressure Increment Correction: Now, given $\mathbf{U}^{k+1}$, find $\Psi^{k+1} \in \mathbb{X}^0\left(\mathcal{T}^{k+1}\right)$ such that for all $W \in \mathbb{X}^0\left(\mathcal{T}^{k+1}\right)$,

$$\int_\Lambda \nabla \Psi^{k+1} \cdot \nabla W d\mathbf{x} = -\frac{\beta_0 \chi}{\Delta t^{k+1}} \int_\Lambda \operatorname{div} \mathbf{U}^{k+1} W d\mathbf{x}. \tag{4.26}$$

(iii) Pressure Update with Rotational Correction: Given $\mathbf{U}^{k+1}$ and Ψ^{k+1}, find $P^{k+1} \in \mathbb{M}\left(\mathcal{T}^{k+1}\right)$ such that for all $Q \in \mathbb{M}\left(\mathcal{T}^{k+1}\right)$,

$$\int_\Lambda P^{n+1} Q d\mathbf{x} = \int_\Lambda \left(P^k + \Psi^{k+1}\right) Q d\mathbf{x} - \min_{\mathbf{x}\in\Lambda} \mu^{k+1}(\mathbf{x}) \int_\Lambda \operatorname{div} \mathbf{U}^{k+1} Q d\mathbf{x} \tag{4.27}$$

With these three steps, we have our new solution $(\mathbf{U}^{k+1}, \Psi^{k+1}, P^{k+1})$.

Remark. We call the algorithm **rotational** when we add $-\min_{\mathbf{x}\in\Lambda}\mu^{k+1}(\mathbf{x})\operatorname{div}\mathbf{U}^{k+1}$ to the pressure update equation (4.17) and in weak form, equation (4.27). The **standard** method leaves this term off and only updates pressure with the pressure correction term, Ψ^{k+1}. See Appendix B for a summary of theory and expect convergence results related to the rotational and standard algorithms with BDF1 and BDF2 time derivative approximations. *end Remark.*

Remark. Finally, we point out that in the case of uniform time steps, the pressure term

$$P^\sharp := \left(P^k + \frac{\Delta t^k}{\Delta t^{k+1}}\frac{\beta_1}{\beta_0^k}\Psi^k + \frac{\Delta t^{k-1}}{\Delta t^{k+1}}\frac{\beta_2}{\beta_0^{k-1}}\Psi^{k-1}\right)$$

of equation (4.25) simplifies to

$$\begin{aligned} P^\sharp &= \frac{7}{3}P^k - \frac{5}{3}P^{k-1} + \frac{1}{3}P^{k-2} \\ &= \frac{1}{3}\left(3P^k - 3P^{k-1} + P^{k-2}\right) + \frac{2}{3}\left(2P^k - P^{k-1}\right) \\ &\approx \frac{1}{3}P^{k+1} + \frac{2}{3}P^{k+1} \\ &\approx P^{k+1} \end{aligned}$$

which is a convex combination of a second and third order extrapolations of P^{k+1}. We have in this calculation ignored the divergence of velocity terms that come from the rotational adjustment. When considered separately and combined via Taylor expansions, they can be viewed as an extra consistent penalty term on the divergence of velocity pushing us towards a divergence free velocity:

$$\frac{4}{3}\mu_{\min}\operatorname{div}\mathbf{U}^k - \frac{1}{3}\mu_{\min}\operatorname{div}\mathbf{U}^{k-1} \approx \mu_{\min}\left(\operatorname{div}\mathbf{U}^{k+1} - \frac{2}{3}\Delta t\operatorname{div}\mathbf{U}_t^{k+1} + \mathcal{O}(\Delta t^3)\right).$$

Thus, by switching to the incremental velocities everywhere, we have improved our pressure-

velocity coupling by using a better approximation to the current pressure in the momentum equation (4.25). *end Remark.*

4.3 Boundary conditions on velocity and pressure increment

There are a number of configurations of boundary conditions on the various functions. It turns out that there is some flexibility in what we want to do and how we model the boundaries. Recall we defined $\partial\Lambda = \partial\Lambda_d \cup \partial\Lambda_n \cup \partial\Lambda_s$ which represented where the Dirichlet (d), Natural (n) and Slip(s) boundary conditions for velocity occurred. Some configurations introduce an artificial boundary layer in pressure while others do not. We review the possible configurations and then discuss how we can obtain optimal order of convergence for each. In the subsequent, we will discuss them as if the entire boundary is either $\partial\Lambda_d$, $\partial\Lambda_n$ or $\partial\Lambda_s$ but in reality this works for mixed boundary conditions and the choices just need to be consistent on each part.

4.3.1 Standard boundary conditions

The first approach is the standard approach in most papers. We have two options for velocity, either a Dirichlet condition or a natural/slip boundary condition. The basic approach says that whichever one we choose, we apply the opposite condition on the pressure correction, ψ term. The follow convergence results hold for the rotational scheme with uniform time step and BDF2 time derivative. We lump the slip and no slip boundary conditions here into the Dirichlet type conditions. Thus if a slip/no-slip condition is applied on the velocity, the pressure correction should have a Natural condition applied. In general we do not consider a slip/no-slip condition on the pressure correction function.

$\mathbf{U}$	Ψ	$\left\|(\mathbf{u}-\mathbf{U})_{\Delta t}\right\|_{\ell^\infty(L^2(\Lambda)^d)}$	$\left\|(\mathbf{u}-\mathbf{U})_{\Delta t}\right\|_{\ell^\infty(H^1(\Lambda)^d)}$	$\left\|(p-P)_{\Delta t}\right\|_{\ell^\infty(L^2(\Lambda))}$
Dirichlet= $\mathbf{u}_d$/Slip/No Slip	Natural = 0	Δt^2	$\Delta t^{3/2}$	$\Delta t^{3/2}$
Natural = $\mathbf{f}_\nu$	Dirichlet = 0	Δt^2	Δt	Δt

The second case is precisely where an artificial boundary layer is created on the pressure boundary, $\partial\Lambda$. We are not so concerned with this since we will be tracking our boundary, Γ far away from $\partial\Lambda$ so the distortions from this boundary layer will have minimal effect on our accuracy. In other words, even though globally we may have a reduced accuracy, we are only concerned with the solution near the boundary $\partial\Omega$ and inside of it where the boundary layer has not affected the solution. Thus we still consider this technique to be viable for our situation.

4.3.1.1 Mean value of pressure

Note that when pure Dirichlet boundary conditions have been used for the velocity, $\mathbf{u}^{k+1}$, and pure Neumann boundary conditions are used on the pressure correction, ψ^{k+1}, then ψ^{k+1} is defined only up to a constant. This means that the pressure, p^{k+1}, is also defined up to a constant. We choose in this case to set the mean value of pressure to be 0 or if we are doing a convergence analysis with known pressure, we set the average value of our pressure equal to the average value of the known pressure. This allows us to observe the predicted L^2-norm convergence rates for pressure. This means that we must remove the null space of constants in some manner to compute a solution for pressure correction and then calculate the pressure update using this solution. There are solvers in the PETSc linear algebra library that will do this automatically, or we can fix one of the degrees of freedom and solve the full rank subsystem using any linear algebra solver. See [50] for a discussion of various methods to solve the Laplacian with Neumann data problem.

Once we have computed the pressure, we then compute it's mean value and subtract it off to reach the desired mean value of pressure. Then we need to adjust the pressure correction to match. That is, if we adjusted the mean value of pressure with

$$p^{k+1} \to p^{k+1} - c$$

then taking our cues from equation (4.17), which we rearrange to be

$$\psi^{k+1} = p^{k+1} - p^k + \left(\min_{\mathbf{x}\in\Lambda} \mu^{k+1}(\mathbf{x}) \right) \operatorname{div} \mathbf{u}^{k+1},$$

we must also adjust the pressure correction

$$\psi^{k+1} \to \psi^{k+1} - c.$$

Thus, we maintain the relationship between all the splitting variables $\mathbf{u}^{k+1}$, ψ^{k+1} and p^{k+1} for the next step. It is always possible to adjust the mean values of pressure and pressure correction in this manner for any combination of boundary conditions but it is necessary in the case of pure Dirichlet on the velocity and pure Neumann on the pressure correction.

4.3.2 Open boundary conditions everywhere

Another possible scenario for the second case is to try to remove the boundary layer by using natural boundary conditions on ψ as well. However for stability reasons we need to add some terms to each equation. Thus if we choose open or natural boundary conditions on both velocity and the pressure increment, then we add a grad-div term to the left hand side of the momentum equation (4.25)

$$- \alpha_0 \nabla \left(\operatorname{div} \left(\frac{\mathbf{U}^{k+1} - \mathbf{U}^k}{\Delta t^{k+1}} \right) \right) \tag{4.28}$$

or in weak form,

$$+ \int_\Lambda \alpha_0 \operatorname{div} \left(\frac{\mathbf{U}^{k+1} - \mathbf{U}^k}{\Delta t^{k+1}} \right) \cdot \operatorname{div} \mathbf{V} d\mathbf{x} \tag{4.29}$$

and a zeroth order term to our pressure correction equation (4.26) with constant, $c > 0$

$$\int_\Lambda \nabla \Psi^{k+1} \cdot \nabla W + c \Psi^{k+1} W d\mathbf{x} = -\frac{\beta_0 \chi}{\Delta t^{k+1}} \int_\Lambda \operatorname{div} \mathbf{U}^{k+1} W d\mathbf{x} \tag{4.30}$$

where $\alpha_0 \geq 1$, is a user defined constant. We note that this grad-div term is consistent since the continuous equations have incompressible velocity i.e. divergence of velocity is zero. We add this term simply to give a bound on the $\|\Psi^{k+1} - \Psi^k\|_{H^1(\Lambda)}$ term which arises in the error analysis of the open boundary system. The momentum equation is then supplemented with the new boundary conditions, for $\mathbf{x} \in \partial\Lambda_n$

$$\left[2\mu\nabla^s \mathbf{U}^{k+1} - \left(P^k - \frac{\Delta t^k}{\Delta t^{k+1}}\frac{\beta_1}{\beta_0^k}\Psi^k - \frac{\Delta t^{k-1}}{\Delta t^{k+1}}\frac{\beta_2}{\beta_0^{k-1}}\Psi^{k-1}\right) I + \alpha_0 \left(\frac{\mathbf{U}^{k+1} - \mathbf{U}^k}{\Delta t^{k+1}}\right)\right] \cdot \boldsymbol{\nu} = f_{\boldsymbol{\nu}}. \tag{4.31}$$

We recommend the paper [47] for further details on this method in which they also prove that with these modification for open boundary conditions, we obtain the optimal order convergence rates:

$\mathbf{U}$	Ψ	$\|(\mathbf{u} - \mathbf{U})_{\Delta t}\|_{\ell^\infty(L^2(\Lambda)^d)}$	$\|(\mathbf{u} - \mathbf{U})_{\Delta t}\|_{\ell^\infty(H^1(\Lambda)^d)}$	$\|(p - P)_{\Delta t}\|_{\ell^\infty(L^2(\Lambda))}$
Natural $= \mathbf{f}_{\partial\Lambda}$	Natural = 0	Δt^2	$\Delta t^{3/2}$	$\Delta t^{3/2}$

4.4 Additional stabilization techniques

It is necessary to add some consistency and stabilization terms to the model in order to obtain the full convergence rates and maintain stability in various circumstances. For instance, this fractional step algorithm works well for small Reynolds number but begins to break down when we are in higher Reynolds number regimes losing the near incompressible-ness of the velocity. Likewise when the convection term dominates the diffusion term in our flow, we behave more like a hyperbolic system and stabilization is necessary. We implement a grad-div stabilization and a Streamlined Upwind/ Petrov-Galerkin (SUPG) stabilization to deal with each of these deficiencies as described below.

4.4.1 Consistent transport term for unconditional stability

We rewrite the convection term

$$\mathbf{u} \cdot \nabla \mathbf{u} \longrightarrow \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{2} (\operatorname{div} \mathbf{u}) \mathbf{u}$$

which is simply adding a consistent term, since at the continuous level (not necessarily discretely) we have $\operatorname{div} \mathbf{u} = 0$. This gives us a skew symmetric formulation of the convection term and allows us to obtain unconditional stability for our time splitting scheme as shown in [45] and later [51].

In our semi-implicit advection case when solving for the time step t^{k+1}, we use the extrapolated velocity term, $\mathbf{U}^*$ and the as of yet unknown velocity term $\mathbf{U}^{k+1}$ in the following manner:

$$\mathbf{U}^* \cdot \nabla \mathbf{U}^{k+1} \longrightarrow \mathbf{U}^* \cdot \nabla \mathbf{U}^{k+1} + \frac{1}{2} (\operatorname{div} \mathbf{U}^*) \mathbf{U}^{k+1}. \tag{4.32}$$

4.4.2 Grad-div stabilization for high Reynolds number

The incremental pressure technique as described above has a deficiency that our velocity is the incremental velocity which is not divergence free. Recall that we have dropped the tilde notation starting from Section 4.2.3 to the present for simplicity, but using the previous notation we have $\operatorname{div} \tilde{\mathbf{u}}^k \neq 0$. This deficiency is particularly true when we have high **Reynolds number** flows, where the Reynolds number, $Re = \frac{\rho V L}{\mu}$ with L the characteristic length scale and V the characteristic velocity scale, is approximated for $\mathbf{x} \in K \in \mathcal{T}^{k+1}$ by

$$Re^{k+1}(\mathbf{x}) = \frac{\rho^{k+1}(\mathbf{x}) \left\| \mathbf{U}^{k+1} \right\|_{L^\infty(K)} (h_K/p)}{\mu^{k+1}(\mathbf{x})}. \tag{4.33}$$

The characteristic Reynolds number is then the maximum of all local Reynolds numbers. To resolve this issue of divergence not being 0, we add a consistent grad-div penalty term

$$-\nabla\left(\alpha^{k+1}\operatorname{div}\mathbf{U}^{k+1}\right) \tag{4.34}$$

to the left hand side of momentum equation which forces the divergence of velocity to 0 with scalar coefficient $\alpha^{k+1} > 0$. Let $\alpha_D > 0$, be a user defined constant, then for each $K \in \mathcal{T}^{k+1}$, define

$$\begin{aligned}\alpha^{k+1}(\mathbf{x})\big|_K &:= \alpha_D\left(\mu^{k+1}(\mathbf{x}) + \rho^{k+1}(\mathbf{x})(h_K/p)\|\mathbf{U}^*\|_{L^\infty(K)}\right)\\ &= \alpha_D\mu^{k+1}(\mathbf{x})\left[1 + Re^{k+1}(\mathbf{x})\right].\end{aligned}$$

With this coefficient, our penalty term matches the units of the momentum equation and scales with the local Reynolds number. The discrete weak term we add to the left hand side of the momentum equation is

$$+\int_\Lambda \alpha^{k+1}\left(\operatorname{div}\mathbf{U}^{k+1}\right)(\operatorname{div}\mathbf{V})\,d\mathbf{x}. \tag{4.35}$$

4.4.3 Stabilization with streamlined upwind / Petrov Galerkin (SUPG) scheme

Our system needs some stabilization especially when we are in a convection dominated flow. We apply the SUPG technique introduced by Brooks and Hughes in 1982 in [52]. A decent review of the work that has been done in SUPG since then is given by Fries in [53]. We will define concepts in abstract at first and then apply the theory to our specific Navier-Stokes problem:

Given an abstract differential equation $\mathcal{L}\mathbf{u} = \mathbf{f}$, the standard approach to solving this in the Galerkin finite element framework is to solve for $\mathbf{u} \in \mathbb{V}$ such that for all test functions,

$\mathbf{w} \in \mathbb{V}^0$ we have

$$\int_\Lambda \mathbf{w} \cdot (\mathcal{L}\mathbf{u} - \mathbf{f})\, d\mathbf{x} = 0. \tag{4.36}$$

The SUPG scheme adds a stabilizing upwind flow to this system by perturbing the test function with an application of the antisymmetric part of the operator $\mathcal{L}$, denoted by $\mathcal{L}_a$. Thus for our SUPG problem, we are solving

$$\int_\Lambda (\mathbf{w} + \tau\mathcal{L}_a\mathbf{w}) \cdot (\mathcal{L}\mathbf{u} - \mathbf{f})\, d\mathbf{x} = 0. \tag{4.37}$$

Note that since we are using piecewise polynomials on our mesh, we only have $C^0(\Lambda)$ globally, where as we have higher smoothness on the interior of each mesh element. Thus in order to avoid applying the differential operator to the edges of elements, where smoothness is not guaranteed, we only actually apply the $\mathcal{L}_a$ to the interior of elements

$$\int_\Lambda \mathbf{w} \cdot (\mathcal{L}\mathbf{u} - \mathbf{f})\, d\mathbf{x} + \sum_{K\in\mathcal{T}} \int_K (\tau\mathcal{L}_a\mathbf{w}) \cdot (\mathcal{L}\mathbf{u} - \mathbf{f})\, d\mathbf{x} = 0. \tag{4.38}$$

In order to keep consistency of units in this system, we must have the units of τ inversely proportional to the units of $\mathcal{L}_a$. We will discuss this more once we have the application to the Navier-Stokes system.

4.4.4 SUPG scheme for time-dependent incompressible Navier-Stokes

In our case, the differential operator is

$$\mathcal{L}(\mathbf{u}, p) = \rho\mathbf{u}^* \cdot \nabla\mathbf{u} - \nabla \cdot (2\mu\nabla^s\mathbf{u})) + \nabla p \tag{4.39}$$

and so the antisymmetric part of the operator is simply the convection term

$$\mathcal{L}_a = \rho\frac{d\mathbf{u}}{dt} + \rho\mathbf{u}^* \cdot \nabla \tag{4.40}$$

which has units (leaving density as itself and using V = L/T)

$$[\mathcal{L}_a] = \frac{\rho V}{L} = \frac{\rho}{T}. \tag{4.41}$$

Remark. We will drop the time derivative part since in reality the SUPG scheme is only designed for stationary problems. There is some evidence but no proof that this will work in our time dependent case as well. It is commonly applied in this manner and we make no justification herein for it's success or failure in this case. We note that in this case our addition is not consistent and we may only get a first order rate. A more appropriate choice would be to use an entropy viscosity stabilization approach as referenced in Chapter 2. *end Remark.*

We desire that our parameter τ should have units $\frac{T}{\rho}$. We follow the scheme referenced by Coupez in [54] and define element-wise for each $K \in \mathcal{T}^{k+1}$

$$\tau^{k+1}\big|_K = \left(\left(\frac{2\rho^{k+1}}{\Delta t} \right)^2 + \left(\frac{2\rho^{k+1} \|\mathbf{u}^*\|_{L^\infty(K)}}{(h_K/p)} \right)^2 + \left(\frac{4\mu^{k+1}}{(h_K/p)^2} \right)^2 \right)^{-1/2} \tag{4.42}$$

where we recall that μ^{k+1} is the dynamic viscosity at time t^{k+1}. Note that this allows a switch between three different type of dominated flows. If we have time dependent dominated flows, the first part will be large and we will have an order $h^{m-1}\Delta t$ stabilization term where m is the order of the approximation of $\int_\Lambda \mathbf{w} \cdot (\mathcal{L}\mathbf{u} - \mathbf{f})\, d\mathbf{x} = 0$. By our CFL condition, this is consistent and we have an order h^m scheme. For convection dominated flows, the second term will be large and we will create an order h^m stabilization term. For diffusion dominated flows, the third term will dominate and the stabilization wasn't really needed in the first place. The stabilization term from the SUPG will be order h^{m+1}. Thus the standard h^m rate will still show up. Finally, we note that the last term can be rewritten

using the Reynolds number, defined in equation (4.33), as

$$\frac{4\mu^{k+1}}{(h_K/p)^2} = \frac{4\rho^{k+1}\|\mathbf{u}^*\|_{L^\infty(K)}}{\mathrm{Re}^{k+1}(h_K/p)}.$$

This shows a little better why we consider the last term to be diffusion dominated flow since for small Reynolds numbers, this is large. Additionally in this form it is simple to see that we indeed have the consistent dimensional scaling between all three terms.

Remark. Note that generalizing the squared and square root parts of the above definition to use a parameter $r \geq 2$ could lead to a sharper transition between the different type of dominated flows. There are some papers for instance, [55], that discuss using higher values of r in

$$\tau^{k+1}\big|_K = \left(\left(\frac{2\rho^{k+1}}{\Delta t}\right)^r + \left(\frac{2\rho^{k+1}\|\mathbf{u}^*\|_{L^\infty(K)}}{(h_K/p)}\right)^r + \left(\frac{4\mu^{k+1}}{(h_K/p)^2}\right)^r\right)^{-1/r}. \tag{4.43}$$

We will stick to using $r = 2$ for now. *end Remark.*

In summary and using the discrete terms now, we add the following SUPG terms to the left and right sides of the momentum equation:

$$\begin{aligned}
&\int_\Lambda \rho^{k+1}\left(\frac{\beta_0}{\Delta t^{k+1}}\mathbf{U}^{k+1} - \frac{\beta_1}{\Delta t^{k+1}}\mathbf{U}^{k} - \frac{\beta_2}{\Delta t^{k+1}}\mathbf{U}^{k-1}\right)\cdot\mathbf{W}^* d\mathbf{x} \\
&\quad + \int_\Lambda \rho^{k+1}\mathbf{U}^*\cdot\nabla\mathbf{U}^{k+1}\cdot\mathbf{W}^* d\mathbf{x} - \int_\Lambda \nabla\cdot\left(2\mu\nabla^s\mathbf{U}^{k+1}\right)\cdot\mathbf{W}^* d\mathbf{x} \\
&\quad = -\int_\Lambda \nabla\left(P^k + \frac{\Delta t^k}{\Delta t^{k+1}}\frac{\beta_1}{\beta_0}\Psi^k + \frac{\Delta t^{k-1}}{\Delta t^{k+1}}\frac{\beta_2}{\beta_0}\Psi^{k-1}\right)\cdot\mathbf{W}^* d\mathbf{x} \\
&\qquad + \int_\Lambda \left(\rho^{k+1}\mathbf{f}^{k+1} + \mathbf{g}^{k+1}\right)\cdot\mathbf{W}^* d\mathbf{x}.
\end{aligned} \tag{4.44}$$

where $\mathbf{V} \in \mathbb{V}^0\left(\mathcal{T}^{k+1}\right)$ is a standard test function so $\mathbf{W}^* = \tau^{k+1}\mathbf{U}^*\cdot\nabla\mathbf{W}$ is the SUPG test function.

Note that the divergence of the symmetric gradient can be broken into two terms

$$\begin{aligned}\nabla \cdot \left(2\nabla^s \mathbf{U}^{k+1}\right) &= \nabla \cdot \left(\nabla \mathbf{U}^{k+1}\right) + \nabla \cdot \left(\left(\nabla \mathbf{U}^{k+1}\right)^T\right) \\ &= \Delta \mathbf{U}^{k+1} + \nabla \operatorname{div} \mathbf{U}^{k+1}\end{aligned} \tag{4.45}$$

so that for variable viscosity, we compute

$$\begin{aligned}-\int_\Lambda \nabla \cdot \left(2\mu^{k+1}\nabla^s \mathbf{U}^{k+1}\right) \cdot \mathbf{W}^* d\mathbf{x} = &-\int_\Lambda \left(2\nabla \mu^{k+1} \cdot \nabla^s \mathbf{U}^{k+1}\right) \cdot \mathbf{W}^* d\mathbf{x} \\ &- \int_\Lambda \mu^{k+1} \Delta \mathbf{U}^{k+1} \cdot \mathbf{W}^* d\mathbf{x} \\ &- \int_\Lambda \mu^{k+1} \nabla \left(\operatorname{div} \mathbf{U}^{k+1}\right) \cdot \mathbf{W}^* d\mathbf{x}.\end{aligned} \tag{4.46}$$

There may be cases where treating viscosity μ as constant is valid in which case the first term is not important. On a practical level, both the Laplacian and the grad-div terms of equation (4.46) are available from the Hessian of our shape functions. Additionally since we are typically using piecewise quadratics for our velocity space, these terms are in fact cell-wise constant. Thus the discrete SUPG stabilization scheme is computable and can reasonably be included without too much extra work.

4.5 Numerical results

The setup for the time dependent test of the Navier-Stokes algorithm above is as follows:

We let $\rho = \mu = 1$ for simplicity and set the exact velocity,

$$\mathbf{u} = \begin{bmatrix} \sin(t+x)\sin(t+y) \\ \cos(t+x)\cos(t+y) \end{bmatrix}$$

and the exact pressure to be

$$p = \sin(t + x - y).$$

Then by the method of manufactured solutions, the right hand side is $\mathbf{f} = \mathbf{0}$ and

$$\begin{aligned}\mathbf{g} &= \mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} - 2\nabla \cdot (\nabla^s \mathbf{u}) + \nabla p \\ &= \begin{bmatrix} \cos(t+x)\left[\sin(t+y) + \sin(t+x)\right] + \sin(t+x)\left[\cos(t+y) + 2\sin(t+y)\right] + \cos(t+x-y) \\ -\cos(t+y)\left[\sin(t+x) + \sin(t+y)\right] - \cos(t+x)\left[\sin(t+y) - 2\cos(t+y)\right] - \cos(t+x-y) \end{bmatrix}\end{aligned}$$

By applying exact Dirichlet boundary conditions to the velocity model and Neumann boundary conditions to the pressure correction and then matching the average pressure value to the average exact pressure value, we obtain the results displayed in Table 4.1, Table 4.2, Table 4.3 and Table 4.4 using the Standard and Rotational models with BDF1 and BDF2 time derivatives.

Table 4.1: Convergence rates for the standard algorithm with BDF1 time derivative, computed with 10 MPI processors

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 L^2}$	rate	$\|v\|_{\ell^\infty L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	3.0249e-04	-	5.1161e-04	-
1	2.5000e-02	40	7.8125e-03	132098	9.2815e-05	1.7045	1.5679e-04	1.7062
2	1.2500e-02	80	7.8125e-03	132098	2.4947e-05	1.8955	4.1680e-05	1.9114
3	6.2500e-03	160	7.8125e-03	132098	6.6799e-06	1.9010	1.0259e-05	2.0224
4	3.1250e-03	320	7.8125e-03	132098	2.4427e-06	1.4514	2.7063e-06	1.9225
5	1.5625e-03	640	7.8125e-03	132098	1.2204e-06	1.0011	1.4747e-06	0.8759
6	7.8125e-04	1280	7.8125e-03	132098	6.4451e-07	0.9211	7.8220e-07	0.9148

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 H^1}$	rate	$\|p\|_{\ell^2 L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	3.2042e-03	-	1.9865e-02	-
1	2.5000e-02	40	7.8125e-03	132098	1.0975e-03	1.5457	7.4017e-03	1.4243
2	1.2500e-02	80	7.8125e-03	132098	3.9443e-04	1.4764	3.0826e-03	1.2637
3	6.2500e-03	160	7.8125e-03	132098	1.4617e-04	1.4322	1.4506e-03	1.0875
4	3.1250e-03	320	7.8125e-03	132098	5.6648e-05	1.3675	7.2170e-04	1.0072
5	1.5625e-03	640	7.8125e-03	132098	2.3158e-05	1.2905	3.6330e-04	0.9902
6	7.8125e-04	1280	7.8125e-03	132098	9.6175e-06	1.2678	1.8222e-04	0.9955

Table 4.2: Convergence rates for the rotational algorithm with BDF1 time derivative, computed with 10 MPI processors

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 L^2}$	rate	$\|v\|_{\ell^\infty L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	1.5336e-04	-	2.3948e-04	-
1	2.5000e-02	40	7.8125e-03	132098	4.4007e-05	1.8011	7.4673e-05	1.6813
2	1.2500e-02	80	7.8125e-03	132098	1.2109e-05	1.8617	2.1733e-05	1.7807
3	6.2500e-03	160	7.8125e-03	132098	4.2189e-06	1.5211	7.1327e-06	1.6074
4	3.1250e-03	320	7.8125e-03	132098	2.1501e-06	0.9724	2.6038e-06	1.4539
5	1.5625e-03	640	7.8125e-03	132098	1.1941e-06	0.8485	1.4650e-06	0.8297
6	7.8125e-04	1280	7.8125e-03	132098	6.4221e-07	0.8948	7.8124e-07	0.9071

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 H^1}$	rate	$\|p\|_{\ell^2 L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	1.6426e-03	-	8.9586e-03	-
1	2.5000e-02	40	7.8125e-03	132098	4.8021e-04	1.7743	4.5006e-03	0.9931
2	1.2500e-02	80	7.8125e-03	132098	1.4462e-04	1.7314	2.4439e-03	0.8809
3	6.2500e-03	160	7.8125e-03	132098	5.6600e-05	1.3534	1.3198e-03	0.8889
4	3.1250e-03	320	7.8125e-03	132098	2.7976e-05	1.0166	6.9343e-04	0.9285
5	1.5625e-03	640	7.8125e-03	132098	1.4569e-05	0.9413	3.5665e-04	0.9593
6	7.8125e-04	1280	7.8125e-03	132098	7.5333e-06	0.9515	1.8107e-04	0.9780

Table 4.3: Convergence rates for the standard algorithm with BDF2 time derivative, computed with 2 MPI processors

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 L^2}$	rate	$\|v\|_{\ell^\infty L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	1.0633e-03	-	1.4991e-03	-
1	2.5000e-02	40	7.8125e-03	132098	2.9229e-04	1.8630	3.9159e-04	1.9367
2	1.2500e-02	80	7.8125e-03	132098	7.5670e-05	1.9496	9.9499e-05	1.9766
3	6.2500e-03	160	7.8125e-03	132098	1.9182e-05	1.9800	2.5056e-05	1.9895
4	3.1250e-03	320	7.8125e-03	132098	4.8237e-06	1.9916	6.2882e-06	1.9945
5	1.5625e-03	640	7.8125e-03	132098	1.2094e-06	1.9959	1.5757e-06	1.9966
6	7.8125e-04	1280	7.8125e-03	132098	3.0272e-07	1.9982	3.9428e-07	1.9987

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 H^1}$	rate	$\|p\|_{\ell^2 L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	6.8439e-03	-	1.7445e-02	-
1	2.5000e-02	40	7.8125e-03	132098	2.3000e-03	1.5732	5.9398e-03	1.5544
2	1.2500e-02	80	7.8125e-03	132098	7.5204e-04	1.6127	1.9429e-03	1.6122
3	6.2500e-03	160	7.8125e-03	132098	2.4854e-04	1.5973	6.3610e-04	1.6109
4	3.1250e-03	320	7.8125e-03	132098	8.4699e-05	1.5531	2.1361e-04	1.5743
5	1.5625e-03	640	7.8125e-03	132098	2.9184e-05	1.5372	7.2709e-05	1.5548
6	7.8125e-04	1280	7.8125e-03	132098	8.9768e-06	1.7009	2.2398e-05	1.6988

Table 4.4: Convergence rates for the rotational algorithm with BDF2 time derivative, computed with 2 MPI processors

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 L^2}$	rate	$\|v\|_{\ell^\infty L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	7.4772e-04	-	1.0446e-03	-
1	2.5000e-02	40	7.8125e-03	132098	2.2789e-04	1.7142	3.1006e-04	1.7523
2	1.2500e-02	80	7.8125e-03	132098	6.3852e-05	1.8355	8.5487e-05	1.8588
3	6.2500e-03	160	7.8125e-03	132098	1.7082e-05	1.9022	2.2657e-05	1.9158
4	3.1250e-03	320	7.8125e-03	132098	4.4525e-06	1.9398	5.8728e-06	1.9478
5	1.5625e-03	640	7.8125e-03	132098	1.1432e-06	1.9615	1.5025e-06	1.9667
6	7.8125e-04	1280	7.8125e-03	132098	2.9087e-07	1.9746	3.8137e-07	1.9781

cycle	Δt	n_timestep	h_min	n_dofs	$\|v\|_{\ell^2 H^1}$	rate	$\|p\|_{\ell^2 L^2}$	rate
0	5.0000e-02	20	7.8125e-03	132098	4.3754e-03	-	8.5353e-03	-
1	2.5000e-02	40	7.8125e-03	132098	1.4180e-03	1.6255	2.8743e-03	1.5702
2	1.2500e-02	80	7.8125e-03	132098	4.2590e-04	1.7353	8.9376e-04	1.6852
3	6.2500e-03	160	7.8125e-03	132098	1.2324e-04	1.7891	2.7020e-04	1.7258
4	3.1250e-03	320	7.8125e-03	132098	3.5051e-05	1.8139	8.1689e-05	1.7258
5	1.5625e-03	640	7.8125e-03	132098	9.9869e-06	1.8114	2.5180e-05	1.6979
6	7.8125e-04	1280	7.8125e-03	132098	3.1416e-06	1.6685	8.1553e-06	1.6265

5. ENERGY FLOW

We are interested in coupling various physics that live on the interface Γ to the two-phase flow system of Chapter 4. In particular, we analyse the minimization of an energy on the interface Γ,

$$e(\Gamma) = \int_\Gamma G(\mathbf{x}, h) d\mathbf{x}$$

for some $G(\mathbf{x}, h)$ a function of space and total curvature, where the forces from minimizing this energy are balanced with the forces from the bulk Navier-Stokes system.

Remark. This approach encompasses a number of models based on the choice of G. We will focus on three choices for G in this work but there are others that could be used. The first is capillarity forces (or surface tension), with $G(\mathbf{x}, h) = \sigma$, the surface tension coefficient. This will be treated in more detail in Chapter 6. The second is the so called Willmore energy $G(\mathbf{x}, h) = \frac{1}{2} k_e h^2$ with $k_e \in \mathbb{R}$, the bending modulus. Thus we will have Willmore flow with constant volume constraint enforced by the incompressible Navier-Stokes flow which will be treated in Chapter 7. Finally in Chapter 8 we will describe the Canham-Helfrich flow. We will define the volume constraint functional using the divergence theorem to be

$$e_v(\Gamma) = -\frac{1}{d} \int_\Gamma \mathbf{x} \cdot \mathbf{n} d\mathbf{x} - V_0, \tag{5.1}$$

and the surface area constraint functional

$$e_{sa}(\Gamma) = \int_\Gamma d\mathbf{x} - A_0, \tag{5.2}$$

and the Willmore energy functional

$$e_w(\Gamma) = \frac{k_e}{2} \int_\Gamma h^2 d\mathbf{x}. \tag{5.3}$$

Introducing the Lagrange multipliers $\lambda_{sa}, \lambda_v \in \mathbb{R}$ which enforce the constraints to be zero, the Canham-Helfrich energy to be minimized is then

$$e(\Gamma) + \lambda_{sa} e_{sa}(\Gamma) + \lambda_v e_v(\Gamma). \tag{5.4}$$

Now because of the coupling with incompressible Navier-Stokes flow, the volume constraint is already being enforced by the pressure, $p(t, \mathbf{x})$ at the continuous level. However we will see that our discrete model does not produce a perfect volume conservation so keeping it in the system is necessary as seen in Figure 7.15. Again for the Canham-Helfrich we will use $G(\mathbf{x}, h) = \frac{1}{2} k_e h^2$ but we will add a method for enforcing the constraints.

Other types of flows not treated here include spatially varying surface tension coefficients which induces the Marangoni effect or for modelling wetting as in [56], or Canham-Helfrich flow with a given spontaneous curvature, $c_0(\mathbf{x})$, modelled by $G(\mathbf{x}, h) = \frac{1}{2} k_e (h - c_0)^2$. *end Remark.*

We couple the energy variation with the two-phase flow through the stress tensor jump on Γ. We will use the notation $\boldsymbol{\sigma} = 2\mu \nabla^s \mathbf{u} - pI$ to represent the Cauchy stress tensor on Γ.

Recall that $\mathbf{n}$ is the outer normal to Γ so using the projector $I - \mathbf{n} \otimes \mathbf{n}$ to project onto the tangential space, we have

$$\boldsymbol{\sigma} \cdot \mathbf{n} = (\boldsymbol{\sigma} \cdot \mathbf{n})(\mathbf{n} \otimes \mathbf{n}) + (\boldsymbol{\sigma} \cdot \mathbf{n})(I - \mathbf{n} \otimes \mathbf{n}) \tag{5.5}$$

$$= (\mathbf{n}^T \boldsymbol{\sigma} \mathbf{n})\mathbf{n} + (\boldsymbol{\sigma} \cdot \mathbf{n})(I - \mathbf{n} \otimes \mathbf{n}). \tag{5.6}$$

Now, making the assumption that stress jumps across Γ are only in the normal direction, that is

$$[\![(\boldsymbol{\sigma} \cdot \mathbf{n})(I - \mathbf{n} \otimes \mathbf{n})]\!] = 0 \tag{5.7}$$

on Γ, and that the velocity, $\mathbf{u}$, is continuous across Γ, ie

$$[\![\mathbf{u}]\!] = 0, \tag{5.8}$$

we have

$$\begin{aligned}
\int_\Gamma [\![\boldsymbol{\sigma}]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} &= \int_\Gamma [\![\boldsymbol{\sigma} \cdot \mathbf{n}]\!] \cdot \mathbf{v} d\mathbf{x} \\
&= \int_\Gamma [\![(\boldsymbol{\sigma} \cdot \mathbf{n})(\mathbf{n} \otimes \mathbf{n}) + (\boldsymbol{\sigma} \cdot \mathbf{n})(I - \mathbf{n} \otimes \mathbf{n})]\!] \cdot \mathbf{v} d\mathbf{x} \\
&= \int_\Gamma [\![\left(\mathbf{n}^T \boldsymbol{\sigma} \mathbf{n}\right) \mathbf{n}]\!] \cdot \mathbf{v} d\mathbf{x} \\
&= \int_\Gamma [\![\left(\mathbf{n}^T \boldsymbol{\sigma} \mathbf{n}\right)]\!] \left(\mathbf{n} \cdot \mathbf{v}\right) d\mathbf{x}.
\end{aligned}$$

If we denote the variation of the energy $e(\Gamma)$ in the direction of velocity, $\mathbf{v}$, to be $\partial e(\Gamma)(\mathbf{v})$ then we will balance the forces from the normal component of the jump tensor with the negative energy variation in the sense that

$$\int_\Gamma [\![\boldsymbol{\sigma}]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} = \int_\Gamma [\![\left(\mathbf{n}^T \boldsymbol{\sigma} \mathbf{n}\right)]\!] \left(\mathbf{n} \cdot \mathbf{v}\right) d\mathbf{x} = -\partial e \left(\Gamma\right) \left(\left(\mathbf{n} \cdot \mathbf{v}\right) \mathbf{n}\right).$$

where we only vary the energy in the normal velocity direction, $(\mathbf{n} \cdot \mathbf{v})\mathbf{n}$ of $\mathbf{v}$. From the gradient flow perspective this contributes a force that will minimize the energy, $e(\Gamma)$, the most along the normal component of the test velocity field, $(\mathbf{n} \cdot \mathbf{v})\mathbf{n}$. This couples the physics of energy minimization to the bulk fluid flow without any forces tangential to Γ.

In applications this corresponds to inextensible surfaces.

Just like the traditional gradient flow algorithms, the flow under this coupling may find surface corresponding to a global minimizer of the energy, but in general we will find a local minimum. The velocity field balances the forces from total energy consisting of kinetic energy, diffusive energy, the interfacial energy, $e(\Gamma)$ and any other external or potential sources of energy, like gravity. The minimum of total energy is achievd when $\mathbf{u} = 0$. Seifert gives an analysis in [74] of the relationship between the shapes of physical surfaces with minimal Canham-Helfrich energy and various parameters including the reduced volume (defined in equation 8.5) related to the ratio between surface area and volume of the surface.

One of the difficulties of the energy, $e(\Gamma)$, is that it requires the ability to integrate along Γ at each time step. This is possible if we parameterize Γ and track it's movement, but simplicity, quality of mesh and changes in topology make this a less desirable approach. Instead, we track Γ implicitly as a certain level set of a function. We use the level set method as described in Chapter 2 with level set function φ and $\Gamma = \{\mathbf{x} \mid \varphi(\mathbf{x}) = 0\}$. Recalling from Chapter 3 that $\varepsilon \sim h_{\min}^{3/4}$, our mesh is a adaptively graded rectangular (hexahedral) unfitted mesh with a fully refined band in an $\omega = c\varepsilon$ neighborhood of Γ where $c > 1$. We will use the smeared Dirac delta function, $\delta_\varepsilon(\varphi)|\nabla\varphi|$ described in Chapter 3 to introduce the approximate energy function

$$E(\varphi) = \int_\Lambda G(\mathbf{x}, h)\delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} \tag{5.9}$$

where h and G are now extended off of Γ to a band where the support of $\delta_\varepsilon(\varphi)$ is located.

Remark. In fact we will not in general distinguish between the terms defined only on Γ and those extended to the bulk domain Λ. However, when needed to aide in understanding and when we use the function φ for the extension, we will use a subscript φ. *end Remark.*

For our level set function, φ, we use the convention that $\varphi > 0$ inside the boundary Γ and $\varphi < 0$ outside. Likewise we have the convention of $\mathbf{n}$ being the outer unit normal vector, so the extension of the normal is defined as

$$\mathbf{n} = \mathbf{n}_\varphi = -\frac{\nabla\varphi}{|\nabla\varphi|}. \tag{5.10}$$

Similarly, we use the convention that the sphere has positive total curvature, ie. for a sphere of radius R embedded in $\mathbb{R}^d$, the total curvature is $h = \frac{d-1}{R}$ and the Gauss curvature is $\kappa = \frac{1}{R^2}$. This corresponds to the convention that

$$h = h_\varphi = \operatorname{div}_\varphi(\mathbf{n}_\varphi) = \operatorname{div}(\mathbf{n}_\varphi) = \operatorname{div}\left(-\frac{\nabla\varphi}{|\nabla\varphi|}\right). \tag{5.11}$$

We have shown in Chapter 3 that

$$\lim_{\varepsilon\to 0} \text{Quad}_\Lambda\left(f\delta_\varepsilon(\varphi)|\nabla\varphi|\right) = \int_\Gamma f(\mathbf{x})d\mathbf{x}$$

with convergence rate $\mathcal{O}(h_{\min}^{3/2})$ when $\varepsilon \sim h_{\min}^{3/4}$ for $f \in W^{6,\infty}(\Lambda)$ and so for a smooth Γ and mesh size, $h_{\min}$ small enough to resolve the curvature of Γ we have

$$\lim_{\varepsilon\to 0} E(\varphi) = e(\Gamma).$$

Thus we replace $e(\Gamma)$ with the approximate energy $E(\varphi)$ in our coupled model. This gives a consistent extension of our energy to Λ and allows us to do all our computations using the bulk finite element system.

Now, if we vary energy function, $e(\Gamma)$, by a velocity field $\mathbf{v}$, then how should we vary our bulk energy functional? We turn to the level set equation to relate Γ and φ. For motion

of φ by a velocity field $\mathbf{v}$, and defining the normal velocity, $v_n = \mathbf{v} \cdot \mathbf{n}_\varphi$, we have

$$\varphi_t = -\mathbf{v} \cdot \nabla\varphi = \mathbf{v} \cdot \left(-\frac{\nabla\varphi}{|\nabla\varphi|} \right) |\nabla\varphi| = \mathbf{v} \cdot \mathbf{n}_\varphi |\nabla\varphi| = v_n |\nabla\varphi|. \tag{5.12}$$

So varying Γ by velocity $\mathbf{v}$ is the same as varying φ by $v_n|\nabla\varphi|$.

We will compute $g \in L^2(\Lambda)$, a representative for the action of $-\partial E(\varphi)$, the negative Gâteaux derivative of energy functional $E(\varphi)$, with $\theta \in H^1(\Lambda)$, ie

$$\int_\Lambda g(\mathbf{x})\theta|\nabla\varphi| d\mathbf{x} = -\partial E(\varphi) \left(\theta|\nabla\varphi|\right).$$

Thus the force balance equation becomes

$$\int_\Gamma [\![2\mu\nabla^s \mathbf{u} - pI]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} = -\partial E(\varphi) \left(\mathbf{v} \cdot \mathbf{n}_\varphi |\nabla\varphi|\right) = \int_\Lambda g(\mathbf{x}) \mathbf{v} \cdot \mathbf{n}_\varphi |\nabla\varphi| d\mathbf{x}. \tag{5.13}$$

We will derive in Section 5.2 the first variation of energy $E(\varphi)$ in the direction $\theta|\nabla\varphi|$. This will be the coupled system to the Navier-Stokes equation. We will go into more detail on the discretization and linearization of these terms with the Navier-Stokes sytem for each of the applications: surface tension in Chapter 6, Willmore flow in Chapter 7 and Canham-Helfrich flow in Chapter 8. The main result of this Chapter is the following theorem which will be restated and proved in Section 5.2.

Theorem 5.2.1. *Given the approximate energy functional*

$$E(\varphi) = \int_\Lambda G(\boldsymbol{x}, h_\varphi)\delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x}$$

where φ is the level set function for a smooth closed surface Γ, h_φ, the total curvature defined by equation (5.11), and $\delta_\varepsilon(\varphi)|\nabla\varphi|$, the approximate Dirac delta function of Theorem 3.3.1, then for any smooth function θ, the variation of the energy in the direction

$\theta|\nabla\varphi|$ *is*

$$\begin{aligned}\partial E(\varphi)(\theta|\nabla\varphi|) &= \int_\Lambda \operatorname{div}\left[G\boldsymbol{n}_\varphi - \frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right]\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x}\\ &= \int_\Lambda\left[\nabla G\cdot\boldsymbol{n}_\varphi\theta + Gh_\varphi\theta + \frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\cdot\frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x}\end{aligned}$$

where $\boldsymbol{n}_\varphi = -\frac{\nabla\varphi}{|\nabla\varphi|}$ *is the extended outward unit normal and* $P_\varphi = I - \boldsymbol{n}_\varphi\otimes\boldsymbol{n}_\varphi = I - \boldsymbol{n}_\varphi\boldsymbol{n}_\varphi^T$ *is the projector onto the tangent space to level sets of* φ*, in particular tangent to the surface,* $\Gamma = \{\boldsymbol{x} \mid \varphi = 0\}$ *and* $\partial_2 G$ *is the partial derivative of* G *with respect to total curvature,* h_φ*, the second argument.*

Remark. The formulation derived does not allow for tangential deformations along Γ. In terms of the coupled model with the velocity, this implies that $\mathbf{u}|_\Gamma = (\mathbf{u}|_\Gamma\cdot\mathbf{n})\,\mathbf{n}$. There are other approaches to coupling which can handle more general velocity coupling, mainly the method of virtual power as is common in some mechanical engineering literature and in the phase field method communities. The two approaches (ours and the method of virtual power) end up with a similar result but come from different assumptions and approaches. We will discuss the similarities for this general energy functional with a method of virtual power approach in Section 5.4. *end Remark.*

5.1 Preliminary computations and some useful terms

Before we can proved the Theorem 5.2.1, we must introduce the notation and prove some useful equalities in the form of Lemma 5.1.2 and 5.1.3. We finish off with Lemma 5.1.4 which relates the resulting energy variation to the true energy variation. All together, these will establish the relationship between the terms that arise in our bulk formulation and the terms that arise from shape calculus on a manifold.

Recall that we defined the extended normal, $\mathbf{n} = -\frac{\nabla\varphi}{|\nabla\varphi|}$ and curvature $h = \operatorname{div}(\mathbf{n})$ in equations (5.10) and (5.11).

We will first compute some terms related to the first variation of our energy functions, then we compute some identities that will aid us in simplifications of the model and in proving stability.

Lemma 5.1.1. *Given $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ and the tangential projector, $P_\varphi = I - \boldsymbol{n}_\varphi \otimes \boldsymbol{n}_\varphi$ $(= I - \boldsymbol{n}_\varphi \boldsymbol{n}_\varphi^T)$, we have*

$$(P_\varphi \boldsymbol{u}) \cdot (P_\varphi \boldsymbol{v}) = (P_\varphi \boldsymbol{u}) \cdot \boldsymbol{v} = \boldsymbol{u} \cdot (P_\varphi \boldsymbol{v}) \tag{5.14}$$

Proof. First note that

$$P_\varphi^T = \left(I - \mathbf{n}_\varphi \mathbf{n}_\varphi^T\right)^T = I - \mathbf{n}_\varphi \mathbf{n}_\varphi^T = P_\varphi$$

and using $\mathbf{n}_\varphi^T \mathbf{n}_\varphi = \mathbf{n}_\varphi \cdot \mathbf{n}_\varphi = 1$

$$P_\varphi^2 = (I - \mathbf{n}_\varphi \mathbf{n}_\varphi^T)(I - \mathbf{n}_\varphi \mathbf{n}_\varphi^T) = I - 2\mathbf{n}_\varphi \mathbf{n}_\varphi^T + \mathbf{n}_\varphi \mathbf{n}_\varphi^T \mathbf{n}_\varphi \mathbf{n}_\varphi^T = I - \mathbf{n}_\varphi \mathbf{n}_\varphi^T = P_\varphi$$

so that

$$\begin{aligned}(P_\varphi \mathbf{u}) \cdot (P_\varphi \mathbf{v}) &= \mathbf{u}^T P_\varphi^T P_\varphi \mathbf{v} = \mathbf{u}^T P_\varphi \mathbf{v} \\ &= (P_\varphi \mathbf{u}) \cdot \mathbf{v} \\ &= \mathbf{u} \cdot (P_\varphi \mathbf{v})\end{aligned}$$

as desired. □

We will denote the extended tangential projector using φ for th extension alternately by P, P_φ or $P[\varphi]$ as needed for understanding.

When we are dealing only with functions defined on Γ, we will denote the differential

operators along the surface Γ (tangential to Γ) with a subscript Γ. The standard ones are the tangential gradient, ∇_Γ, tangential divergence, div_Γ, and the Laplace-Beltrami operator or tangential Laplacian, $\Delta_\Gamma = \mathrm{div}_\Gamma(\nabla_\Gamma)$. For a scalar function $v : \Gamma \to \mathbb{R}$ which admits a C^1 extension, $\tilde{v}$, to a tubular neighborhood of Γ, the tangential gradient is

$$\nabla_\Gamma v := \nabla \tilde{v}|_\Gamma - (\mathbf{n} \cdot \nabla \tilde{v}|_\Gamma)\, \mathbf{n} = P\, \nabla \tilde{v}|_\Gamma \,. \tag{5.15}$$

The tangential gradient can also be defined for a vector function with gradient transpose applied to each component so that the i-th row is the transpose of the gradient of the i-th component,

$$\nabla_\Gamma \mathbf{v} = P\, \nabla \mathbf{v}|_\Gamma \,. \tag{5.16}$$

Likewise the tangential divergence for a vector field $\mathbf{v} : \Gamma \to \mathbb{R}^d$ which admits a C^1 extension $\tilde{\mathbf{v}}$ to a tubular neighborhood around Γ is

$$\mathrm{div}_\Gamma \mathbf{v} = \mathrm{div}(\tilde{\mathbf{v}})|_\Gamma - \mathbf{n} \cdot (\nabla \tilde{\mathbf{v}}|_\Gamma \cdot \mathbf{n}) = \mathrm{Trace}\,(\nabla_\Gamma \mathbf{v}) \tag{5.17}$$

Finally, the tangential laplacian can be written for $v : \Gamma \to \mathbb{R}$ allowing a C^2 extension and with a constant in normal direction extension, as

$$\Delta_\Gamma v = \mathrm{div}_\Gamma(\nabla_\Gamma v) = \Delta \tilde{v}|_\Gamma - \mathbf{n} \cdot \left(D^2 \tilde{v}\big|_\Gamma \mathbf{n} \right) - (\nabla \tilde{v}|_\Gamma \cdot \mathbf{n})\, \mathrm{div}(\tilde{\mathbf{n}}). \tag{5.18}$$

These are standard elements in any book on surface calculus, for instance [57]. However, once we change over to using an implicit level set function, φ, and extend to narrow bands around Γ we in fact have many implicit surfaces at once and so we change our notation to reflect this. We will instead use a subscript φ to denote the tangential differential operators

on the band as in ∇_φ, div_φ and $\Delta_\varphi = \mathrm{div}_\varphi(\nabla_\varphi)$. To be specific, given the levelset function, φ our implicit function, scalar function $v : \Lambda \to \mathbb{R}$ and extended normal $\mathbf{n}_\varphi$, then

$$\nabla_\varphi v := \nabla v - (\mathbf{n}_\varphi \cdot \nabla v)\, \mathbf{n}_\varphi = P_\varphi \nabla v. \tag{5.19}$$

Likewise the tangential divergence for a vector function $\mathbf{v} : \Lambda \to \mathbb{R}^d$ is

$$\begin{aligned} \mathrm{div}_\varphi \mathbf{v} &= \mathrm{div}(\mathbf{v}) - \mathbf{n}_\varphi^T (\nabla \mathbf{v}) \mathbf{n}_\varphi = \mathrm{Trace}\,(\nabla_\varphi \mathbf{v}) = \mathrm{Trace}\,(P_\varphi \nabla \mathbf{v}) \\ &= \frac{1}{|\nabla \varphi|}\, \mathrm{div}(\mathbf{v} |\nabla \varphi|) \end{aligned} \tag{5.20}$$

In addition the tangential laplacian can be written as

$$\begin{aligned} \Delta_\varphi v &= \mathrm{div}_\varphi(\nabla_\varphi v) = \mathrm{div}(P_\varphi \nabla v) - \mathbf{n}_\varphi^T \nabla (P_\varphi \nabla v) \mathbf{n}_\varphi \\ &= \frac{1}{|\nabla \varphi|}\, \mathrm{div}\,(P_\varphi \nabla \mathbf{v} |\nabla \varphi|)\,. \end{aligned} \tag{5.21}$$

We are now ready to do some calculations in preparation for the calculation of variations of our approximate energy functional.

Lemma 5.1.2. *The following variational terms hold using the extended normal* $\boldsymbol{n} = \boldsymbol{n}_\varphi = -\frac{\nabla \varphi}{|\nabla \varphi|}$*, the tangential projector* $P_\varphi = I - \boldsymbol{n}_\varphi \otimes \boldsymbol{n}_\varphi \ \ (= I - \boldsymbol{n}_\varphi \boldsymbol{n}_\varphi^T)$ *with* $|\nabla \varphi| > 0$ *and*

smooth scalar function, ψ:

$$\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|\right)\Big|_{\eta=0} = \frac{\nabla\varphi\cdot\nabla\psi}{|\nabla\varphi|} \tag{5.22}$$

$$\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|^{-1}\right)\Big|_{\eta=0} = -\frac{\nabla\varphi\cdot\nabla\psi}{|\nabla\varphi|^3} \tag{5.23}$$

$$\frac{d}{d\eta}\left(\boldsymbol{n}_{\varphi+\eta\psi}\right)\Big|_{\eta=0} = -\frac{P_\varphi\nabla\psi}{|\nabla\varphi|} \tag{5.24}$$

$$\frac{d}{d\eta}\left(h_{\varphi+\eta\psi}\right)\Big|_{\eta=0} = \operatorname{div}\left(-\frac{P_\varphi\nabla\psi}{|\nabla\varphi|}\right) \tag{5.25}$$

$$\frac{d}{d\eta}\delta_\varepsilon\left(\varphi+\eta\psi\right)\Big|_{\eta=0} = \delta'_\varepsilon\left(\varphi\right)\psi \tag{5.26}$$

Proof. Using $|\mathbf{u}| = \sqrt{\mathbf{u}\cdot\mathbf{u}}$, and the chain rule, we calculate

$$\begin{aligned}
\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|\right)\Big|_{\eta=0} &= \left(\frac{d}{d\eta}\sqrt{\nabla\varphi\cdot\nabla\varphi+2\eta\nabla\varphi\cdot\nabla\psi+\eta^2\nabla\psi\cdot\nabla\psi}\right)\Big|_{\eta=0}\\
&= \frac{1}{2}\frac{\left(2\nabla\varphi\cdot\nabla\psi+2\eta\nabla\psi\cdot\nabla\psi\right)}{|\nabla\varphi+\eta\nabla\psi|}\Big|_{\eta=0}\\
&= \frac{\nabla\varphi\cdot\nabla\psi}{|\nabla\varphi|}.
\end{aligned}$$

We use this result with the chain rule to calculate

$$\begin{aligned}
\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|^{-1}\right)\Big|_{\eta=0} &= -\left|\nabla\varphi+\eta\nabla\psi\right|^{-2}\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|\right)\Big|_{\eta=0}\\
&= -\frac{\nabla\varphi\cdot\nabla\psi}{|\nabla\varphi|^3}.
\end{aligned}$$

Again, we use this in combination with the product rule to obtain

$$\begin{aligned}
\frac{d}{d\eta}\left(\mathbf{n}_{\varphi+\eta\psi}\right)\Big|_{\eta=0} &= \frac{d}{d\eta}\left(-\frac{\nabla\varphi+\eta\nabla\psi}{|\nabla\varphi+\eta\nabla\psi|}\right)\Big|_{\eta=0} \\
&= -\frac{\nabla\psi}{|\nabla\varphi+\eta\nabla\psi|}\Big|_{\eta=0} - (\nabla\varphi+\eta\nabla\psi)\frac{d}{d\eta}\left(|\nabla\varphi+\eta\nabla\psi|^{-1}\right)\Big|_{\eta=0} \\
&= -\frac{\nabla\psi}{|\nabla\varphi|} + \nabla\varphi\frac{\nabla\varphi\cdot\nabla\psi}{|\nabla\varphi|^3} \\
&= -\frac{1}{|\nabla\varphi|}\left(\nabla\psi - (\mathbf{n}_\varphi\cdot\nabla\psi)\mathbf{n}_\varphi\right) \\
&= -\frac{P_\varphi\nabla\psi}{|\nabla\varphi|}
\end{aligned}$$

Next, we note that $\frac{d}{d\eta}$ can freely pass through the spatial $\mathrm{div}(\cdot)$ operator, so that

$$\begin{aligned}
\frac{d}{d\eta}\left(h_{\varphi+\eta\psi}\right)\Big|_{\eta=0} &= \frac{d}{d\eta}\,\mathrm{div}\left(\mathbf{n}_{\varphi+\eta\psi}\right)\Big|_{\eta=0} \\
&= \mathrm{div}\left(\frac{d}{d\eta}\mathbf{n}_{\varphi+\eta\psi}\Big|_{\eta=0}\right) \\
&= \mathrm{div}\left(-\frac{P_\varphi\nabla\psi}{|\nabla\varphi|}\right).
\end{aligned}$$

Finally we have

$$\begin{aligned}
\frac{d}{d\eta}\delta_\varepsilon\left(\varphi+\eta\psi\right)\Big|_{\eta=0} &= \delta_\varepsilon'\left(\varphi+\eta\psi\right)|_{\eta=0}\,\psi \\
&= \delta_\varepsilon'\left(\varphi\right)\psi
\end{aligned}$$

which ends our calculations. □

Lemma 5.1.3. *The following equalities hold for scalars $u \in C^2(\Lambda)$ and $\varphi \in C^4(\Lambda)$, where φ is the level set function with $|\nabla\varphi| > 0$, $\boldsymbol{n}_\varphi = -\frac{\nabla\varphi}{|\nabla\varphi|}$ and $P_\varphi = I - \boldsymbol{n}_\varphi\boldsymbol{n}_\varphi^T\,(= I - \boldsymbol{n}_\varphi\otimes\boldsymbol{n}_\varphi)$,*

the tangential projector matrix:

$$\nabla_\varphi \delta_\varepsilon(\varphi) = P_\varphi \nabla \left(\delta_\varepsilon(\varphi)\right) = \mathbf{0} \tag{5.27}$$

$$\nabla \left(|\nabla\varphi|\right) = \frac{D^2\varphi\nabla\varphi}{|\nabla\varphi|} \tag{5.28}$$

$$\nabla \left(|\nabla\varphi|^{-1}\right) = -\frac{D^2\varphi\nabla\varphi}{|\nabla\varphi|^3} \tag{5.29}$$

In addition, we have the following relations concerning the gradient of the normal, $D\boldsymbol{n}_\varphi$*, where* $(D\boldsymbol{n}_\varphi)_{ij} := \partial_j(\boldsymbol{n}_\varphi)_i$*,*

$$D\boldsymbol{n}_\varphi = -\frac{P_\varphi D^2\varphi}{|\nabla\varphi|} \tag{5.30}$$

$$D\boldsymbol{n}_\varphi = \nabla_\varphi \boldsymbol{n}_\varphi = P_\varphi \nabla \boldsymbol{n}_\varphi \tag{5.31}$$

$$D\boldsymbol{n}_\varphi^T \boldsymbol{n}_\varphi = \mathbf{0} \tag{5.32}$$

$$\frac{\nabla_\varphi \left(|\nabla\varphi|\right)}{|\nabla\varphi|} = D\boldsymbol{n}_\varphi \boldsymbol{n}_\varphi \tag{5.33}$$

$$\nabla u \cdot (D\boldsymbol{n}_\varphi \boldsymbol{n}_\varphi) = -\boldsymbol{n}_\varphi^T \nabla (P_\varphi \nabla u) \boldsymbol{n}_\varphi \tag{5.34}$$

$$\operatorname{div}(D\boldsymbol{n}_\varphi \boldsymbol{n}_\varphi) = \nabla h_\varphi \cdot \boldsymbol{n}_\varphi + D\boldsymbol{n}_\varphi^T : D\boldsymbol{n}_\varphi. \tag{5.35}$$

If in addition, $\frac{\nabla\varphi}{|\nabla\varphi|} = \nabla d$*, which is true in the case* $\varphi(\boldsymbol{x}) = f(d(\boldsymbol{x}))$ *where* $f(t) \in C^2(\mathbb{R})$ *is the level set filter with* $f(0) = 0$ *and* $f'(t) \geq 0$*, then the following hold*

$$D\boldsymbol{n}_\varphi^T = D\boldsymbol{n}_\varphi \tag{5.36}$$

$$\nabla_\varphi \left(|\nabla\varphi|\right) = P_\varphi \nabla \left(|\nabla\varphi|\right) = \mathbf{0} \tag{5.37}$$

$$\nabla h_\varphi \cdot \boldsymbol{n}_\varphi = -|\nabla_\varphi \boldsymbol{n}_\varphi|^2. \tag{5.38}$$

Proof. To start with, we compute the surface gradient of the approximate Dirac measure

as

$$\nabla_\varphi \delta_\varepsilon(\varphi) = P_\varphi \nabla \left(\delta_\varepsilon(\varphi)\right) = P_\varphi \delta_\varepsilon'(\varphi) \nabla\varphi = \delta_\varepsilon'(\varphi) P_\varphi \nabla\varphi = \mathbf{0}.$$

Next we compute

$$\nabla \left(|\nabla\varphi|\right) = \nabla \left(\sqrt{\nabla\varphi \cdot \nabla\varphi}\right) = \frac{D^2\varphi \nabla\varphi}{|\nabla\varphi|}$$

and

$$\nabla \left(|\nabla\varphi|^{-1}\right) = -|\nabla\varphi|^{-2} \nabla \left(|\nabla\varphi|\right) = -\frac{D^2\varphi \nabla\varphi}{|\nabla\varphi|^3}.$$

Next we compute using Einstein notation and symmetry of $D^2\varphi$ (dropping the subscript notation on $\mathbf{n} = \mathbf{n}_\varphi$ for simplicity with the Einstein notation)

$$\begin{aligned}
(D\mathbf{n}_\varphi)_{ij} &= \partial_j (\mathbf{n}_\varphi)_i \\
&= \partial_j \left(-\frac{\partial_i \varphi}{|\nabla\varphi|}\right) \\
&= -\frac{\partial_{ij}\varphi}{|\nabla\varphi|} - \partial_i\varphi \partial_j \left(|\nabla\varphi|^{-1}\right) \\
&= -\frac{\partial_{ij}\varphi}{|\nabla\varphi|} - \partial_i\varphi \frac{\partial_{jk}\varphi \partial_k\varphi}{|\nabla\varphi|^3} \\
&= -\left(\frac{\partial_{ij}\varphi}{|\nabla\varphi|} - \mathbf{n}_i\mathbf{n}_k \frac{\partial_{jk}\varphi}{|\nabla\varphi|}\right) \\
&= -\left(\frac{\partial_{ij}\varphi}{|\nabla\varphi|} - \mathbf{n}_i\mathbf{n}_k \frac{\partial_{kj}\varphi}{|\nabla\varphi|}\right) \\
&= -(\delta_{ik} - \mathbf{n}_i\mathbf{n}_k) \frac{\partial_{kj}\varphi}{|\nabla\varphi|}
\end{aligned}$$

so that

$$D\mathbf{n}_\varphi = -\frac{P_\varphi D^2\varphi}{|\nabla\varphi|}.$$

By Lemma 5.1.1, $P_\varphi^2 = P_\varphi$. Thus using equation (5.19) and recognizing the equivalent notations, $\nabla\mathbf{n}_\varphi = D\mathbf{n}_\varphi$, we have

$$\begin{aligned}
\nabla_\varphi\mathbf{n}_\varphi &= P_\varphi\nabla\mathbf{n}_\varphi \\
&= P_\varphi D\mathbf{n}_\varphi \\
&= P_\varphi\left(-\frac{P_\varphi D^2\varphi}{|\nabla\varphi|}\right) \\
&= -\frac{P_\varphi D^2\varphi}{|\nabla\varphi|} \\
&= D\mathbf{n}_\varphi.
\end{aligned}$$

Differentiating $|\mathbf{n}_\varphi|^2 = 1$ we have $0 = \partial_j(\mathbf{n}_i\mathbf{n}_i) = 2\mathbf{n}_i\partial_j\mathbf{n}_i$ so that

$$D\mathbf{n}_\varphi^T\mathbf{n}_\varphi = \mathbf{0}.$$

In general, we don't have that $D\mathbf{n}_\varphi = D\mathbf{n}_\varphi^T$ or $D\mathbf{n}_\varphi\mathbf{n}_\varphi = 0$ but in the case that $\varphi = f(d(\mathbf{x}))$ and $f' \geq 0$ then $|\nabla\varphi| = f'(d(\mathbf{x}))$ and $\mathbf{n}_\varphi = -\nabla d$ so that

$$\begin{aligned}
(D\mathbf{n}_\varphi)_{ij} &= \partial_j\left(\mathbf{n}_i\right) \\
&= -\partial_{ji}d \\
&= -\partial_{ij}d \\
&= \partial_i\left(\mathbf{n}_j\right) \\
&= (D\mathbf{n}_\varphi)_{ji}.
\end{aligned}$$

Now, we also have

$$\begin{aligned}\frac{\nabla_\varphi\left(|\nabla\varphi|\right)}{|\nabla\varphi|} &= \frac{P_\varphi\nabla\left(|\nabla\varphi|\right)}{|\nabla\varphi|} = P_\varphi\left(\frac{D^2\varphi\cdot\nabla\varphi}{|\nabla\varphi|^2}\right)\\ &= \left(-\frac{P_\varphi D^2\varphi}{|\nabla\varphi|}\right)\cdot\left(-\frac{\nabla\varphi}{|\nabla\varphi|}\right)\\ &= D\mathbf{n}_\varphi\mathbf{n}_\varphi.\end{aligned}$$

Thus, when $\mathbf{n}_\varphi = -\nabla d$ as in cases listed above, we have $D\mathbf{n}_\varphi = D\mathbf{n}_\varphi^T$ so that

$$\nabla_\varphi\left(|\nabla\varphi|\right) = \cancelto{0}{D\mathbf{n}_\varphi^T\mathbf{n}_\varphi}|\nabla\varphi| = \mathbf{0}.$$

Next we have (again dropping the subscript on $\mathbf{n} = \mathbf{n}_\varphi$ for simplicity with the Einstein notation)

$$\begin{aligned}\mathbf{n}_\varphi^T\nabla\left(P_\varphi\nabla u\right)\mathbf{n}_\varphi &= \mathbf{n}_i\partial_j\left((P\nabla u)_i\right)\mathbf{n}_j\\ &= \mathbf{n}_i\partial_j\left(\partial_i u - (\mathbf{n}_k\partial_k u)\mathbf{n}_i\right)\mathbf{n}_j\\ &= \mathbf{n}_i\left(\partial_{ij}u - \partial_j\mathbf{n}_k\partial_k u\mathbf{n}_i - \mathbf{n}_k\partial_{jk}u\mathbf{n}_i - (\mathbf{n}_k\partial_k u)\partial_j\mathbf{n}_i\right)\mathbf{n}_j\\ &= \mathbf{n}_i\partial_{ij}u\mathbf{n}_j - \cancelto{1}{(\mathbf{n}_i\mathbf{n}_i)}\partial_j\mathbf{n}_k\partial_k u\mathbf{n}_j - \cancelto{1}{(\mathbf{n}_i\mathbf{n}_i)}\mathbf{n}_k\partial_{jk}u\mathbf{n}_j - \cancelto{0}{(\partial_j\mathbf{n}_i\mathbf{n}_i)}(\mathbf{n}_k\partial_k u)\mathbf{n}_j\\ &= -\mathbf{n}_j\partial_j\mathbf{n}_k\partial_k u\\ &= -\mathbf{n}_\varphi^T D\mathbf{n}_\varphi^T\nabla u,\end{aligned}$$

so that recognizing these as scalars, we have

$$\nabla u\cdot\left(D\mathbf{n}_\varphi\mathbf{n}_\varphi\right) = -\mathbf{n}_\varphi^T\nabla\left(P_\varphi\nabla u\right)\mathbf{n}_\varphi.$$

Finally,

$$\begin{aligned}
\operatorname{div}(D\mathbf{n}_\varphi \mathbf{n}_\varphi) &= \partial_i \left(\partial_j \mathbf{n}_i \mathbf{n}_j\right) \\
&= \partial_{ij} \mathbf{n}_i \mathbf{n}_j + \partial_j \mathbf{n}_i \partial_i \mathbf{n}_j \\
&= \partial_j (\partial_i \mathbf{n}_i) \mathbf{n}_j + D\mathbf{n}^T : D\mathbf{n} \\
&= \partial_j h \mathbf{n}_j + D\mathbf{n}^T : D\mathbf{n} \\
&= \nabla h_\varphi \cdot \mathbf{n}_\varphi + D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi.
\end{aligned}$$

Thus if $\mathbf{n}_\varphi = -\nabla d$ as described above, then $D\mathbf{n}_\varphi = D\mathbf{n}_\varphi^T$ and

$$\begin{aligned}
\operatorname{div}(D\mathbf{n}_\varphi \mathbf{n}_\varphi) &= \nabla h_\varphi \cdot \mathbf{n}_\varphi + D\mathbf{n}_\varphi : D\mathbf{n}_\varphi \\
&= \nabla h_\varphi \cdot \mathbf{n}_\varphi + |D\mathbf{n}_\varphi|^2 \\
&= \nabla h_\varphi \cdot \mathbf{n}_\varphi + |\nabla_\varphi \mathbf{n}_\varphi|^2,
\end{aligned}$$

and in particular $\operatorname{div}(D\mathbf{n}_\varphi \mathbf{n}_\varphi) = \operatorname{div}(D\mathbf{n}_\varphi^T \mathbf{n}_\varphi) = 0$ so that

$$\nabla h_\varphi \cdot \mathbf{n}_\varphi = -|\nabla_\varphi \mathbf{n}_\varphi|^2.$$

□

Now to finish out the useful lemmata, we show that the term we will get from our variation of energy, Theorem 5.2.1 can be simplified back to use some recognizable tangential differential operators. In particular, this will help us relate our Willmore energy model to one that would be derived using the ideas of the continuum surface force model. We will discuss this more fully in Section 7.4.1.

Lemma 5.1.4. *Given a functional $G = G(\boldsymbol{x}, h_\varphi)$ a function of space and curvature, then*

$$-\operatorname{div}\left(\frac{P_\varphi \nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right) + \operatorname{div}(G\boldsymbol{n}_\varphi) = -\Delta_\varphi(\partial_2 G) + Gh_\varphi - \partial_2 G D\boldsymbol{n}_\varphi^T : D\boldsymbol{n}_\varphi + \partial_{\mathbf{1}}\mathbf{G}\cdot\boldsymbol{n}_\varphi \tag{5.39}$$

so that if $\boldsymbol{n}_\varphi = -\nabla d$ as in the case of our filtered level set, then $D\boldsymbol{n}_\varphi^T = D\boldsymbol{n}_\varphi$ and we obtain

$$-\operatorname{div}\left(\frac{P_\varphi \nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right) + \operatorname{div}(G\boldsymbol{n}_\varphi) = -\Delta_\varphi(\partial_2 G) + Gh_\varphi - \partial_2 G|\nabla_\varphi \boldsymbol{n}_\varphi|^2 + \partial_{\mathbf{1}}\mathbf{G}\cdot\boldsymbol{n}_\varphi. \tag{5.40}$$

where $\partial_2 G$ is the derivative of G with respect to total curvature h_φ and $\partial_{\mathbf{1}}\mathbf{G}$ is the gradient with respect to $\boldsymbol{x}$ of all terms that are not a function of curvature.

For example, the Willmore energy with spontaneous curvature, $G(\mathbf{x}, h_\varphi) = \frac{1}{2}(h_\varphi - c_0(\mathbf{x}))^2$ has $\partial_2 G = h_\varphi - c_0$ and $\partial_{\mathbf{1}}\mathbf{G} = -(h_\varphi - c_0)\nabla c_0$.

Proof. We will use many of the results from Lemma 5.1.3. By equation (5.34) with $u = \partial_2 G$, we have $-\nabla(\partial_2 G)^T D\mathbf{n}_\varphi\mathbf{n}_\varphi = \nabla(\partial_2 G)^T D\mathbf{n}_\varphi\mathbf{n}_\varphi$ and by equation (5.35),

$$\partial_2 G \operatorname{div}(D\mathbf{n}_\varphi\mathbf{n}_\varphi) = \partial_2 G(\nabla h_\varphi^T\mathbf{n}_\varphi + D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi).$$

Thus we have

$$\begin{aligned}
-\operatorname{div}\left(\frac{P_\varphi \nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right) &= -\operatorname{div}(P_\varphi\nabla(\partial_2 G)) - \operatorname{div}\left(\partial_2 G\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\right)\\
&= -\operatorname{div}(\nabla_\varphi(\partial_2 G)) - \operatorname{div}(\partial_2 G D\mathbf{n}_\varphi\mathbf{n}_\varphi)\\
&= -\operatorname{div}(\nabla_\varphi(\partial_2 G)) - \nabla(\partial_2 G)^T D\mathbf{n}_\varphi\mathbf{n}_\varphi - \partial_2 G\operatorname{div}(D\mathbf{n}_\varphi\mathbf{n}_\varphi)\\
&= -\operatorname{div}(\nabla_\varphi(\partial_2 G)) + \mathbf{n}_\varphi^T\nabla(P_\varphi\nabla(\partial_2 G))\mathbf{n}_\varphi\\
&\qquad - \partial_2 G(\nabla h_\varphi^T\mathbf{n}_\varphi + D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi)\\
&= -\Delta_\varphi(\partial_2 G) - \partial_2 G(\nabla h_\varphi^T\mathbf{n}_\varphi + D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi).
\end{aligned}$$

Now, using $\nabla G\cdot\mathbf{n}_\varphi = \partial_{\mathbf{1}}\mathbf{G}\cdot\mathbf{n}_\varphi + \partial_2 G\nabla h_\varphi^T\mathbf{n}_\varphi$ and $h_\varphi = \operatorname{div}\mathbf{n}_\varphi$ then

$$\begin{aligned}
\operatorname{div}(G\mathbf{n}_\varphi) &= \nabla G\cdot\mathbf{n}_\varphi + G\operatorname{div}(\mathbf{n}_\varphi)\\
&= \partial_{\mathbf{1}}\mathbf{G}^T\mathbf{n}_\varphi + \partial_2 G\nabla h_\varphi^T\mathbf{n}_\varphi + Gh_\varphi,
\end{aligned}$$

and so

$$\begin{aligned}
-\operatorname{div}\left(\frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right) + \operatorname{div}(G\mathbf{n}_\varphi) &= -\Delta_\varphi(\partial_2 G) - \partial_2 G(\nabla h_\varphi^T\mathbf{n}_\varphi + D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi)\\
&\qquad + \partial_{\mathbf{1}}\mathbf{G}^T\mathbf{n}_\varphi + \partial_2 G\nabla h_\varphi^T\mathbf{n}_\varphi + Gh_\varphi\\
&= -\Delta_\varphi(\partial_2 G) - \partial_2 G D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi\\
&\qquad + \partial_{\mathbf{1}}\mathbf{G}^T\mathbf{n}_\varphi + Gh_\varphi,
\end{aligned}$$

as desired. If in addition $\mathbf{n}_\varphi = -\nabla d$ as when we have our filtered level set, then $D\mathbf{n}_\varphi^T = D\mathbf{n}_\varphi$ and we reduce to

$$-\Delta_\varphi(\partial_2 G) - \partial_2 G|D\mathbf{n}_\varphi|^2 + \partial_{\mathbf{1}}\mathbf{G}^T\mathbf{n}_\varphi + Gh_\varphi.$$

□

5.2 Computing the variation of energy, $\partial E(\varphi)(\theta|\nabla\varphi|)$

We now derive the first variation of our energy, $E(\varphi)$, which will give us the gradient flow terms in the coupling to the Navier-Stokes model. We will use many of the results of Lemma 5.1.2 and other equalities from Section 5.1.

Theorem 5.2.1. *Given the approximate energy functional*

$$E(\varphi) = \int_\Lambda G(\boldsymbol{x}, h_\varphi)\delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x}$$

where φ is the level set function for a smooth closed surface Γ with $\Gamma \cap \partial\Lambda = \emptyset$ and $\varepsilon > 0$ suficiently small so that $\delta_\varepsilon|_{\partial\Lambda} = 0$. Given $h = h_\varphi$, the total curvature defined in equation (5.11) and $\delta_\varepsilon(\varphi)|\nabla\varphi|$, the approximate Dirac delta function defined in Theorem 3.3.1, then for any smooth function, θ, the variation of the energy in the direction $\theta|\nabla\varphi|$ is

$$\begin{aligned}\partial E(\varphi)(\theta|\nabla\varphi|) &= \int_\Lambda \operatorname{div}\left[G\boldsymbol{n}_\varphi - \frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right]\theta\delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x} \qquad (5.41)\\ &= \int_\Lambda \left[\nabla G\cdot\boldsymbol{n}_\varphi\theta + Gh_\varphi\theta + \frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\cdot\frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x},\end{aligned}$$

where $\boldsymbol{n}_\varphi$ is the outward unit normal and $P_\varphi = I - \boldsymbol{n}_\varphi\otimes\boldsymbol{n}_\varphi = I - \boldsymbol{n}_\varphi\boldsymbol{n}_\varphi^T$ is the projector onto the tangent space to the level sets of φ and in particular the tangent space to the surface $\Gamma = \{\boldsymbol{x} \mid \varphi = 0\}$. Finally, $\partial_2 G$ is the partial derivative of G with respect to total curvature, h_φ, the second argument.

In addition, when $\varphi = f(d(\boldsymbol{x}))$ then by equation (5.37), $P_\varphi\nabla(|\nabla\varphi|) = \mathbf{0}$ and the

energy variation reduces to

$$\begin{aligned}\partial E(\varphi)\left(\theta|\nabla\varphi|\right) &= \int_{\Lambda} \operatorname{div}\left[G\boldsymbol{n}_{\varphi} - P_{\varphi}\nabla\left(\partial_2 G\right)\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x} \\ &= \int_{\Lambda}\left[\nabla G\cdot\boldsymbol{n}_{\varphi}\theta + Gh_{\varphi}\theta + P_{\varphi}\nabla(\partial_2 G)\cdot\nabla\theta\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x}.\end{aligned} \tag{5.42}$$

Proof. To begin, given θ smooth enough, we compute the Gâteaux derivative

$$\begin{aligned}\partial E(\varphi)\left(\theta|\nabla\varphi|\right) &= \frac{\partial}{\partial\eta}E(\varphi+\eta\theta|\nabla\varphi|)\Big|_{\eta=0} \\ &= \int_{\Lambda}\frac{\partial}{\partial\eta}G\left(\mathbf{x}, h_{\varphi+\eta\theta|\nabla\varphi|}\right)\Big|_{\eta=0}\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x} \\ &\quad + \int_{\Lambda} G(\mathbf{x},h)\,\frac{\partial}{\partial\eta}\delta_{\varepsilon}\left(\varphi+\eta\theta|\nabla\varphi|\right)\Big|_{\eta=0}|\nabla\varphi| d\mathbf{x} \\ &\quad\quad + \int_{\Lambda} G(\mathbf{x},h_{\varphi})\delta_{\varepsilon}(\varphi)\,\frac{\partial}{\partial\eta}\left|\nabla\varphi+\eta\nabla\left(\theta|\nabla\varphi|\right)\right|\Big|_{\eta=0} d\mathbf{x} \\ &= I + II + III.\end{aligned}$$

The remainder of the proof will be presented in a few Lemmata, Lemma 5.2.2 and Lemma 5.2.3 which will respectively simplify the terms I and $II + III$ to the desired results. Finally we will put them together and discuss the simplifications in Lemma 5.2.4 when $|\nabla\varphi| = 1$. □

Lemma 5.2.2. *Under the assumption that $\delta_{\varepsilon}(\varphi)|_{\partial\Lambda} = 0$ as in Theorem 5.2.1, then the term, I in the proof of Theorem 5.2.1 simplifies to*

$$\begin{aligned}I &= \int_{\Lambda}\frac{\partial}{\partial\eta}G\left(\boldsymbol{x}, h_{\varphi+\eta\theta|\nabla\varphi|}\right)\Big|_{\eta=0}\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x} \\ &= -\int_{\Lambda}\operatorname{div}\left[\frac{P_{\varphi}\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\right]\theta\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x}.\end{aligned}$$

Proof. By the chain rule and Lemma 5.1.2 with $\psi = \theta|\nabla\varphi|$, we obtain

$$\begin{aligned}\frac{\partial}{\partial\eta}G\left(\mathbf{x}, h_{\varphi+\eta\theta|\nabla\varphi|}\right)\Big|_{\eta=0} &= \frac{\partial}{\partial h}G(\mathbf{x}, h_\varphi)\ \frac{d}{d\eta}\left(h_{\varphi+\eta\psi}\right)\Big|_{\eta=0} \\ &= \partial_2 G \operatorname{div}\left(\frac{-P_\varphi\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}\right).\end{aligned}$$

Putting this into I and integrating by parts (using ν for the outer unit normal on $\partial\Lambda$), we obtain

$$\begin{aligned}I &= -\int_\Lambda \partial_2 G \operatorname{div}\left(\frac{-P\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}\right)\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x} \\ &= \int_\Lambda \nabla\left(\partial_2 G\delta_\varepsilon(\varphi)|\nabla\varphi|\right)\cdot\frac{P_\varphi\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}d\mathbf{x} \\ &\qquad -\int_{\partial\Lambda}\partial_2 G\cancelto{0}{\delta_\varepsilon(\varphi)}|\nabla\varphi|\frac{P_\varphi\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}\cdot\nu d\mathbf{x} \\ &= \int_\Lambda P_\varphi\nabla\left(\partial_2 G\delta_\varepsilon(\varphi)|\nabla\varphi|\right)\cdot\frac{\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}d\mathbf{x} \\ &= \int_\Lambda \frac{P_\varphi\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\cdot\frac{\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\end{aligned}$$

since $P_\varphi\nabla\left(\delta_\varepsilon(\varphi)\right) = \mathbf{0}$ as per equation (5.27) of Lemma 5.1.3.

This stage is where we will normally end modifying term I in actual models, but we will continue by integrating by parts once more to expose the θ term. Again using $P_\varphi^2 = P_\varphi$, $P_\varphi\nabla\delta_\varepsilon(\varphi) = \mathbf{0}$, and $\delta_\varepsilon(\varphi) = 0$ on $\partial\Lambda$ we obtain

$$
\begin{aligned}
I &= -\int_\Lambda \operatorname{div}\left(\frac{P_\varphi \nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\delta_\varepsilon(\varphi)\right)\theta|\nabla\varphi|d\mathbf{x} \\
&\quad + \int_{\partial\Lambda} \frac{P_\varphi \nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\cdot \nu\cancelto{0}{\delta_\varepsilon(\varphi)}\theta|\nabla\varphi|d\mathbf{x} \\
&= -\int_\Lambda \operatorname{div}\left(\frac{P_\varphi \nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x} \\
&\quad - \int_\Lambda \frac{P_\varphi \nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\cdot \cancelto{\mathbf{0}}{P_\varphi \nabla(\delta_\varepsilon(\varphi))\theta}|\nabla\varphi|d\mathbf{x} \\
&= -\int_\Lambda \operatorname{div}\left(\frac{P_\varphi \nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.
\end{aligned}
$$

□

Lemma 5.2.3. *Under the assumption that $\delta_\varepsilon(\varphi)|_{\partial\Lambda} = 0$ as in Theorem 5.2.1, then the sum of terms, $II + III$ in the proof of Theorem 5.2.1 simplifies to*

$$
\begin{aligned}
II + III &= \int_\Lambda G(\boldsymbol{x}, h)\,\frac{\partial}{\partial\eta}\delta_\varepsilon\left(\varphi + \eta\theta|\nabla\varphi|\right)\Big|_{\eta=0}|\nabla\varphi|d\boldsymbol{x} \\
&\qquad + \int_\Lambda G(\boldsymbol{x}, h_\varphi)\delta_\varepsilon(\varphi)\,\frac{\partial}{\partial\eta}\left|\nabla\varphi + \eta\nabla\left(\theta|\nabla\varphi|\right)\right|\Big|_{\eta=0}d\boldsymbol{x} \\
&= \int_\Lambda \operatorname{div}\left[G\boldsymbol{n}_\varphi\right]\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x} \\
&= \int_\Lambda \left(\nabla G\cdot\boldsymbol{n}_\varphi + Gh_\varphi\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x}.
\end{aligned}
$$

Proof. To simplify II and III, we use the chain rule and Lemma 5.1.2 again with $\psi = \theta|\nabla\varphi|$ to get

$$
\begin{aligned}
\frac{\partial}{\partial\eta}\delta_\varepsilon\left(\varphi + \eta\theta|\nabla\varphi|\right)\Big|_{\eta=0} &= \delta_\varepsilon'(\varphi)\theta|\nabla\varphi| \\
&= \nabla\delta_\varepsilon(\varphi)\cdot\frac{\nabla\varphi}{|\nabla\varphi|}\theta
\end{aligned}
$$

and

$$\frac{\partial}{\partial\eta}\left|\nabla\varphi+\eta\nabla\left(\theta|\nabla\varphi|\right)|\right|_{\eta=0}=\frac{\nabla\varphi\cdot\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}.$$

Combining these results, recognizing $\mathbf{n}_\varphi=-\frac{\nabla\varphi}{|\nabla\varphi|}$ and integrating by parts we have

$$\begin{aligned}
II+III &= \int_\Lambda -G(\mathbf{x},h_\varphi)\nabla\delta_\varepsilon(\varphi)\cdot\mathbf{n}_\varphi\theta|\nabla\varphi|d\mathbf{x} \\
&\qquad -\int_\Lambda G(\mathbf{x},h_\varphi)\delta_\varepsilon(\varphi)\mathbf{n}_\varphi\cdot\nabla\left(\theta|\nabla\varphi|\right)d\mathbf{x} \\
&= -\int_\Lambda G(\mathbf{x},h_\varphi)\mathbf{n}_\varphi\cdot\nabla\left(\theta\delta_\varepsilon(\varphi)|\nabla\varphi|\right)d\mathbf{x} \\
&= \int_\Lambda \operatorname{div}\left(G(\mathbf{x},h_\varphi)\mathbf{n}_\varphi\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x} \\
&\qquad -\int_{\partial\Lambda} G(\mathbf{x},h_\varphi)\mathbf{n}_\varphi\cdot\nu\theta\cancelto{0}{\delta_\varepsilon(\varphi)}|\nabla\varphi|d\mathbf{x} \\
&= \int_\Lambda \operatorname{div}\left(G\mathbf{n}_\varphi\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}
\end{aligned}$$

where we have again used the assumption that $\delta_\varepsilon(\varphi)=0$ on $\partial\Lambda$. □

Lemma 5.2.4. *Using the results of Lemma 5.2.2 and Lemma 5.2.3, the variation of energy in the direction $\theta|\nabla\varphi|$ is*

$$\begin{aligned}
\partial E(\varphi)\left(\theta|\nabla\varphi|\right) &= \int_\Lambda \operatorname{div}\left[G\boldsymbol{n}_\varphi-\frac{P_\varphi\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x} \qquad (5.43)\\
&= \int_\Lambda\left[\nabla G\cdot\boldsymbol{n}_\varphi\theta+Gh_\varphi\theta+\frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\cdot\frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\boldsymbol{x}.
\end{aligned}$$

Again, when the levelset is $\varphi=f(d(\boldsymbol{x}))$ then $P_\varphi\nabla(|\nabla\varphi|)=\mathbf{0}$ and the energy variation

reduces to

$$\begin{aligned}\partial E(\varphi)\left(\theta|\nabla\varphi|\right) &= \int_{\Lambda} \operatorname{div}\left[G\boldsymbol{n}_{\varphi} - P_{\varphi}\nabla\left(\partial_2 G\right)\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x} \\ &= \int_{\Lambda}\left[\nabla G\cdot\boldsymbol{n}_{\varphi}\theta + Gh_{\varphi}\theta + P_{\varphi}\nabla(\partial_2 G)\cdot\nabla\theta\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\boldsymbol{x},\end{aligned} \tag{5.44}$$

thus completing the proof of Theorem 5.2.1.

Proof. We have

$$\begin{aligned}\partial E(\varphi)\left(\theta|\nabla\varphi|\right) &= I + II + III \\ &= -\int_{\Lambda}\operatorname{div}\left(\frac{P_{\varphi}\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right)\theta\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x} \\ &\quad + \int_{\Lambda}\operatorname{div}\left(G\mathbf{n}_{\varphi}\right)\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x} \\ &= \int_{\Lambda}\operatorname{div}\left[G\mathbf{n}_{\varphi} - \frac{P_{\varphi}\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x}\end{aligned}$$

as desired. If $\varphi = f(d(\mathbf{x}))$, then in addition $P_{\varphi}\nabla(|\nabla\varphi|) = \mathbf{0}$, that is that the level set profile only varies in the normal direction. Under this additional assumption, then the energy variation, specifically term I simplifies to

$$\partial E(\varphi)\left(\theta|\nabla\varphi|\right) = \int_{\Lambda}\operatorname{div}\left[G\mathbf{n}_{\varphi} - P_{\varphi}\nabla\left(\partial_2 G\right)\right]\delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x}. \tag{5.45}$$

A more useful weak form of this variation for computations comes from going back to our intermediate solution for the term I of Lemma 5.2.2 before the final integration by parts. We also apply the product rule to the other term and use that $h_{\varphi} = \operatorname{div}(\mathbf{n}_{\varphi})$. In that

case,

$$\partial E(\varphi)\,(\theta|\nabla\varphi|) = \int_\Lambda \left(\nabla G\cdot\mathbf{n}_\varphi + Gh_\varphi\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x} + \int_\Lambda \frac{P_\varphi\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\cdot\frac{\nabla\left(\theta|\nabla\varphi|\right)}{|\nabla\varphi|}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}$$

and when we make the assumption that $\varphi(\mathbf{x}) = f(d(\mathbf{x}))$, then $P_\varphi\nabla(|\nabla\varphi|) = \mathbf{0}$ and this simplifies to

$$\partial E(\varphi)\,(\theta|\nabla\varphi|) = \int_\Lambda \left[\nabla G\cdot\mathbf{n}_\varphi\theta + Gh_\varphi\theta + P\nabla(\partial_2 G)\cdot\nabla\theta\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.$$

□

Remark. This simplification using $\varphi(\mathbf{x}) = f(d(\mathbf{x}))$ is the model will often be used in implementation. We spend resources to push φ to the distance function, d on the band of importance using the filter of equation (2.12). We have tested the full model and the simplified model and find them to give equivalent evolutions and steady states when we spend adequate resources to preserve the levelset shape. Thus we feel it reasonable to make this assumption and use this simplified model. *end Remark.*

Remark. If Γ is close to or intersect the boundary $\partial\Lambda$ then an extra term does not cancel out which accounts for the contact physics. This is expected as there are always additional terms for contact angle in surface tension or wetting problems (see [58]). We will not treat these cases here as we are typically considering a closed vesicle in a large area and our region with boundary is considered to be merely a snapshot of the larger region. If the $\delta_\varepsilon(\varphi)$ function gets close to the boundary but does not intersect it, it may lose accuracy due to the area under the approximate delta function in the normal direction no longer adding up to 1. Some work has been done by Tornberg in the Extensions section of [29] to adjust δ_ε to be consistent in this case which might be necessary in the case of flow

through an actual enclosed region, for instance when modelling the micropipette aspiration of biological cells, or flow of a blood cell through a vein. *end Remark.*

5.3 Stability of the force balance algorithm

We will now show how this coupling leads to stability estimates in our models. We begin with a simple lemma for handling the material derivative.

Lemma 5.3.1. *Given a smooth solution of the (in)compressible Navier-Stokes system with variable density $\rho = \rho(t, \boldsymbol{x})$ governed by $\rho_t + \operatorname{div}(\rho \boldsymbol{u}) = 0$, we have*

$$\begin{aligned}\int_\Lambda \rho D_t \boldsymbol{u} \cdot \boldsymbol{u} d\boldsymbol{x} &= \int_\Lambda \rho \boldsymbol{u}_t \cdot \boldsymbol{u} d\boldsymbol{x} + \int_\Lambda (\rho \boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{u} d\boldsymbol{x} \\ &= \int_\Lambda \frac{d}{dt}\left(\frac{1}{2}\rho(t)|\boldsymbol{u}(t)|^2\right) d\boldsymbol{x} + \int_{\partial\Lambda} \frac{\rho}{2}(\boldsymbol{u}\cdot\boldsymbol{n})|\boldsymbol{u}|^2 d\boldsymbol{x}\end{aligned}$$

where D_t is the material derivative. Thus if $\boldsymbol{u} = 0$ or $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on $\partial\Lambda$, then this term reduces to the rate of change of kinetic energy,

$$\int_\Lambda \rho D_t \boldsymbol{u} \cdot \boldsymbol{u} d\boldsymbol{x} = \frac{d}{dt}\int_\Lambda \frac{1}{2}\rho(t)|\boldsymbol{u}(t)|^2 d\boldsymbol{x}.$$

Proof. We have a standard equality from expansion of $\nabla(\mathbf{A}\cdot\mathbf{B})$ with $\mathbf{A} = \rho\mathbf{u}$ and $\mathbf{B} = \mathbf{u}$,

$$\rho\mathbf{u}\cdot\nabla\mathbf{u} = \nabla\left(\rho|\mathbf{u}|^2\right) - \mathbf{u}\cdot\nabla(\rho\mathbf{u}) - \rho\mathbf{u}\times(\nabla\times\mathbf{u}) - \mathbf{u}\times(\nabla\times(\rho\mathbf{u}))\,.$$

Now, since $\mathbf{u}$ is orthogonal to these last two terms, we have

$$\begin{aligned}\rho\mathbf{u}\cdot\nabla\mathbf{u}\cdot\mathbf{u} &= \nabla\left(\rho|\mathbf{u}|^2\right)\cdot\mathbf{u} - \mathbf{u}\cdot\nabla(\rho\mathbf{u})\cdot\mathbf{u} \\ &= \operatorname{div}\left(\rho|\mathbf{u}|^2\mathbf{u}\right) - \rho|\mathbf{u}|^2\operatorname{div}(\mathbf{u}) - \mathbf{u}\cdot\nabla\rho|\mathbf{u}|^2 - \rho\mathbf{u}\cdot\nabla\mathbf{u}\cdot\mathbf{u} \\ &= \operatorname{div}\left(\rho|\mathbf{u}|^2\mathbf{u}\right) - \operatorname{div}(\rho\mathbf{u})|\mathbf{u}|^2 - \rho\mathbf{u}\cdot\nabla\mathbf{u}\cdot\mathbf{u} \\ &= \operatorname{div}\left(\rho|\mathbf{u}|^2\mathbf{u}\right) + \rho_t|\mathbf{u}|^2 - \rho\mathbf{u}\cdot\nabla\mathbf{u}\cdot\mathbf{u},\end{aligned}$$

using $\rho_t + \operatorname{div}(\rho\mathbf{u}) = 0$, so that

$$\begin{aligned}\int_\Lambda \rho\mathbf{u}\cdot\nabla\mathbf{u}\cdot\mathbf{u}d\mathbf{x} &= \frac{1}{2}\int_\Lambda \operatorname{div}\left(\rho|\mathbf{u}|^2\mathbf{u}\right)d\mathbf{x} + \frac{1}{2}\int_\Lambda \rho_t|\mathbf{u}|^2 d\mathbf{x} \\ &= \frac{1}{2}\int_{\partial\Lambda}\rho|\mathbf{u}|^2\mathbf{u}\cdot\mathbf{n}d\mathbf{x} + \frac{1}{2}\int_\Lambda \rho_t|\mathbf{u}|^2 d\mathbf{x},\end{aligned}$$

where we have used the Divergence theorem to arrive at this last step. Thus all together we have

$$\begin{aligned}\int_\Lambda \rho\mathbf{u}_t\cdot\mathbf{u}d\mathbf{x} + \int_\Lambda(\rho\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u}d\mathbf{x} &= \frac{1}{2}\int_\Lambda \rho(t)\frac{d}{dt}\left(|\mathbf{u}(t)|^2\right)d\mathbf{x} + \frac{1}{2}\int_\Lambda \frac{d}{dt}(\rho(t))|\mathbf{u}(t)|^2 d\mathbf{x} \\ &\quad + \int_{\partial\Lambda}\frac{1}{2}\rho|\mathbf{u}|^2\mathbf{u}\cdot\mathbf{n}d\mathbf{x} \\ &= \frac{d}{dt}\int_\Lambda \frac{1}{2}\rho(t)|\mathbf{u}(t)|^2 d\mathbf{x} + \int_{\partial\Lambda}\frac{1}{2}\rho|\mathbf{u}|^2\mathbf{u}\cdot\mathbf{n}d\mathbf{x},\end{aligned}$$

so that if $\mathbf{u} = 0$ or $\mathbf{u}\cdot\mathbf{n} = 0$ on $\partial\Lambda$ then

$$\int_\Lambda \rho\mathbf{u}_t\cdot\mathbf{u}d\mathbf{x} + \int_\Lambda(\rho\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u}d\mathbf{x} = \frac{d}{dt}\int_\Lambda \frac{1}{2}\rho(t)|\mathbf{u}(t)|^2 d\mathbf{x}.$$

□

5.3.1 Stability of coupled system

We show the stability of the continuous Navier-Stokes system with the energy flow coupling

$$\int_\Gamma [\![2\mu\nabla^s\mathbf{u} - pI]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} = -\partial E(\varphi)(\mathbf{v} \cdot \mathbf{n}_\varphi |\nabla\varphi|)$$

with $\mathbf{f} = \mathbf{0}$. We have the Navier-Stokes system of equation (4.2) for $(\mathbf{u}, p)$:

$$\begin{aligned}\int_\Lambda \rho\mathbf{u}_t \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda \rho\mathbf{u} \cdot \nabla\mathbf{u} \cdot \mathbf{v} d\mathbf{x} + \int_\Lambda 2\mu\nabla^s\mathbf{u} \cdot \nabla^s\mathbf{v} d\mathbf{x} \\ - \int_\Gamma [\![2\mu\nabla^s\mathbf{u} - pI]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} - \int_\Lambda p \operatorname{div} \mathbf{v} d\mathbf{x} = 0\end{aligned}$$

with the true solution $\mathbf{v} = \mathbf{u}$. Using Lemma 5.3.1 with our slip or no-slip conditions for velocity, then we have the estimate

$$\frac{d}{dt}\int_\Lambda \frac{1}{2}\rho|\mathbf{u}|^2 d\mathbf{x} + \int_\Lambda 2\mu|\nabla^s\mathbf{u}|^2 d\mathbf{x} + \partial E(\varphi)(\mathbf{u} \cdot \mathbf{n}_\varphi|\nabla\varphi|) = 0.$$

Finally, we recall that $\varphi_t = -\mathbf{u} \cdot \nabla\varphi = \mathbf{u} \cdot \mathbf{n}_\varphi|\nabla\varphi|$, so that

$$\partial E(\varphi)(-\mathbf{u} \cdot \mathbf{n}_\varphi|\nabla\varphi|) = \partial E(\varphi)(\varphi_t) = \frac{d}{dt}E(\varphi).$$

Theorem 5.3.2. *Testing the continuous Navier-Stokes system with approximate energy flow against the true solution, $\boldsymbol{u}(t)$, and with no external sources, $\boldsymbol{f} = 0$, yields the stability estimate*

$$\frac{d}{dt}\int_\Lambda \frac{1}{2}\rho(t)|\boldsymbol{u}(t)|^2 d\boldsymbol{x} + \int_\Lambda 2\mu(t)|\nabla^s\boldsymbol{u}(t)|^2 d\boldsymbol{x} + \frac{d}{dt}E(\varphi(t)) = 0. \tag{5.46}$$

5.4 Comparison to the method of virtual power in continuum mechanics

Now instead of coupling the variation to the jump in the stress tensor, another method common in the physics and continuum mechanics disciplines is to simply consider the forces from the surface to be just another force in the derivation of the Navier-Stokes system to be balanced. Following the notation of [59], we will call it the forces from the curvature of membrane, and denote it by F_c. Then the constitutive equations for $(\mathbf{u}, \varphi)$ are

$$\begin{aligned} \rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) - \operatorname{div}(2\eta \nabla^s \mathbf{u}) + \nabla p &= \mathbf{F}_c \\ \operatorname{div}(\mathbf{u}) &= 0 \\ \varphi_t + \mathbf{u} \cdot \nabla \varphi &= 0. \end{aligned}$$

The vector force $\mathbf{F}_c$ is computed using the method of virtual power where given an energy functional $E(\varphi)$, we compute

$$\int_\Lambda \mathbf{F}_c \cdot \mathbf{u} d\mathbf{x} = -\frac{d}{dt} E(\varphi) = -\partial E(\varphi)(\varphi_t) = \partial E(\varphi)(\mathbf{u} \cdot \nabla \varphi).$$

where we have used the levelset relation $\varphi_t + \mathbf{u} \cdot \nabla \varphi = 0$ in the last step. In [59], the method of virtual power has been used for the energy in the form $E(\varphi) = \int_\Lambda G(h) \delta_\varepsilon(\varphi) |\nabla \varphi| d\mathbf{x}$ where they consider the Willmore forces, $G(h) = \frac{1}{2} h^2$. The variational form of the energy that they obtain is in fact the same as ours although our derivations are approached in a completely different manner. They obtain $\mathbf{F}_c$ as

$$\mathbf{F}_c = \operatorname{div}\left[-G(h_\varphi) \frac{\nabla \varphi}{|\nabla \varphi|} + \frac{P \nabla \left(G'(h_\varphi) |\nabla \varphi| \right)}{|\nabla \varphi|} \right] \delta_\varepsilon(\varphi) \nabla \varphi$$

whereas we obtain in Section 7.2 that

$$\int_\Lambda g\theta|\nabla\varphi|d\mathbf{x} = -\int_\Lambda \operatorname{div}\left[G\mathbf{n}_\varphi - \frac{P_\varphi\nabla\left(\partial_2 G|\nabla\varphi|\right)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}$$
$$= -\int_\Lambda\left[\nabla G\cdot\mathbf{n}_\varphi\theta + Gh_\varphi\theta + \frac{P_\varphi\nabla(\partial_2 G|\nabla\varphi|)}{|\nabla\varphi|}\cdot\frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.$$

Notice that

$$\int_\Lambda g\theta|\nabla\varphi|d\mathbf{x} = \int_\Lambda \mathbf{F}_c\cdot\left(-\frac{\nabla\varphi}{|\nabla\varphi|}\right)\theta d\mathbf{x} = \int_\Lambda \mathbf{F}_c\cdot\mathbf{n}_\varphi\theta d\mathbf{x}$$

or more generally since $\mathbf{F}_c$ is colinear to $\nabla\varphi$ and thus to $\mathbf{n}_\varphi$, testing against a general velocity field, $\mathbf{v}$, yields

$$\int_\Lambda \mathbf{F}_c\cdot\mathbf{v}d\mathbf{x} = \int_\Lambda \mathbf{F}_c\cdot\mathbf{n}_\varphi(\mathbf{v}\cdot\mathbf{n}_\varphi)d\mathbf{x} = \int_\Lambda g(\mathbf{v}\cdot\mathbf{n}_\varphi)|\nabla\varphi|d\mathbf{x} = \int_\Lambda (g\mathbf{n}_\varphi|\nabla\varphi|)\cdot\mathbf{v}d\mathbf{x}.$$

Thus, the force terms can be directly related, $\mathbf{F}_c = g|\nabla\varphi|\mathbf{n}_\varphi = -g\nabla\varphi$. We can view $\mathbf{F}_c$ as the vector normal force from the energy on the membrane, where as our $g \in L^2(\Lambda)$ is the scalar normal force from the energy on the membrane. In our opinion both formulations are comparable. We couple the energy minimizing force to the jump in the stress tensor across the surface whereas they couple theirs as an external bulk force.

Another major difference is in the approach to discretizing the approach. They do not directly discuss their method of discretizing and solving their system in [59] although in some subsequent papers by another group including [60], [61] and [62], some more detailed approaches to discretizing this model are included. In particular, the force function F_c is computed explicitly from the distance function and other intermediate terms which smooth terms to obtain a usable F_c. They focus on using higher order finite element approximations to the level set to get better approximations of the surface, curvature and the

Willmore force F_c. They also allow for the levelset to stretch and deviate from the distance function and they add an additional energy to be minimized which penalizes stretching.

We will present two methods for discretizing the system in Chapter 7. Our first method of Section 7.2 uses a semi-implicit formulation to couple the curvature, and thus the Willmore forces, to the velocity and decouple them from the levelset. Our second method of Section 7.3 uses a subiterating scheme which is also different than their approach. In addition we provide a method for enforcing the surface area and volume constraints directly through the use of Lagrange multipliers (see Chapter 8), thereby extending the Willmore energy flow to the full Canham-Helfrich energy flow.

6. ENERGY FLOW APPLICATION: SURFACE TENSION FLOW

We now consider the case of a simple surface tension model with

$$e_{st}(\Gamma) = \sigma_{st} \int_{\Gamma} d\mathbf{x} \tag{6.1}$$

with $\sigma_{st} \in \mathbb{R}^+$, the coefficient of surface tension. We will not go into the specifics of what assumptions are made to arrive from basic physics to this energy formulation, but take it as our model as is. Undoubtedly there are more complicated approaches and energies that capture the various interesting effects that can happen under the umbrella of surface tension or more generally under the effect of capillarity forces, but we will not delve into these at this time. This simple model captures the most basic type of surface tension effect and so suits our needs. We refer the reader to [56] for a more detailed discussion of energy based flows related to surface tension, capillarity and wetting.

Using the notation of Chapter 5, we set

$$G(\mathbf{x}, h_\varphi) = \sigma_{st} \in \mathbb{R}^+, \tag{6.2}$$

a constant function. Using shape calculus, the variation of $e(\Gamma)$ (with Γ a C^2-surface) under the velocity field $\mathbf{v}$ is

$$\partial e(\Gamma)(\mathbf{v}) = \sigma_{st} \int_{\Gamma} h\mathbf{n} \cdot \mathbf{v} d\mathbf{x}. \tag{6.3}$$

Then we introduce the approximate energy functional

$$E(\varphi) = \sigma_{st} \int_{\Lambda} \delta_\varepsilon(\varphi) |\nabla \varphi| d\mathbf{x}$$

which using Theorem 5.2.1, has variation for $\theta \in H^1(\Lambda)$,

$$\partial E(\varphi)(\theta|\nabla\varphi|) = \int_\Lambda \operatorname{div}(\sigma_{st}\mathbf{n}_\varphi)\,\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x} = \sigma_{st}\int_\Lambda h_\varphi\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}. \tag{6.4}$$

Later we will use $\theta = \mathbf{v}\cdot\mathbf{n}_\varphi$ so that

$$\partial E(\varphi)(\mathbf{v}\cdot\mathbf{n}_\varphi|\nabla\varphi|) = \sigma_{st}\int_\Lambda h_\varphi\mathbf{n}_\varphi\cdot\mathbf{v}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}$$

and we immediately see the similarities between $\partial e(\Gamma)$ and $\partial E(\varphi)$.

We will first treat in Section 6.1.1 on the fully implicit algorithm that comes from using this energy variation in the force balance equation. Recall that we have already shown in Chapter 5.3 that all the energy variation approaches using a semi-discrete fully implicit algorithm have an energy stability related to the them. We will then in Section 6.1.2, present a semi-implicit algorithm that allows us to alternate between computing a velocity that takes into account the surface tension forces and updating the levelset. We will then compare this algorithm with some other approaches in Section 6.2 and finally we will present some standard numerical results for surface tension in Section 6.3.

6.1 Implementation of surface tension

We describe some algorithms for computing the two-phase flow with surface tension using the energy flow formulations of Chapter 5. We start with the fully implicit algorithm and then introduce a semi-implicit splitting to couple the surface tension forces with the velocity and decouple them from the level set. This allows us to maintain the algorithm on each time step of first solving for the velocity field and then transporting the levelset with that velocity field.

6.1.1 *Fully implicit scheme*

Given a time discretization $t^0 < t^1 < \cdots < t^k < \cdots < t^N$, we denote the times step by superscript, ie $f^k \approx f\left(t^k\right)$. Suppose we are at time t^k and want to solve for solutions at t^{k+1}. Following the notation of Chapter 5, we introduce a scalar function, $h_\varphi^{k+1} = \operatorname{div} \mathbf{n}_\varphi^{k+1}$, for the total curvature at time t^{k+1}. Then given (φ^k, $\mathbf{u}^k$) we must solve for $(\varphi^{k+1}, \mathbf{u}^{k+1}, h_\varphi^{k+1})$ such that:

$$\frac{\partial \varphi}{\partial t}(t^{k+1}) + \mathbf{u}^{k+1} \cdot \nabla \varphi^{k+1} = 0, \tag{6.5}$$

an implicit version of equation (2.1) and the fully implicit Navier-Stokes system (4.2) with the energy force balance coupling

$$\int_\Gamma \left[\!\left[2\mu \nabla^s \mathbf{u}^{k+1} - p^{k+1} I \right]\!\right] \cdot \mathbf{n}^{k+1} \cdot \mathbf{v} d\mathbf{x} = \sigma_{st} \int_\Lambda h_\varphi^{k+1} \mathbf{n}_\varphi^{k+1} \cdot \mathbf{v} |\nabla \varphi^{k+1}| d\mathbf{x}$$

and

$$\int_\Lambda h_\varphi^{k+1} \theta d\mathbf{x} = \int_\Lambda \frac{\nabla \varphi^{k+1}}{|\nabla \varphi^{k+1}|} \cdot \nabla \theta d\mathbf{x} - \int_{\partial \Lambda} \frac{\nabla \varphi^{k+1}}{|\nabla \varphi^{k+1}|} \cdot \nu \theta d\mathbf{x},$$

a weak formulation of total curvature, $h_\varphi = \operatorname{div} \mathbf{n}_\varphi$.

6.1.2 *Semi-implicit scheme*

The previous system is nonlinear and the curvature is dependent on the levelset function directly and only circuitously related to the velocity field. We will implement a splitting scheme that separates the levelset update from the others:

(i) Given $(\varphi^k, \mathbf{u}^k, \mathbf{u}^{k-1})$ solve for $(\mathbf{u}^{k+1}, h^{k+1})$

(ii) Given $(\varphi^k, \varphi^{k-1}, \mathbf{u}^{k+1})$ solve for φ^{k+1}.

To achieve this, we follow and idea of Bansch and decouple h_φ^{k+1} from φ^{k+1} and couple it directly to the velocity, which we do by introducing the scalar function,

$$u_n^{k+1} = -\mathbf{u}^{k+1} \cdot \nabla \varphi^k,$$

the normal velocity. Recall that the surface normal $\mathbf{n}_\varphi^k = -\frac{\nabla \varphi^k}{|\nabla \varphi^k|}$ so that this is the normal velocity when $|\nabla \varphi^k| = 1$, otherwise it is not scaled not quite right. However we filter the levelset φ so that it is a distance function with $|\nabla \varphi| = 1$ in the region that matters. In weak form, we seek $u_n^{k+1} \in L^2(\Lambda)$ such that for all $\theta \in L^2(\Lambda)$

$$\int_\Lambda u_n^{k+1} \theta d\mathbf{x} = -\int_\Lambda \mathbf{u}^{k+1} \cdot \nabla \varphi^k \theta d\mathbf{x}.$$

Expanding the levelset equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = 0,$$

we have

$$\begin{aligned} \frac{\partial \varphi}{\partial t}\left(t^k\right) &= -\mathbf{u}\left(t^k\right) \cdot \nabla \varphi\left(t^k\right) \\ &= -\mathbf{u}\left(t^{k+1}\right) \cdot \nabla \varphi\left(t^k\right) + \Delta t \frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi\left(t^k\right) \end{aligned}$$

for some $\xi \in \left(t^k, t^{k+1}\right)$. Then discretizing the time derivative of φ with a forward difference, we obtain

$$\frac{\varphi^{k+1} - \varphi^k}{\Delta t} = -\mathbf{u}^{k+1} \cdot \nabla \varphi^k + \Delta t \frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi\left(t^k\right) + \frac{\Delta t}{2} \frac{\partial^2 \varphi}{\partial t^2}\left(t^k\right)$$

so that using the normal velocity, u_n^{k+1}, we have the first order update equation with Δt^2 error

$$\varphi^{k+1} = \left(\varphi^k + \Delta t u_n^{k+1}\right) + \Delta t^2 \left(\frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi^k + \frac{1}{2}\frac{\partial^2 \varphi}{\partial t^2}\left(t^k\right)\right). \tag{6.6}$$

Now, we can replace φ^{k+1} with $\varphi^k + \Delta t u_n^{k+1}$ in some places to couple h^{k+1} to $\mathbf{u}^{k+1}$ instead of φ^{k+1}. We substitute all the $|\nabla \varphi^{k+1}|$ terms with $|\nabla \varphi^k|$ evaluated at the previous time and do the same with the Dirac measure so that they are all defined on φ^k. We can thus write $\mathbf{n}_\varphi^{k+1} = -\frac{\nabla \varphi^{k+1}}{|\nabla \varphi^{k+1}|} \approx -\frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|}$. Thus our semi-implicit scheme for curvature is

$$\begin{aligned}
\int_\Lambda h_\varphi^{k+1} \theta d\mathbf{x} &= \int_\Lambda \operatorname{div}(\mathbf{n}_\varphi^{k+1}) \theta d\mathbf{x} \\
&= -\int_\Lambda \mathbf{n}_\varphi^{k+1} \cdot \nabla \theta d\mathbf{x} + \int_{\partial\Lambda} \mathbf{n}_\varphi^{k+1} \cdot \nu \theta d\mathbf{x} \\
&= \int_\Lambda \frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} - \int_{\partial\Lambda} \frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x}
\end{aligned}$$

or separating the implicit and explicit terms we have

$$\begin{aligned}
\int_\Lambda h_\varphi^{k+1} \theta d\mathbf{x} - \Delta t \int_\Lambda \frac{\nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} &= \int_\Lambda \frac{\nabla \varphi^k}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} \\
+\Delta t \int_{\partial\Lambda} \frac{\nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x} \qquad &- \int_{\partial\Lambda} \frac{\nabla \varphi^k}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x}.
\end{aligned} \tag{6.7}$$

6.1.2.1 Summary of semi-implicit scheme

To summarize, we have the Navier-Stokes system of equations with force balance coupling

$$\begin{aligned}
\int_\Gamma [\![2\mu \nabla^s \mathbf{u}^{k+1} - p^{k+1} I]\!] \cdot \mathbf{n}^k \cdot \mathbf{v} d\mathbf{x} &= \sigma_{st} \int_\Lambda h_\varphi^{k+1} \mathbf{n}_\varphi^k \cdot \mathbf{v} \delta_\varepsilon(\varphi^k) |\nabla \varphi^k| d\mathbf{x} \\
&= -\sigma_{st} \int_\Lambda h_\varphi^{k+1} \frac{\nabla \varphi^k}{|\nabla \varphi^k|} \cdot \mathbf{v} \delta_\varepsilon(\varphi^k) |\nabla \varphi^k| d\mathbf{x}
\end{aligned}$$

And the other coupled equations

$$\int_\Lambda u_n^{k+1}\theta d\mathbf{x} + \int_\Lambda \mathbf{u}^{k+1}\cdot\nabla\varphi^k\theta d\mathbf{x} = 0$$

$$\begin{aligned}\int_\Lambda h_\varphi^{k+1}\theta d\mathbf{x} - \Delta t\int_\Lambda \frac{\nabla u_n^{k+1}}{|\nabla\varphi^k|}\cdot\nabla\theta d\mathbf{x} &= \int_\Lambda \frac{\nabla\varphi^k}{|\nabla\varphi^k|}\cdot\nabla\theta d\mathbf{x}\\ +\Delta t\int_{\partial\Lambda}\frac{\nabla u_n^{k+1}}{|\nabla\varphi^k|}\cdot\nu\theta d\mathbf{x} &\quad -\int_{\partial\Lambda}\frac{\nabla\varphi^k}{|\nabla\varphi^k|}\cdot\nu\theta d\mathbf{x}.\end{aligned}$$

These are all solved simultaneously which provides a velocity with which to advect the levelset. There are $d+2$ components which gives a 3x3-block linear algebra system of the form

$$\begin{bmatrix}\mathbf{S}_{ns} & \mathbf{0} & \sigma_{st}\mathbf{M}_\delta\\ \mathbf{M}_{\nabla\varphi}^T & M & 0\\ \mathbf{0} & \Delta t S_h & M\end{bmatrix}\begin{bmatrix}\mathbf{U}^{k+1}\\ U_n^{k+1}\\ H^{k+1}\end{bmatrix} = \begin{bmatrix}\mathbf{F}_{ns}\\ 0\\ F_h\end{bmatrix}. \tag{6.8}$$

One could use a parallel direct solver to solve this non-symmetric system. This is quick but requires a large amount of memory to perform the LU decomposition. A better approach is to use a preconditioned GMRES iterative solver. We have found that a block diagonal preconditioner composed of an Algebraic Multigrid (AMG) solver for the first (Navier-Stokes) diagonal block and then an SSOR preconditioner for each of the second and third diagonal blocks seems to work well. Even though the block preconditioner is only applied to the diagonal blocks, this seems to work well enough. It solves quite quickly and the number of outer GMRES steps is relatively constant.

6.2 Connections to other models

At this stage, we return to how our model relates to other models in the literature for surface tension. There are more approaches than could reasonable be compared here so we focus on two other approaches. The first is the current energy model with the exact

energy functional. We show that in the limit as $\varepsilon \to 0$, our approximate model converges to the original model.

Theorem 6.2.1. *Supposing that the interface $\Gamma \subset \Lambda$ defined implicitly through the level set function, φ, is smooth with $\Gamma \cap \partial\Lambda = \emptyset$, then for smooth velocity field $\boldsymbol{v}$*

$$\partial E_{st}(\varphi)(\boldsymbol{v} \cdot \boldsymbol{n}_\varphi |\nabla\varphi|) = \sigma_{st} \int_\Lambda h_\varphi \boldsymbol{v} \cdot \boldsymbol{n}_\varphi \delta_\varepsilon(\varphi) |\nabla\varphi| d\boldsymbol{x} \overset{\varepsilon-\lim}{=} \sigma_{st} \int_\Gamma h\boldsymbol{n} \cdot \boldsymbol{v} d\boldsymbol{x} = \partial e_{st}(\Gamma)(\boldsymbol{v}).$$

Proof. Using the convergence of $\delta_\varepsilon(\varphi)|\nabla\varphi|$ to δ_Γ as in Chapter 3, in particular supposing that $h\mathbf{n} \cdot \mathbf{v} \in W^{6,\infty}$ near the surface Γ with two bounded derivatives in the normal direction, we get the convergence of the functionals. This is reasonable for flows with medium to high surface tension values since physically, the surfaces stay quite smooth so the curvature, h is also smooth as well as the normal velocity, $\mathbf{n} \cdot \mathbf{v}$. The final equality comes from standard shape calculus tools applied to $e_{st}(\Gamma) = \sigma_{st} \int_\Gamma d\mathbf{x}$. □

Mirroring the stability estimate, Theorem 5.3.2, we have the following stability for surface tension.

Theorem 6.2.2. *Testing the continuous Navier-Stokes system (4.1) with force balance,*

$$\int_\Gamma [\![2\mu \nabla^s \boldsymbol{u} - pI]\!] \cdot \boldsymbol{n} \cdot \boldsymbol{v} d\boldsymbol{x} = -\partial E_{sa}(\varphi)(\boldsymbol{v} \cdot \boldsymbol{n}_\varphi |\nabla\varphi|) := \sigma_{st} \int_\Lambda h_\varphi \boldsymbol{n}_\varphi \cdot \boldsymbol{v} |\nabla\varphi| d\boldsymbol{x}$$

against the true velocity solution, $\boldsymbol{v} = \boldsymbol{u}$, and with no external sources, $\boldsymbol{f} = \mathbf{0}$, yields the stability estimate

$$\frac{d}{dt} \int_\Lambda \frac{1}{2} \rho(t) |\boldsymbol{u}(t)|^2 d\boldsymbol{x} + \int_\Lambda 2\mu(t) |\nabla^s \boldsymbol{u}(t)|^2 d\boldsymbol{x} + \frac{d}{dt} E_{sa}(\varphi(t)) = 0. \tag{6.9}$$

6.2.1 A more efficient model for surface tension

While the above model is useful for proving theorems and understanding what is happening geometrically, it produces a large block system for the velocity update which is not as desirable to handle as other formulations. In the case of simple surface tension, there are many formulations out there which each have their benefits and issues. We will now describe an equivalent model which is preferable when solving for surface tension effects since it avoids the extra components needed above. This method was first introduced by Bansch in [63] and has been used in many other papers since then. The key is to use the well known identity from differential geometry that vector total curvature $\mathbf{h} := h\mathbf{n} = \Delta_\Gamma X$ is the Laplace-Beltrami applied to the identity function $X(\mathbf{x}) = \mathbf{x}$ on the surface, Γ. The resulting implicit formulations are completely equivalent to our method but the semi-implicit methods utilize a different set of tools to couple surface tension with velocity instead of the levelset. In experiments, the results are the same, but the amount of work required to solve this model is much smaller than in our model described above. The redeeming quality of our approach is that it provides a stability estimate, Theorem 6.2.2 for the implicit version of all of these other models and an intuition for the flow since the surface tension is balancing the minimization of a scaled surface area energy with the other inertial and external forces. This is already generally understood by practitioners but it provides a formalism to the matter. Likewise our previous model computes the total curvature directly whereas this model does not.

To begin with, we have shown that

$$\partial E_{st}(\varphi)(\theta|\nabla\varphi|) = \sigma_{st}\int_\Lambda h_\varphi \theta \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}.$$

Then using that $\Delta_{\Gamma_t} X = h\mathbf{n}$ for the identity $X(\mathbf{x}) = \mathbf{x}$ on each Γ_t and using the co-area

formula, we rewrite the force balance equation

$$\begin{aligned}\int_\Gamma [\![2\mu\nabla^s\mathbf{u} - pI]\!]\cdot\mathbf{n}\cdot\mathbf{v}d\mathbf{x} &= -\partial E_{st}(\varphi)\left(\mathbf{n}_\varphi\cdot\mathbf{v}|\nabla\varphi|\right)\\ &= -\sigma_{st}\int_\Lambda h_\varphi\mathbf{n}_\varphi\cdot\mathbf{v}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &= -\sigma_{st}\int_{-\infty}^{\infty}\delta_\varepsilon(t)\int_{\Gamma_t} h\mathbf{n}\cdot\mathbf{v}d\mathbf{s}dt\\ &= -\sigma_{st}\int_{-\infty}^{\infty}\delta_\varepsilon(t)\int_{\Gamma_t}\Delta_{\Gamma_t}X\cdot\mathbf{v}d\mathbf{s}dt\\ &= \sigma_{st}\int_{-\infty}^{\infty}\delta_\varepsilon(t)\int_{\Gamma_t}\nabla_{\Gamma_t}X:\nabla_{\Gamma_t}\mathbf{v}d\mathbf{s}dt\\ &= \sigma_{st}\int_\Lambda\nabla_\varphi X:\nabla_\varphi\mathbf{v}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.\end{aligned}$$

Here we have used the vector integration by parts formula on a manifold, Γ_t,

$$-\int_{\Gamma_t}\Delta_{\Gamma_t}X\cdot\mathbf{v}d\mathbf{s} = \int_{\Gamma_t}\nabla_{\Gamma_t}X:\nabla_{\Gamma_t}\mathbf{v}d\mathbf{s} - \int_{\partial\Gamma_t}\mathbf{v}\cdot\left(\nabla_{\Gamma_t}X\cdot(\boldsymbol{\tau}\times\boldsymbol{\nu})\right)d\mathbf{s}$$

and that supp$\{\delta_\varepsilon(\varphi)\}\cap\partial\Lambda = \emptyset$. This guarantees that each Γ_t, for $t\in[-\varepsilon,\varepsilon]$ in the support of $\delta_\varepsilon(\varphi)$, is indeed closed and $\partial\Gamma_t = \emptyset$. This coupled with the two-phase flow model above constitutes the nonlinear implicit model where we use each variable evaluated at time t^{k+1}.

6.2.1.1 Semi-implicit formulation

To formulate the semi-implicit model and decouple the surface tension from the implicit levelset variable, we denote

$$X^k := \{\mathbf{x}\in\Gamma^k\} = \{\mathbf{x}\in\mathbb{R}^d \mid \varphi^k(\mathbf{x}) = 0\} \tag{6.10}$$

We linearize the position dependence X^{k+1} as

$$X^{k+1} = X^k + \Delta t^{k+1}\mathbf{u}^{k+1}.$$

where we recall that $\mathbf{u}^{k+1}$ is the as yet unknown velocity of the fluid flow at time t^{k+1}. We use the previous value φ^k in the $\delta_\varepsilon(\varphi)|\nabla\varphi|$ terms as well as in the ∇_{φ^k} tangential gradient. Then substituting the linearization, we obtain

$$\begin{aligned}\int_\Gamma [\![2\mu\nabla^s\mathbf{u}^{k+1} - p^{k+1}I]\!]\cdot\mathbf{n}^{k+1}\cdot\mathbf{v}d\mathbf{x} &= \sigma_{st}\int_\Lambda \nabla_{\varphi^k}X^{k+1} : \nabla_{\varphi^k}\mathbf{v}\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\ &= \sigma_{st}\int_\Lambda \nabla_{\varphi^k}X^{k} : \nabla_{\varphi^k}\mathbf{v}\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\ &\quad + \Delta t^{k+1}\sigma_{st}\int_\Lambda \nabla_{\varphi^k}\mathbf{u}^{k+1} : \nabla_{\varphi^k}\mathbf{v}\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}.\end{aligned}$$

Notice that everything here is defined with regards to the levelset φ^k at time t^k. We finally simplify by using that $\nabla_{\varphi^k} = (I - \mathbf{n}_\varphi^k\otimes\mathbf{n}_\varphi^k)\nabla = P_\varphi^k\nabla$ and so $\nabla_{\varphi^k}X^k = P_\varphi^k\nabla X^k = P_\varphi^k I = P_\varphi^k$, thus the force balance for surface tension is

$$\begin{aligned}\int_\Gamma [\![2\mu\nabla^s\mathbf{u}^{k+1} - p^{k+1}I]\!]\cdot\mathbf{n}^{k+1}\cdot\mathbf{v}d\mathbf{x} &= \sigma_{st}\int_\Lambda \left(P_\varphi^k : P_\varphi^k\nabla\mathbf{v}\right)\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\ &\quad + \Delta t^{k+1}\sigma_{st}\int_\Lambda \left(P_\varphi^k\nabla\mathbf{u}^{k+1} : P_\varphi^k\nabla\mathbf{v}\right)\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}.\end{aligned}$$

Thus the surface tension force, $h_\varphi^{k+1} = h\left(\mathbf{u}^{k+1},\Gamma^k\right)$ is dependent on the velocity $\mathbf{u}^{k+1}(\varphi^k)$ based on previous levelset function φ^k and evaluated around $\Gamma^k = \{\mathbf{x}\mid\varphi^k(\mathbf{x}) = 0\}$. It is a first order approximation to surface tension, but results in a linear addition to the existing two-phase flow algorithm. In addition, there are no extra components needed since curvature does not need to be explicitly constructed. This only requires the dim components of velocity and the single component of levelset and so is preferable to the previous algorithm in terms of memory and work and thus time. However, if the total (mean) curvature is necessary then the above algorithm is better since it calculates total curvature in a semi-implicit form, which results in a more smooth calculation of curvature than an explicit algorithm directly from $h_\varphi^k = \mathrm{div}(\mathbf{n}_\varphi^k)$ is able do.

6.3 Numerical experiments and validations

We will give some of the classic surface tension benchmark tests as described by Hysing in [64]. They are both 2D simulations of a bubble rising under gravity and different amounts of surface tension. The physical setting is a domain $\Lambda = [0,1] \times [0,2]$ with a circular bubble of radius $r_0 = 0.25$ centered $(0.5, 0.5)$. The simulation is run from $t = 0.0$ until $t = 3.0$. The exterior liquid and interior liquid have the physical density and viscosity as in Table 6.1. The two cases have different surface tension parameters, σ and the

Case	ρ_{out}	ρ_{in}	μ_{out}	μ_{in}	g	σ	Re	Eo	$\rho_{\text{out}}/\rho_{\text{in}}$	$\mu_{\text{out}}/\mu_{\text{in}}$
1	1000	100	10	1	-0.98	24.5	35	10	10	10
2	1000	1	10	0.1	-0.98	1.65	35	125	1000	100

Table 6.1: Parameters for 2 test cases of surface tension.

two-phases move under gravity in direction $(0, g)$ where g is the non dimensional gravity constant. The characteristic length scale is $L = 2r_0 = 0.5$. Then the gravitational velocity is $U_g = \sqrt{|g|2r_0}$. The Reynolds number is thus computed using the heavier fluid,

$$Re = \frac{\rho_{\text{out}} U_g L}{\mu_{\text{out}}}, \tag{6.11}$$

describing the ratio of inertial to viscous forces is also displayed as well as the Eötvös number,

$$Eo = \frac{\rho_{\text{out}} U_g^2 L}{\sigma}, \tag{6.12}$$

giving the ratio of gravitational forces to surface tension. We impose the no-slip boundary condition $\mathbf{u} = 0$ on the top and bottom of the region's boundary. The free slip boundary, $\mathbf{u} \cdot \mathbf{n} = 0$, is imposed on the left and right boundaries. We compute the center of mass of

the bubble

$$\mathbf{X}_c = \frac{\int_\Lambda \mathbf{x} H_{\varepsilon(\varphi)d\mathbf{x}}}{\int_\Lambda 1 H_\varepsilon(\varphi) d\mathbf{x}}$$

using the approximate characteristic function $H_\varepsilon(\varphi)$ as in equation (4.11). We also compute the rise velocity,

$$u_c = \frac{\int_\Lambda \mathbf{u}_y H_{\varepsilon(\varphi)d\mathbf{x}}}{\int_\Lambda 1 H_\varepsilon(\varphi) d\mathbf{x}}$$

where $\mathbf{u}_y$ is the y-component of velocity, and the circularity (in 2D),

$$circ = \frac{2\pi\sqrt{\frac{\int_\Lambda H(\varphi)d\mathbf{x}}{\pi}}}{\int_\Gamma d\mathbf{x}} \approx \frac{2\pi\sqrt{\frac{\int_\Lambda H_\varepsilon(\varphi)d\mathbf{x}}{\pi}}}{\int_\Lambda \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}}$$

measuring the deviation from circular shape. This is measured calculating the ratio of the perimeter of a circle of equivalent volume to the interior over the actual perimeter of the interior. The circularity takes a value between 0 and 1 with the value of 1 meaning it is a circle. Of course, since we are using an approximate characteristic and Dirac function to compute the circularity, the value may exceed 1 somewhat.

Using semi-implicit version of coupling as in Section 6.1.2, we display some of these statistics for a sequence of mesh-sizes and timesteps in Figure 6.1. We also track the volume and surface area of the simulations over time for case 1 as displayed in Figure 6.2. We see that the smaller mesh size maintains the volume much better as is desired with the incompressible velocity. The time series of case 1 with $h = 0.003125$ and $\Delta t = 0.002$ is displayed in Figure 6.3.

In case 2, we display our final result which can be compared to many other benchmark simulations in [64]. They are qualitatively similar. Because we continually reinitialize the profile, it is more difficult for the domain to fracture and the dragged parts stay fairly smooth.

Finally we perform a simulation of case 1 in 3D and simulate from $t = 0.0$ until

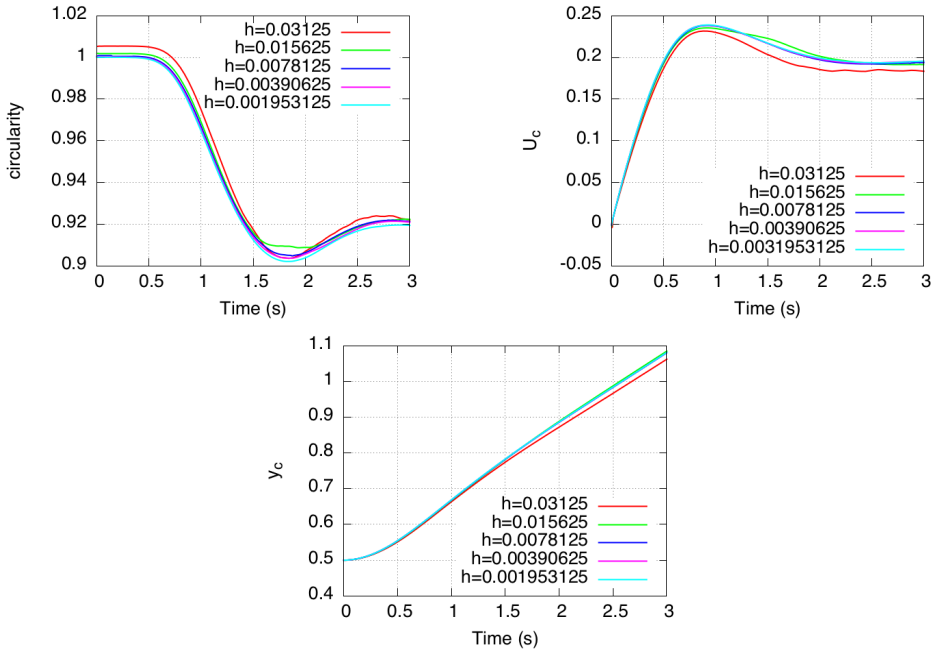


Figure 6.1: The circularity, rise velocity and center of mass in case 1.

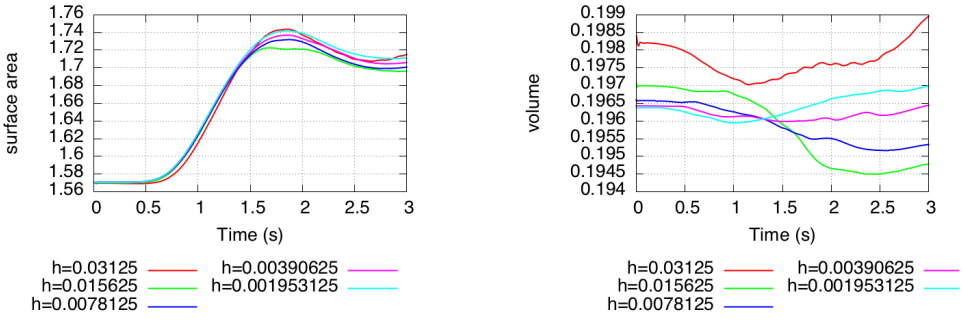


Figure 6.2: The surface area and volume of interior region in case 1.

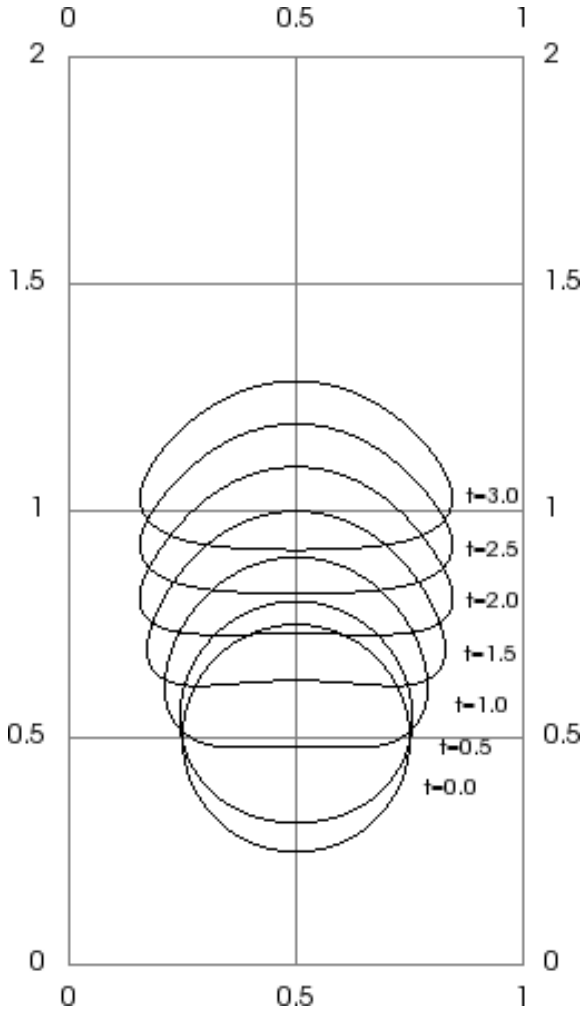


Figure 6.3: Case 1 timeseries.

$t = 2.0$ since in 3D, the bubble rises much faster than in 2D. The initial and final results are displayed in Figure 6.5. Notice that we obtain the same steady state profile of the bubble as in 2D.

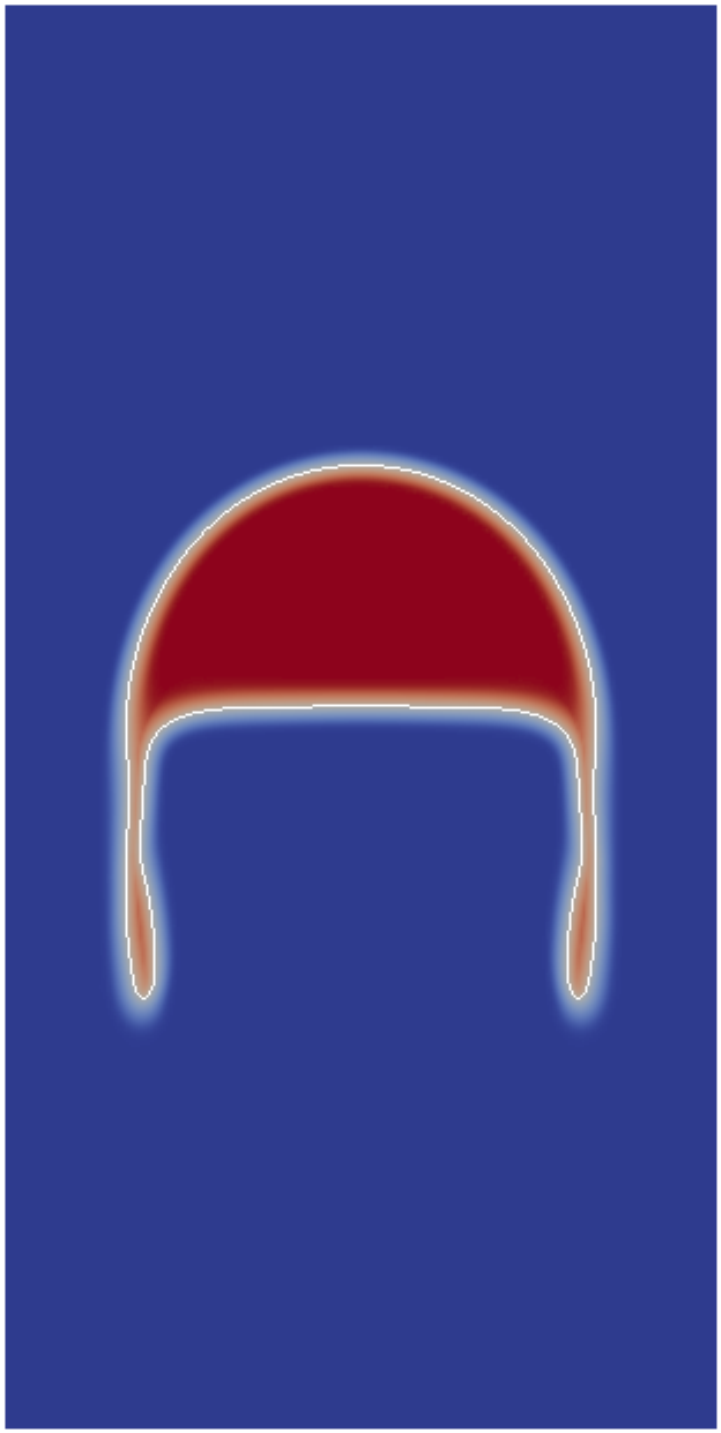


Figure 6.4: Final state of simulation with semi-implicit version for small surface tension coefficient, $\sigma_{st} = 1.96$. This solution compares well with the other benchmark solutions published in [64].

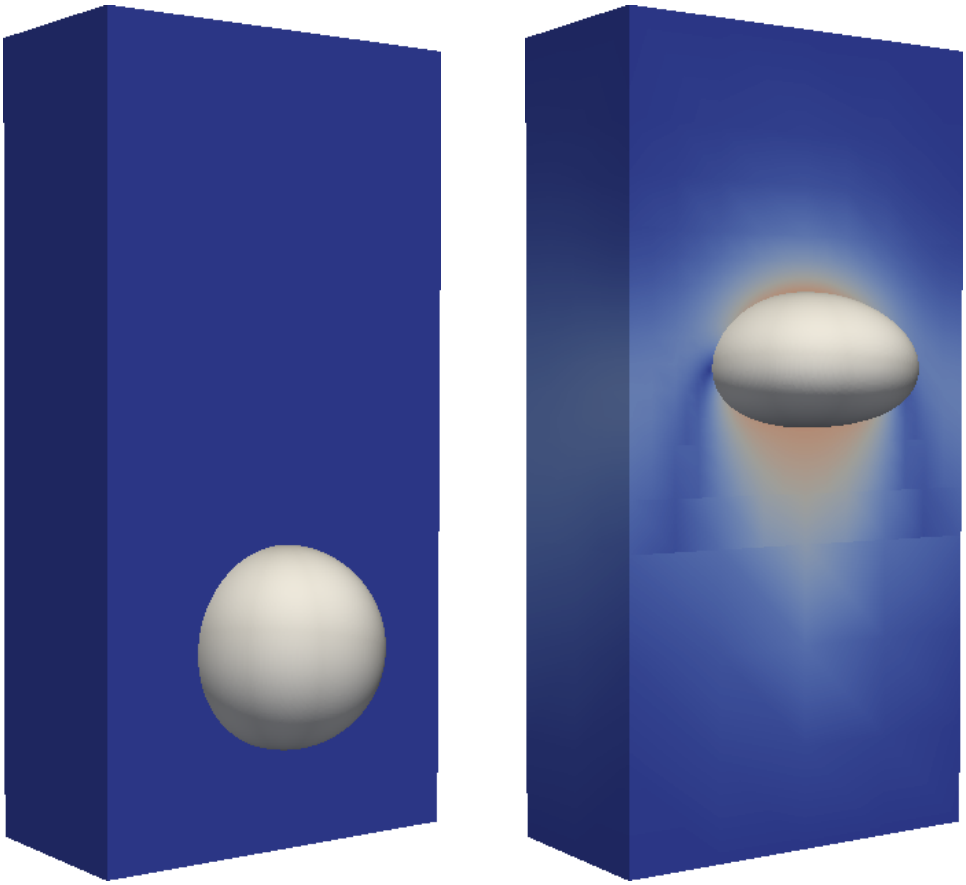


Figure 6.5: The initial $(t = 0.0)$ and final $(t = 2.0)$ states of simulation of case 1 in 3D with half the domain cut away to see the interior bubble.

7. ENERGY FLOW APPLICATION: WILLMORE FLOW

When considering the modelling of a red blood cell, or in general a biological vesicle with a lipid bilayer membrane, a dominant force governing the shape of the vesicle relates to minimizing the bending of the membrane. The bilayer membrane resists bending which combined with a surface area and volume constraint leads the red blood cells to have their characteristic pin cushion-like shape. Following [65], the bending energy can be modeled as the Willmore or curvature energy with κ_1 and κ_2, the principle curvatures of the surface, and k_1 and k_2 the bending or elastic moduli in each direction,

$$e_{\text{bending}}(\Gamma) = \int_\Gamma \frac{1}{2}k_1\kappa_1^2 + \frac{1}{2}k_2\kappa_2^2 d\mathbf{x}.$$

In an isotropic material these bending moduli are the same and will be denoted by k_e, the bending modulus. Now using $h = \kappa_1 + \kappa_2$, the total curvature and $\kappa = \kappa_1\kappa_2$, the Gauss curvature, we can simplify the bending energy to

$$e_{\text{bending}}(\Gamma) = \frac{1}{2}k_e \int_\Gamma h^2 d\mathbf{x} + k_e \int_\Gamma \kappa d\mathbf{x}.$$

Helfrich in [2] proposed a more general energy,

$$e_{\text{bending}}(\Gamma) = \frac{1}{2}k_e \int_\Gamma (h - c_0)^2 d\mathbf{x} + \bar{k} \int_\Gamma \kappa d\mathbf{x},$$

where c_0 is the spontaneous curvature or characteristic curvature of the surface, Γ. This may be non-zero if the lipid compositions of the membrane are different on the inside and outside. The moduli k_e and $\bar{k}$ are referred to as the "bending modulus" and the "saddle splay modulus" respectively. In the subsequent, we will assume that $c_0 = 0$.

When we consider a vesicle that maintains a constant topology, we can simplify this energy somewhat. The Gauss-Bonnet theorem from differential geometry tells us that the integral of Gauss curvature is directly related to the geometry of the surface, Γ, by

$$\int_{\Lambda} \kappa d\mathbf{x} = 2\pi\chi(\Gamma).$$

Where $\chi(\Gamma)$ is the Euler characteristic of the surface Γ which is constant for a fixed topology. Thus minimizing the simpler Willmore energy,

$$e_w(\Gamma) = \frac{1}{2}k_e \int_{\Gamma} h^2 d\mathbf{x},$$

is the same as minimizing the bending energy, $e_{\text{bending}}(\Gamma)$ when there are no topology changes.

7.1 Willmore energy

We will thus consider the coupling of the Willmore energy

$$e_w(\Gamma) = \frac{1}{2}k_e \int_{\Gamma} h^2 d\mathbf{x} \tag{7.1}$$

to the incompressible Navier-Stokes system which corresponds to using the integrand

$$G_w(\mathbf{x}, h) = \frac{1}{2}k_e h^2.$$

We follow the pattern of Chapter 5 and use the approximate Willmore energy

$$E_w(\varphi) = \frac{1}{2}k_e \int_{\Lambda} h_{\varphi}^2 \delta_{\varepsilon}(\varphi)|\nabla\varphi| d\mathbf{x}. \tag{7.2}$$

Using Theorem 5.2.1, with the computations $\partial_2 G_w = k_e h_\varphi$, $\nabla G_w = k_e h_\varphi \nabla h_\varphi$ and $\boldsymbol{\partial_1 G_w} = \mathbf{0}$, we have the following corollary.

Corollary 7.1.1. *Using the approximate Willmore energy* $E_w(\varphi) = \frac{1}{2}k_e \int_\Lambda h_\varphi^2 \delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x}$*, we have the energy variation for* $\theta \in H^1(\Lambda)$*,*

$$\partial E_w(\varphi)(\theta|\nabla\varphi|) = k_e \int_\Lambda \operatorname{div}\left[\frac{1}{2}h_\varphi^2 \boldsymbol{n}_\varphi - \frac{P_\varphi \nabla(h_\varphi|\nabla\varphi|)}{|\nabla\varphi|}\right] \theta\delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x} \tag{7.3}$$

$$= k_e \int_\Lambda \left[h_\varphi \nabla h_\varphi \cdot \boldsymbol{n}_\varphi \theta + \frac{1}{2}h_\varphi^3 \theta + \frac{P_\varphi \nabla(h_\varphi|\nabla\varphi|)}{|\nabla\varphi|} \cdot \frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right] \delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x}. \tag{7.4}$$

Using Lemma 5.1.4, we have

$$\partial E_w(\varphi)(\theta|\nabla\varphi|) = k_e \int_\Lambda \left[-\Delta_\varphi h_\varphi + \frac{1}{2}h_\varphi^3 - h_\varphi D\mathbf{n}_\varphi^T : D\mathbf{n}_\varphi\right] \theta\delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}.$$

Now assuming that $\varphi(\mathbf{x}) = f(d(\mathbf{x}))$ as in Chapter 2 and using $\theta = \mathbf{n}_\varphi \cdot \mathbf{v}$ where $\mathbf{v} \in [H^1(\Gamma)]^d$, this simplifies to

$$\partial E_w(\varphi)(\mathbf{v} \cdot \mathbf{n}_\varphi|\nabla\varphi|) = k_e \int_\Lambda \left[-\Delta_\varphi h_\varphi + \frac{1}{2}h_\varphi^3 - h_\varphi|\nabla_\varphi \mathbf{n}_\varphi|^2\right] \mathbf{n}_\varphi \cdot \mathbf{v}\delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}.$$

This compares nicely to the variation of the Willmore energy functional,

$$\partial e_w(\Gamma)(\mathbf{v}) = k_e \int_\Gamma \left[-\Delta_\Gamma h + \frac{1}{2}h^3 - h|\nabla_\Gamma \mathbf{n}|^2\right] \mathbf{n} \cdot \mathbf{v} d\mathbf{x}.$$

Thus using the convergence of $\delta_\varepsilon(\varphi)|\nabla\varphi| \to \delta_\Gamma$ we have the following Corollary.

Corollary 7.1.2. *Assuming that* Γ *is closed and smooth and implicitly defined by* $\varphi(\boldsymbol{x}) =$

$f(d(\boldsymbol{x}))$ *for* $f \in C^2(\mathbb{R})$ *with* $f(0) = 0$ *and* $f'(t) \geq 0$ *then*

$$\lim_{\varepsilon \to 0} \partial E_w(\varphi)(\boldsymbol{v} \cdot \boldsymbol{n}_\varphi |\nabla \varphi|) = \partial e_w(\Gamma)(\boldsymbol{v}).$$

When we couple this Willmore energy with the incompressible Navier-Stokes system, we are minimizing bending energy under a constant volume constraint. This is different than the standard Willmore energy flow which only minimizes the bending energy. Without the volume constraint, the bending energy will eventually make the manifold disappear however with the volume constraint, the minimal bending energy will occur when curvature is constant, that is when the manifold is a sphere in 3D or a circle in 2D. However since this force is governed by the action of the negative Gâteaux derivative which can be seen as a form of gradient descent, we can often get stuck in a local minimum. It is not clear that Willmore flow with volume constraint but without surface area constraint models anything of interest. The remainder of this chapter will consider how to discretize the variation term and couple it with the Navier-Stokes and level set equations. We will give two formulations in Section 7.2 and Section 7.3. Recall that the stability estimates of Section 5.3 hold for the implicit Willmore energy flow scheme described. We will then compare our approximations of the Willmore energy flow with some other existing models in Section 7.4. We finish with some numerical simulations of the Willmore energy flow with volume constraint in Section 7.5.

We finally emphasize again the stability of the continuous fully implicit Willmore flow algorithm with Theorem 7.1.3 which mirrors the general Theorem 5.3.2 of Chapter 5.

Theorem 7.1.3. *Testing the continuous Navier-Stokes system with approximate Willmore energy flow against the true solution,* $\boldsymbol{u}(t)$*, and with no external sources,* $\boldsymbol{f} = 0$*, yields the*

energy stability estimate,

$$\frac{d}{dt}\int_\Lambda \frac{1}{2}\rho(t)|\boldsymbol{u}(t)|^2 d\boldsymbol{x} + \int_\Lambda 2\mu(t)|\nabla^s \boldsymbol{u}(t)|^2 d\boldsymbol{x} + \frac{d}{dt}E_w(\varphi(t)) = 0. \tag{7.5}$$

7.2 Semi-implicit form for Willmore flow

We will let $g_w \in L^2(\Lambda)$ represent the negative variation of our energy in the sense that

$$k_e \int_\Lambda g_w \psi d\mathbf{x} = -\partial E_w(\varphi)(\psi) \tag{7.6}$$

where we will use $\psi = \theta|\nabla\varphi|$ with $\theta \in H^1(\Lambda)$. Thus,

$$\int_\Lambda g_w\theta|\nabla\varphi|d\mathbf{x} = -k_e\int_\Lambda \left[h_\varphi \nabla h_\varphi \cdot \mathbf{n}_\varphi \theta + \frac{1}{2}h_\varphi^3\theta + \frac{P_\varphi\nabla(h_\varphi|\nabla\varphi|)}{|\nabla\varphi|}\cdot\frac{\nabla(\theta|\nabla\varphi|)}{|\nabla\varphi|}\right]\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}$$

or expanding into it's various terms,

$$\begin{aligned}
\int_\Lambda g_w\theta|\nabla\varphi|d\mathbf{x} = &-k_e\int_\Lambda h_\varphi\theta\nabla h_\varphi\cdot\mathbf{n}_\varphi\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\
&- k_e\int_\Lambda \frac{1}{2}h_\varphi^3\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\
&- k_e\int_\Lambda P_\varphi\nabla h_\varphi\cdot\nabla\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\
&- k_e\int_\Lambda h_\varphi\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\cdot\nabla\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\
&- k_e\int_\Lambda \theta\nabla h_\varphi\cdot\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\
&- k_e\int_\Lambda h_\varphi\theta\left|\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\right|^2\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.
\end{aligned}$$

The important terms for the physics of motion are the first three terms since the last three terms contain action against $\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}$, the tangential derivative of $|\nabla\varphi|$ which we are pushing to zero with the level set reinitialization and only account for the deviations of the

level set function from the distance function.

Remark. We are spending resources in the levelset method pushing φ toward the exact profile with $|\nabla\varphi| = 1$ on the band of support of the Dirac delta function, $\delta_\varepsilon(\varphi)$. Because of this it is possible to drop the last three terms of the above system since then, $P\nabla(|\nabla\varphi|) = \mathbf{0}$. In practice we have noticed little to no difference between the model with and without these terms so many of the simulations have been run without them. Unless otherwise specified, we assume that the full model has been used for the simulation. *end Remark.*

This system coupled with the velocity assumes that $(\varphi, \mathbf{u}, h, g_w)$ are all implicitly coupled, but this becomes a nonlinear fourth order system which is not practical to solve. In Section 7.2.1, we introduce some simplifications which will allow us to decouple φ from the rest and make h implicitly defined through $\mathbf{u}$ instead of φ. Then in Section 7.2.2, we will introduce an operator splitting on g_w to obtain a linear system for $(\mathbf{u}, h, g_w)$ decoupled from φ. Finally, in Section 7.2.3, we will summarize the entire semi-implicit algorithm for Willmore flow.

7.2.1 Coupling velocity and curvature and decoupling curvature from the level set

We use superscripts to denote the time step and consider that we have solutions at time t^k and are solving for solutions of (φ^{k+1}, $\mathbf{u}^{k+1}$, h_φ^{k+1}, g_w^{k+1}) at time t^{k+1}. We will break this update into two steps:

(i) Given (φ^k, $\mathbf{u}^k$, h_φ^k), compute ($\mathbf{u}^{k+1}$, h_φ^{k+1}, g_w^{k+1}).

(ii) Given (φ^k, $\mathbf{u}^{k+1}$), compute φ^{k+1}.

We have defined $h_\varphi^{k+1} = \operatorname{div}\left(\mathbf{n}_\varphi^{k+1}\right) = \operatorname{div}\left(-\frac{\nabla\varphi^{k+1}}{|\nabla\varphi^{k+1}|}\right)$, which means curvature is coupled with the level set function, but we want to decouple it from the level set function and couple it instead with the velocity in order to keep our two step update as described above. As in Section 6.1.2, we decouple h_φ^{k+1} from φ^{k+1} and couple it directly to the velocity by

introducing the scalar function,

$$u_n^{k+1} = -\mathbf{u}^{k+1} \cdot \nabla \varphi^k,$$

the normal velocity. Recall that the surface normal $\mathbf{n}_\varphi^k = -\frac{\nabla \varphi^k}{|\nabla \varphi^k|}$ so that this is the normal velocity when $|\nabla \varphi^k| = 1$, in the region that matters. In weak form, we seek $u_n^{k+1} \in L^2(\Lambda)$ such that for all $\theta \in L^2(\Lambda)$

$$\int_\Lambda u_n^{k+1} \theta d\mathbf{x} = -\int_\Lambda \mathbf{u}^{k+1} \cdot \nabla \varphi^k \theta d\mathbf{x}.$$

Expanding the levelset equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = 0,$$

we have

$$\begin{aligned} \frac{\partial \varphi}{\partial t}\left(t^k\right) &= -\mathbf{u}\left(t^k\right) \cdot \nabla \varphi\left(t^k\right) \\ &= -\mathbf{u}\left(t^{k+1}\right) \cdot \nabla \varphi\left(t^k\right) + \Delta t \frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi\left(t^k\right) \end{aligned}$$

for some $\xi \in \left(t^k, t^{k+1}\right)$. Then discretizing the time derivative of φ with a forward difference, we obtain

$$\frac{\varphi^{k+1} - \varphi^k}{\Delta t} = -\mathbf{u}^{k+1} \cdot \nabla \varphi^k + \Delta t \frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi\left(t^k\right) + \frac{\Delta t}{2} \frac{\partial^2 \varphi}{\partial t^2}(\xi_2), \tag{7.7}$$

for some $\xi_2 \in [t^k, t^{k+1}]$. Thus, using the normal velocity, u_n^{k+1}, we have the first order update equation with Δt^2 error

$$\varphi^{k+1} = \left(\varphi^k + \Delta t u_n^{k+1}\right) + \Delta t^2 \left(\frac{\partial \mathbf{u}}{\partial t}(\xi) \cdot \nabla \varphi^k + \frac{1}{2}\frac{\partial^2 \varphi}{\partial t^2}(\xi_2)\right).$$

Now, we can replace φ^{k+1} with $\varphi^k + \Delta t u_n^{k+1}$ in some places to couple h_φ^{k+1} to $\mathbf{u}^{k+1}$ instead of φ^{k+1}. We substitute all the $|\nabla \varphi^{k+1}|$ terms with $|\nabla \varphi^k|$ evaluated at the previous time and do the same with the Dirac measure so that they are all defined on φ^k. We can thus write $\mathbf{n}_\varphi^{k+1} = -\frac{\nabla \varphi^{k+1}}{|\nabla \varphi^{k+1}|} \approx -\frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|}$. Then our semi-implicit scheme for curvature (using ν for the outer unit normal on $\partial\Lambda$) is

$$\begin{aligned}
\int_\Lambda h_\varphi^{k+1} \theta d\mathbf{x} &= \int_\Lambda \operatorname{div}(\mathbf{n}_\varphi^{k+1}) \theta d\mathbf{x} \\
&= -\int_\Lambda \mathbf{n}_\varphi^{k+1} \cdot \nabla \theta d\mathbf{x} + \int_{\partial\Lambda} \mathbf{n}_\varphi^{k+1} \cdot \nu \theta d\mathbf{x} \\
&= \int_\Lambda \frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} - \int_{\partial\Lambda} \frac{\nabla \varphi^k + \Delta t \nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x},
\end{aligned}$$

or separating the implicit and explicit terms we have

$$\begin{aligned}
\int_\Lambda h_\varphi^{k+1} \theta d\mathbf{x} - \Delta t \int_\Lambda \frac{\nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} &= \int_\Lambda \frac{\nabla \varphi^k}{|\nabla \varphi^k|} \cdot \nabla \theta d\mathbf{x} \\
+\Delta t \int_{\partial\Lambda} \frac{\nabla u_n^{k+1}}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x} \qquad &- \int_{\partial\Lambda} \frac{\nabla \varphi^k}{|\nabla \varphi^k|} \cdot \nu \theta d\mathbf{x}.
\end{aligned} \tag{7.8}$$

7.2.2 *A semi-implicit in time splitting for Willmore flow*

Again using superscripts to represent the time step, we would like to be able to calculate, for all $\theta \in H^1(\Lambda)$,

$$\begin{aligned}\int_\Lambda g_w^{k+1}\theta|\nabla\varphi^{k+1}|d\mathbf{x} = &-k_e\int_\Lambda h_\varphi^{k+1}\theta\nabla h_\varphi^{k+1}\cdot\mathbf{n}_\varphi^{k+1}\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}\\ &-k_e\int_\Lambda\frac{1}{2}(h_\varphi^{k+1})^3\theta\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}\\ &-k_e\int_\Lambda P[\varphi^{k+1}]\nabla h_\varphi^{k+1}\cdot\nabla\theta\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}\\ &-k_e\int_\Lambda h_\varphi^{k+1}\frac{P[\varphi^{k+1}]\nabla(|\nabla\varphi^{k+1}|)}{|\nabla\varphi^{k+1}|}\cdot\nabla\theta\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}\\ &-k_e\int_\Lambda\theta\nabla h_\varphi^{k+1}\cdot\frac{P[\varphi^{k+1}]\nabla(|\nabla\varphi^{k+1}|)}{|\nabla\varphi^{k+1}|}\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}\\ &-k_e\int_\Lambda h_\varphi^{k+1}\theta\left|\frac{P[\varphi^{k+1}]\nabla(|\nabla\varphi^{k+1}|)}{|\nabla\varphi^{k+1}|}\right|^2\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|d\mathbf{x}.\end{aligned}$$

but this is a highly non-linear system implicit in velocity and in the level set equation. We will modify the terms as in curvature to obtain a linear first-order splitting in time. Notice that by Lemma 5.1.3, $\mathbf{n}_\varphi\cdot\nabla h_\varphi$ tends to $-|\nabla_\varphi\mathbf{n}_\varphi|^2 = 0$ as $\mathbf{n}_\varphi \to -\nabla d$ which we are enforcing by our level set reinitialization scheme likewise $P_\varphi\nabla(|\nabla\varphi|)$ tends to $\mathbf{0}$ under the same conditions so that the first and last three terms are being pushed to 0. Thus the important terms are the second and third terms which we will keep implicit. The rest will be treated fully explicitly. In order to split the second term and keep it linear, we choose to keep one of the h_φ^{k+1} implicit and the rest will be explicit. The Dirac delta terms will all be set to explicit as well as the $|\nabla\varphi|$ terms. To handle the third term, we introduce a splitting

so that we have a full gradient on the implicit term:

$$
\begin{aligned}
-P\left[\varphi^k\right]\nabla h_\varphi^{k+1}\cdot\nabla\theta &= -\nabla h_\varphi^{k+1}\cdot\nabla\theta + \left(I - P\left[\varphi^k\right]\right)\nabla h_\varphi^{k+1}\cdot\nabla\theta \\
&\approx -\nabla h_\varphi^{k+1}\cdot\nabla\theta + \left(I - P\left[\varphi^k\right]\right)\nabla h_\varphi^{k}\cdot\nabla\theta.
\end{aligned}
$$

Thus our final system for g_w^{k+1} with implicit terms on the left and explicit on the right is

$$
\begin{aligned}
\int_\Lambda g_w^{k+1}\theta|\nabla\varphi^k|d\mathbf{x} \quad &= \quad -k_e\int_\Lambda h_\varphi^k\theta\nabla h_\varphi^k\cdot\mathbf{n}_\varphi^k\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \\
+k_e\int_\Lambda \nabla h_\varphi^{k+1}\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \quad &\quad + k_e\int_\Lambda (I - P[\varphi^k])\nabla h_\varphi^k\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \\
+k_e\int_\Lambda \frac{1}{2}h_\varphi^{k+1}(h_\varphi^k)^2\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \quad &\quad - k_e\int_\Lambda h_\varphi^k\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \\
&\quad - k_e\int_\Lambda \theta\nabla h_\varphi^k\cdot\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \\
&\quad - k_e\int_\Lambda h_\varphi^k\theta\left|\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\right|^2\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}.
\end{aligned}
$$

7.2.3 Summary of semi-implicit Willmore system

Now that we have all the equations developed, we summarize the entire system. Given $(\varphi^k, \mathbf{u}^k, h_\varphi^k)$ compute $(\mathbf{u}^{k+1}, u_n^{k+1}, h_\varphi^{k+1}, g_w^{k+1})$ so that for all $\theta \in H^1(\Lambda)$ and $\mathbf{v} \in \left(H^1(\Lambda)\right)^d$, we have $\mathbf{u}^{k+1}$ satisfying the Navier-Stokes system of Chapter 4 with the force balance term

$$
-\int_\Gamma [\![2\mu\nabla^S\mathbf{u}^{k+1} - pI]\!]\cdot\mathbf{n}^k\cdot\mathbf{v}d\mathbf{x} = -\int_\Lambda g_w^{k+1}\mathbf{n}_\varphi^k\cdot\mathbf{v}|\nabla\varphi^k|d\mathbf{x} = \int_\Lambda g_w^{k+1}\nabla\varphi^k\cdot\mathbf{v}d\mathbf{x} \quad (7.9)
$$

coupled with the following equations:

$$
\int_\Lambda u_n^{k+1}\theta d\mathbf{x} + \int_\Lambda \mathbf{u}^{k+1}\cdot\nabla\varphi^k\theta d\mathbf{x} \quad = \quad 0 \qquad (7.10)
$$

$$\int_\Lambda h_\varphi^{k+1}\theta d\mathbf{x} - \Delta t \int_\Lambda \frac{\nabla u_n^{k+1}}{|\nabla\varphi^k|}\cdot\nabla\theta d\mathbf{x} = \int_\Lambda \frac{\nabla\varphi^k}{|\nabla\varphi^k|}\cdot\nabla\theta d\mathbf{x} \qquad (7.11)$$
$$+\Delta t\int_{\partial\Lambda}\frac{\nabla u_n^{k+1}}{|\nabla\varphi^k|}\cdot\nu\theta d\mathbf{x} \qquad -\int_{\partial\Lambda}\frac{\nabla\varphi^k}{|\nabla\varphi^k|}\cdot\nu\theta d\mathbf{x}$$

and

$$\begin{aligned}
\int_\Lambda g_w^{k+1}\theta|\nabla\varphi^k|d\mathbf{x} \quad &= \quad -k_e\int_\Lambda h_\varphi^k\theta\nabla h_\varphi^k\cdot\mathbf{n}_\varphi^k\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\
+k_e\int_\Lambda \nabla h_\varphi^{k+1}\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \quad &+ k_e\int_\Lambda (I-P[\varphi^k])\nabla h_\varphi^k\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\
+k_e\int_\Lambda \frac{1}{2}h_\varphi^{k+1}(h_\varphi^k)^2\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x} \quad &- k_e\int_\Lambda h_\varphi^k\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\cdot\nabla\theta\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\
&- k_e\int_\Lambda \theta\nabla h_\varphi^k\cdot\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}\\
&- k_e\int_\Lambda h_\varphi^k\theta\left|\frac{P[\varphi^k]\nabla(|\nabla\varphi^k|)}{|\nabla\varphi^k|}\right|^2\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|d\mathbf{x}.
\end{aligned} \qquad (7.12)$$

These are all solved simultaneously which provides a velocity with which to advect the levelset. There are $d+3$ components which gives a 4x4-block linear algebra system. Using the notation of Section 7.2.4 for the discrete solutions this can be written in matrix form as

$$\begin{bmatrix} \mathbf{S}_{ns} & \mathbf{0} & \mathbf{0} & \mathbf{M}_{\nabla\varphi} \\ \mathbf{M}_{\nabla\varphi}^T & M & 0 & 0 \\ \mathbf{0} & -\Delta t S_{u_n} & M & 0 \\ \mathbf{0} & 0 & k_e S_h & M_{|\nabla\varphi|} \end{bmatrix} \begin{bmatrix} \mathbf{U}^{k+1} \\ U_n^{k+1} \\ H^{k+1} \\ G_w^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{ns} \\ 0 \\ F_h \\ k_e F_g \end{bmatrix} \qquad (7.13)$$

The matrix has a circular block structure dependence. The matrix $M_{\nabla\varphi}^T$ represents the mapping from velocity to normal velocity and in particular contains an inner product of velocity against $\nabla\varphi^k$. The matrix $M_{|\nabla\varphi|}$ is a mass matrix with the $|\nabla\varphi^k|$ weight. This weight tends to 0 outside the band of interest and so we will often add a 1 to the diagonal

element of the row for the degrees of freedom where $|\nabla\varphi^k|$ is too small so that discrete coercivity is maintained. This makes the full system invertible and does not effect the solution since the other terms on that row are integrated against $\delta_\varepsilon(\varphi^k)|\nabla\varphi^k|$ which is 0 in all of these cases where we perturb the mass matrix. In the support of $\delta_\varepsilon(\varphi^k)$ we are pushing the level set function to have $|\nabla\varphi^k| = 1$. We proceed with this strategy under the intuition that the mass matrix has exponential decay in dependence between degrees of freedom based on their distance apart and the term $|\nabla\varphi^k|$ is positive for a few more layers of cells beyond where the support of $\delta_\varepsilon(\varphi^k)$.

This matrix turns out to be rather difficult to solve numerically. The best method that we have found for 2D simulations has been to use a parallel direct solver such as SuperLU_DIST (see [66], [67] and [7]). This solves it in a few seconds but does require a large amount of memory.

Remark. It is well understood in the direct solver community of computer scientists that using a direct solver to invert a finite element approximation to the Laplacian operator on a well behaved mesh has time (measured in flops) and space (measured in numbers to be stored in memory) complexity as in Table 7.1. The complexity has been computed using the method of nested dissection introduced in [68] and which has been shown to have almost minimal complexity in [69]. In other words, even in the optimal case and using a

	2D	3D
Space(fill-in)	$\mathcal{O}(n\log(n))$	$\mathcal{O}(n^{4/3})$
Time (flops)	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^2)$

Table 7.1: Time and Space complexity of inverting an $n \times n$ matrix representing finite element approximation to Laplacian operator on a well behaved mesh with a direct solver using method of nested dissection.

completely distributed parallel sparse solver, the time required to solve even the smallest practical problems in 3D is on the order of hours per time step. For instance, if we reduce to $n = 40,000$ degrees of freedom per processor we require on the order of a minimum of 8 million flops in 2D but a minimum of 1.6 billion flops in 3D. This is either a very small problem in 3D or requires way more processors than are available to us. *end Remark.*

There is currently no known preconditioner to make this feasible for an iterative solver. The matrix is non symmetric so we must use an iterative solver like the General Minimum Residual (GMRes) solver. We have attempted many approaches for the preconditioner including a global Algebraic Multigrid (AMG) preconditioner an incomplete LU (ILU(k)) and incomplete LU with thresholding (ILUT(k)), a block diagonal preconditioner with various combinations of AMG and Symmetric Succesive Over Relaxation (SSOR), and a Schur complement approach breaking the lumping the last 3 blocks into 1 block since we have a good preconditioner for the Navier Stokes block. It is however fascinating that we can use a Gaussian elimination approach to reduce this block system to the following non-block system where we first solve for velocity

$$\begin{aligned}(S_{ns} + \Delta t k_e M_{\nabla\varphi} M^{-1} S_h M^{-1} S_{u_n} M^{-1} M_{\nabla\varphi}^T)\mathbf{U}^{k+1} \\ = F_{ns} - k_e M_{\nabla\varphi} M^{-1} F_g + k_e M_{\nabla\varphi} M^{-1} S_h M^{-1} F_h\end{aligned}$$

then update the other components

$$\begin{aligned}U_n^{k+1} &= -M^{-1} M_{\nabla\varphi}^T \mathbf{U}^{k+1} \\ H^{k+1} &= M^{-1} F_h - \Delta t M^{-1} S_{u_n} M^{-1} M_{\nabla\varphi}^T \mathbf{U}^{k+1} \\ G_w^{k+1} &= k_e M^{-1} F_g - k_e M^{-1} S_h M^{-1} F_h + \Delta t k_e M^{-1} S_h M^{-1} M_{\nabla\varphi}^T \mathbf{U}^{k+1}.\end{aligned}$$

This is not, however, a practical system to solve and it is not clear how to simplify it to

use as an approximate inverse. The Schur complement approach results in series of matrix applications similar to this. We did find that using the ILU(k) algorithm as a preconditioner with k large did eventually solve the system but it was too expensive in time and memory usage to be practical.

At the current state, this algorithm is only practical for solving the Willmore (and Canham-Helfrich) system in 2D.

7.2.4 Spatial discretization of semi-implicit Willmore flow

We will discretize the domain Λ into a mesh $\mathcal{T}_h$ with minimal mesh size $h_{\min}$ as defined in equation (2.17). Then we use the standard Taylor-Hood finite element space for our velocity-pressure solutions $(\mathbf{U}^{k+1}, P^{k+1}) \in \left(\mathbb{Q}^2\left(\mathcal{T}^{k+1}\right)\right)^d \times \mathbb{Q}^1\left(\mathcal{T}^{k+1}\right)$. We let our levelset, normal velocity, curvature, Willmore and test functions $\Phi^{k+1}, (U_n)^{k+1}$, $H^{k+1}, G_w^{k+1}, \Theta$ be in $\mathbb{Q}^2\left(\mathcal{T}^{k+1}\right)$ as well. Otherwise, everything is as is described in Section 7.2.3 where we have replaced $(\mathbf{u}^k, p^k, u_n^k, h^k, g_w^k, \varphi^k)$ with their discrete counterparts $(\mathbf{U}^k, P^k, (U_n)^k, H^k, G_w^k, \Phi^k)$ and likewise for solutions at timestep t^{k+1}.

7.3 Sub-iterating Willmore flow scheme

Since we have been unsuccessful at finding a method for solving the semi-implicit scheme in 3D in any reasonable time period, we will approach this problem from a new direction. The trouble is that the Willmore force has the dominant term $-\Delta_\Gamma h_\varphi$ where h_φ is the total curvature. Recall that $h_\varphi = \operatorname{div} \mathbf{n}_\varphi = -\operatorname{div}\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right)$ is a second derivative of the levelset, so that in the continuous form, we require the 4th derivative of the levelset and there are a number of terms that are nonlinear in h_φ. Thus using an explicit scheme is not going to be practical. However, in each timestep, if we introduce a sub-iteration scheme between the velocity and levelset using the current levelset to compute the energy flow terms, for small enough timestep the levelset and velocity will work themselves forward until they are both sitting at the new time value and it is as if we had an implicit scheme.

We terminate the sub-iteration step when the relative error in velocity is less than 1%. In the beginning of simulations when velocities are extremely large this can be around 20 or 30 substeps but quickly this tends to 3 or 4 substeps once the model has run for a bit. It is necessary to play with the time step size to find where this sub-iteration scheme is stable, but typically we have found it to be stable around minimal meshsize $h_{\min}$ to the $3/2$, that is $\Delta t \approx 1/3h_{\min}^{1.5}$. This is not unreasonable since our δ_ε has accuracy $h^{3/2}$ and so this is consistent with the previous semi-implicit model where we introduced a linear splitting in time and also chose $\Delta t \sim ch_{\min}^{3/2}$ to match temporal and spatial errors. The benefit of this system is that we no longer need the $dim+3$ components in the velocity system. We merely need to compute the extra right hand side term corresponding to the gradient flow term and have only the dim components of velocity. An AMG preconditioner with the GMRES iterative method solves this system efficiently. We will discuss the computation of the right hand side term in Section 7.3.1.1. The sub-iterating scheme is given in Algorithm 6

7.3.1 Sub-iterating scheme to approximate fully-implicit algorithm

The sub-iterating scheme computes the forces explicitly using the given levelset function. The key to why this can work is that by alternating between the velocity and the levelset inside each timestep as in Algorithm 6, both the velocity and levelset move forward until it is as if they are solving the implicitly coupled scheme at the next time value. Since the Willmore energy term is a fourth order nonlinear term, it is somewhat finicky to approximate explicitly. We discuss in Section 7.3.1.1 the method we use to calculate the force, g_w, from a given levelset, φ.

7.3.1.1 Explicit energy gradient flow right hand side

Given a levelset function, φ, we must compute the energy representative $g_w(\mathbf{x})$ defined over Λ that can be used to calculate the Willmore energy force to be coupled with the stress

Algorithm 6: Sub-iterating scheme for Navier-Stokes/ Level Set system with Willmore flow.

Data: $(\mathbf{u}^k, p^k, \varphi^k)$
Result: $(\mathbf{u}^{k+1}, p^{k+1}, \varphi^{k+1})$

begin
  $\mathbf{u}^{k+1,0} \leftarrow \mathbf{u}^k$;
  $\varphi^{k+1,0} \leftarrow \varphi^k$;
  for $n = 1, \ldots,$ *MAXSUBSTEPS* **do**
    refine_mesh$(\varphi^k, \varphi^{k+1,n-1})$;
    $F^{n-1}_{\text{energy}} \leftarrow$ compute_energy_rhs$(\varphi^{k+1,n-1})$;
    $\mathbf{u}^{k+1,n} \leftarrow$ compute_velocity $\left(\mathbf{u}^k, \varphi^{k+1,n-1}, F^{n-1}_{\text{energy}}\right)$;
    $\varphi^{k+1,n} \leftarrow$ compute_levelset $\left(\varphi^k, \mathbf{u}^{k+1,n}\right)$;
    if $\|\boldsymbol{U}^{k+1,n} - \boldsymbol{U}^{k+1,n-1}\|_{\ell^2} < 0.01\|\boldsymbol{U}^{k+1,n-1}\|_{\ell^2}$ **then**
      exit with convergence;
    end
  end
  if *exit with convergence* **then**
    $\mathbf{u}^{k+1} \leftarrow \mathbf{u}^{k+1,n}$;
    $p^{k+1} \leftarrow$ compute_pressure$(p^k, \mathbf{u}^{k+1})$;
    $\varphi^{k+1} \leftarrow \varphi^{k+1,n}$;
  else
    Error, no convergence. Increase MAXSUBSTEPS or decrease Δt.;
  end
end

jump along Γ from the Navier-Stokes system

$$\int_\Gamma [\![2\mu\nabla^s\mathbf{u} - pI]\!] \cdot \mathbf{n} \cdot \mathbf{v} d\mathbf{x} = \int_\Lambda g_w \mathbf{n} \cdot \mathbf{v} |\nabla\varphi| d\mathbf{x} \tag{7.14}$$

where $g \in L^2(\Lambda)$ is defined using φ, $\mathbf{n}_\varphi = -\frac{\nabla\varphi}{|\nabla\varphi|}$ and $h_\varphi = \mathrm{div}(\mathbf{n}_\varphi)$ as

$$\begin{aligned}\int_\Lambda g_w\theta|\nabla\varphi|d\mathbf{x} = &-k_e\int_\Lambda h_\varphi\theta\nabla h_\varphi\cdot\mathbf{n}_\varphi\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &-k_e\int_\Lambda\frac{1}{2}(h_\varphi)^3\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &-k_e\int_\Lambda P_\varphi\nabla h_\varphi\cdot\nabla\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &-k_e\int_\Lambda h_\varphi\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\cdot\nabla\theta\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &-k_e\int_\Lambda\theta\nabla h_\varphi\cdot\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}\\ &-k_e\int_\Lambda h_\varphi\theta\left|\frac{P_\varphi\nabla(|\nabla\varphi|)}{|\nabla\varphi|}\right|^2\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.\end{aligned}$$

We find that the curvature needs to be extremely smooth for g_w to be meaningful. Otherwise the velocity tends to blow up within a few sub-iterative steps. The standard weak form for curvature, $h_\varphi = \mathrm{div}(\mathbf{n}_\varphi)$, is to multiply by a test function $\theta \in H^1(\Lambda)$ and integrate by parts. Then we seek $h_\varphi \in H^1(\Lambda)$ such that for all $\theta \in H^1(\Lambda)$

$$\begin{aligned}\int_\Lambda h_\varphi\theta d\mathbf{x} &= -\int_\Lambda\mathbf{n}_\varphi\cdot\nabla\theta d\mathbf{x} + \int_{\partial\Lambda}\mathbf{n}_\varphi\cdot\nu\theta d\mathbf{x}\\ &= \int_\Lambda\frac{\nabla\varphi}{|\nabla\varphi|}\cdot\nabla\theta d\mathbf{x} - \int_{\partial\Lambda}\frac{\nabla\varphi}{|\nabla\varphi|}\cdot\nu\theta d\mathbf{x}.\end{aligned}$$

The gradient of curvature that results from this formulation when using Lagrange finite elements of degree 2, that is $h_\varphi, \varphi, \theta \in \mathbb{Q}^2(\mathcal{T})$, is quite oscillatory and the resulting g_w is not good. To resolve this, we add a smoothing term to this equation that scales by the mesh size, $h_{\min}$. Since the curvature should be smooth in the tangential direction and to avoid bleeding between the levelsets, we only apply smoothing in the tangential direction.

Thus we are solving for curvature that satisfies

$$h_\varphi - c\Delta_\varphi h_\varphi = \operatorname{div} n_\varphi$$

with $c = h_{\min}$, on the order of the meshsize. In weak form, we seek $h_\varphi \in H^1(\Lambda)$ such that for all $\theta \in H^1(\Lambda)$,

$$\int_\Lambda h_\varphi \theta d\mathbf{x} + c\int_\Lambda P_\varphi \nabla h_\varphi \cdot P_\varphi \nabla\theta |\nabla\varphi| d\mathbf{x} = = \int_\Lambda \frac{\nabla\varphi}{|\nabla\varphi|}\cdot\nabla\theta d\mathbf{x} - \int_{\partial\Lambda}\frac{\nabla\varphi}{|\nabla\varphi|}\cdot\nu\theta d\mathbf{x}.$$

Remark. This smoothing term adds a first order, $\mathcal{O}(h_{\min})$, consistency error to our curvature computation. Thus we don't get the full convergence rate of $h_{\min}^{3/2}$ that is hoped for when using the $\delta_\varepsilon(\varphi)$ as in Theorem 3.3.1.

end Remark.

Remark. When we compute the curvature $h = \operatorname{div}\mathbf{n}$ using the weak formulation and finite elements, what order convergence should we expect? Suppose that $\varphi_h \in \mathbb{Q}^n(\mathcal{T}_h)$ is a finite element approximation of the levelset φ on a regular family of meshes $\{\mathcal{T}_h\}_{h>0}$, with meshsize h, with $\|\varphi - \varphi_h\|_{L^2(\Lambda)} + h\|\nabla\varphi - \nabla\varphi_h\|_{L^2(\Lambda)} \le ch^{n+1}\|\varphi\|_{H^{k+1}(\Lambda)}$. Then a heuristic to calculate the expected convergence rate of the finite element curvature $H_h = \operatorname{div}(\mathbf{n}_h) \in \mathbb{Q}^k(\mathcal{T}_h)$ to the true curvature $H = \operatorname{div}(\mathbf{n})$ goes as follows. (Note that we have used a capital letter for curvature to distinguish from the meshsize, h.)

$$\begin{aligned}
\|H - H_h\|_{H^{-1}(\Lambda)} &:= \sup_{\theta\in H_0^1(\Lambda)} \frac{|\langle H - H_h, \theta\rangle_{H^{-1}(\Lambda)-H^1(\Lambda)}|}{\|\theta\|_{H_0^1(\Lambda)}} \\
&= \sup_{\theta\in H^1(\Lambda)} \frac{|\langle \mathbf{n} - \mathbf{n}_h, \nabla\theta\rangle_{L^2(\Lambda)-L^2(\Lambda)}|}{\|\theta\|_{H_0^1(\Lambda)}} \\
&\le C\|\mathbf{n} - \mathbf{n}_h\|_{L^2(\Lambda)}
\end{aligned}$$

Using that $\mathbf{n}_h = -\frac{\nabla\varphi_h}{|\nabla\varphi_h|}$, we can also expect $||\nabla\varphi_h| - 1| \le c_1 h^n$ and $\|\mathbf{n} - \mathbf{n}_h\|_{L^2(\Lambda)} \le c_2 h^n$

with c_1, c_2 constants independent of mesh size. Thus,

$$\|H - H_h\|_{H^{-1}(\Lambda)} \leq ch^n$$

and so we can expect

$$\|H - H_h\|_{L^2(\Lambda)} \leq ch^{n-1}.$$

So if we use $\varphi_h \in \mathbb{Q}^2(\mathcal{T}_h)$ then we expect curvature to exhibit linear convergence in the L^2-norm. *end Remark.*

7.4 Comparison to other models

We have already discussed the similarities between our model and the method of Maitre, [59], using the method of virtual power in Section 5.4. There are a couple other models of Willmore flow worthy of noting as well.

7.4.1 A continuum surface forces approach

We have shown in Section 5.3 that the implicit scheme has energy stability. We will now show that our model can be equated to another approach (see [70] by Laadhari, et al) related to the use of continuum surface force technique of directly extending the force balance equation to a narrow band against an approximate Dirac measure to calculate the Willmore forces balance. They use a sub-iterating scheme between the levelset and velocity computation with the Willmore force right hand side and surface area constraints computed explicitly from the levelset. This is similar in fashion to our sub-iterating scheme of Section 7.3 but is not formulated using an energy flow but merely using the continuum surface force approach to derive the model. As will be seen, if our levelset is such that $\frac{\nabla\varphi}{|\nabla\varphi|} = \nabla d$ for d, a true distance function, then the two models are equivalent, thus

providing this other approach with a sense of stability per Theorem 7.1.3. Likewise, this calculation shows that the variation of our approximate energy does in fact limit to the variation of the exact energy functional.

Theorem 7.4.1. *Let Γ be closed, $\Gamma \cap \partial\Lambda = \emptyset$, $\theta \in H^1(\Lambda)$ and $\varphi = f(d(\boldsymbol{x}))$ so that $\frac{\nabla\varphi}{|\nabla\varphi|} = \nabla d$ and $\nabla h_\varphi \cdot \boldsymbol{n}_\varphi = -|P_\varphi \nabla \boldsymbol{n}_\varphi|^2$. Then using convergence of $\delta_\varepsilon(\varphi)|\nabla\varphi|$ to δ_Γ,*

$$\begin{aligned}
\partial E_w(\varphi)(\theta|\nabla\varphi|) &= k_e \int_\Lambda \left(-h_\varphi |P_\varphi \nabla \boldsymbol{n}_\varphi|^2 \theta + \frac{1}{2} h_\varphi^3 \theta + P_\varphi \nabla h_\varphi \cdot \nabla\theta \right) \delta_\varepsilon(\varphi)|\nabla\varphi| d\boldsymbol{x} \\
&\overset{\varepsilon-\lim}{=} k_e \int_\Gamma -h|\nabla_\Gamma \boldsymbol{n}|^2\theta + \frac{1}{2}h^3\theta + \nabla_\Gamma h \cdot \nabla_\Gamma \theta d\boldsymbol{x} \\
&= \partial e_w(\Gamma)\left(\theta \boldsymbol{n}\right).
\end{aligned}$$

Proof. Using Theorem 5.2.1 with $G = \frac{1}{2}k_e h_\varphi^2$, we have

$$\begin{aligned}
\partial E_w(\varphi)(\theta|\nabla\varphi|) &= k_e \int_\Lambda \left(P_\varphi \nabla(h_\varphi|\nabla\varphi|) \cdot \nabla(\theta|\nabla\varphi|) + \frac{1}{2} h_\varphi^3 \theta + h_\varphi \nabla h_\varphi \cdot \mathbf{n}_\varphi \theta \right) \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} \\
&= k_e \int_\Lambda \left(-\operatorname{div}(P_\varphi \nabla h_\varphi) + \frac{1}{2} h_\varphi^3 + h_\varphi \nabla h_\varphi \cdot \mathbf{n}_\varphi \right) \theta \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} \\
&= k_e \int_\Lambda \left(-\Delta_\varphi h_\varphi + \frac{1}{2} h_\varphi^3 + h_\varphi \nabla h_\varphi \cdot \mathbf{n}_\varphi \right) \theta \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}
\end{aligned}$$

As discussed in Lemma 5.1.3, $\nabla h_\varphi \cdot \mathbf{n}_\varphi$ does not in general reduce to something recognizable and useful, but in the case that $\frac{\nabla\varphi}{|\nabla\varphi|} = \nabla d$ (for instance when the level set function is a filtered distance function, $\varphi(\mathbf{x}) = f(d(\mathbf{x}))$) then we do have

$$\nabla h_\varphi \cdot \mathbf{n}_\varphi = -|P_\varphi \nabla \mathbf{n}_\varphi|^2 = -|\nabla_\varphi \mathbf{n}_\varphi|^2.$$

Thus we obtain in this case

$$\begin{aligned}
\partial E_w(\varphi)(\theta|\nabla\varphi|) &= k_e \int_\Lambda \left(-\Delta_\varphi h_\varphi + \frac{1}{2}h_\varphi^3 + h_\varphi\left(-|\nabla_\varphi \mathbf{n}_\varphi|^2\right)\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} \\
&= k_e \int_\Lambda \left(-\Delta_\varphi h_\varphi + \frac{1}{2}h_\varphi^3 - h_\varphi|\nabla_\varphi \mathbf{n}_\varphi|^2\right)\theta\delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} \\
&\overset{\varepsilon\to 0}{=} k_e \int_\Gamma \left(-\Delta_\Gamma h + \frac{1}{2}h^3 - h|\nabla_\Gamma \mathbf{n}|^2\right)\theta d\mathbf{x} \\
&= \partial e_w(\Gamma)(\theta\mathbf{n}).
\end{aligned}$$

So when we have a level set function, $\varphi(\mathbf{x}) = f(d(\mathbf{x}))$, the variation of our energy function $E_w(\varphi)$ in the direction $\theta|\nabla\varphi|$ is a regularization of the variation of $e_w(\Gamma)$ in the direction $\theta\mathbf{n}$.

□

7.4.2 A level set formulation for the Willmore flow

In [71], Droske and Rumpf present a nice formulation of Willmore energy flow with the level set function, minimizing the energy functional,

$$E[\varphi] = \int_\Lambda h_\varphi^2 |\nabla\varphi| d\mathbf{x}.$$

This paper was some of the inspiration to look at energy flows using a smeared energy functional. In addition, our weak form splitting for the higher order term, $\int_\Lambda P_\varphi \nabla h_\varphi \cdot \nabla\theta\delta_\varepsilon(\varphi) d\mathbf{x}$, representing the Laplace-Beltrami of curvature was inspired by their approach, where $\theta \in H^1(\Lambda)$ is a test function. Their model is purely an L^2- gradient flow algorithm under a special metric that is derived using the co-area formula. The model equation for Willmore flow of the level set function they present is

$$\int_\Lambda \frac{\varphi_t\theta}{|\nabla\varphi|} d\mathbf{x} = \int_\Lambda \left(-\frac{1}{2}\frac{(|\nabla\varphi|h_\varphi)^2}{|\nabla\varphi|^3}\nabla\varphi\cdot\nabla\theta + \frac{P_\varphi\nabla(|\nabla\varphi|h_\varphi)\cdot\nabla\theta}{|\nabla\varphi|}\right) d\mathbf{x}. \tag{7.15}$$

There are some similarities in the terms we obtain, but we couple the resulting forces to the incompressible Navier-Stokes system to produce a velocity field where as they evolve the levelset directly.

7.5 Numerical experiments and validations

We implement a few classic Willmore flow problems in 2D and reserve the 3D simulations for the Canham-Helfrich model. Recall that when the forces from the Willmore flow are coupled with the incompressible Navier-Stokes fluid we have a constant volume constraint implicitly being imposed. So we have here a Willmore flow with volume constraint. It has been shown in [72] that the closed surface of genus 0 in 3 dimensions (ie, topologically similar to a sphere) with minimal bending energy is the sphere. Furthermore, the minimal bending energy of any sphere is always $\frac{1}{2}\int_\Gamma h^2 d\mathbf{x} = 8\pi$, regardless of the radius. *Remark.* In this dissertation, we have used the convention that h is the total curvature defined as the sum of the principal curvatures, κ_1 and κ_2 at each point on the surface Γ. Then our bending energy is defined as $\frac{1}{2}\int_\Gamma h^2 d\mathbf{x}$. Sometimes the mean curvature, defined as the average of the principle curvatures, $H = \frac{\kappa_1+\kappa_2}{2}$, is used instead, and the minimal bending energy is reported as $\frac{1}{2}\int_\Gamma H^2 d\mathbf{x} = \frac{1}{2}\int_\Gamma \left(\frac{\kappa_1+\kappa_2}{2}\right)^2 d\mathbf{x} = 2\pi$. *end Remark.*

In two dimensions, a circle of radius r has constant curvature, $\kappa = \frac{1}{r}$ so that the bending energy is

$$\frac{1}{2}\int_\Gamma h^2 d\mathbf{x} = \frac{1}{2}(2\pi r)\left(\frac{1}{r}\right)^2 = \frac{\pi}{r}. \tag{7.16}$$

We give three examples of evolutions from an ellipse shaped initial interface with different aspect ratios between the two axes. The first is an ellipse where the x axis edge is 2 times as long as the y axis edge, which we will refer to as a 2×1 ellipse. The second is an initial 4×1 ellipse with one axis 4 times longer than the other. The third is an 8×1 ellipse with the x-axis 8 times longer than the y-axis. We evolve the Willmore flow system without external source until steady state. We note that the magnitude of velocity is typically

largest in the beginning when the greatest changes in shape seem to be taking place. We track the kinetic energy, $\int_\Lambda \rho|\mathbf{u}|^2 d\mathbf{x}$, the bending energy $\int_\Lambda \frac{1}{2}h_\varphi^2 \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x}$, the diffusive energy, $\int_\Lambda 2\mu|\nabla^s \mathbf{u}|^2 d\mathbf{x}$, along with their sum, the total energy.

Due to the stability result of Theorem 7.1.3 we expect the total energy to be decreasing overtime, however there are a few caveats. First the stability result holds for the continuous fully implicit algorithm. Given a spatial discretization of the Navier-Stokes with Willmore flow system, it isn't too had to see that the energy estimate holds for the implicit semi-discrete model. Our semi-implicit model of Section 7.2 doesn't necessarily have this estimate and the subiterating scheme of Section 7.3 approximates the fully implicit algorithm but we don't yet have a stability estimate for the temporal discretization. Finally, the stability estimate is best understood if we drop the diffusive energy term. Then the sum of kinetic and Willmore energy are non-increasing, that is

$$\int_\Lambda \rho(t)|\mathbf{u}(t)|^2 d\mathbf{x} + E_w(\varphi(t)) \geq \int_\Lambda \rho(s)|\mathbf{u}(s)|^2 d\mathbf{x} + E_w(\varphi(s)), \quad \text{for } t \geq s. \tag{7.17}$$

We will see in the subsequent simulations and then similarly the simulations of Section 8.3 that the total energy does tend to decrease over time but there are times for instance in Figure 7.1 where after a period, the diffusive energy increases for a time but then settles down again. This is still consistent with the stability estimate. An interesting symptom of the pressure correction splitting of the incompressible Navier-Stokes system as described in Chapter 4 is that the iniial conditions of the discretized system often do not perfectly satisfy the governing equations and it takes a few steps for it to compensate and reach equilibrium. Thus we often see some bobbles in the solutions at the beginning. We can compensate by starting everything at 0, but then it might be prudent to ramp up the bending modulus from 0 as well. Regardless, this can lead to bobbles in the energies at the beginning as well. Other than these two caveats, we observe that the equation 7.17 holds

for our discretizations as well. We will report for each simulation, the total energy and the partial total energy consisting of the kinetic and interfacial energies only.

We also track the surface area (perimeter) of the interface and the volume of the interior calculated using the approximate Dirac delta function, $\delta_\varepsilon(\varphi)$ of Chapter 3 and the related approximate Heaviside (characteristic) function of equation (4.11).

We have observed that the volume of interior region is not perfectly conserved in our system so we provide the same simulations where we explicitly enforce the volume constraint. We will introduce in Chapter 8 the Newton-like approach of enforcing the surface area and/or volume constraint explicitly on each time step. As will be seen, this method does well at enforcing the volume constraints and we will show, for example in Figure 7.15, that the steady state shapes are different when the volume constraint is not enforced. In fact, since the system continually loses mass over time, we never reach a steady state when the volume constraint is not being enforced. Thus it is imperative that the volume be conserved by some manner to find accurate final shapes.

The following computations are computed using the semi-implicit Willmore scheme described in Section 7.2. The linear algebra system is solved using the direct parallel solver SuperLU_dist [7] version 5.1.2 accessed through the Amesos wrapper class of Trilinos [6] version 12.10.1. The finite element system is constructed using the deal.II library [3] version 8.5.0 with p4est [5] version 1.1 for handling the completely distributed mesh. Everything is programmed in the C++ language. We use MPI for communicating between the distributed processors. The number of processors is chosen to keep the maximum number of degrees of freedom per processor between 40,000 and 120,000. By doing this, we balance solvability of the linear algebra system and memory footprint per processor with the overhead of passing data between the processors.

7.5.1 *The* 2×1 *ellipse in 2D*

We let $\Lambda = [-2, 2] \times [-1.5, 1.5]$ and subdivide our initial mesh with 4 times in the x direction and 3 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the approximate distance function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{2}\right)^2 + \left(\frac{y}{1}\right)^2}. \tag{7.18}$$

We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a timestep of $\Delta t = 0.002$ which is on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the semi-implicit Willmore scheme described in Section 7.2 for this computation with 2 MPI processors.

The bending energy (using equation (7.16)) of a circle with equivalent volume, that is with radius $r = \frac{1}{2}\sqrt{2}$ is $\frac{\pi}{r} = \sqrt{2}\pi \approx 4.442882937$.

We plot the energy, surface area and volume over time in Figure 7.1. The initial and final shapes plotted using the characteristic function are given in Figure 7.3. The surface has evolved to the shape of a circle. Since we can clearly see in Figure 7.1 that the capsule is losing mass (volume) over time (mainly due to the reinitialization algorithm of the level

set equation of Chapter 2.2), we also include the Willmore flow with an explicit volume constraint being enforced. We will introduce the method for doing this in Chapter 8. We enforce the volume at any given time step to within 10^{-6} of the initial computed volume. The statistics for the evolution of the 2×1 ellipse with explicit volume constraint are given in Figure 7.2 and the shape at times $t = 0$ and $t = 4$ are given in Figure 7.4. Notice that it has found steady state in the shape of a perfect circle, the shape with minimal bending energy.

7.5.2 The 4×1 ellipse in 2D

We let $\Lambda = [-4, 4] \times [-2, 2]$ and subdivide our initial mesh with 8 times in the x direction and 4 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set filter parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{4}\right)^2 + \left(\frac{y}{1}\right)^2} \tag{7.19}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a timestep of $\Delta t = 0.002$ which is on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step with $R = 1$ steps of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing reinitialization is $c_L = 0.1$.

We use the semi-implicit Willmore scheme described in Section 7.2 for this computa-

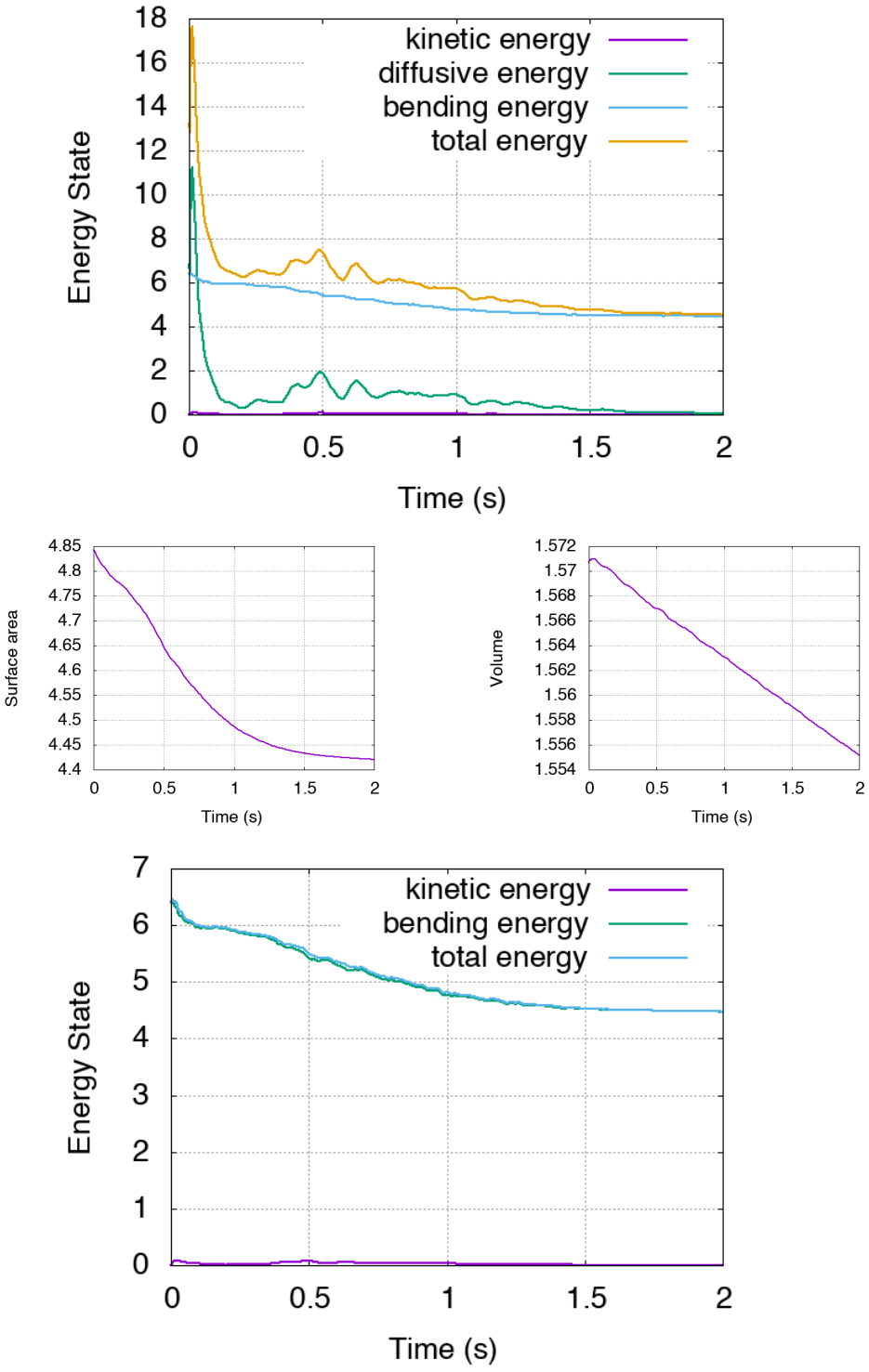


Figure 7.1: The energy, surface area and volume statistics for the 2x1 ellipse with Willmore force.

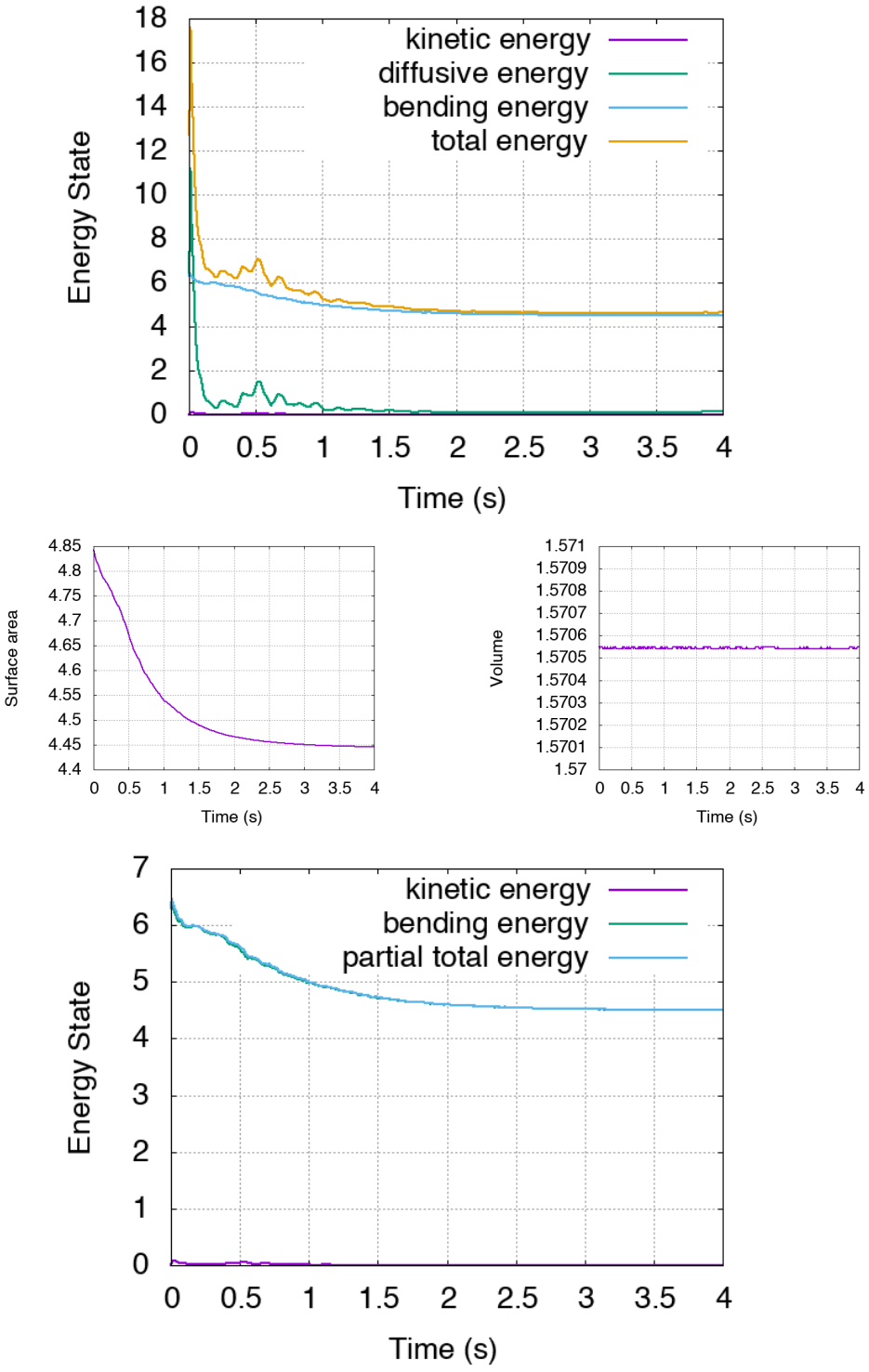


Figure 7.2: The energy, surface area and volume statistics for the 2x1 ellipse with Willmore force and explicit volume constraint.

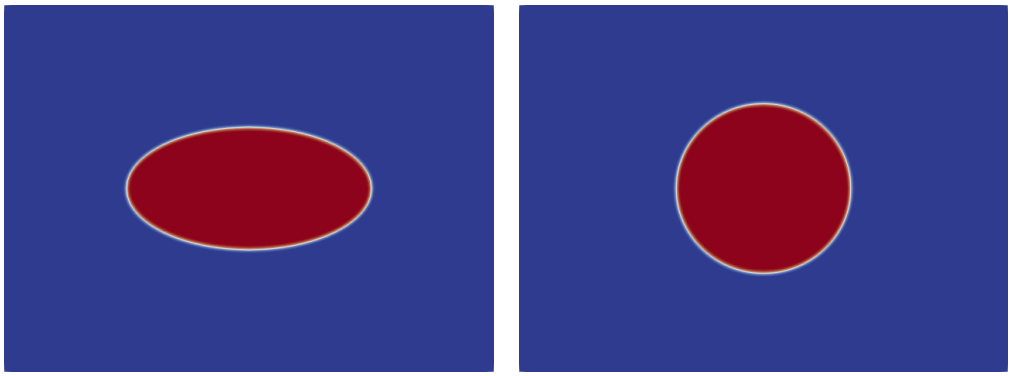


Figure 7.3: The initial and final shapes of 2×1 ellipse with Willmore flow at $t = 0$ and $t = 2$.

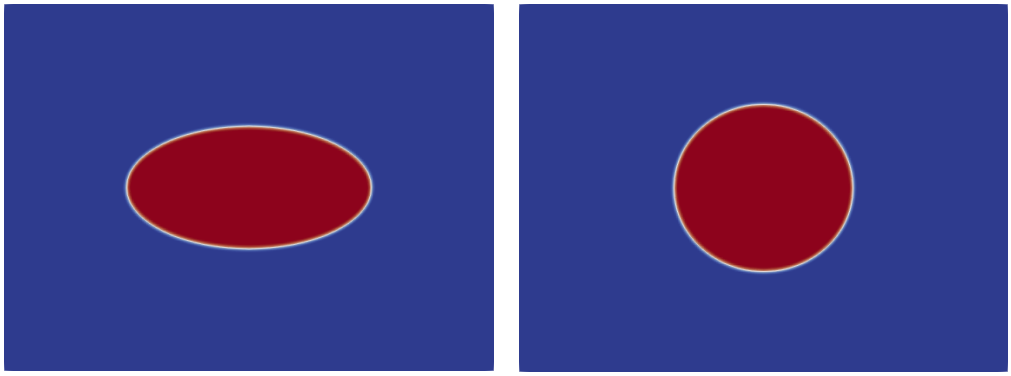


Figure 7.4: The initial and final shapes of 2×1 ellipse with Willmore flow and explicit volume constraint at $t = 0$ and $t = 4$.

tion with 4 MPI processors.

The bending energy (using equation (7.16)) of a circle with equivalent volume, that is with radius $r = \frac{1}{2}\sqrt{4} = 1$ is $\frac{\pi}{r} = \pi \approx 3.1415926$.

The energies for this 4×1 ellipse and the volume and surface area are given in Figure 7.5. The initial and final states are given in Figure 7.6. The profile initially tends toward a local minimum in the shape of a dog bone but as time goes on, enough volume is lost to be able to exit this local minimum toward the circular shape.

We also display the energies, surface area and volume for the Willmore flow of the 4×1 ellipse with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 7.7. The initial and final shapes are displayed in Figure 7.8 and some plots of the velocity field through the evolution are given in Figure 7.9. It appears that the evolution with volume constraint got stuck in a local minimum since the bending energy of the circle with an equivalent volume is π which is smaller than the final bending energy here of around 6. Notice that when the volume constraint is enforced, we do not get the same steady state shape as without.

7.5.3 The 8×1 ellipse in 2D

We let $\Lambda = [-6, 6] \times [-3, 3]$ and subdivide our initial mesh with 12 times in the x direction and 6 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{8}\right)^2 + \left(\frac{y}{1}\right)^2} \tag{7.20}$$

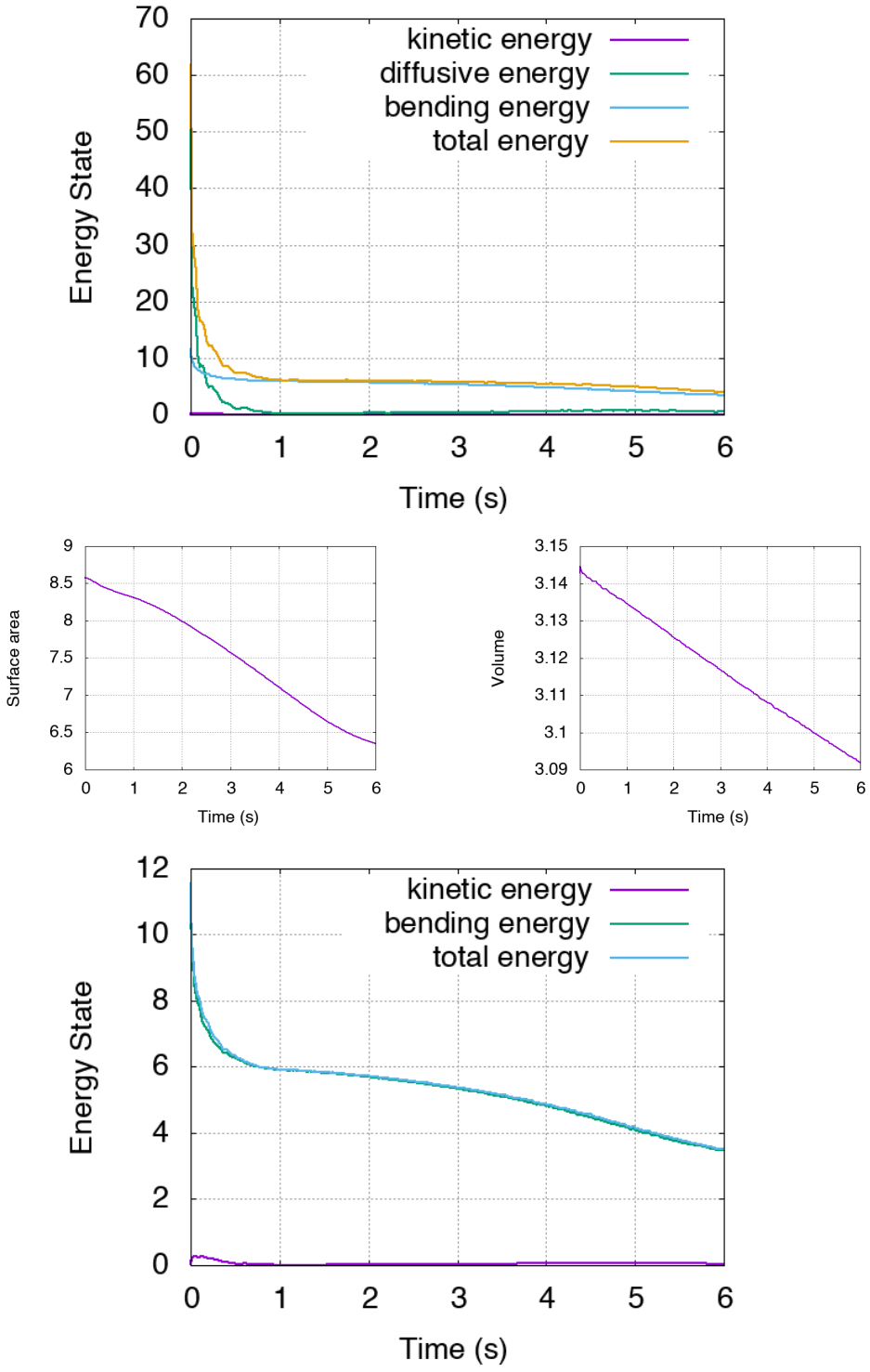


Figure 7.5: The energy, surface area and volume statistics for the 4x1 ellipse with Willmore force.

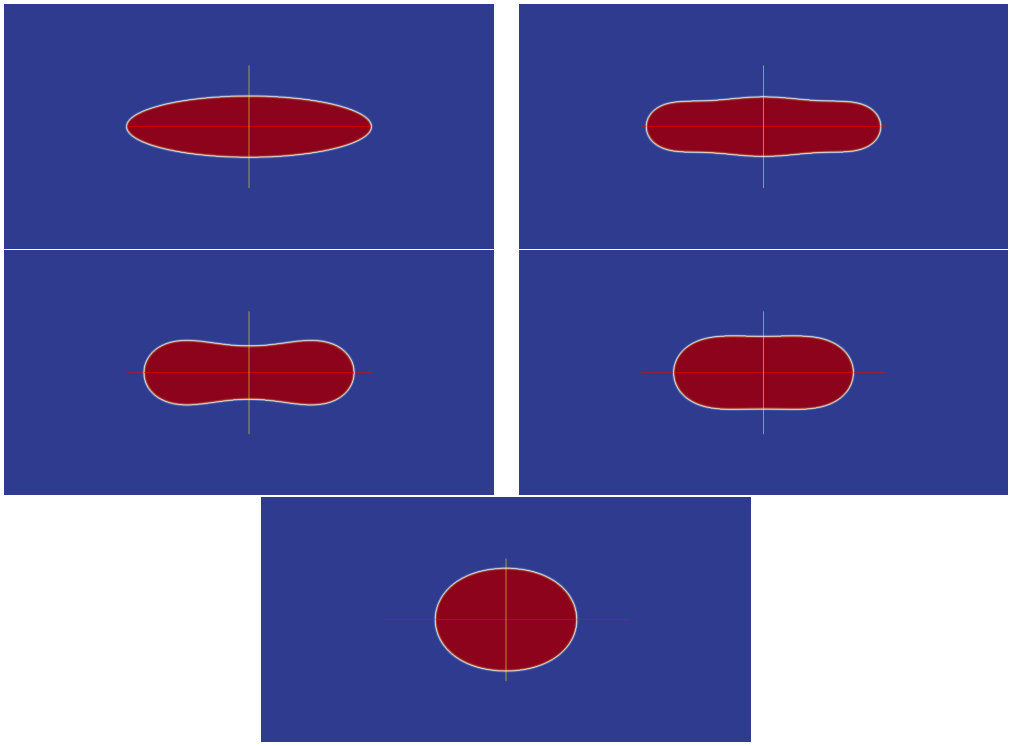


Figure 7.6: The shapes of 4×1 ellipse with Willmore flow are displayed at $t = 0$, $t = 0.2$, $t = 2.0$, $t = 4.0$ and $t = 6.0$ (from left to right and top to bottom). It appears to be converging toward the final profile of a circle with bending energy, π.

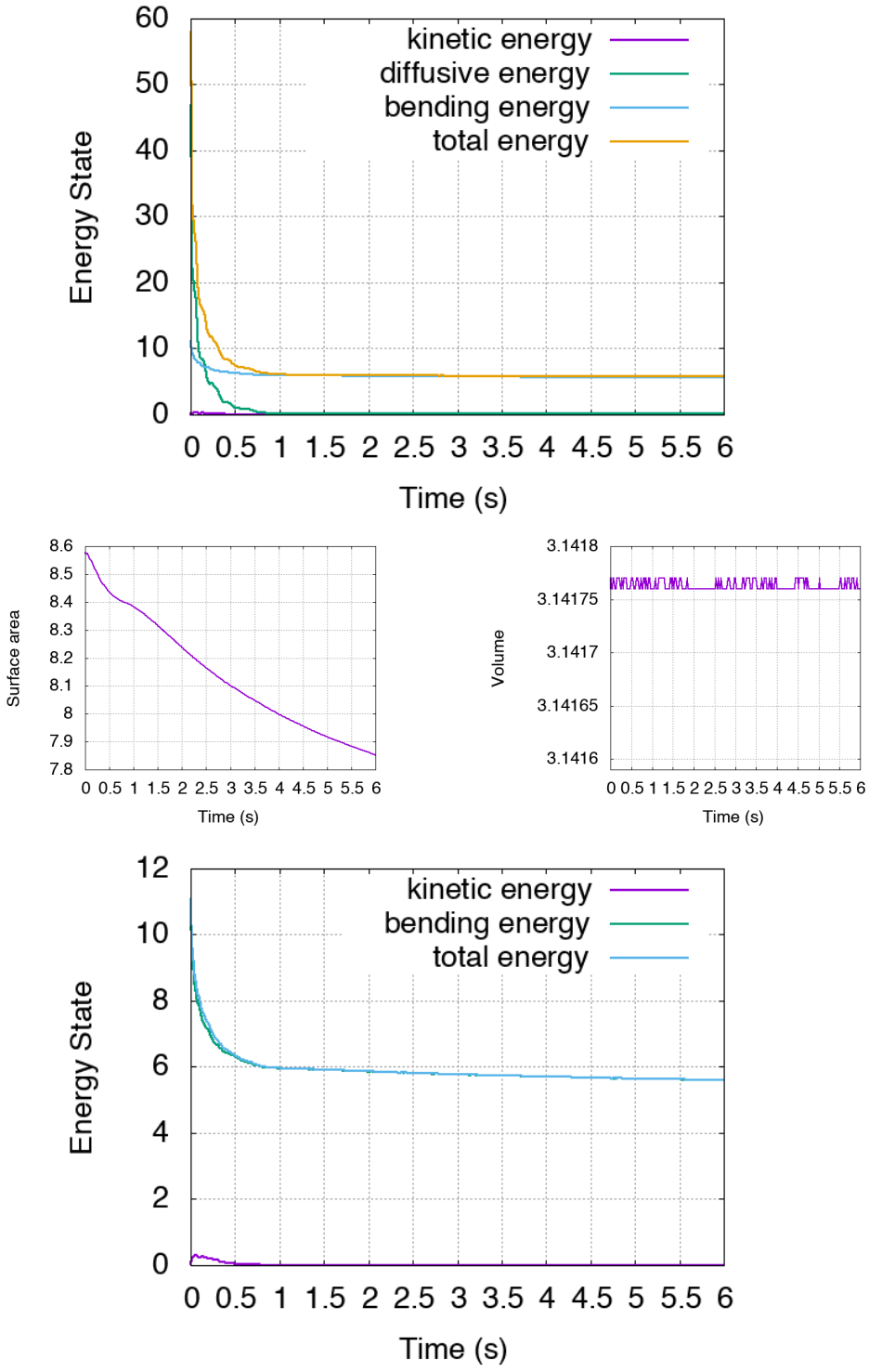


Figure 7.7: The energy, surface area and volume statistics for the 4x1 ellipse with Willmore force and explicit volume constraint.

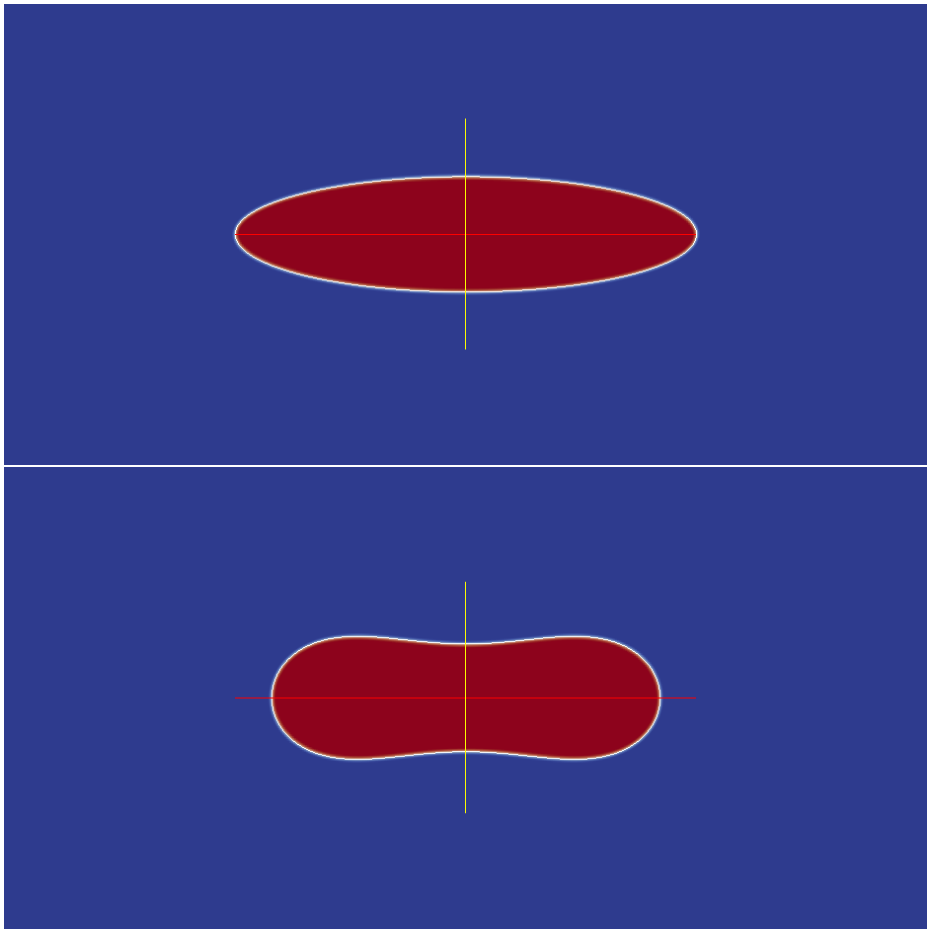


Figure 7.8: The initial and final shapes of 4×1 ellipse with Willmore flow with volume constraint at $t = 0$ and $t = 6$.

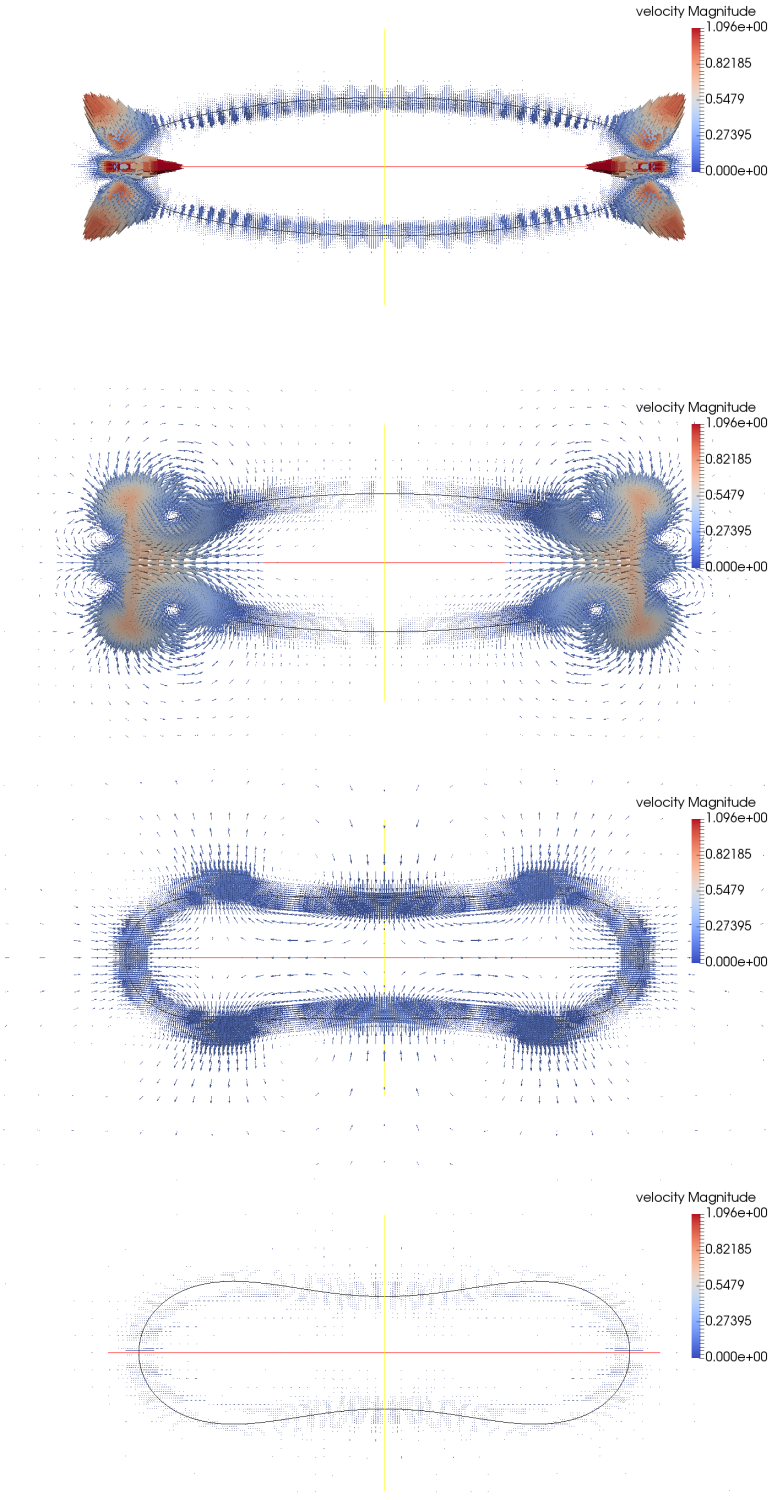


Figure 7.9: The velocity field of 4×1 ellipse with Willmore flow and explicit volume constraint at (from top to bottom) $t = 0.002$, $t = 0.02$, $t = 0.4$ and $t = 2.0$.

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a smaller timestep, $\Delta t = 0.001$, than for the previous cases since the velocities are correspondingly larger in the beginning. This is still on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the semi-implicit Willmore scheme described in Section 7.2 for this computation with 4 MPI processors.

The bending energy (using equation (7.16)) of a circle with equivalent volume, that is with radius $r = \frac{1}{2}\sqrt{8} = \sqrt{2}$ is $\frac{\pi}{r} = \frac{\pi}{\sqrt{2}} \approx 2.22144146$.

The energies for this 8×1 ellipse and the volume and surface area are given in Figure 7.10. The initial and final states are given in Figure 7.11. It appears that the evolution got stuck in a local minimum since the energy of the circle with an equivalent volume is $\frac{\pi}{\sqrt{2}}$ which is smaller than the final bending energy here.

We also display the energies, surface area and volume for the Willmore flow of the 8×1 ellipse with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 7.7. The Heaviside function for the initial, some intermediate and the final shape are displayed in Figure 7.13. The velocity fields showing how the Willmore forces induce a rotating field are also displayed in Figure 7.14. We can see in Figure 7.15 by directly comparing the final states that enforcing the volume constraint has a big effect on the steady state shape and is indeed vital to understand the minimal energy shapes.

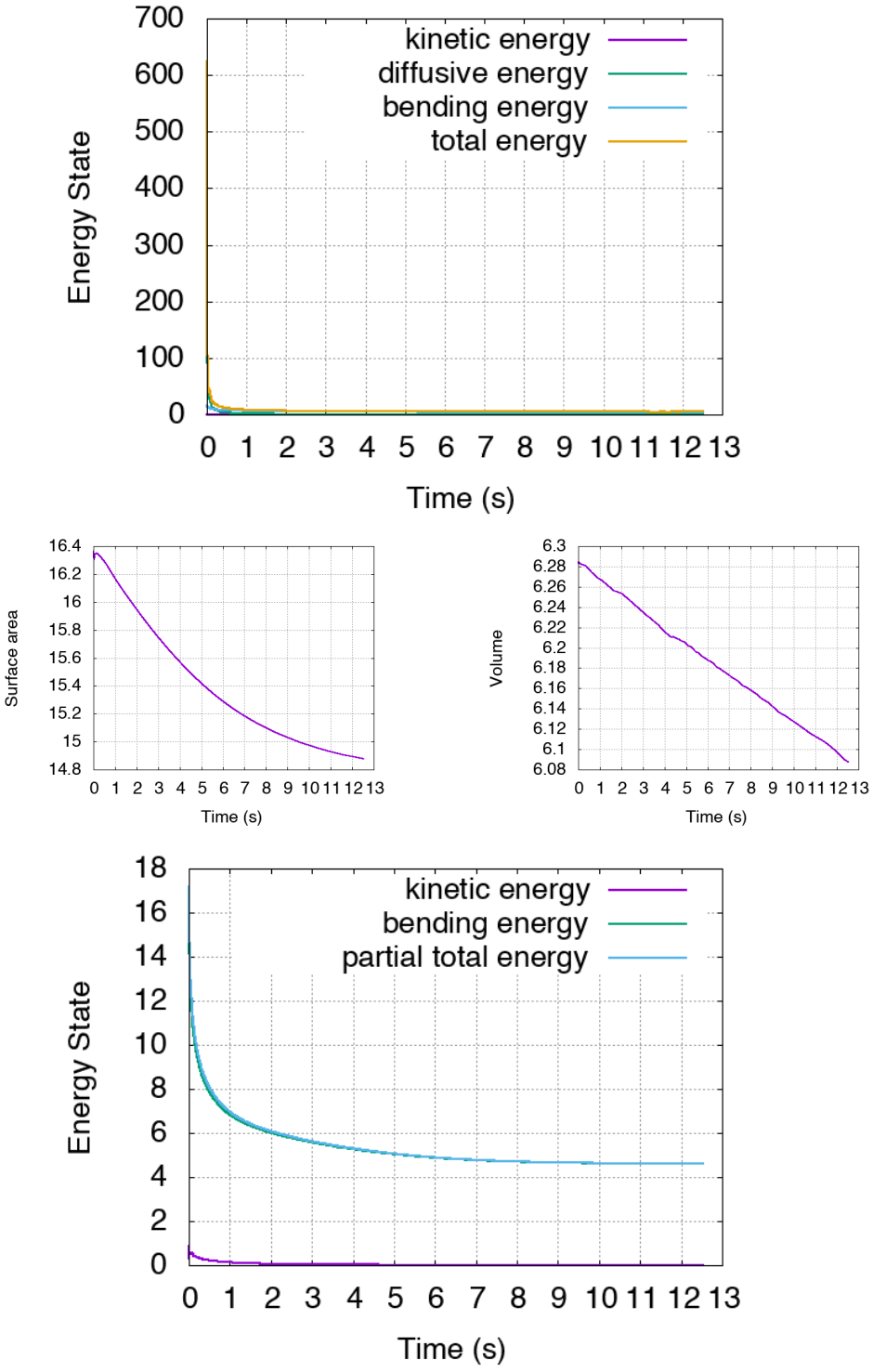


Figure 7.10: The energy, surface area and volume statistics for the 8×1 ellipse with Willmore force.

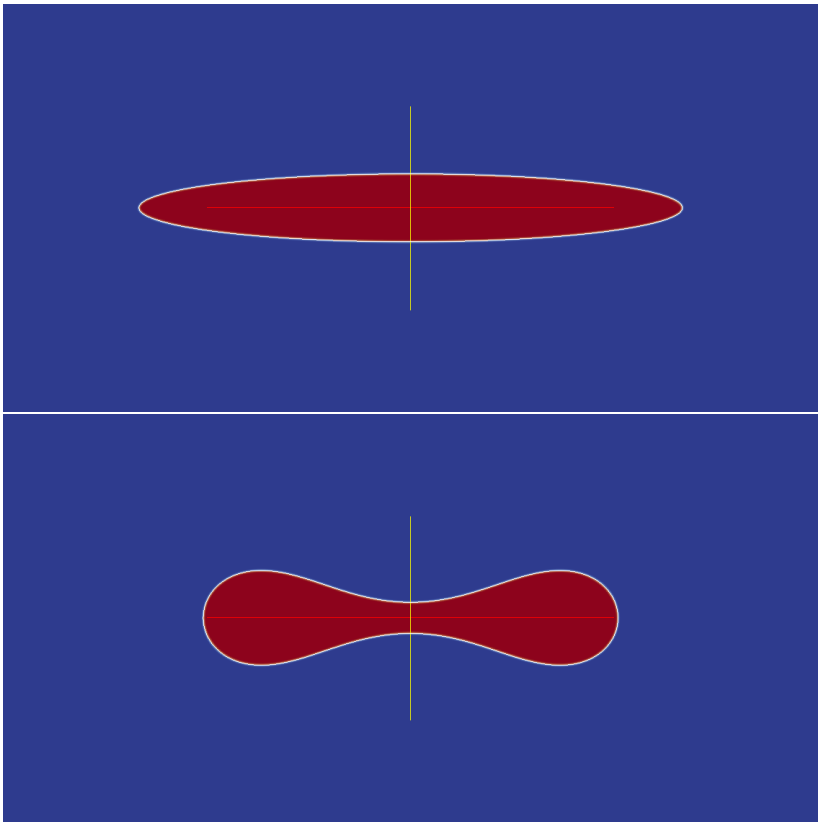


Figure 7.11: The initial and final shapes of 8×1 ellipse with Willmore flow at $t = 0.0$ and $t = 10.0$.

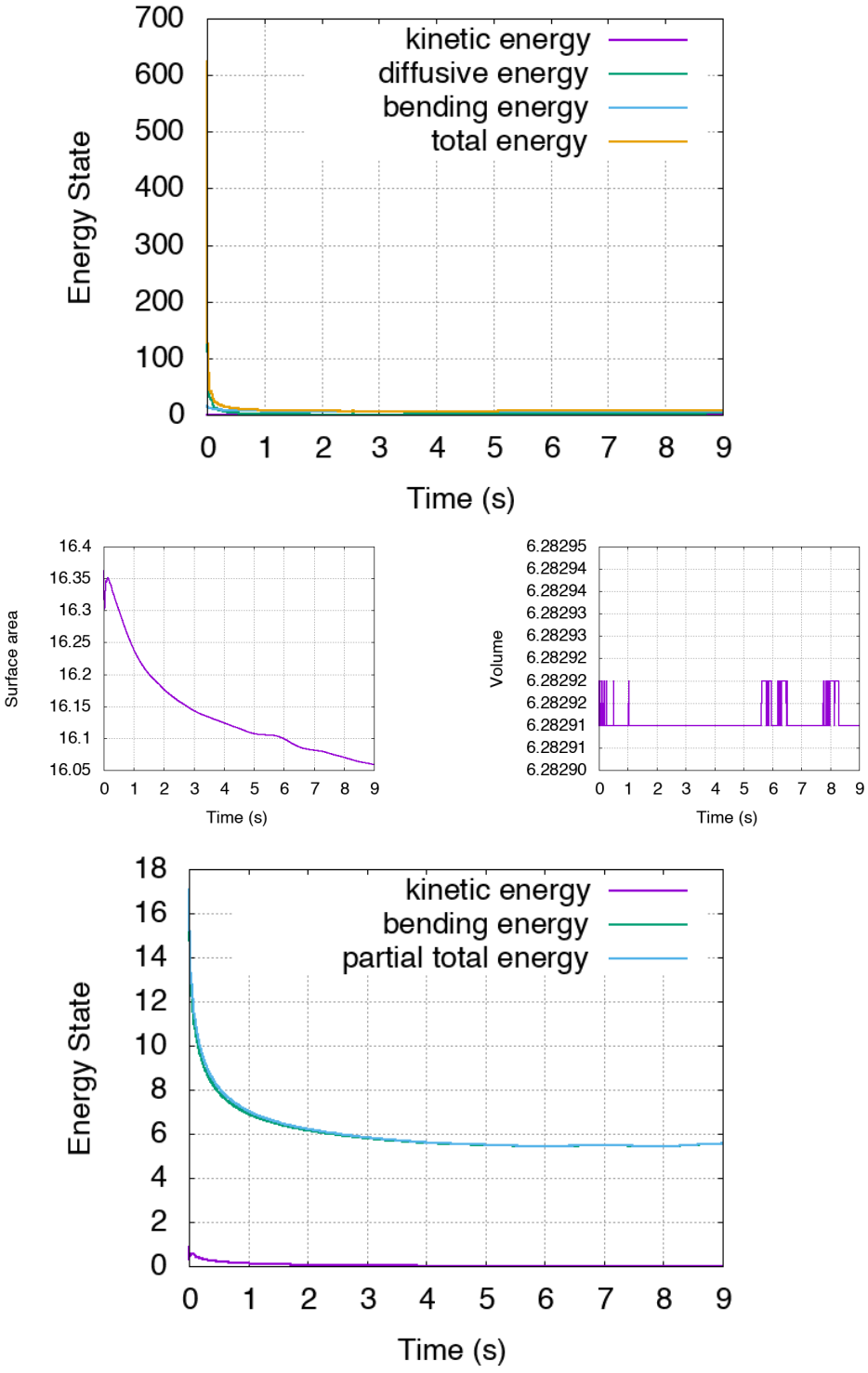


Figure 7.12: The energy, surface area and volume statistics for the 8×1 ellipse with Willmore force and explicit volume constraint.

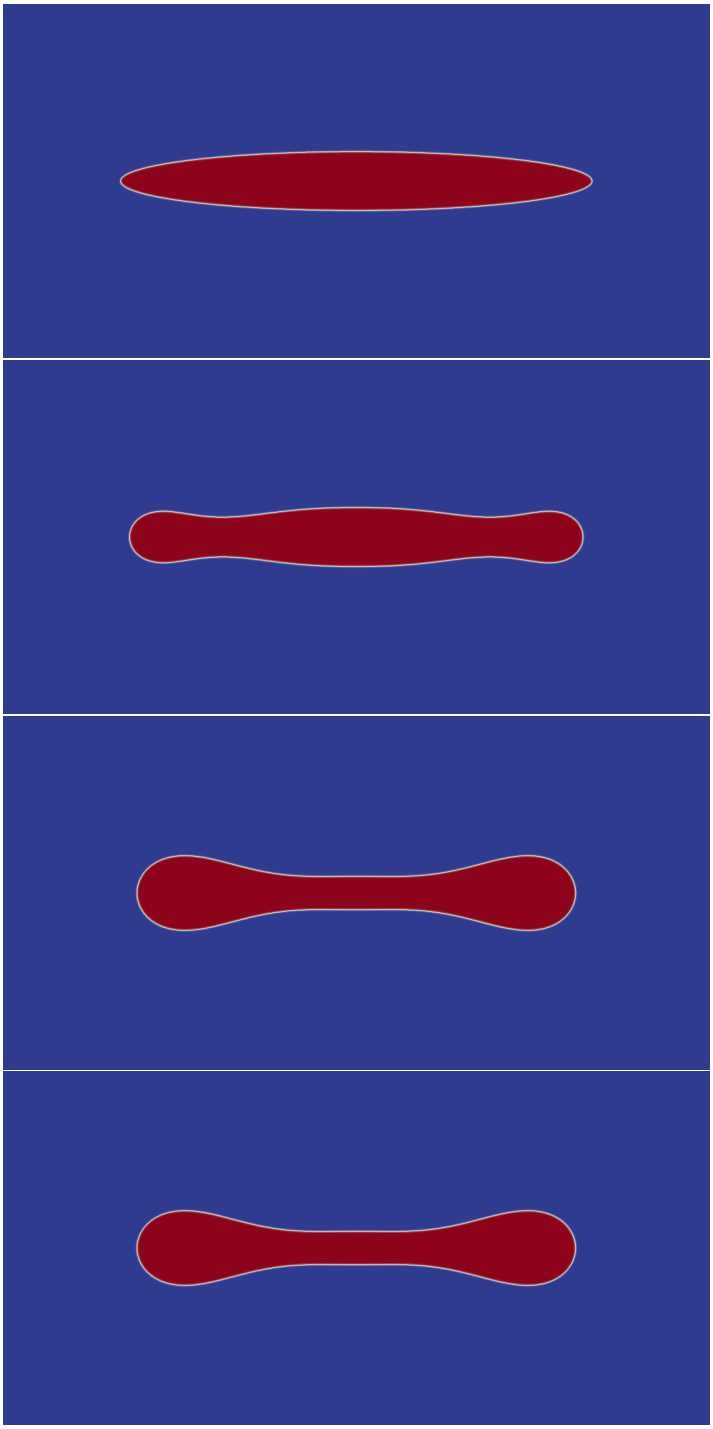


Figure 7.13: The shapes of 8×1 ellipse with Willmore flow and explicit volume constraint at (top to bottom) $t = 0.0$, $t = 1.0$, $t = 3.0$ and $t = 6.0$. By $t = 6.0$ we have reached a steady state shape.

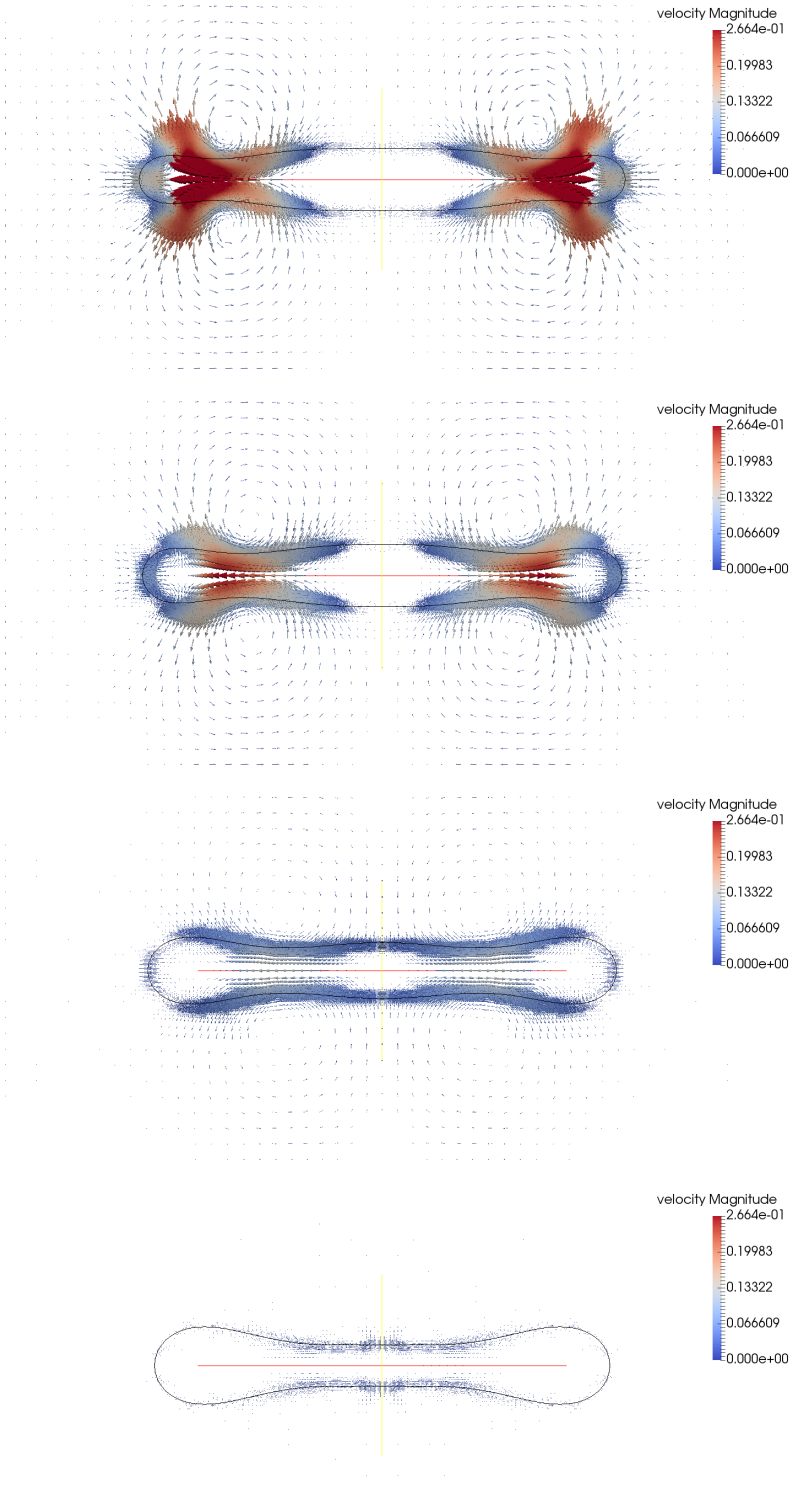


Figure 7.14: The velocity field of 8×1 ellipse with Willmore flow and explicit volume constraint at (from top to bottom) $t = 0.5$, $t = 1.0$, $t = 3.0$ and $t = 10.0$.

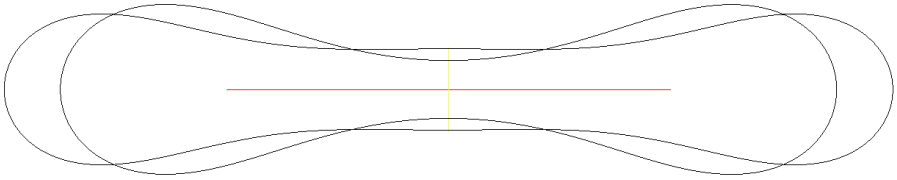


Figure 7.15: A comparison of the final shape for 8×1 ellipse with and without the volume constraint being explicitly enforced. Without enforcing the volume constraint, mass is lost over time leading to a different steady state shape. The wider one is with the volume constraint being enforced.

8. ENERGY FLOW APPLICATION: CANHAM-HELFRICH FLOW

The final application of our energy flow will be the Canham-Helfrich energy which is the bending energy defined in Chapter 7 combined with a surface area and volume constraint.

At the continuous level, the forces from the two-phase flow will balance with the Willmore force on Γ and the surface area and volume constraints with Lagrange multipliers $\lambda_{sa}, \lambda_v \in \mathbb{R}$ to enforce the constraints,

$$\int_\Gamma (\llbracket \mu \nabla^s \mathbf{u} - p \rrbracket \cdot \mathbf{n}) \cdot \mathbf{v} d\mathbf{x} = -k_e \partial e_w(\Gamma)(\mathbf{v}) - \lambda_{sa} \partial e_{sa}(\Gamma)(\mathbf{v}) - \lambda_v \partial e_v(\Gamma)(\mathbf{v})$$

where $e_w(\Gamma)$ is the Willmore energy functional defined in equation (7.1), $e_a(\Gamma)$ is the surface area functional,

$$e_{sa}(\Gamma) = \int_\Gamma 1 d\mathbf{x} - \int_{\Gamma^0} 1 d\mathbf{x}$$

and $e_v(\Gamma)$ is the volume functional, using Ω the interior region with $\partial\Omega = \Gamma$ and the divergence theorem,

$$e_v(\Gamma) = \int_\Omega d\mathbf{x} - \int_{\Omega^0} d\mathbf{x} = \frac{1}{d} \int_\Gamma \mathbf{x} \cdot \mathbf{n} d\mathbf{x} - \frac{1}{d} \int_{\Gamma^0} \mathbf{x} \cdot \mathbf{n} d\mathbf{x}.$$

Following the pattern of Chapter 5, we replace the exact surface area and volume functionals with their approximate counterparts using the smeared Dirac delta function. Thus, we introduce the approximate surface area constraint functional as

$$E_{sa}(\varphi) = \int_\Lambda \delta_\varepsilon(\varphi) |\nabla \varphi| d\mathbf{x} - A_0, \tag{8.1}$$

where A_0 is the approximate surface area of the initial surface defined by φ^0, and the

approximate volume constraint functional as

$$E_v(\varphi) = \int_\Lambda H_\varepsilon(\varphi) d\mathbf{x} - V_0, \tag{8.2}$$

where V_0 is the volume of the initial surface and $H_\varepsilon(\varphi)$ is the approximate Heaviside (characteristic) function of equation (4.11). Then using the scalar Lagrange multipliers $\lambda_{sa}, \lambda_v \in \mathbb{R}$ which will be chosen to enforce the surface area and volume constraints respectively and using the Willmore energy $E_w(\varphi)$ of equation (7.2), we have the approximate Canham-Helfrich energy,

$$\begin{aligned} E_{ch}(\varphi, \lambda_{sa}, \lambda_v) :&= E_w(\varphi) + \lambda_{sa} E_{sa}(\varphi) + \lambda_v E_v(\varphi) \qquad (8.3) \\ &= \frac{k_e}{2} \int_\Lambda h_\varphi^2 \delta_\varepsilon(\varphi) |\nabla \varphi| d\mathbf{x} \\ &\quad + \lambda_{sa} \left(\int_\Lambda \delta_\varepsilon(\varphi) |\nabla \varphi| d\mathbf{x} - A_0 \right) \\ &\quad + \lambda_v \left(\int_\Lambda H_\varepsilon(\varphi) d\mathbf{x} - V_0 \right). \qquad (8.4) \end{aligned}$$

This Canham-Helfrich energy is used to model the flow of blood cells and is a dominant component of the dynamics for any lipid bilayer vesicle or membrane in the short time frame. That is on the time frame where osmosis through the cell is minimal or the folding, deletion or addition of membrane surface is not occurring.

It was observed in [73] and [74] that the bending energy minimizing shape of a vesicle including the shapes of stomatocytes, oblate and prolate ellipsoids, can be directly linked to a parameter called the reduced volume, V_{red}, defined as

$$V_{\text{red}} := \frac{V}{4\pi R_0^3/3} \leq 1 \tag{8.5}$$

where V is the volume of the cell and R_0 is the radius of the sphere of equivalent surface

area,

$$A = 4\pi R_0^2. \tag{8.6}$$

Interestingly, the minimal bending energy absent a volume or surface area constraint in 3 dimensions becomes a sphere, when $V_{\text{red}} = 1$, but common shapes of vesicles observed in nature like prolate ellipsoids, dumbbells, oblate ellipsoids, discocytes and stomatocytes are the minimal bending energies for other fixed values of $V_{\text{red}} < 1$. See [74] for many fascinating phase diagrams relating the shapes of the minimizing bending energy surfaces to parameters like spontaneous curvature, reduced volume and various types of up-down symmetries/asymmetries. This fact has motivated our choice of initial profile shapes in the numerical results section, Section 8.3. We choose initial shapes to be an ellipsoid with various relative ratios of length of major and minor axes that have reduced volume, $V_{\text{red}} < 1$. These should evolve toward the discocytes or dumbbell shapes in 2D and 3D.

We will now describe an algorithm for enforcing the surface area and/or volume constraints at each time step. This allows us to model the Canham-Helfrich energy when added to the algorithms of minimizing Willmore energy of Chapter 7. The algorithm consists of a Newton-like iteration for λ_{sa} and λ_v to enforce $E_{sa}(\varphi) = 0$ and/or $E_v(\varphi) = 0$. The algorithm will be described in Section 8.2 and then some simulations of the Canham-Helfrich flow in 2D and 3D will be provided in Section 8.3.

8.1 Volume constraints

Observe that when we couple the Willmore forces to the incompressible Navier-Stokes system (4.1), the pressure variable is already acting as a Lagrange multiplier for the volume constraint. That is, our velocity is pushed by the pressure toward being divergence free, thereby imposing a volume constraint on the solution over time. Thus, we do not need to explicitly enforce again the volume constraint as we will the surface area constraint. However, we have observed that the level set reinitialization scheme introduces

mass loss so that explicitly enforcing the volume constraint may still be a useful option for longer simulations. We have already observed in Figure 7.15 of Chapter 7 that the volume loss does affect the steady state shape and so we recommend that explicitly enforcing the volume constraint is a necessary component of this algorithm. To this end, we will discuss the implementation of the volume constraint in conjunction with the surface area constraint.

8.2 Surface area and volume constraints

We will now describe the method of choosing $\lambda_{sa}, \lambda_v \in \mathbb{R}$ and incorporate the constraint forces into the Navier-Stokes system.

8.2.1 Variation of surface area functional

Notice that the surface area energy functionals is similar to the surface tension functionals of Chapter 6. We can thus compute that the shape variation of $e_{sa}(\Gamma)$ under velocity field $\mathbf{v}$ is

$$\partial e_{sa}(\Gamma)(\mathbf{v}) = \int_\Gamma h\mathbf{n}\cdot\mathbf{v}d\mathbf{x}$$

and using Theorem 5.2.1 with $G(\mathbf{x}, h) = 1$ and $\theta = \mathbf{n}\cdot\mathbf{v}$, we have

$$\partial E_{sa}(\varphi)(\mathbf{n}\cdot\mathbf{v}|\nabla\varphi|) = \int_\Lambda h_\varphi \mathbf{n}_\varphi \cdot \mathbf{v}\delta_\varepsilon(\varphi)|\nabla\varphi|d\mathbf{x}.$$

8.2.2 Variation of volume functional

Following the approaches of Chapter 6, we compute the variation of the volume functional.

Theorem 8.2.1. *Given $\theta \in H^1(\Lambda)$, and assuming that $\delta_\varepsilon(\varphi) = 0$ on $\partial\Lambda$, then the variation*

of the volume constraint in the direction $\varphi_t = \theta|\nabla\varphi|$ *is*

$$\partial E_v(\varphi)(\theta|\nabla\varphi|) = \int_\Lambda \delta_\varepsilon(\varphi)|\nabla\varphi|\theta d\boldsymbol{x}. \tag{8.7}$$

Proof. We compute the variation, using $H'_\varepsilon(\varphi) = \delta_\varepsilon(\varphi)$ to be

$$\begin{aligned}
\partial E_v(\varphi)(\theta|\nabla\varphi|) &= \frac{\partial}{\partial\eta} E_v\left(\varphi + \eta\theta|\nabla\varphi|\right)\Big|_{\eta=0} \\
&= \int_\Lambda \frac{\partial}{\partial\eta}\left(H_\varepsilon(\varphi + \eta\theta|\nabla\varphi|)\right|_{\eta=0} d\mathbf{x} \\
&= \int_\Lambda H'_\varepsilon(\varphi)\theta|\nabla\varphi| d\mathbf{x} \\
&= \int_\Lambda \delta_\varepsilon(\varphi)|\nabla\varphi|\theta d\mathbf{x}
\end{aligned}$$

□

Thus, using $\theta = \mathbf{v}\cdot\mathbf{n}$, we are varying in the direction $\varphi_t = \mathbf{v}\cdot\mathbf{n}|\nabla\varphi|$

$$\partial E_v(\varphi)(\mathbf{v}\cdot\mathbf{n}|\nabla\varphi|) = \int_\Lambda \delta_\varepsilon(\varphi)|\nabla\varphi|\mathbf{v}\cdot\mathbf{n}d\mathbf{x}. \tag{8.8}$$

Remark. Note that this is equivalent to the variation of the true volume functional in the direction $\mathbf{x}_t = \mathbf{v}$,

$$\partial e_v(\Gamma)(\mathbf{v}) = \int_\Gamma \mathbf{v}\cdot\mathbf{n}d\mathbf{x} \tag{8.9}$$

which can be computed using standard shape calculus techniques as in [57]. *end Remark.*

8.2.3 Linearization of system with constraints

Our force balance relationship for Navier-Stokes with Willmore flow and constrained surface area and volume is

$$\int_\Gamma [\![2\mu\nabla^s\mathbf{u} - pI]\!]\cdot\mathbf{n}\cdot\mathbf{v}dx = -k_e\partial E_w(\varphi)(\mathbf{n}_\varphi\cdot\mathbf{v}|\nabla\varphi|) \\ - \lambda_{sa}\partial E_{sa}(\varphi)(\mathbf{n}_\varphi\cdot\mathbf{v}|\nabla\varphi|) - \lambda_v\partial E_v(\varphi)(\mathbf{n}_\varphi\cdot\mathbf{v}|\nabla\varphi|).$$

Note that λ_{sa} and λ_v are unknown variables and multiplied by terms containing curvature and the level set, making this is a nonlinear term. When solving for the solutions at time t^{k+1}, we will linearize the surface area and volume constraints by keeping $\lambda_{sa} = \lambda_{sa}^{k+1}$ and $\lambda_v = \lambda_v^{k+1}$ implicit, but the terms in the energy variations explicit,

$$\lambda_{sa}^{k+1}\partial E_{sa}(\varphi^k)(\mathbf{n}_\varphi^k\cdot\mathbf{v}|\nabla\varphi^k|) = \lambda_{sa}^{k+1}\int_\Lambda \delta_\varepsilon(\varphi^k)|\nabla\varphi^k|h_\varphi^k\mathbf{v}\cdot\mathbf{n}_\varphi^k d\mathbf{x}. \tag{8.10}$$

and

$$\lambda_v^{k+1}\partial E_v(\varphi^k)(\mathbf{n}_\varphi^k\cdot\mathbf{v}|\nabla\varphi^k|) = \lambda_v^{k+1}\int_\Lambda \delta_\varepsilon(\varphi^k)|\nabla\varphi^k|\mathbf{v}\cdot\mathbf{n}_\varphi^k d\mathbf{x}. \tag{8.11}$$

Since λ_{sa}^{k+1} and λ_v^{k+1} are still unknown, we will now discuss the manner in which we will compute them at each time step. The full algorithm is described in Section 8.2.6. In Section 8.2.4, we will discuss the use of the semi-implicit Willmore algorithm of Section 7.2 to compute the terms needed in Section 8.2.6. In Section 8.2.5, we will discuss the use of the sub-iterating Willmore algorithm of Section 7.3 to compute the terms needed in Section 8.2.6.

8.2.4 Semi-implicit Canham-Helfrich algorithm

We denote by

$$A_{ns+w}((\mathbf{u}^{k+1}, p^{k+1}, u_n^{k+1}, h_\varphi^{k+1}, g_w^{k+1}), (\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k),$$

the bilinear form related to our linearized semi-implicit splitting of the two-phase flow incompressible Navier-Stokes system with Willmore forces as discussed in Chapter 4 and Section 7.2. We have denoted the explicit dependence on the previous level set, φ^k to remind us that we are computing these terms on the explicit surface, Γ_k defined by φ^k. We also denote by

$$\begin{aligned} F((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g), \lambda_{sa}, \lambda_v; \varphi^k) := & F_{ns+w}((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\ & + \lambda_{sa} F_{sa}((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\ & + \lambda_v F_v((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k), \end{aligned} \tag{8.12}$$

the right hand side of the Navier-Stokes with Willmore flow and constraints, where following Section 8.2.3, the terms

$$\begin{aligned} F_{sa}((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) &= -\partial E_{sa}(\varphi^k)(\mathbf{v} \cdot \mathbf{n}_\varphi^k |\nabla \varphi^k|) \\ &= -\int_\Lambda h_\varphi^k \delta_\varepsilon(\varphi^k) |\nabla \varphi^k| \mathbf{v} \cdot \mathbf{n}^k d\mathbf{x} \\ &= \int_\Lambda h_\varphi^k \delta_\varepsilon(\varphi^k) \mathbf{v} \cdot \nabla \varphi^k d\mathbf{x} \end{aligned} \tag{8.13}$$

and

$$
\begin{aligned}
F_v((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) &= -\partial E_v(\varphi^k)(\mathbf{v} \cdot \mathbf{n}_\varphi^k |\nabla \varphi^k|) \\
&= -\int_\Lambda \delta_\varepsilon(\varphi^k) |\nabla \varphi^k| \mathbf{v} \cdot \mathbf{n}^k d\mathbf{x} \\
&= \int_\Lambda \delta_\varepsilon(\varphi^k) \mathbf{v} \cdot \nabla \varphi^k d\mathbf{x}
\end{aligned} \tag{8.14}
$$

are the right hand sides representing the constraint variations of equation (8.10) and equation (8.11). Notice that the previous curvature has been computed and is available as data for surface area right hand side, F_{sa}. Given λ_{sa} and λ_v, we seek a solution

$$
(\mathbf{u}^{k+1}, p^{k+1}, u_n^{k+1}, h_\varphi^{k+1}, g_w^{k+1})
$$

to the system

$$
\begin{aligned}
A_{ns+w}((\mathbf{u}^{k+1}, p^{k+1}, u_n^{k+1}, h_\varphi^{k+1}, g_w^{k+1}), (\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\
= F((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g), \lambda_{sa}, \lambda_v; \varphi^k).
\end{aligned} \tag{8.15}
$$

Remark. The method of choosing λ_{sa} and λ_v will be described in Section 8.2.6. *end Remark.*

Using the linearity of A_{ns+w} and the explicitness of the constraint variations in the right hand sides F_{sa} and F_v, we can decompose our total velocity $\mathbf{u}^{k+1}$ into three velocity terms: the first consisting of the velocity from the Navier-Stokes with the Willmore system and the second and third velocities from the surface area constraint and the volume constraint respectively,

$$
\mathbf{u}^{k+1} = \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa} \mathbf{u}_{sa}^{k+1} + \lambda_v \mathbf{u}_v^{k+1}.
$$

Since we have designed the system to be linear in velocity, we can compute each of the

three velocities separately and the add them using the superposition principle. The bilinear form is exactly the same for each case and only needs to be assembled once but the right hand sides will be different.

First we compute the solution set, $(\mathbf{u}_{ns+w}^{k+1}, p_{ns+w}^{k+1}, (u_n)_{ns+w}^{k+1}, h_{ns+w}^{k+1}, (g_w)_{ns+w}^{k+1})$ such that for all test functions $(\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g)$ defined in their appropriate spaces,

$$\begin{aligned} A_{ns+w}((\mathbf{u}_{ns+w}^{k+1}, p_{ns+w}^{k+1}, (u_n)_{ns+w}^{k+1}, h_{ns+w}^{k+1}, (g_w)_{ns+w}^{k+1}), (\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\ = F_{ns+w}((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k), \end{aligned} \tag{8.16}$$

representing the coupling of the Navier-Stokes equation and Willmore forces as described in Chapter 7.2.

Next we compute the surface area constraint solution set,

$$(\mathbf{u}_{sa}^{k+1}, p_{sa}^{k+1}, (u_n)_{sa}^{k+1}, h_{sa}^{k+1}, (g_w)_{sa}^{k+1})$$

such that for all test functions $(\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g)$ defined in their appropriate spaces,

$$\begin{aligned} A_{ns+w}((\mathbf{u}_{sa}^{k+1}, p_{sa}^{k+1}, (u_n)_{sa}^{k+1}, h_{sa}^{k+1}, (g_w)_{sa}^{k+1}), (\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\ = F_{sa}((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k, h^k). \end{aligned} \tag{8.17}$$

Finally, the velocity constraint solutions, $(\mathbf{u}_v^{k+1}, p_v^{k+1}, (u_n)_v^{k+1}, h_v^{k+1}, (g_w)_v^{k+1})$ solves the system

$$\begin{aligned} A_{ns+w}((\mathbf{u}_v^{k+1}, p_v^{k+1}, (u_n)_v^{k+1}, h_v^{k+1}, (g_w)_v^{k+1}), (\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \\ = F_v((\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g); \varphi^k) \end{aligned} \tag{8.18}$$

for all test functions $(\mathbf{v}, q, \theta_{u_n}, \theta_h, \theta_g)$ defined in their appropriate spaces.

We solve for each of these three solution sets before we begin searching for the Lagrange multipliers λ_{sa}^{k+1} and λ_{v}^{k+1}.

Remark. Notice that all of these systems use the same matrix representation but different right hand sides. Thus using a direct solver, we can compute all the solutions with the same LU decomposition. In other words, we do not add any complexity by adding in these additional systems to be solved. *end Remark.*

We will see in Section 8.2.6 that we will need the $\mathbf{u}_{sa}^{k+1}$ and $\mathbf{u}_{v}^{k+1}$ velocities to compute the optimal λ_{sa}^{k+1} and λ_{v}^{k+1} for enforcing the surface area and volume constraint.

Remark. Notice that if we are only using one of the constraints, we only need to compute the appropriate right hand side and velocity. We will give the algorithm for using both, but it is simple to drop one of the constraints by not computing the data and setting the Lagrange multiplier to 0 if it is not being used. *end Remark.*

8.2.5 Sub-iterating Canham-Helfrich algorithm

In the case of the sub-iterating scheme, we have a little more complication because of the sub-iterating algorithm of Section 7.3. The system for each sub-iterating step, $n > 0$ solves

$$A_{ns+w}((\mathbf{u}^{k+1,n+1}, p^{k+1,n+1}), (\mathbf{v}, q); \varphi^{k+1,n}) = F((\mathbf{v}, q), \lambda_{sa}, \lambda_{v}; \varphi^{k+1,n}, \varphi^{k})$$

where we denote by A_{ns+w}, the bilinear form implementing the sub-iterating two-phase flow incompressible Navier-Stokes scheme with Willmore flow of Chapter 4 and Section 7.3. We have denoted the right hand side of the Navier-Stokes with Willmore flow

and constraints following Section 8.2.3 as

$$\begin{aligned} F((\mathbf{v},q),\lambda_{sa},\lambda_v;\varphi^{k+1,n},\varphi^k) = F_{ns+w}((\mathbf{v},q);\varphi^{k+1,n}) \\ + \lambda_{sa}F_{sa}((\mathbf{v},q);\varphi^k) \\ + \lambda_v F_v((\mathbf{v},q);\varphi^k), \end{aligned} \tag{8.19}$$

where the terms

$$\begin{aligned} F_{sa}((\mathbf{v},q);\varphi^k) &= -\partial E_{sa}(\varphi^k)(\mathbf{v}\cdot\mathbf{n}_\varphi^k|\nabla\varphi^k|) \\ &= -\int_\Lambda h_\varphi^k \delta_\varepsilon(\varphi^k)|\nabla\varphi^k|\mathbf{v}\cdot\mathbf{n}^k d\mathbf{x} \\ &= \int_\Lambda h_\varphi^k \delta_\varepsilon(\varphi^k)\mathbf{v}\cdot\nabla\varphi^k d\mathbf{x} \end{aligned} \tag{8.20}$$

and

$$\begin{aligned} F_v((\mathbf{v},q);\varphi^k) &= -\partial E_v(\varphi^k)(\mathbf{v}\cdot\mathbf{n}_\varphi^k|\nabla\varphi^k|) \\ &= -\int_\Lambda \delta_\varepsilon(\varphi^k)|\nabla\varphi^k|\mathbf{v}\cdot\mathbf{n}_\varphi^k d\mathbf{x} \\ &= \int_\Lambda \delta_\varepsilon(\varphi^k)\mathbf{v}\cdot\nabla\varphi^k d\mathbf{x} \end{aligned} \tag{8.21}$$

are the right hand sides representing the constraint variations of equation (8.10) and equation (8.11). Notice that we use $\varphi^{k+1,n}$ in the Navier-Stokes with Willmore flow bilinear form and right hand side, but the explicit term φ^k for the constraints right hand side. In the course of computing the Willmore forces, the curvature term, h_φ^k, has also be computed based on φ^k and so is available as data for the surface area constraint.

Using the linearity of A_{ns+w} and the explicitness of our constraint variations, we can decompose our total velocity into three velocity terms: the first consisting of the velocity

from the Navier-Stokes with Willmore flow and the second and third velocities from the surface area constraint and the volume constraint respectively,

$$\mathbf{u}^{k+1} = \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa}\mathbf{u}_{sa}^{k+1} + \lambda_v \mathbf{u}_v^{k+1}.$$

The sub-iterating scheme is applied only on the Navier-Stokes with Willmore flow bilinear form and right hand side so we can first use the sub-iterating scheme to solve for the final $(\mathbf{u}_{ns+w}^{k+1,n+1}, p_{ns+w}^{k+1,n+1})$ and $\varphi_{ns+w}^{k+1,n+1}$ as described in Section 7.3 solving

$$A_{ns+w}((\mathbf{u}_{ns+w}^{k+1,n+1}, p_{ns+w}^{k+1,n+1}), (\mathbf{v}, q); \varphi_{ns+w}^{k+1,n}) = F_{ns+w}((\mathbf{v}, q); \varphi_{ns+w}^{k+1,n}). \tag{8.22}$$

for all test functions $(\mathbf{v}, q)$ in their appropriate velocity and pressure spaces. Once we have the final sub-iterating bilinear form and velocity solution $\mathbf{u}_{ns+w}^{k+1} = \mathbf{u}_{ns+w}^{k+1,n+1}$, we can solve for the constraint velocities $(\mathbf{u}_{sa}^{k+1}, p_{sa}^{k+1})$ (using the existing matrix representation of $A_{ns+w}(\cdot, \cdot; \varphi_{ns+w}^{k+1,n})$) as solution to

$$A_{ns+w}((\mathbf{u}_{sa}^{k+1}, p_{sa}^{k+1}), (\mathbf{v}, q); \varphi_{ns+w}^{k+1,n}) = F_{sa}((\mathbf{v}, q); \varphi^k). \tag{8.23}$$

for all test functions $(\mathbf{v}, q)$. Finally we also compute $(\mathbf{u}_v^{k+1}, p_v^{k+1})$ such that for all test functions $(\mathbf{v}, q)$,

$$A_{ns+w}((\mathbf{u}_v^{k+1}, p_v^{k+1}), (\mathbf{v}, q); \varphi_{ns+w}^{k+1,n}) = F_v((\mathbf{v}, q); \varphi^k). \tag{8.24}$$

We solve for these three terms before we begin searching for the Lagrange multipliers λ_{sa}^{k+1} and λ_v^{k+1}.

Remark. Notice that all of these systems use the same matrix representation but different right hand sides. The Willmore term is treated internally as an explicit force so the

system matrix is the matrix of Chapter 4 for which the Algebraic Multi-grid (AMG) preconditioner solves this system well in conjunction with the General Minimum Residual (GMRes) solver. Solving for the three velocities does not increase the complexity over just solving for one. *end Remark.*

We will see in Section 8.2.6 that we will need the $\mathbf{u}_{sa}^{k+1}$ and $\mathbf{u}_{v}^{k+1}$ velocities to compute the optimal λ_{sa}^{k+1} and λ_{v}^{k+1} for enforcing the surface area and volume constraint.

Remark. Notice that if we are only using one of the constraints, we only need to compute the appropriate right hand side and velocity. We will give the algorithm for using both, but it is simple to drop one of the constraints by not computing the data and setting the Lagrange multiplier to 0 if it is not being used. *end Remark.*

8.2.6 The Newton-like algorithm for enforcing constraints

We assume from this point forward that the velocities $\mathbf{u}_{ns+w}^{k+1}$, $\mathbf{u}_{sa}^{k+1}$ and $\mathbf{u}_{v}^{k+1}$ have already been computed using the appropriate Navier-Stokes with Willmore force algorithm as in Sections 8.2.4 or 8.2.5.

We now introduce the algorithm for selecting the pair $(\lambda_{sa}, \lambda_v)$ at each time step. We will compute it as a zero of the function

$$\mathbf{f}\,(\lambda_{sa}, \lambda_v) := \begin{pmatrix} \int_\Lambda \delta_\varepsilon(\varphi)|\nabla\varphi| d\mathbf{x} - A_0 \\ \int_\Lambda H_\varepsilon(\varphi) d\mathbf{x} - V_0 \end{pmatrix} = \begin{pmatrix} E_{sa}(\varphi) \\ E_v(\varphi) \end{pmatrix} \tag{8.25}$$

where the dependence on $(\lambda_{sa}, \lambda_v)$ is through the level set function $\varphi = \varphi(\lambda_{sa}, \lambda_v)$. Note that our velocity field, $\mathbf{u}$ has a linear dependence on λ_{sa} and λ_v as seen in our decomposition, $\mathbf{u} = \mathbf{u}_{ns+w} + \lambda_{sa}\mathbf{u}_{sa} + \lambda_v\mathbf{u}_v$, and the level set function advects according to the velocity $\mathbf{u}$, so that we have the relationship

$$\varphi_t + (\mathbf{u}_w + \lambda_{sa}\mathbf{u}_{sa} + \lambda_v\mathbf{u}_v) \cdot \nabla\varphi = 0. \tag{8.26}$$

Linearizing at time t^k with implicit velocity field $\mathbf{u}^{k+1}$ as per equation (7.7), we have

$$\varphi^{k+1} \approx \varphi^k - \Delta t \left(\mathbf{u}_w^{k+1} + \lambda_{sa} \mathbf{u}_{sa}^{k+1} + \lambda_v \mathbf{u}_v^{k+1} \right) \cdot \nabla \varphi^k. \tag{8.27}$$

Thus we can consider the level set function, φ^{k+1} to have a linear relationship with respect to λ_{sa} and λ_v. Furthermore, we compute

$$\frac{d\varphi^{k+1}}{d\lambda_{sa}} \approx -\Delta t \mathbf{u}_{sa}^{k+1} \cdot \nabla \varphi^k \tag{8.28}$$

and

$$\frac{d\varphi^{k+1}}{d\lambda_v} \approx -\Delta t \mathbf{u}_v^{k+1} \cdot \nabla \varphi^k. \tag{8.29}$$

Supposing that φ^{k+1} is the solution at time t^{k+1} from previous solution φ^k evolving under the velocity field

$$\mathbf{u}^{k+1} = \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa} \mathbf{u}_{sa}^{k+1} + \lambda_v \mathbf{u}_v^{k+1},$$

then the Jacobian of the function $\mathbf{f}(\lambda_{sa}, \lambda_v)$ is

$$D\mathbf{f}(\lambda_{sa}, \lambda_v) := \begin{pmatrix} \frac{\partial E_{sa}(\varphi^{k+1})}{\partial \lambda_{sa}} & \frac{\partial E_{sa}(\varphi^{k+1})}{\partial \lambda_v} \\ \frac{\partial E_v(\varphi^{k+1})}{\partial \lambda_{sa}} & \frac{\partial E_v(\varphi^{k+1})}{\partial \lambda_v} \end{pmatrix}. \tag{8.30}$$

If we define the Clement interpolants, $(u_{sa})_n^{k+1} := \mathcal{I}_{cl} \left(\mathbf{u}_{sa}^{k+1} \cdot \nabla \varphi^k \right)$ and $(u_v)_n^{k+1} :=$

$\mathcal{I}_{cl}\left(\mathbf{u}_v^{k+1}\cdot\nabla\varphi^k\right)$, then we can compute an approximation to the Jacobian terms as

$$\frac{\partial E_{sa}(\varphi^{k+1})}{\partial\lambda_{sa}} = -\Delta t\int_\Lambda \delta_\varepsilon'(\varphi^{k+1})|\nabla\varphi^{k+1}|(u_{sa})_n^{k+1} + \delta_\varepsilon(\varphi^{k+1})\frac{\nabla\varphi^{k+1}}{|\nabla\varphi^{k+1}|}\cdot\nabla(u_{sa})_n^{k+1}d\mathbf{x} \tag{8.31}$$

$$\frac{\partial E_{sa}(\varphi^{k+1})}{\partial\lambda_{v}} = -\Delta t\int_\Lambda \delta_\varepsilon'(\varphi^{k+1})|\nabla\varphi^{k+1}|(u_{v})_n^{k+1} + \delta_\varepsilon(\varphi^{k+1})\frac{\nabla\varphi^{k+1}}{|\nabla\varphi^{k+1}|}\cdot\nabla(u_{v})_n^{k+1}d\mathbf{x} \tag{8.32}$$

$$\frac{\partial E_{v}(\varphi^{k+1})}{\partial\lambda_{sa}} = -\Delta t\int_\Lambda \delta_\varepsilon(\varphi^{k+1})\mathbf{u}_{sa}^{k+1}\cdot\nabla\varphi^k d\mathbf{x} \tag{8.33}$$

$$\frac{\partial E_{v}(\varphi^{k+1})}{\partial\lambda_{v}} = -\Delta t\int_\Lambda \delta_\varepsilon(\varphi^{k+1})\mathbf{u}_{v}^{k+1}\cdot\nabla\varphi^k d\mathbf{x}. \tag{8.34}$$

This is true since we can take the partial derivative $\frac{\partial E_{sa}(\varphi^{k+1})}{\partial\lambda}$ for $\lambda=\lambda_{sa}$ or $\lambda=\lambda_v$, so that

$$\begin{aligned}\frac{\partial E_{sa}(\varphi^{k+1})}{\partial\lambda} &= \int_\Lambda \frac{\partial}{\partial\lambda}\left(\delta_\varepsilon(\varphi^{k+1})|\nabla\varphi^{k+1}|\right)d\mathbf{x}\\ &= \int_\Lambda \delta_\varepsilon'(\varphi^{k+1})\frac{\partial\varphi^{k+1}}{\partial\lambda}|\nabla\varphi^{k+1}| + \delta_\varepsilon(\varphi^{k+1})\frac{\nabla\varphi^{k+1}}{|\nabla\varphi^{k+1}|}\nabla\left(\frac{\partial\varphi^{k+1}}{\partial\lambda}\right)d\mathbf{x}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial E_{v}(\varphi^{k+1})}{\partial\lambda} &= \int_\Lambda \frac{\partial}{\partial\lambda}\left(H_\varepsilon(\varphi^{k+1})\right)d\mathbf{x}\\ &= \int_\Lambda H_\varepsilon'(\varphi^{k+1})\frac{\partial\varphi^{k+1}}{\partial\lambda}d\mathbf{x}\\ &= \int_\Lambda \delta_\varepsilon(\varphi^{k+1})\frac{\partial\varphi^{k+1}}{\partial\lambda}d\mathbf{x}.\end{aligned}$$

Now using the approximations to the $\frac{\partial\varphi^{k+1}}{\partial\lambda}$ terms in equations (8.28) and (8.29), and noting that since we must take the gradient of these terms, we can apply the Clement interpolant to the scalar terms $\frac{\partial\varphi^{k+1}}{\partial\lambda}$ which maps them into $H^1(\Lambda)$, we obtain the result.

We will use a Newton-like iteration on $\mathbf{f}(\lambda_{sa}, \lambda_v)$ to find the zero of $\mathbf{f}$ at $(\lambda_{sa}^{k+1}; \lambda_v^{k+1})$. The very first step of evolution, we will initialize with $\lambda_{sa}^0 = 0$ and $\lambda_v^0 = 0$ and then to move from timestep t^k to time step t^{k+1} with $k \geq 0$, we initialize the Newton method with $(\lambda_{sa}^{k+1,0}, \lambda_v^{k+1,0}) = (\lambda_{sa}^k, \lambda_v^k)$. Then for $n = 0, \ldots, MAXITER$ we compute

$$\begin{pmatrix} \lambda_{sa}^{k+1,n+1} \\ \lambda_v^{k+1,n+1} \end{pmatrix} = \begin{pmatrix} \lambda_{sa}^{k+1,n} \\ \lambda_v^{k+1,n} \end{pmatrix} - D\mathbf{f}(\lambda_{sa}^{k+1,n}, \lambda_v^{k+1,n})^{-1}\mathbf{f}(\lambda_{sa}^{k+1,n}, \lambda_v^{k+1,n}) \tag{8.35}$$

where following equation (8.25), $\mathbf{f}$ is defined as

$$\mathbf{f}\left(\lambda_{sa}^{k+1,n}, \lambda_v^{k+1,n}\right) = \begin{pmatrix} \int_\Lambda \delta_\varepsilon(\varphi^{k+1,n})|\nabla\varphi^{k+1,n}|d\mathbf{x} - A_0 \\ \int_\Lambda H_\varepsilon(\varphi^{k+1,n})d\mathbf{x} - V_0 \end{pmatrix} \tag{8.36}$$

and the Jacobian $D\mathbf{f}$ as in equation (8.30) using $\varphi^{k+1} = \varphi^{k+1,n}$. Once we have achieved $\left|\mathbf{f}\left(\lambda_{sa}^{k+1,n}, \lambda_v^{k+1,n}\right)\right| < TOL$, we finish the sub-iteration and move on to the next time step. We will typically choose $TOL = \mathcal{O}(h_{\min}^{3/2})$ so that the constraint errors are consistent with the use of $\delta_\varepsilon(\varphi)$ to compute them. See Theorem 3.3.1 for more details on the convergence rates of our approximate Dirac delta function.

A pseudo code summary for the linearized constraint algorithm using the semi-implicit algorithm for computing the Willmore flow is given in Algorithm 7 which also uses Algorithm 9 to describe the Newton-like algorithm for choosing the Lagrange multipliers. Likewise a psuedo-code summary for the linearized constraint algorithm using the sub-iterating Willmore scheme is given in Algorithm 8 which also uses Algorithm 9 to break the sub-iteration scheme and Newton-like algorithm into distinct pieces.

Remark. When using the pressure correction algorithm for the Navier-Stokes system, the pressure update can in fact be deferred until after the Lagrange multipliers have been found and the final velocity, $\mathbf{u}^{k+1} = \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa}^{k+1}\mathbf{u}_{sa}^{k+1} + \lambda_v^{k+1}\mathbf{u}_v^{k+1}$, has been set. Doing

this saves us from solving the pressure update system (a Poisson equation) 3 separate times and then adding them together like the velocity. Since the system is linear, there is no problem doing either approach but the deferred pressure correction method is more efficient. We use the deferred pressure update in the sub-iterating Algorithm 8 where we have denoted the pressure update algorithm as compute_pressure$(p^k, \mathbf{u}^{k+1})$. In the semi-implicit scheme we have not used the deferred pressure update but it can be done just the same. *end Remark.*

Algorithm 7: The Newton-like method for imposing constraints on the Navier-Stokes system with the semi-implicit Willmore flow algorithm.

Data: $(\mathbf{u}^k, p^k, u_n^k, h^k, g^k, \varphi^k, \lambda_{sa}^k, A_0, \lambda_v^k, V_0, MAXITER, TOL)$
Result: $(\mathbf{u}^{k+1}, p^{k+1}, u_n^{k+1}, h^{k+1}, g^{k+1}, \varphi^{k+1}, \lambda_{sa}^{k+1}, \lambda_v^{k+1})$

begin
  /* Using $(\mathbf{u}^k, p^k, u_n^k, h^k, g^k, \varphi^k)$ compute A_{ns+w}, F_{ns+w}, F_{sa} and F_v, the Navier-Stokes with semi-implicit Willmore system matrix and right hand side, the surface area right hand side and the volume right hand side of equations (8.16), (8.17) and (8.18). */
  /* Compute the intermediate components $(\mathbf{u}_{ns+w}^{k+1}$, $p_{ns+w}^{k+1}, (u_n)_{ns+w}^{k+1}$, h_{ns+w}^{k+1}, $g_{ns+w}^{k+1})$, $(\mathbf{u}_{sa}^{k+1}$, p_{sa}^{k+1}, $(u_n)_{sa}^{k+1}$, h_{sa}^{k+1}, $g_{sa}^{k+1})$ and $(\mathbf{u}_v^{k+1}$, p_v^{k+1}, $(u_n)_v^{k+1}$, h_v^{k+1}, $g_v^{k+1})$ from these three systems. */
  /* Using Algorithm 9, update Lagrange multipliers. */
  $\left(\lambda_{sa}^{k+1}, \lambda_v^{k+1}\right) \leftarrow$ update_lagrange_multipliers $\left(\mathbf{u}_{ns+w}^{k+1}, \lambda_{sa}^k, \mathbf{u}_{sa}^{k+1}, \lambda_v^k, \mathbf{u}_v^{k+1}, \varphi^k, A_0, V_0\right)$;
  if *exit with convergence of Newton's method* **then**
    /* Transfer solutions to prepare for next step */
    $\mathbf{u}^{k+1} \leftarrow \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa}^{k+1}\mathbf{u}_{sa}^{k+1} + \lambda_v^{k+1}\mathbf{u}_v^{k+1}$;
    $u_n^{k+1} \leftarrow (u_n)_{ns+w}^{k+1} + \lambda_{sa}^{k+1}(u_n)_{sa}^{k+1} + \lambda_v^{k+1}(u_n)_v^{k+1}$;
    $p^{k+1} \leftarrow p_{ns+w}^{k+1} + \lambda_{sa}^{k+1}p_{sa}^{k+1} + \lambda_v^{k+1}p_v^{k+1}$
    $h^{k+1} \leftarrow h_{ns+w}^{k+1} + \lambda_{sa}^{k+1}h_{sa}^{k+1} + \lambda_v^{k+1}h_v^{k+1}$
    $g^{k+1} \leftarrow g_{ns+w}^{k+1} + \lambda_{sa}^{k+1}g_{sa}^{k+1} + \lambda_v^{k+1}g_v^{k+1}$
    $\varphi^{k+1} \leftarrow$ transport$(\varphi^k, \mathbf{u}^{k+1})$;
  end
end

Algorithm 8: The Newton-like method for imposing constraints on the sub-iterating Navier-Stokes/Level Set system with Willmore flow.

Data: $(\mathbf{u}^k, p^k, \varphi^k, \lambda_{sa}^k, \lambda_v^k)$
Result: $(\mathbf{u}^{k+1}, p^{k+1}, \varphi^{k+1}, \lambda_{sa}^{k+1}, \lambda_v^{k+1})$
begin
  $\mathbf{u}_{ns+w}^{k+1,0} \leftarrow \mathbf{u}^k$;
  $\varphi_{ns+w}^{k+1,0} \leftarrow \varphi^k$;
  for $n = 0, \ldots,$ *MAXSUBSTEPS* **do**
    refine_mesh$(\varphi^k, \varphi^{k+1,n})$; /* ensure mesh is fully refined in band around both solutions */
    $F_{\text{energy}}^n \leftarrow$ compute_energy_rhs$(\varphi_{ns+w}^{k+1,n})$;
    $\mathbf{u}_{ns+w}^{k+1,n+1} \leftarrow$ compute_velocity $\left(\mathbf{u}^k, \varphi_{ns+w}^{k+1,n}, F_{\text{energy}}^n\right)$;
    $\varphi_{ns+w}^{k+1,n+1} \leftarrow$ compute_levelset $\left(\varphi^k, \mathbf{u}_{ns+w}^{k+1,n+1}\right)$;
    /* Use coefficient vectors, $\mathbf{U}_{ns+w}$, to measure convergence of sub-iterating scheme */
    if $\|\boldsymbol{U}_{ns+w}^{k+1,n+1} - \boldsymbol{U}_{ns+w}^{k+1,n}\|_{\ell^2} < 0.01\|\boldsymbol{U}_{ns+w}^{k+1,n}\|_{\ell^2}$ **then**
      exit sub-iteration scheme with convergence;
    end
  end
  if *exit sub-iteration scheme with convergence* **then**
    $\mathbf{u}_{ns+w}^{k+1} \leftarrow \mathbf{u}_{ns+w}^{k+1,n+1}$;
    $\varphi_{ns+w}^{k+1} \leftarrow \varphi_{ns+w}^{k+1,n+1}$;
    /* Begin Newton iteration */
    $\left(F_{sa}^{k+1}, F_v^{k+1}\right) \leftarrow$ compute_constraint_rhs(φ^k);
    $\left(\mathbf{u}_{sa}^{k+1}, \mathbf{u}_v^{k+1}\right) \leftarrow$ compute_constraint_velocity$(\mathbf{u}_{ns+w}^{k+1}, \varphi_{ns+w}^{k+1}, F_{sa}^{k+1}, F_v^{k+1})$;
    /* Using Algorithm 9, update Lagrange multipliers */
    $\left(\lambda_{sa}^{k+1}, \lambda_v^{k+1}\right) \leftarrow$
    update_lagrange_multipliers $\left(\mathbf{u}_{ns+w}^{k+1}, \lambda_{sa}^k, \mathbf{u}_{sa}^{k+1}, \lambda_v^k, \mathbf{u}_v^{k+1}, \varphi^k, A_0, V_0\right)$;
    if *exit with convergence of Newton's method* **then**
      /* Update final solution */
      $\mathbf{u}^{k+1} \leftarrow \mathbf{u}_{ns+w}^{k+1} + \lambda_{sa}^{k+1}\mathbf{u}_{sa}^{k+1} + \lambda_v^{k+1}\mathbf{u}_v^{k+1}$;
      $p^{k+1} \leftarrow$ compute_pressure$(p^k, \mathbf{u}^{k+1})$;
      $\varphi^{k+1} \leftarrow$ transport$(\varphi^k, \mathbf{u}^{k+1})$;
    end
  else
    Error, no convergence. Either increase MAXSUBSTEPS or decrease Δt.;
  end
end

Algorithm 9: update_lagrange_multipliers($\mathbf{u}_{ns+w}, \lambda_{sa}^0, \mathbf{u}_{sa}, \lambda_v^0, \mathbf{u}_v, \varphi^0, A_0, V_0$)

Data: $(\mathbf{u}_{ns+w}, \lambda_{sa}^0, \mathbf{u}_{sa}, \lambda_v^0, \mathbf{u}_v, \varphi^0, A_0, V_0)$, MAXNEWTONSTEPS,TOL
Result: $(\lambda_{sa}, \lambda_v)$

begin
  /* Precompute normal velocity terms for Df */
  $(u_{sa})_n \leftarrow \mathcal{I}_{cl}\left(\mathbf{u}_{sa} \cdot \nabla\varphi^0\right)$;
  $(u_v)_n \leftarrow \mathcal{I}_{cl}\left(\mathbf{u}_v \cdot \nabla\varphi^0\right)$;
  /* Begin Newton iteration */
  for $k = 0, \ldots,$ *MAXNEWTONSTEPS* **do**
    $\mathbf{u}^k \leftarrow \mathbf{u}_{ns+w} + \lambda_{sa}^k \mathbf{u}_{sa} + \lambda_v^k \mathbf{u}_v$;
    $\varphi^k \leftarrow$ transport$(\varphi^0, \mathbf{u}^k)$;
    $\varphi^k \leftarrow$ post-process(φ^k);
    $\mathbf{f}\left(\lambda_{sa}^k, \lambda_v^k\right) \leftarrow \begin{pmatrix} \int_\Lambda \delta_\varepsilon\left(\varphi^k\right)\left|\nabla\varphi^k\right| d\mathbf{x} - A_0 \\ \int_\Lambda H_\varepsilon\left(\varphi^k\right) d\mathbf{x} - V_0 \end{pmatrix}$;
    if $|\boldsymbol{f}(\lambda_{sa}^k, \lambda_v^k)| <$*TOL* **then**
      exit with convergence;
    else
      /* Compute $D\mathbf{f}$ as per equation (8.30). */
      $\begin{pmatrix} \lambda_{sa}^{k+1} \\ \lambda_v^{k+1} \end{pmatrix} = \begin{pmatrix} \lambda_{sa}^k \\ \lambda_v^k \end{pmatrix} - D\mathbf{f}(\lambda_{sa}^k, \lambda_v^k)^{-1}\mathbf{f}(\lambda_{sa}^k, \lambda_v^k)$;
    end
  end
  if $|\boldsymbol{f}(\lambda_{sa}^k, \lambda_v^k)| \geq$ *TOL* **then**
    Error, non-convergence of Newton's method
  else
    $(\lambda_{sa}, \lambda_v) \leftarrow (\lambda_{sa}^k, \lambda_v^k)$;
    return $(\lambda_{sa}, \lambda_v)$
  end
end

8.3 Numerical results and validations

To start with, we perform the same 2D simulations with additional surface area (and volume constraints) using the initial surface of an ellipse with 2×1, 4×1 and 8×1 aspect between the major and minor axes as described in the Numerical Results, Section 7.5, of the Willmore Flow, Chapter 7. We then introduce some 3D computations using an ellipsoid shape with similar description of axis length based on relative length in x,y and z directions. The first is a $3 \times 3 \times 1$ ellipsoid which evolves to the standard red blood cell shape. This $3 \times 3 \times 1$ ellipsoid has evolved to steady state shape. We also give the beginnings of the simulations of a $5 \times 5 \times 1$ ellipsoid, a $7 \times 7 \times 1$ ellipsoid and an $8 \times 1 \times 1$ ellipsoid. They are currently ongoing simulations and the full evolution will be included in a future work.

The following computations in 2D are computed using the semi-implicit Canham-Helfrich scheme described in Section 8.2.4 and the 3D simulations are performed using the sub-iterating Canham-Helfrich scheme described in Section 8.2.5. The linear algebra system for the semi-implicit scheme is solved using the direct parallel solver SuperLU_dist [7] version 5.1.2 accessed through the Amesos wrapper class of Trilinos [6] version 12.10.1. The sub-iterating scheme is solved using the GMRes solver and the Algebraic Multi-grid (AMG) preconditioner in Trilinos [6] version 12.10.1. The finite element system is constructed using the deal.II library [3] version 8.5.0 with p4est [5] version 1.1 for handling the completely distributed mesh. Everything is programmed in the C++ language. We use MPI for communicating between the distributed processors. The number of processors is chosen to keep the maximum number of degrees of freedom per processor between 40,000 and 120,000. By doing this, we balance solvability of the linear algebra system and memory footprint per processor with the overhead of passing data between the processors.

8.3.1 The 2×1 ellipse in 2D

We let $\Lambda = [-2, 2] \times [-1.5, 1.5]$ and subdivide our initial mesh with 4 times in the x direction and 3 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the approximate distance function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{2}\right)^2 + \left(\frac{y}{1}\right)^2}. \tag{8.37}$$

We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a timestep of $\Delta t = 0.002$ which is on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the semi-implicit Canham-Helfrich scheme described in Section 7.2 and Algorithm 7 for this computation with 2 MPI processors.

We plot the energy, surface area and volume over time in Figure 8.1. The initial and final shapes plotted using the characteristic function are given in Figure 8.2. The surface has evolved to the shape of a circle. Since we can clearly see in Figure 8.1 that the capsule is losing mass (volume) over time (mainly due to the reinitialization algorithm of the level set equation of Chapter 2.2), we also include the Willmore flow with an explicit volume constraint being enforced. We will introduce the method for doing this in Chapter 8. We

enforce the volume at any given time step to within 10^{-6} of the initial computed volume. The statistics for the evolution of the 2×1 ellipse with explicit volume constraint are given in Figure 8.3 and the shape at times $t = 0$ and $t = 0.5$ are given in Figure 8.4. Since the simulation time is short enough, the final solutions with and without volume constraint are visually the same, as seen in Figure 8.5.

8.3.2 The 4×1 ellipse in 2D

We let $\Lambda = [-4, 4] \times [-2, 2]$ and subdivide our initial mesh with 8 times in the x direction and 4 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set filter parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{4}\right)^2 + \left(\frac{y}{1}\right)^2} \tag{8.38}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a timestep of $\Delta t = 0.002$ which is on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step with $R = 1$ steps of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing reinitialization is $c_L = 0.1$.

We use the semi-implicit Canham-Helfrich scheme described in Section 7.2 and Algorithm 7 for this computation with 4 MPI processors.

The energies for this 4×1 ellipse and the volume and surface area are given in Fig-

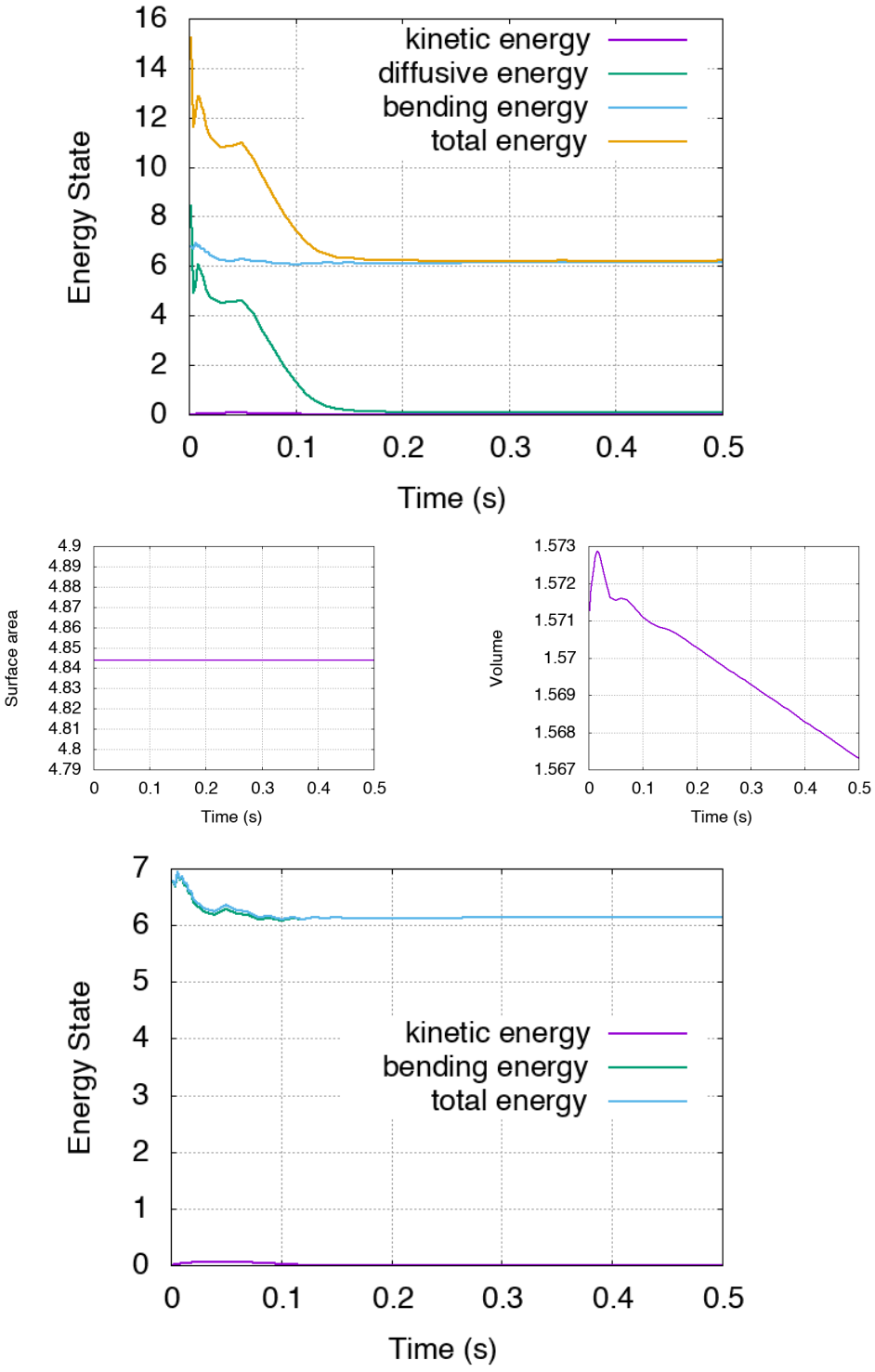


Figure 8.1: The energy, surface area and volume statistics for the 2x1 ellipse with Canham-Helfrich force.

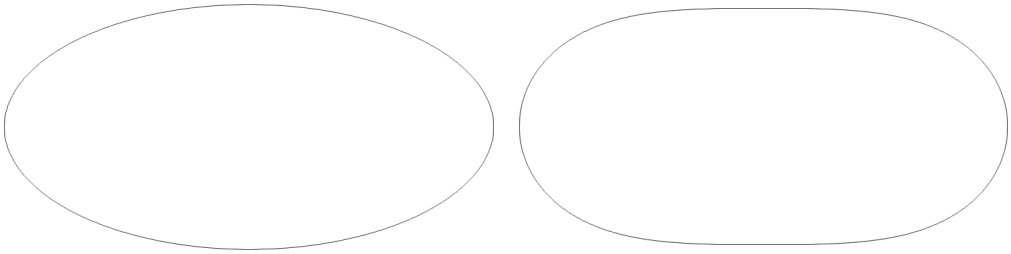


Figure 8.2: The initial and final shapes of 2×1 ellipse with Canham-Helfrich flow at $t = 0$ and $t = 0.5$.

ure 8.6. The initial and final states are given in Figure 8.7.

We also display the energies, surface area and volume for the Willmore flow of the 4×1 ellipse with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.8. The initial and final shapes are displayed in Figure 8.9.A comparison between the solutions with and without constraint at the final time $t = 6.0$ is given in Figure 8.10.

8.3.3 The 8×1 ellipse in 2D

We let $\Lambda = [-6, 6] \times [-3, 3]$ and subdivide our initial mesh with 12 times in the x direction and 6 times in the y direction so that each element is square. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 1.2h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 6 and the minimal refinement level is 2. This corresponds to a minimal mesh size of $h_{\min} = 0.0156125$ and a maximal mesh size of $h_{\max} = 0.25$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{2} - \sqrt{\left(\frac{x}{8}\right)^2 + \left(\frac{y}{1}\right)^2} \tag{8.39}$$

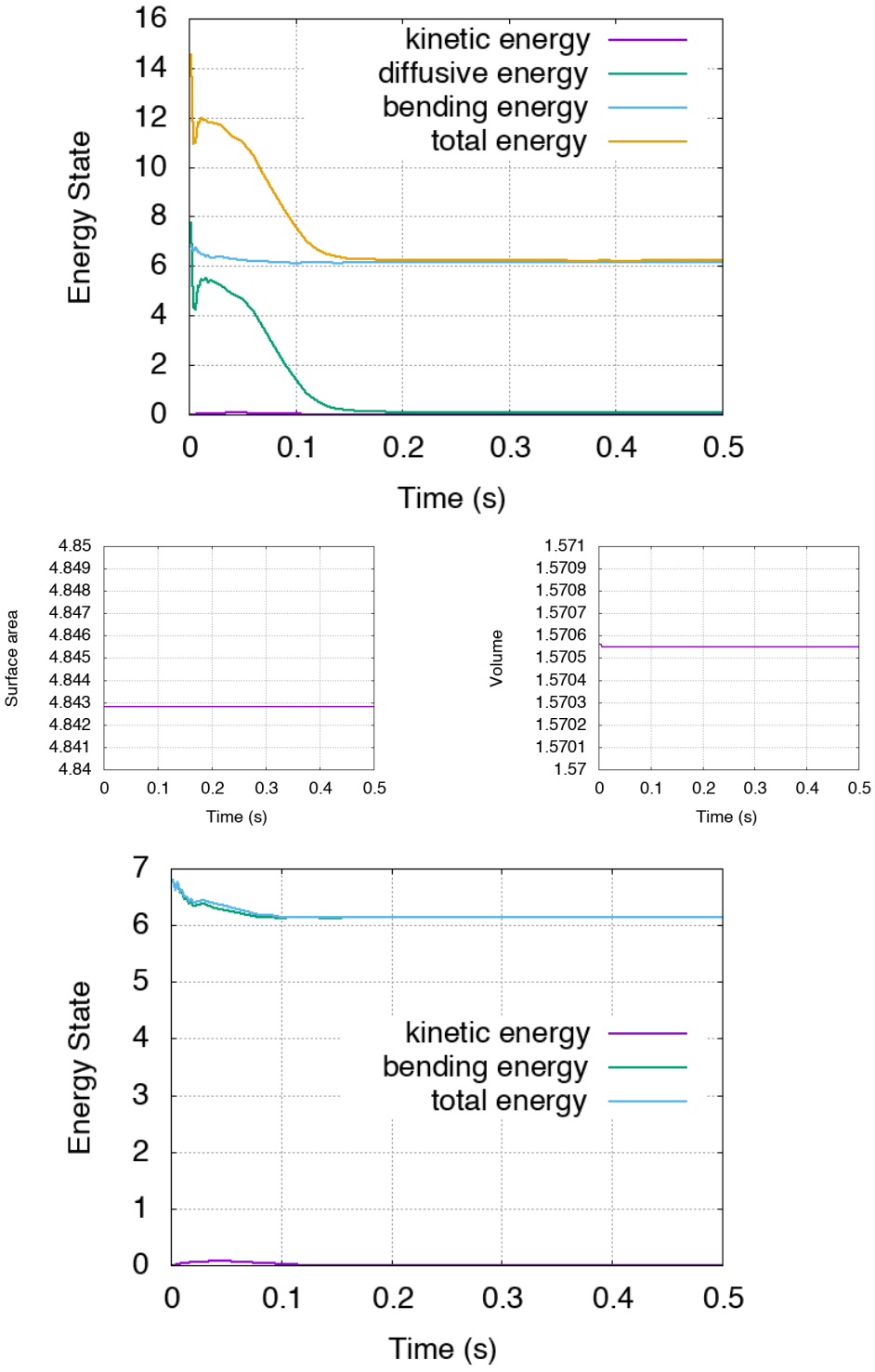


Figure 8.3: The energy, surface area and volume statistics for the 2x1 ellipse with Canham-Helfrich force and explicit volume constraint.

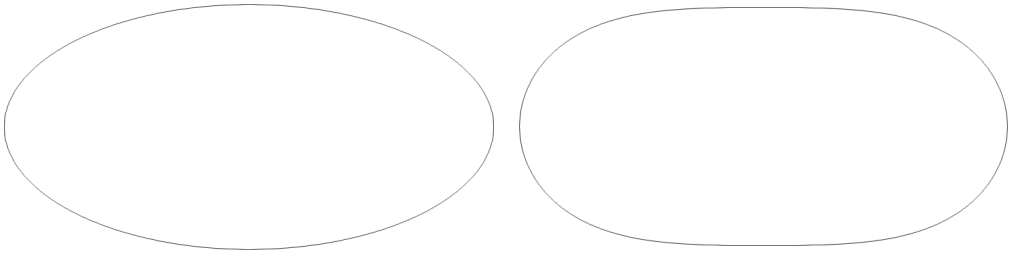


Figure 8.4: The initial and final shapes of 2×1 ellipse with Canham-Helfrich flow and explicit volume constraint at $t = 0$ and $t = 0.5$.

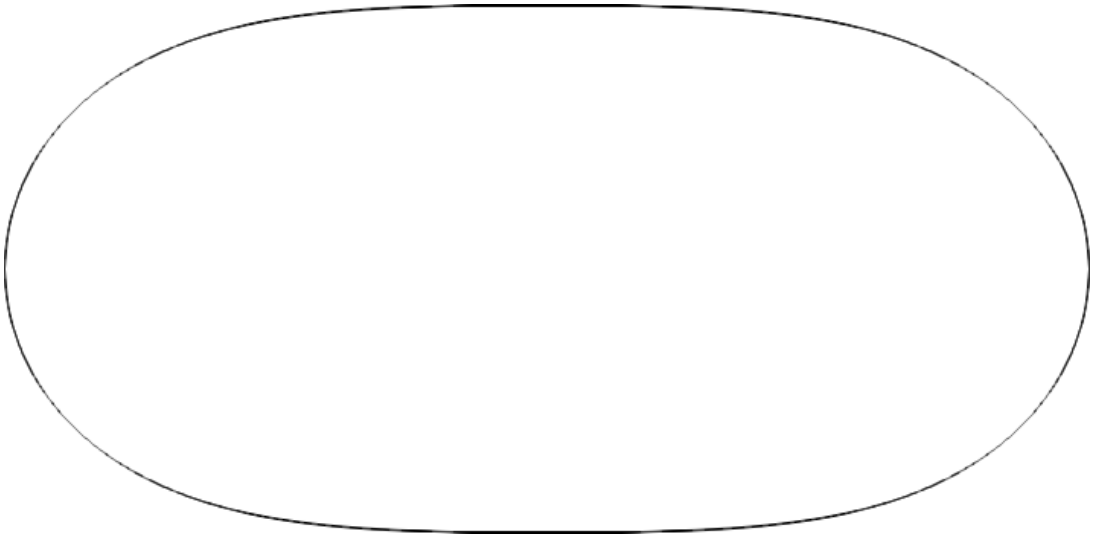


Figure 8.5: The comparison between Canham-Helfrich flow with and without explicit volume constraint does not make a difference when the time length is short enough.

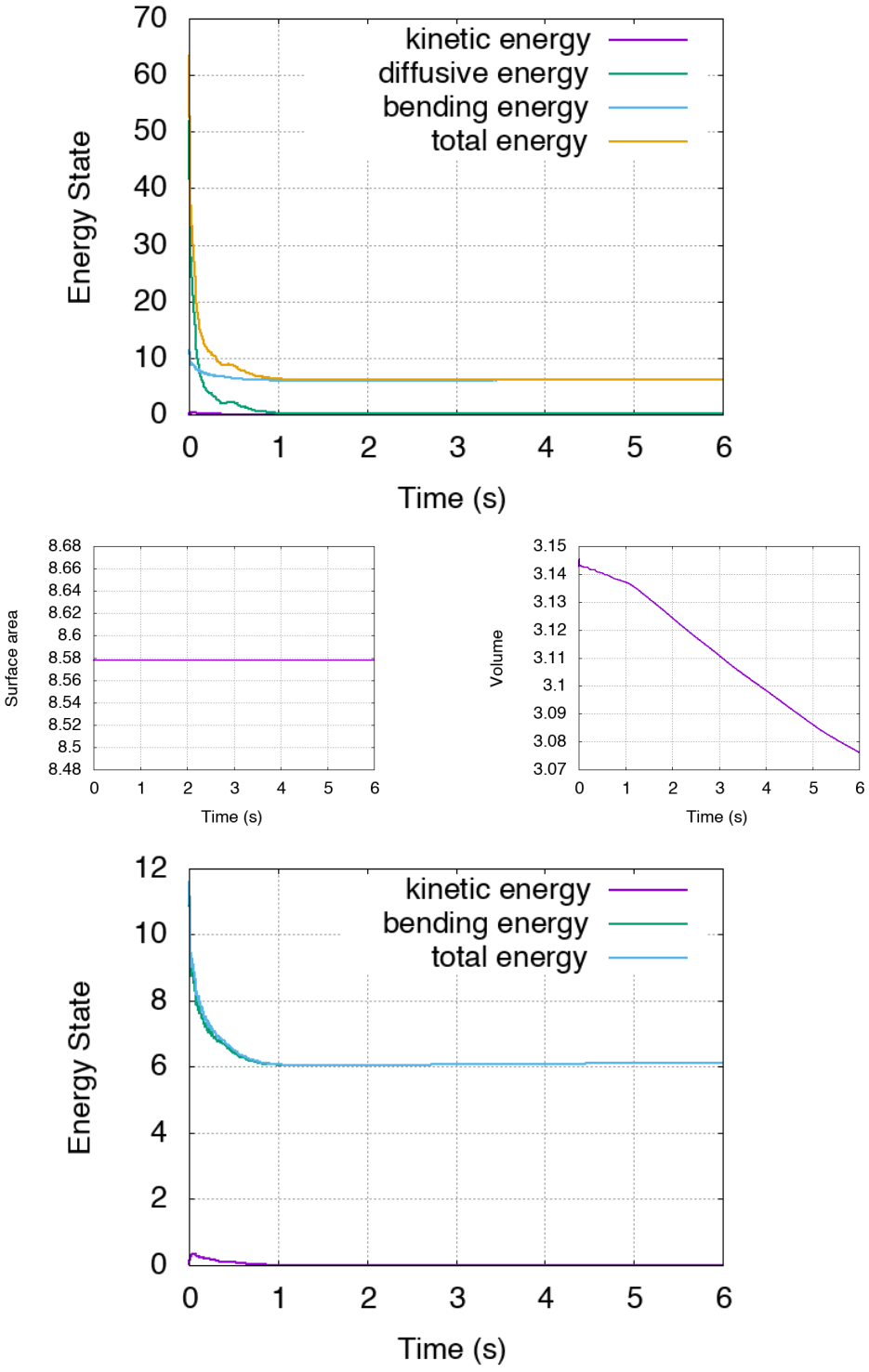


Figure 8.6: The energy, surface area and volume statistics for the 4x1 ellipse with Canham-Helfrich force.

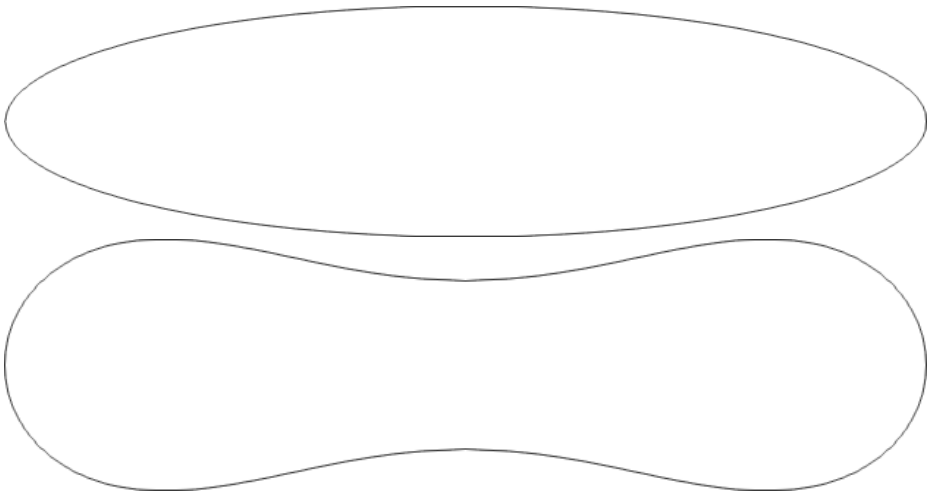


Figure 8.7: The shapes of 4×1 ellipse with Canham-Helfrich flow are displayed at $t = 0$ and $t = 6.0$.

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use a smaller timestep, $\Delta t = 0.001$, than for the previous cases since the velocities are correspondingly larger in the beginning. This is still on the order of $\Delta t \sim h_{\min}^{3/2}$. The Dirac and Heaviside widths are $\varepsilon = h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the semi-implicit Canham-Helfrich scheme described in Section 7.2 and Algorithm 7 for this computation with 4 MPI processors.

The energies for this 8×1 ellipse and the volume and surface area are given in Figure 8.11. The initial and final states are given in Figure 8.12.

We also display the energies, surface area and volume for the Willmore flow of the 8×1 ellipse with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.8. The Heaviside function for the initial, some intermediate and the

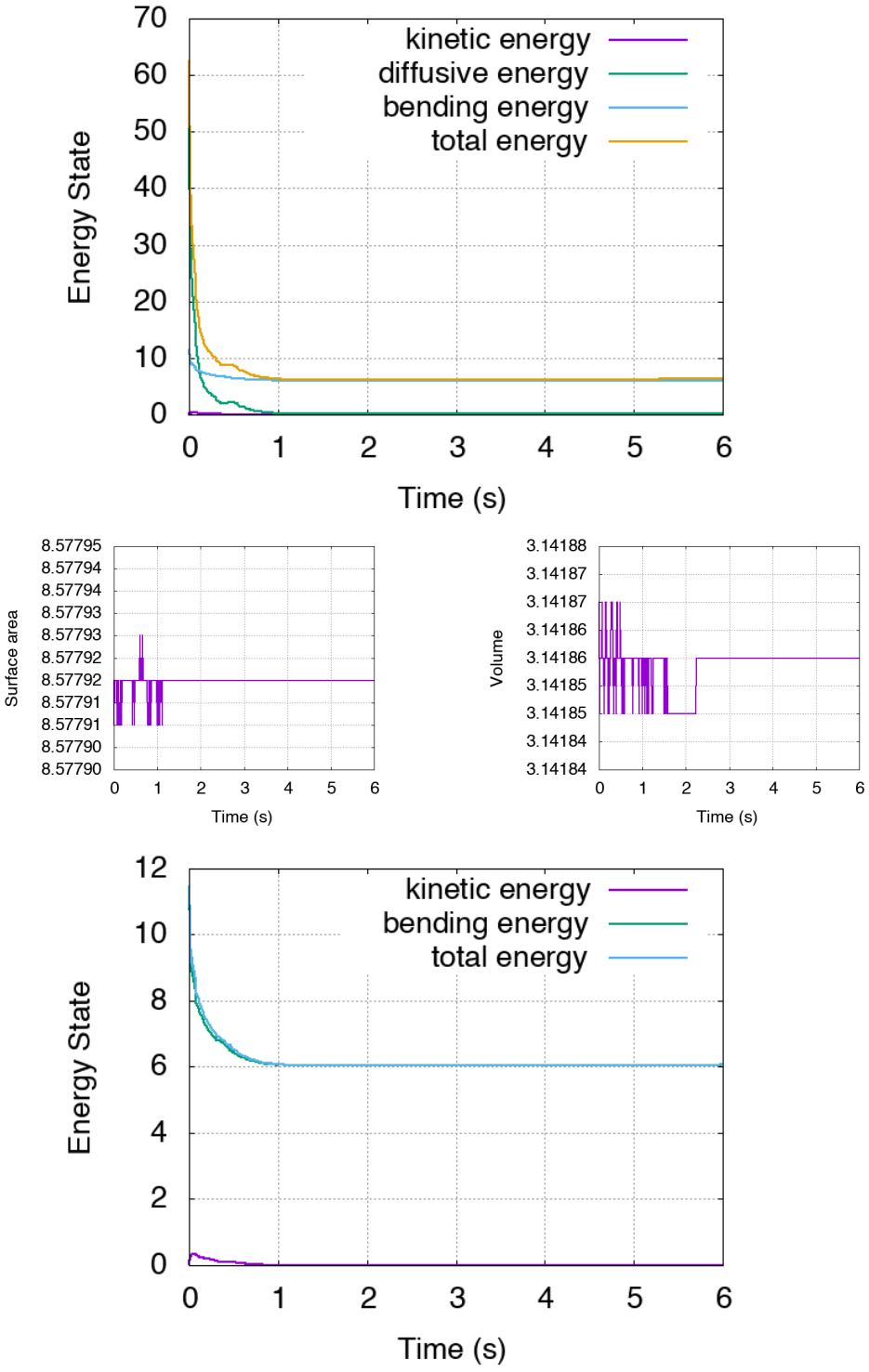


Figure 8.8: The energy, surface area and volume statistics for the 4x1 ellipse with Canham-Helfrich force and explicit volume constraint.

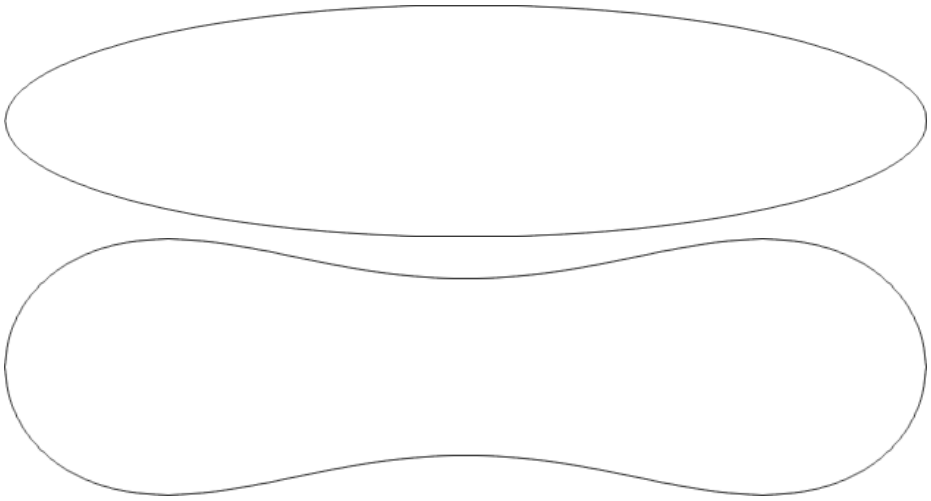


Figure 8.9: The initial and final shapes of 4×1 ellipse with Canham-Helfrich flow with explicit volume constraint at $t = 0$ and $t = 6$.

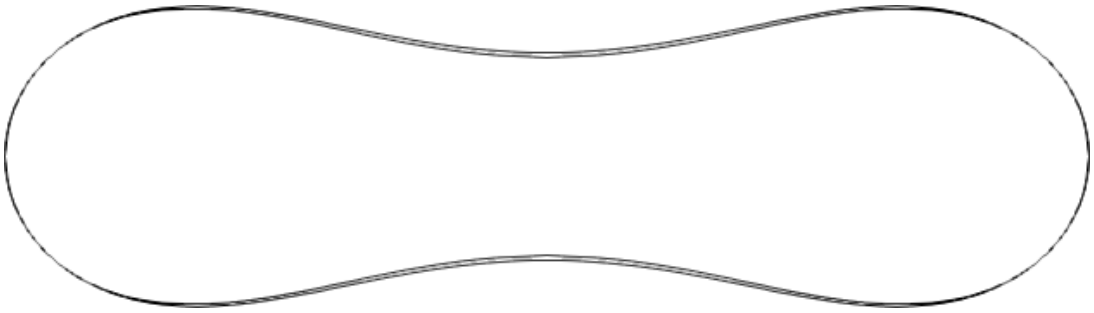


Figure 8.10: A comparison at time $t = 6.0$ between the Canham-Helfrich solutions of 4×1 ellipse with and without explicit volume constraint.

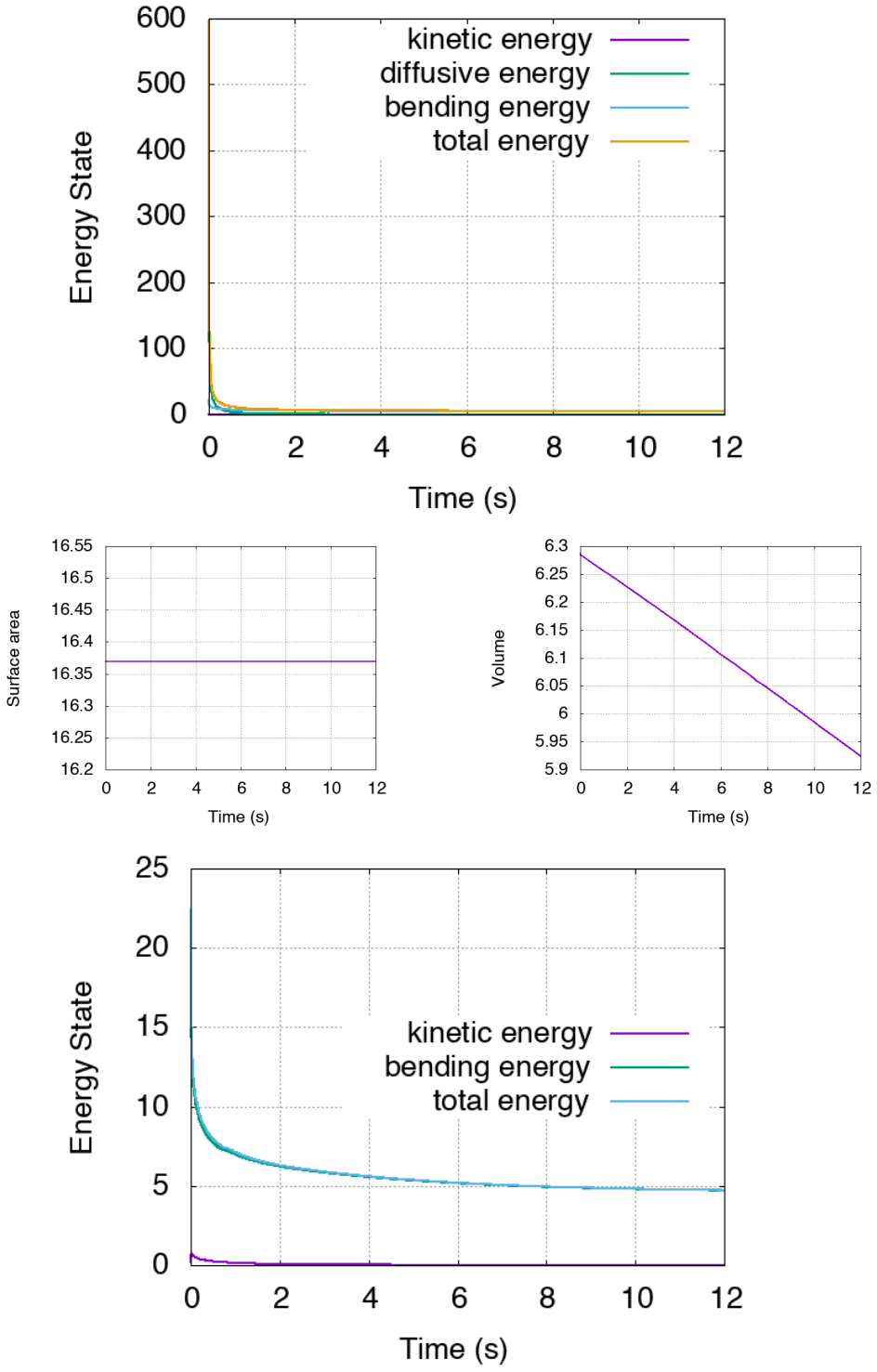


Figure 8.11: The energy, surface area and volume statistics for the 8×1 ellipse with Canham-Helfrich force.

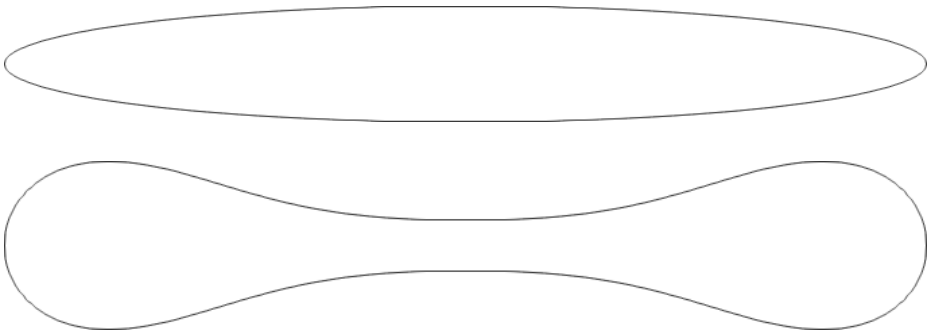


Figure 8.12: The initial and final shapes of 8×1 ellipse with Canham-Helfrich flow at $t = 0.0$ and $t = 12.0$.

final shape are displayed in Figure 8.14. We can see in Figure 8.15 by directly comparing the final states that enforcing the volume constraint has a somewhat smaller effect on the steady state shape for the Canham-Helfrich flow than the Willmore flow (Figure 7.15) but is still important in reaching a steady state shape.

8.3.4 The $3 \times 3 \times 1$ ellipsoid in 3D

We let $\Lambda = [-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 0.6h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 7 and the minimal refinement level is 3. This corresponds to a minimal mesh size of $h_{\min} = 0.0078125$ and a maximal mesh size of $h_{\max} = 0.125$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{10} - \sqrt{\left(\frac{x}{3}\right)^2 + \left(\frac{y}{3}\right)^2 + \left(\frac{z}{1}\right)^2} \tag{8.40}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use the timestep, $\Delta t = 0.0001$ which is somewhere between the order of $\Delta t \sim h_{\min}^{3/2}$ and $\Delta t \sim h_{\min}^2$. The Dirac and Heaviside widths are $\varepsilon = \frac{1}{2}h_{\min}^{3/4}$. We use the level

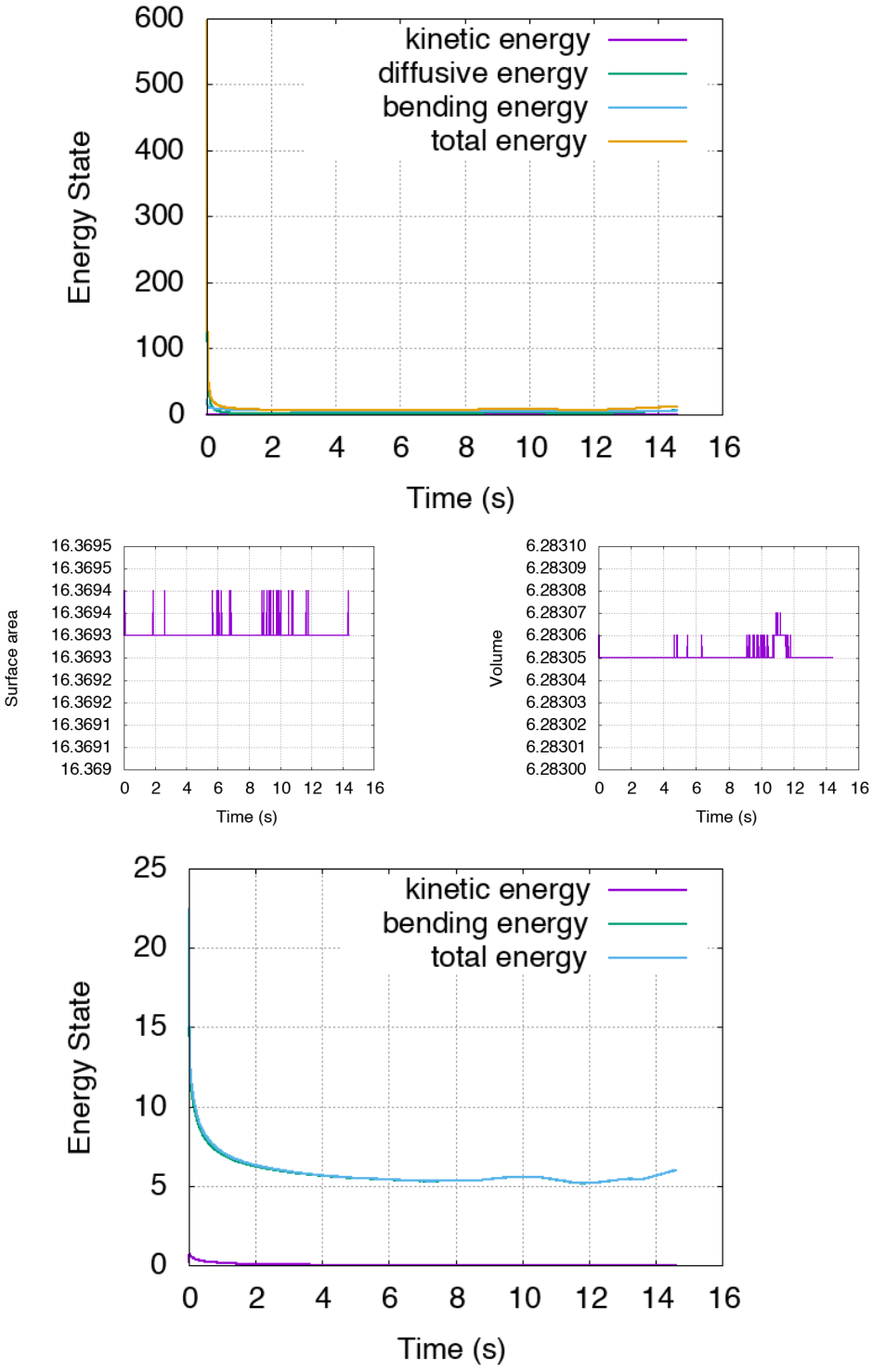


Figure 8.13: The energy, surface area and volume statistics for the 8×1 ellipse with Canham-Helfrich force and explicit volume constraint.

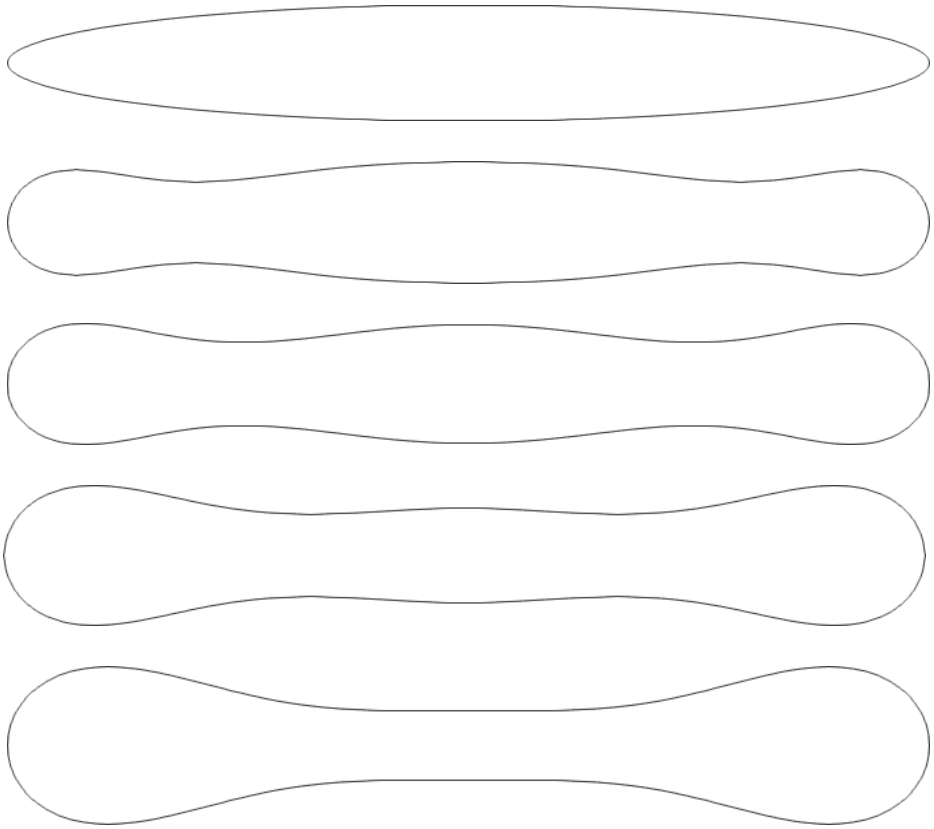


Figure 8.14: The shapes of 8×1 ellipse with Canham-Helfrich flow and explicit volume constraint at (top to bottom) $t = 0.0$, $t = 1.0$, $t = 2.0$ and $t = 5.0$ and $t = 15.0$. By $t = 15.0$ we have reached a steady state shape.

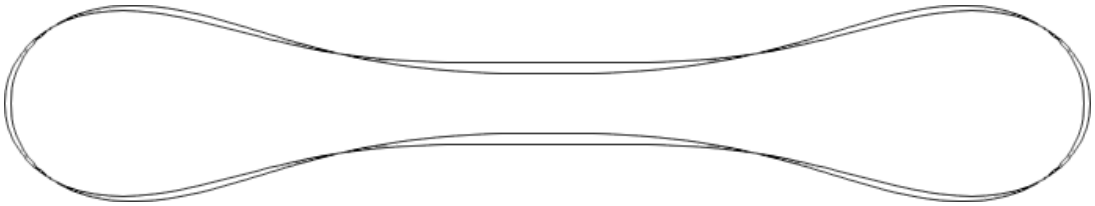


Figure 8.15: A comparison of the final shape at $t = 15.0$ for the 8×1 ellipse with and without the volume constraint being explicitly enforced with the Canham-Helfrich flow. Without enforcing the volume constraint, mass is lost over time leading to a different steady state shape. The wider necked shape has the explicit volume constraint being enforced.

set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the sub-iterating Willmore flow of Section 7.3 with constraints enforced via the Canham-Helfrich algorithm described in Algorithm 8 and use 32 MPI processors for this computation.

The energies for this $3 \times 3 \times 1$ ellipsoid and the volume and surface area with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.18. The Heaviside function for the initial and the final shape are displayed in Figure 8.16. A view of the surface in relation to the full three dimensional mesh is provided in Figure 8.17.

8.3.5 The $5 \times 5 \times 1$ ellipsoid in 3D

We let $\Lambda = [-1.5, 1.5] \times [-1.5, 1.5] \times [-1.0, 1.0]$ which we refine into 3 intervals in the x and y directions and 2 intervals in the z direction so that the initial mesh is composed of 18 cubes with equal length edges. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.90$ and level set parameters $c_d = 0.6h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 7 and the minimal refinement level is 3.

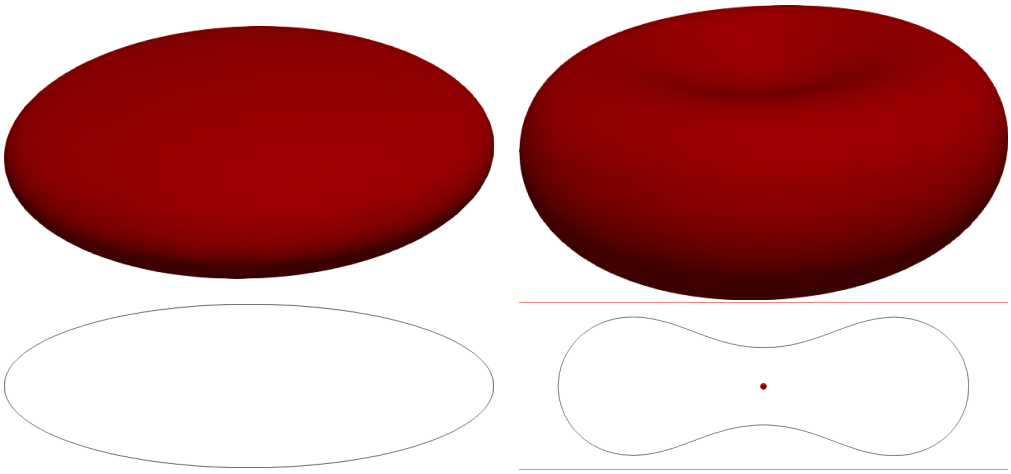


Figure 8.16: The initial and final shapes of $3 \times 3 \times 1$ ellipsoid with Canham-Helfrich flow and explicit volume constraint at $t = 0.0$ and $t = 0.112$.

This corresponds to a minimal mesh size of $h_{\min} = 0.0078125$ and a maximal mesh size of $h_{\max} = 0.125$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{3}{20} - \sqrt{\left(\frac{x}{5}\right)^2 + \left(\frac{y}{5}\right)^2 + \left(\frac{z}{1}\right)^2} \tag{8.41}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use the timestep, $\Delta t = 0.0001$ which is somewhere between the order of $\Delta t \sim h_{\min}^{3/2}$ and $\Delta t \sim h_{\min}^2$. The Dirac and Heaviside widths are $\varepsilon = \frac{1}{2} h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla \varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the sub-iterating Willmore flow of Section 7.3 with constraints enforced via the Canham-Helfrich algorithm described in Algorithm 8 and use 128 MPI processors for

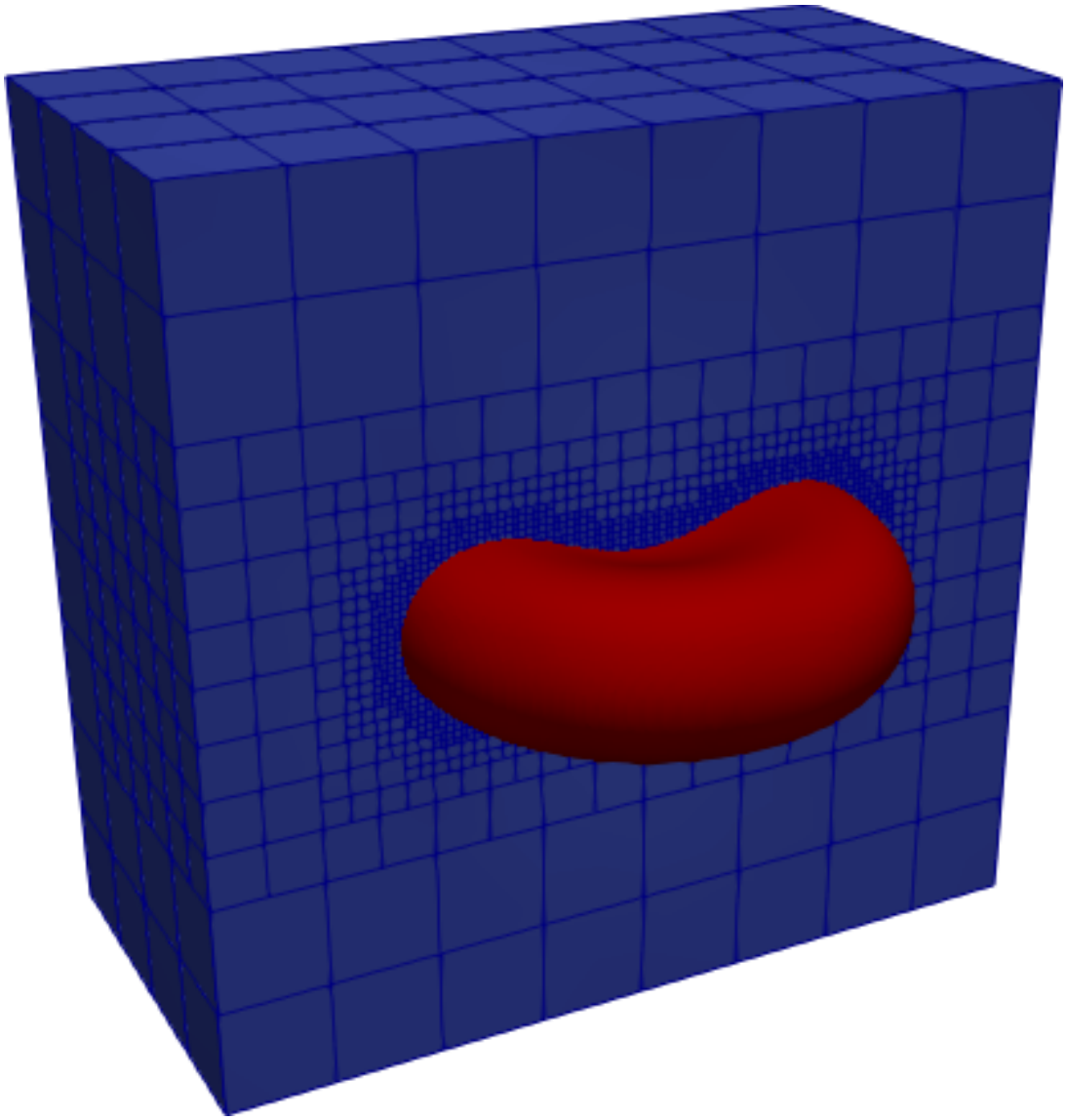


Figure 8.17: A view of the surface, Γ, evolution of $3 \times 3 \times 1$ ellipsoid at $t = 0.112$ with respect to the cut away bulk mesh.

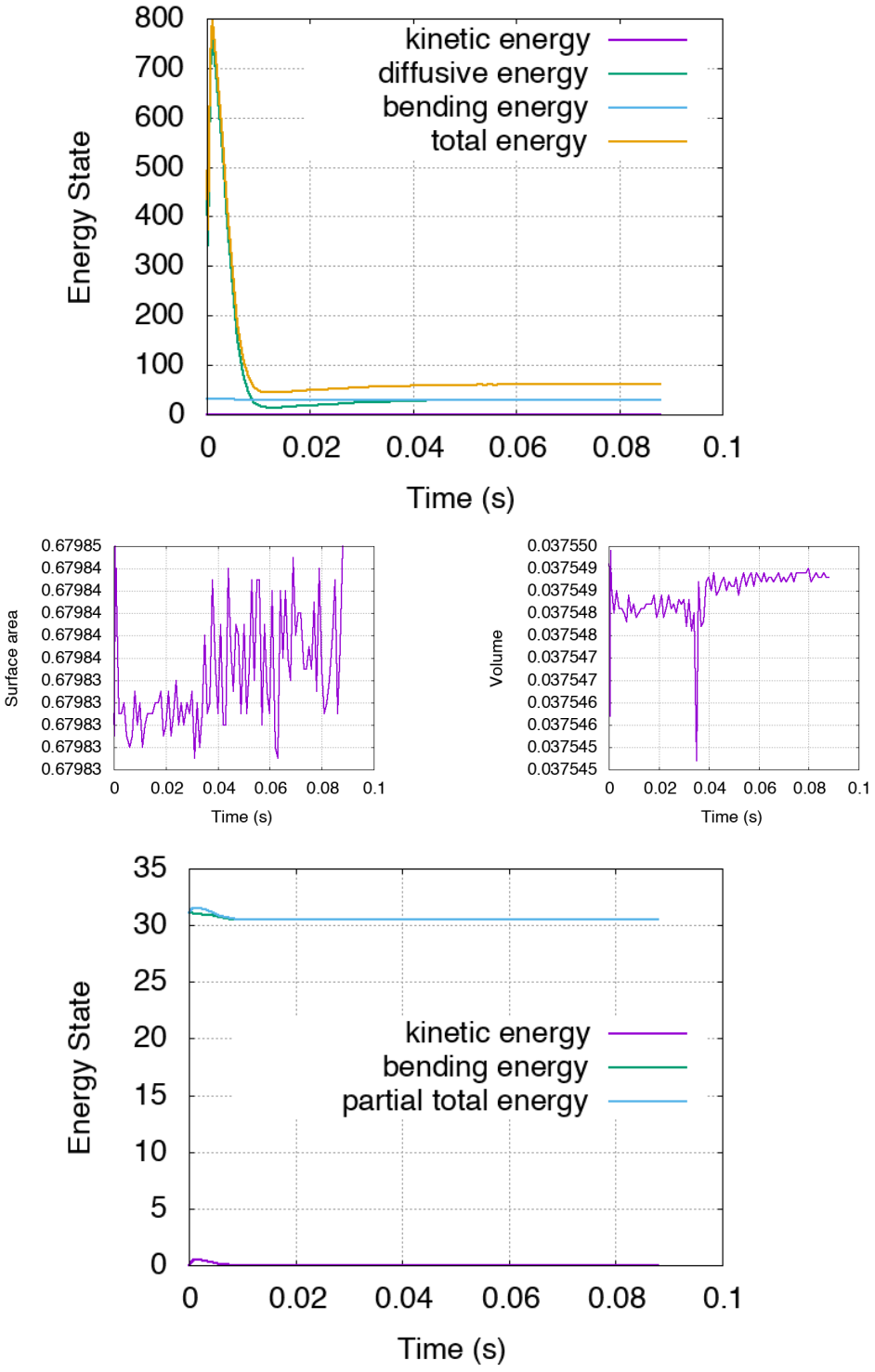


Figure 8.18: The energy, surface area and volume statistics for the $3 \times 3 \times 1$ ellipsoid with Canham-Helfrich force and explicit volume constraint.

this computation.

The energies for this $5 \times 5 \times 1$ ellipsoid with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.19. The profile for the initial and an intermediate timestep $t = 0.0159$ are displayed in Figure 8.20. A view of the surface at this time $t = 0.0159$ in relation to the full three dimensional mesh is provided in Figure 8.21.

8.3.6 The $7 \times 7 \times 1$ ellipsoid in 3D

We let $\Lambda = [-1.5, 1.5] \times [-1.5, 1.5] \times [-1.0, 1.0]$ which we refine into 3 intervals in the x and y directions and 2 intervals in the z direction so that the initial mesh is composed of 18 cubes with equal length edges. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.90$ and level set parameters $c_d = 0.6h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 7 and the minimal refinement level is 3. This corresponds to a minimal mesh size of $h_{\min} = 0.0078125$ and a maximal mesh size of $h_{\max} = 0.125$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{3}{20} - \sqrt{\left(\frac{x}{7}\right)^2 + \left(\frac{y}{7}\right)^2 + \left(\frac{z}{1}\right)^2} \tag{8.42}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use the timestep, $\Delta t = 0.0001$ which is somewhere between the order of $\Delta t \sim h_{\min}^{3/2}$ and $\Delta t \sim h_{\min}^2$. The Dirac and Heaviside widths are $\varepsilon = \frac{1}{2}h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the sub-iterating Willmore flow of Section 7.3 with constraints enforced via the Canham-Helfrich algorithm described in Algorithm 8 and use 128 MPI processors for

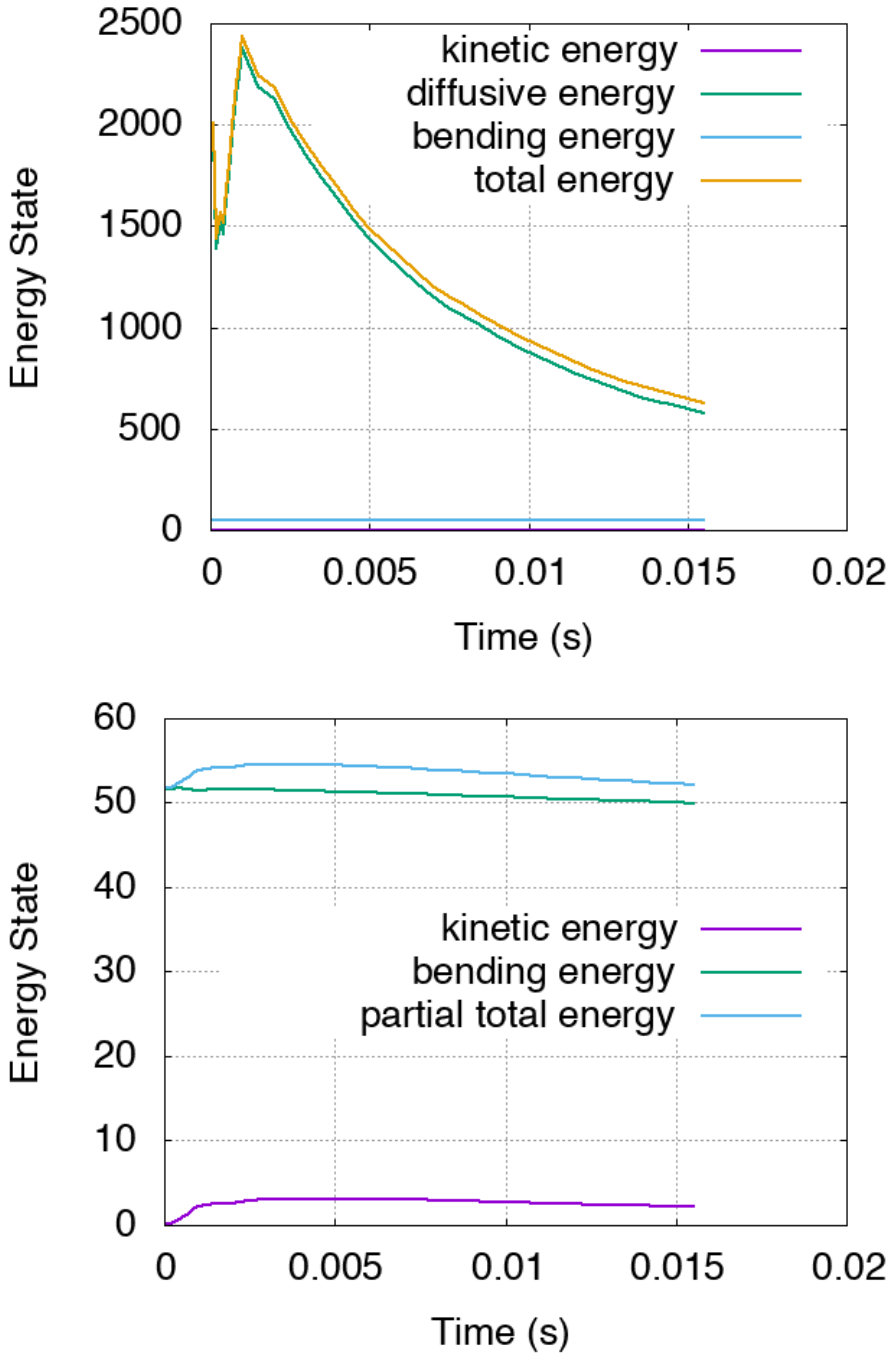


Figure 8.19: The energy statistics for the $5 \times 5 \times 1$ ellipsoid with Canham-Helfrich force and explicit volume constraint. We see that the diffusive energy dominates this portion of the simulation whereas the bending energy is much smaller, but still decreasing.

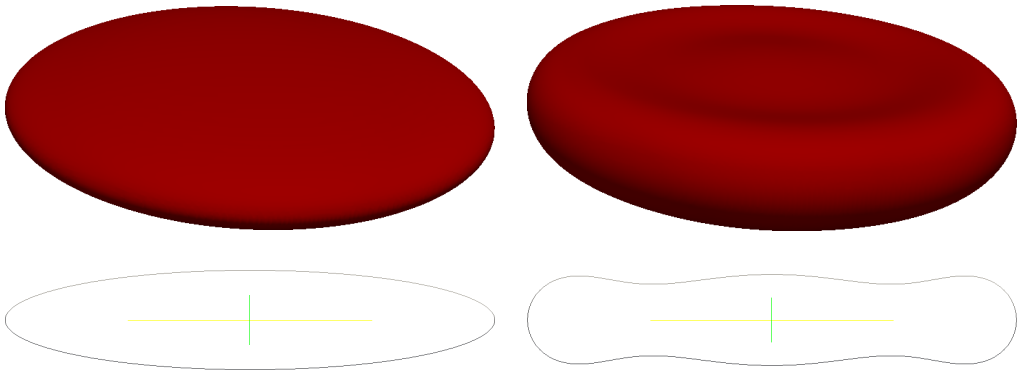


Figure 8.20: The initial and intermediate shapes of $5 \times 5 \times 1$ ellipsoid with Canham-Helfrich flow at $t = 0.0$ and $t = 0.0159$.

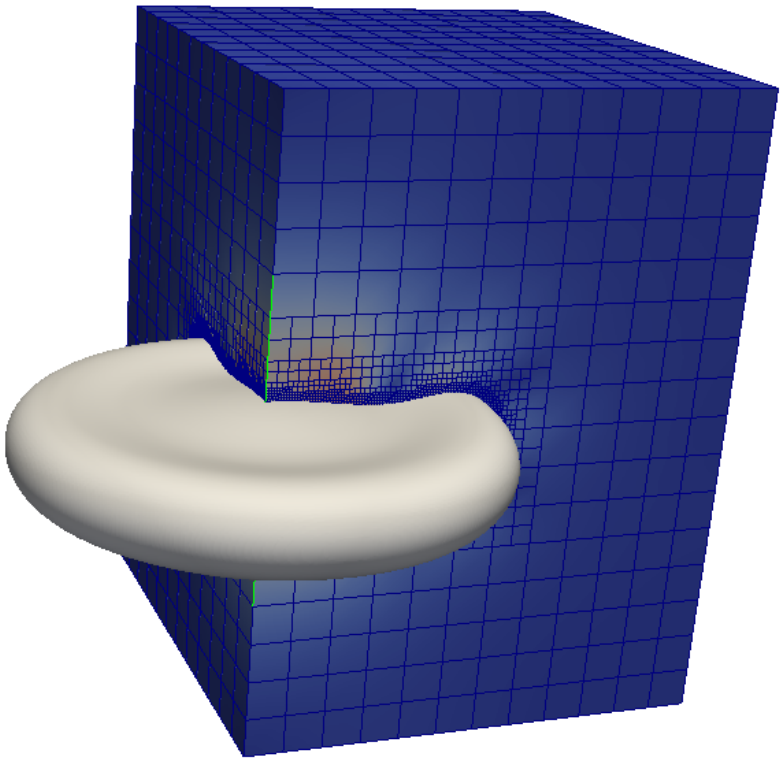


Figure 8.21: A view of the surface, Γ, evolution of $5 \times 5 \times 1$ ellipsoid at $t = 0.1$ with respect to the cut away bulk mesh.

this computation.

The energies for this $7 \times 7 \times 1$ ellipsoid with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.22. The profile for the initial and an intermediate timestep $t = 0.0133$ are displayed in Figure 8.23. A view of the surface at this time $t = 0.0133$ in relation to the full three dimensional mesh is provided in Figure 8.24 and the velocity field at the same time along a slice of the surface is given in Figure 8.25.

8.3.7 The $8 \times 1 \times 1$ ellipsoid in 3D

We let $\Lambda = [-1.5, 1.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$ which we refine into 3 intervals in the x direction so that the initial mesh is composed of 3 cubes with equal length edges. We apply 4 initial refinements and then adaptively refine using the method of Section 2.3 with $c_a = 0.95$ and level set parameters $c_d = 0.8h_{\min}^{3/4}$ and $c_f = 2h_{\min}$ until the maximal refinement level is 7 and the minimal refinement level is 3. This corresponds to a minimal mesh size of $h_{\min} = 0.0078125$ and a maximal mesh size of $h_{\max} = 0.125$. The starting levelset is initialized using the function

$$\varphi_{\text{initial}} = \frac{1}{8} - \sqrt{\left(\frac{x}{8}\right)^2 + \left(\frac{y}{1}\right)^2 + \left(\frac{z}{1}\right)^2} \tag{8.43}$$

to approximate the distance function. We apply 25 timesteps of the levelset reinitialization algorithm with linear viscosity coefficient $c_L = 0.2$ to obtain the starting levelset equation. We use the timestep, $\Delta t = 0.0005$ which is somewhere between the order of $\Delta t \sim h_{\min}^{3/2}$ and $\Delta t \sim h_{\min}^2$. The Dirac and Heaviside widths are $\varepsilon = 0.6h_{\min}^{3/4}$. We use the level set reinitialization scheme as a post processing step after transport with $R = 1$ step of reinitialization after every $T = 1$ steps of transport. This is applied to keep $|\nabla\varphi|$ smooth and close to 1 on the fully refined band. The linear viscosity coefficient for this post processing is $c_L = 0.1$.

We use the sub-iterating Willmore flow of Section 7.3 with constraints enforced via

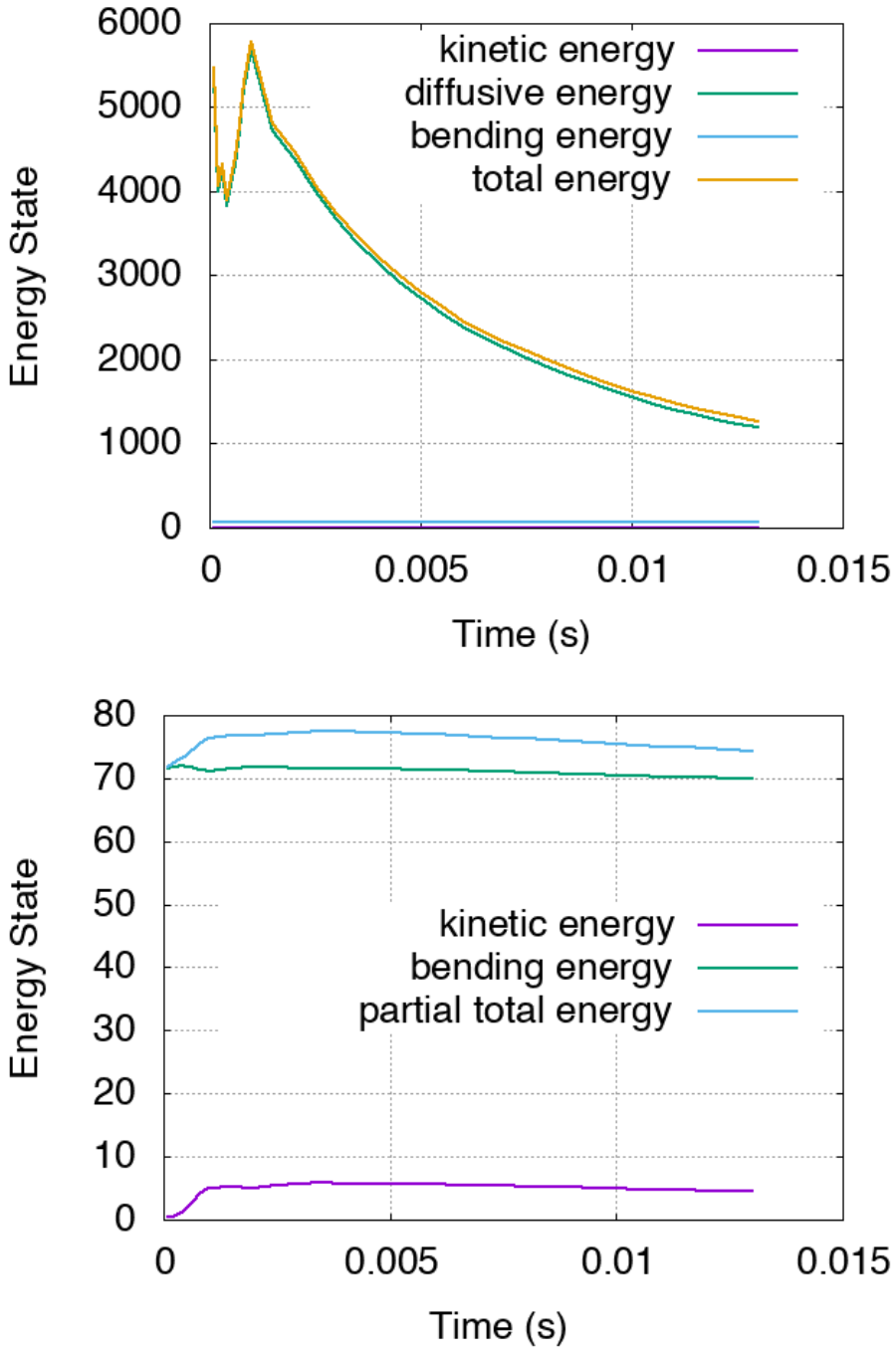


Figure 8.22: The energy statistics for the $7 \times 7 \times 1$ ellipsoid with Canham-Helfrich force and explicit volume constraint. We see that the diffusive energy dominates this portion of the simulation whereas the bending energy is much smaller, but still decreasing.

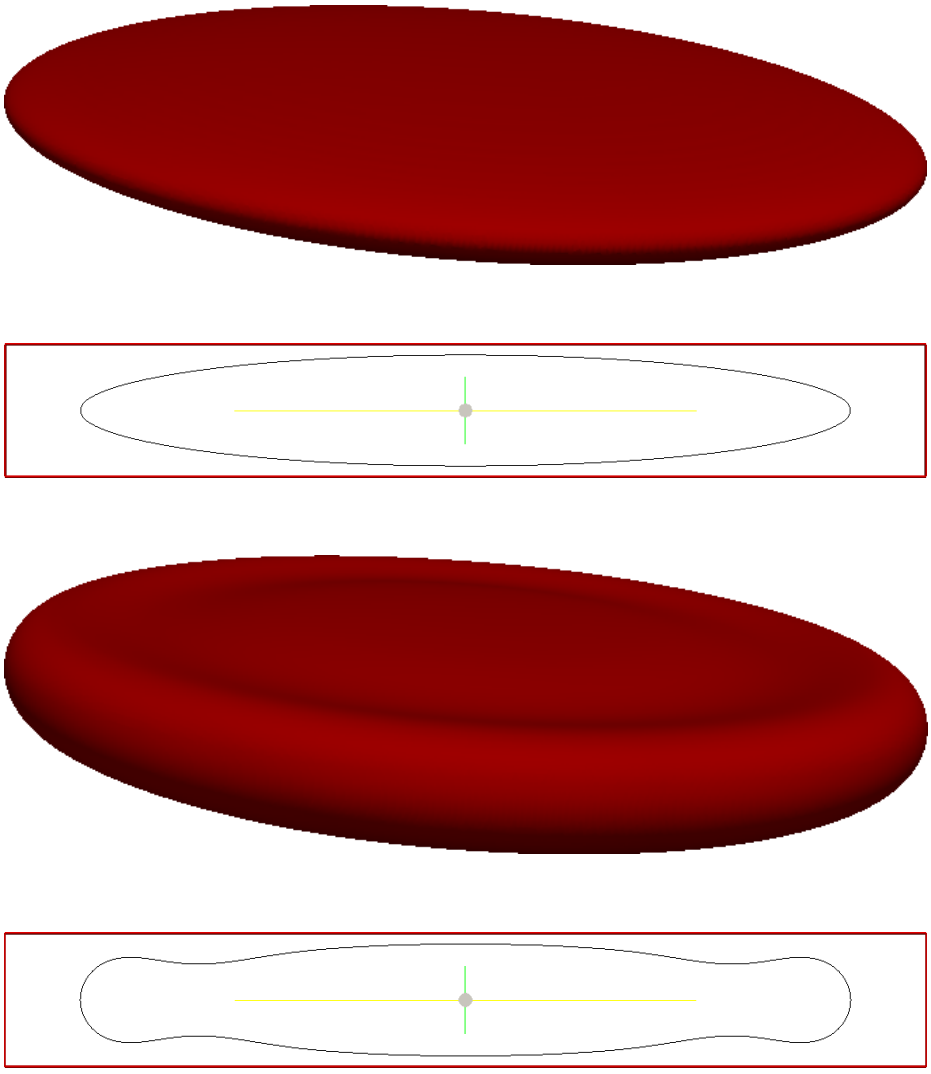


Figure 8.23: The initial and intermediate shapes of $7 \times 7 \times 1$ ellipsoid with Canham-Helfrich flow and explicit volume constraint at $t = 0.0$ and $t = 0.0133$.

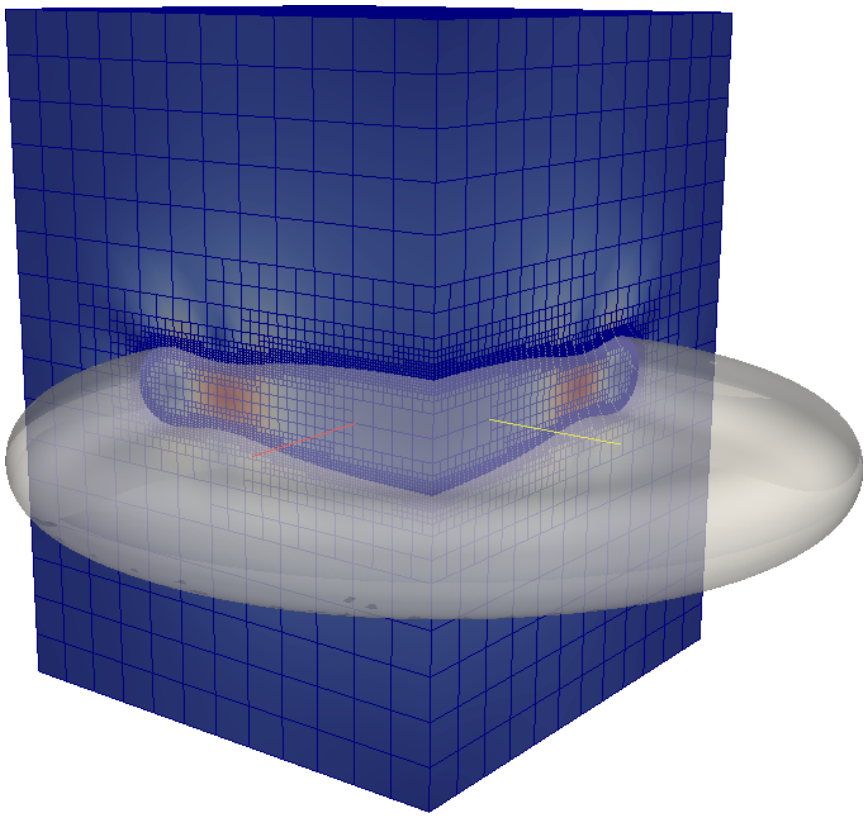


Figure 8.24: A view of the surface, Γ, evolution of $7 \times 7 \times 1$ ellipsoid at $t = 0.0133$ with respect to the cut away bulk mesh.

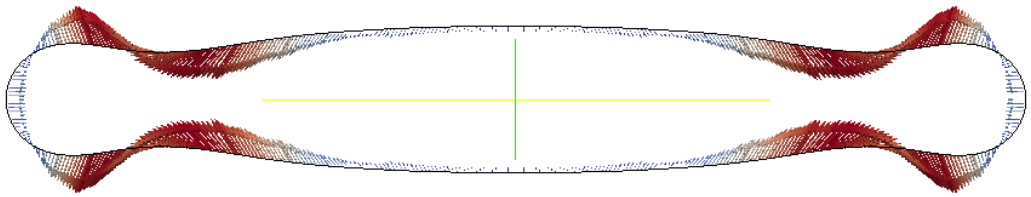


Figure 8.25: A view of the velocity field for a slice of surface, Γ, an evolution of $7 \times 7 \times 1$ ellipsoid at $t = 0.0133$.

the Canham-Helfrich algorithm described in Algorithm 8 and use 64 MPI processors for this computation.

The energies for this $8 \times 1 \times 1$ ellipsoid with the volume constraint explicitly enforced to accuracy of 10^{-6} as per Chapter 8 in Figure 8.26. The profile for the initial and an intermediate timestep $t = 0.0122$ are displayed in Figure 8.27. A view of the surface at this time $t = 0.0122$ in relation to the full three dimensional mesh is provided in Figure 8.28 and the velocity field at the same time along a slice of the surface is given in Figure 8.29.

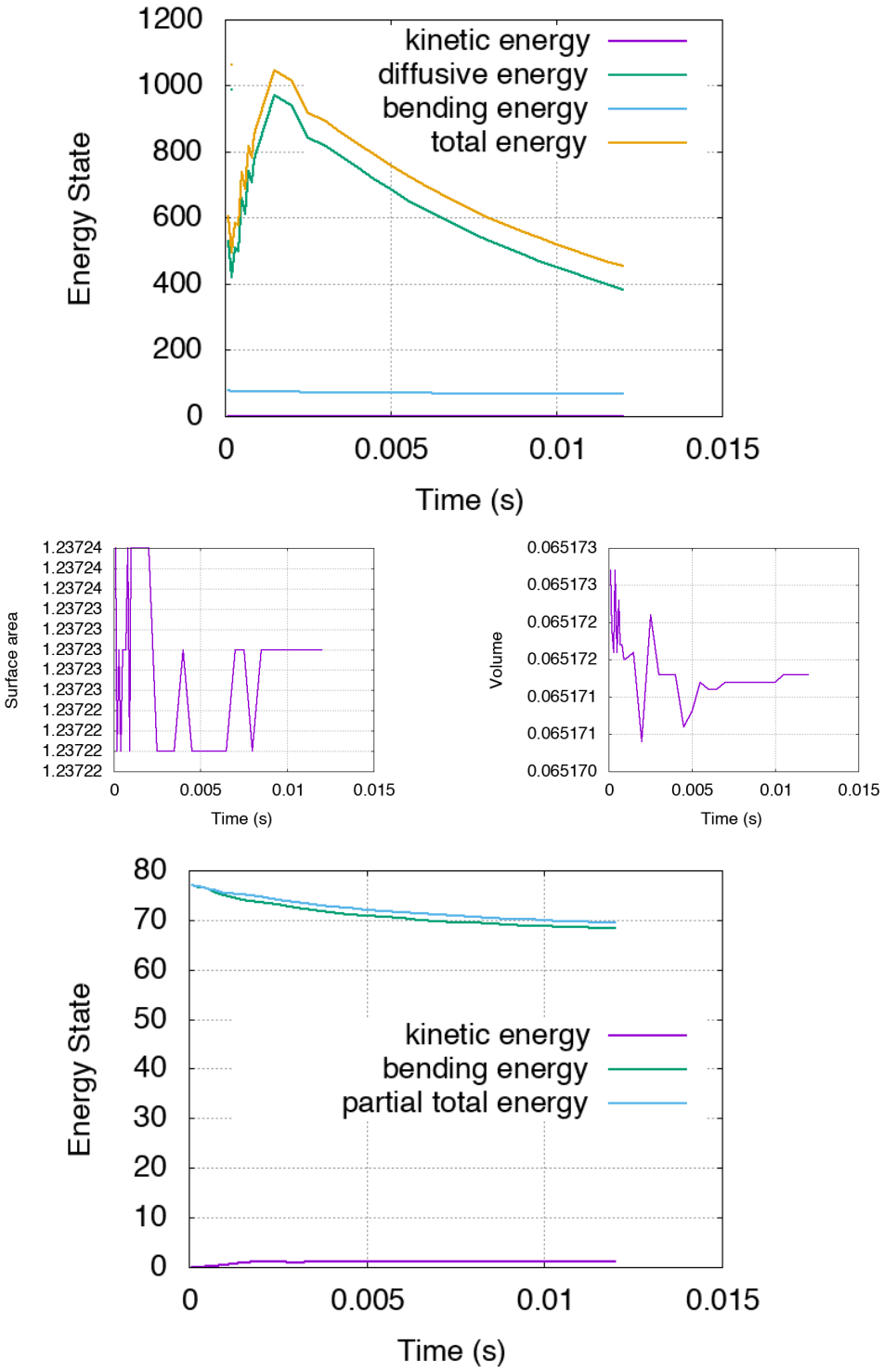


Figure 8.26: The energy statistics for the $8 \times 1 \times 1$ ellipsoid with Canham-Helfrich force and explicit volume constraint. We see that the diffusive energy dominates this portion of the simulation whereas the bending energy is much smaller, but still decreasing.

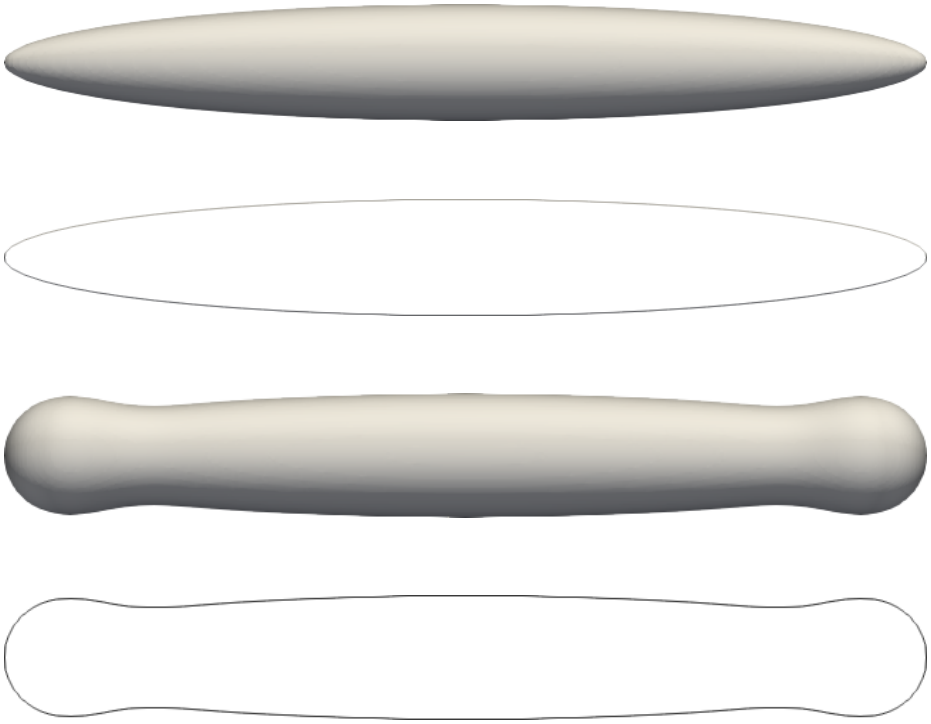


Figure 8.27: The initial and intermediate shapes of $8 \times 1 \times 1$ ellipsoid with Canham-Helfrich flow and explicit volume constraint at $t = 0.0$ and $t = 0.0122$.

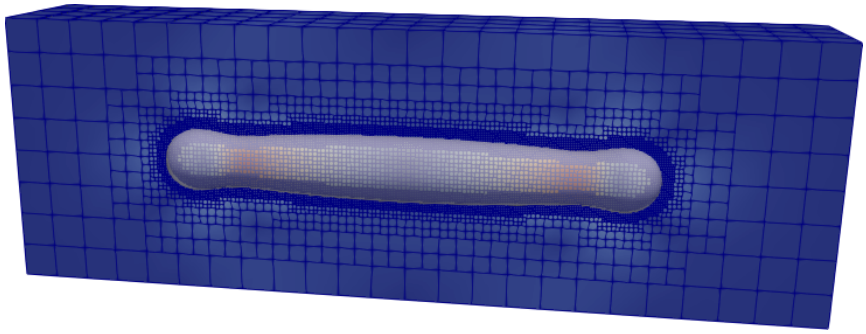


Figure 8.28: A view of the surface, Γ, evolution of $8 \times 1 \times 1$ ellipsoid at $t = 0.0122$ with respect to the cut away bulk mesh.

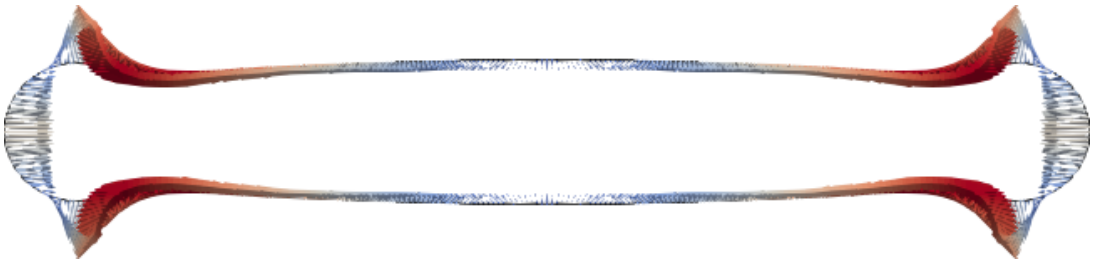


Figure 8.29: A view of the velocity field for a slice of surface, Γ, an evolution of $8 \times 1 \times 1$ ellipsoid at $t = 0.0122$.

9. CONCLUSIONS

We have developed a model for an incompressible Navier-Stokes two-phase flow coupled with physics on the interface between the two-phases. In particular, we have described a method of coupling the forces minimizing an energy defined on the interface to the Navier-Stokes system. The goal has been to provide a model for the dynamics of a blood cell which balances the Canham-Helfrich energy on the vesicle membrane with the dynamics of the fluid itself. We have described a method for generating a velocity field that balances these forces and a method for transporting the interface by that velocity field. In addition we have provided a method for ensuring that the surface area of the interface and the volume of the interior are preserved over time. We have provided various 2 dimensional and 3 dimensional simulations with the Canham-Helfrich energy as well as provided simulations of the two-phase flow with surface tension and with the Willmore energy coupled into the flow. We have discovered that conserving the volume is necessary for obtaining consistent steady state shapes of the vesicle.

We have introduced two approaches to discretizing the force balance of the Navier-Stokes system with the minimization of the Willmore energy. The first is a semi-implicit method (Section 7.2) and so far works well in 2 dimensions but is not yet suitable for simulations in 3 dimensions due to the block linear algebra system being difficult to solve numerically. The second (Section 7.3) performs a subiteration loop for each time step which approximates the fully implicit system. It is well solvable in 3 dimensions at the expense of more work per step due to the subiteration. In addition, since the forces are treated explicitly, the computation of the curvature along the interface is highly dependent on the surface being numerically smooth and so it has been necessary to perturb the curvature system by adding a first order tangential smoothing term. We hypothesize that a

hybrid model combining the semi-implicit algorithm described in Section 7.2 for computing the curvature with the subiteration scheme could resolve this issue and provide a fully consistent model solvable in both 2 and 3 dimensions. There is also room for improvement to the reinialization procedure (Section 8.5) of the levelset function for preserving $|\nabla\varphi| = 1$ in the ε-band around Γ.

The simulations of our application chapters, Chapter 6-8 have provided a few insights and observations which we will iterate here. First, the enforcement of the volume constraint is absolutely necessary to obtain consistent steady state shapes as seen best in Figure 7.15. This is important for longer term simulations but is important for any simulation especially in 3 dimensions where the volume can be lost quickly.

We are interested to see that the Willmore flow with volume constraint can in fact get stuck in local minima as seen in the 4×1 and 8×1 simulations of Section 7.5. It will require some more research to understand how this could be overcome or if these local minima occur naturally.

We hope that our Canham-Helfrich simulations of the $5 \times 5 \times 1$ or $7 \times 7 \times 1$ or $8 \times 1 \times 1$ ellipsoids will lead to the membrane self-intersecting. We are interested to see what happens since the levelset can in theory handle the topological changes. It is difficult to predict what would happen since we have used the Gauss-Bonnett theorem to drop the topological invariant, $\int_\Gamma \kappa d\mathbf{x}$, from the Canham-Helfrich energy as described in the introduction of Chapter 7.

Finally, in [74], the discussion of how the reduced volume parameter (equation 8.5) relates to the steady state shapes of minimal bending energy motives the search for initial surfaces that could lead to other steady state shapes observed in nature. Along these lines, it will also be necessary to verify that the the existing models match the predicted shapes based on their reduced volume.

Finally there are many directions of research that could be taken from this model to

add various components of the biological cells for instance, chemical signals to model chemotaxis or the actin-myosin structure of the cytoplasm for modelling cell blebbing. In addition, the simulation of more complex environments possibly with multiple structures and external flow of the fluid would be an interesting application of this basic model of a cell.

REFERENCES

[1] P. B. Canham, "The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell," *Journal of theoretical biology*, vol. 26, no. 1, pp. 61IN777–76IN881, 1970.

[2] W. Helfrich, "Elastic properties of lipid bilayers: theory and possible experiments," *Zeitschrift für Naturforschung C*, vol. 28, no. 11-12, pp. 693–703, 1973.

[3] W. Bangerth, R. Hartmann, and G. Kanschat, "deal.II – a general purpose object oriented finite element library," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 24/1–24/27, 2007.

[4] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, *et al.*, "Open mpi: Goals, concept, and design of a next generation mpi implementation," in *European Parallel Virtual Machine/Message Passing Interface UsersâĂŹ Group Meeting*, pp. 97–104, Springer, 2004.

[5] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[6] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[7] X. Li, J. Demmel, J. Gilbert, iL. Grigori, M. Shao, and I. Yamazaki, "SuperLU Users' Guide," Tech. Rep. LBNL-44289, Lawrence Berkeley National Laboratory, September 1999. `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011.

[8] Brazos Computational Resource, Academy for Advanced Telecommunications and Learning Technologies, Texas A&M University.

[9] C. W. Hirt and B. D. Nichols, "Volume of fluid (vof) method for the dynamics of free boundaries," *Journal of computational physics*, vol. 39, no. 1, pp. 201–225, 1981.

[10] J. E. Welch, F. H. Harlow, J. P. Shannon, and B. J. Daly, "The mac method-a computing technique for solving viscous, incompressible, transient fluid-flow problems involving free surfaces," tech. rep., Los Alamos Scientific Lab., Univ. of California, N. Mex., 1965.

[11] C. S. Peskin, "The immersed boundary method," *Acta numerica*, vol. 11, pp. 479–517, 2002.

[12] S. Leung and H. Zhao, "A grid based particle method for moving interface problems," *Journal of Computational Physics*, vol. 228, no. 8, pp. 2993–3024, 2009.

[13] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.

[14] G. J. Fix, "Phase field methods for free boundary problems," 1982.

[15] J. Langer, "Models of pattern formation in first-order phase transitions," *Directions in Concensed Matter Physics: Memorial Volume in Honor of Shang-Keng Ma. Edited by GRINSTEIN S ET AL. Published by World Scientific Publishing Co. Pte. Ltd., 1986. ISBN# 9789814415309, pp. 165-186*, pp. 165–186, 1986.

[16] J. W. Barrett, H. Garcke, and R. Nürnberg, "A stable parametric finite element discretization of two-phase navier–stokes flow," *Journal of Scientific Computing*, vol. 63, no. 1, pp. 78–117, 2015.

[17] J. W. Barrett, H. Garcke, and R. Nürnberg, "Phase field models versus parametric front tracking methods: Are they accurate and computationally efficient?," *Communications in Computational Physics*, vol. 15, no. 02, pp. 506–555, 2014.

[18] J. Donea, S. Giuliani, and J.-P. Halleux, "An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions," *Computer methods in applied mechanics and engineering*, vol. 33, no. 1-3, pp. 689–723, 1982.

[19] T. Coupez, "Convection of local level set function for moving surfaces and interfaces in forming flow," in *Materials Processing and Design, Modeling, Simulation and Applications, NUMIFORM'07: 9th International Conference on Numerical Methods in Industrial Forming Processes*, vol. 908, pp. pages–61, 2007.

[20] L. Ville, L. Silva, and T. Coupez, "Convected level set method for the numerical simulation of fluid buckling," *International Journal for numerical methods in fluids*, vol. 66, no. 3, pp. 324–344, 2011.

[21] S. Lee and A. J. Salgado, "Stability analysis of pressure correction schemes for the navier–stokes equations with traction boundary conditions," *Computer Methods in Applied Mechanics and Engineering*, vol. 309, pp. 307–324, 2016.

[22] A. Bonito, J.-L. Guermond, and B. Popov, "Stability analysis of explicit entropy viscosity methods for non-linear scalar conservation equations," *Mathematics of Computation*, vol. 83, no. 287, pp. 1039–1062, 2014.

[23] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, "High order strong stability preserving time discretizations," *Journal of Scientific Computing*, vol. 38, no. 3, pp. 251–289,

2009.

[24] I. Higueras, "Representations of runge–kutta methods and strong stability preserving methods," *SIAM journal on numerical analysis*, vol. 43, no. 3, pp. 924–948, 2005.

[25] J.-L. Guermond and M. Nazarov, "A maximum-principle preserving c0 finite element method for scalar conservation equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 272, pp. 198–213, 2014.

[26] J.-L. Guermond, M. Nazarov, B. Popov, and Y. Yang, "A second-order maximum principle preserving lagrange finite element technique for nonlinear scalar conservation equations," *SIAM Journal on Numerical Analysis*, vol. 52, no. 4, pp. 2163–2182, 2014.

[27] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W.-S. Don, "The theoretical accuracy of runge–kutta time discretizations for the initial boundary value problem: a study of the boundary error," *SIAM Journal on Scientific Computing*, vol. 16, no. 6, pp. 1241–1252, 1995.

[28] D. Pathria, "The correct formulation of intermediate boundary conditions for runge–kutta time integration of initial boundary value problems," *SIAM Journal on Scientific Computing*, vol. 18, no. 5, pp. 1255–1266, 1997.

[29] A.-K. Tornberg, "Multi-dimensional quadrature of singular and discontinuous functions," *BIT Numerical Mathematics*, vol. 42, no. 3, pp. 644–669, 2002.

[30] C. Lehrenfeld, "High order unfitted finite element methods on level set domains using isoparametric mappings," *Computer Methods in Applied Mechanics and Engineering*, vol. 300, pp. 716–733, 2016.

[31] P. Smereka, "The numerical approximation of a delta function with application to level set methods," *Journal of Computational Physics*, vol. 211, no. 1, pp. 77–90,

2006.

[32] J. D. Towers, "Two methods for discretizing a delta function supported on a level set," *Journal of Computational Physics*, vol. 220, no. 2, pp. 915–931, 2007.

[33] J. T. Beale, "A proof that a discrete delta function is second-order accurate," *Journal of Computational Physics*, vol. 227, no. 4, pp. 2195–2197, 2008.

[34] J. D. Towers, "A convergence rate theorem for finite difference approximations to delta functions," *Journal of Computational Physics*, vol. 227, no. 13, pp. 6591–6597, 2008.

[35] B. Müller, F. Kummer, and M. Oberlack, "Highly accurate surface and volume integration on implicit domains by means of moment-fitting," *International Journal for Numerical Methods in Engineering*, vol. 96, no. 8, pp. 512–528, 2013.

[36] R. Saye, "High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. A993–A1019, 2015.

[37] X. Wen, "High order numerical methods to a type of delta function integrals," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1952–1967, 2007.

[38] X. Wen, "High order numerical quadratures to one dimensional delta function integrals," *SIAM Journal on Scientific Computing*, vol. 30, no. 4, pp. 1825–1846, 2008.

[39] X. Wen, "High order numerical methods to two dimensional delta function integrals in level set methods," *Journal of Computational Physics*, vol. 228, no. 11, pp. 4273–4290, 2009.

[40] X. Wen, "High order numerical methods to three dimensional delta function integrals in level set methods," *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1288–1309, 2010.

[41] B. Engquist, A.-K. Tornberg, and R. Tsai, "Discretization of dirac delta functions in level set methods," *Journal of Computational Physics*, vol. 207, no. 1, pp. 28–51, 2005.

[42] S. Zahedi and A.-K. Tornberg, "Delta function approximations in level set methods by distance function extension," *Journal of Computational Physics*, vol. 229, no. 6, pp. 2199–2219, 2010.

[43] A. J. Chorin, "Numerical solution of the navier-stokes equations," *Mathematics of computation*, vol. 22, no. 104, pp. 745–762, 1968.

[44] L. Timmermans, P. Minev, and F. Van De Vosse, "An approximate projection scheme for incompressible flow using spectral elements," *International Journal for Numerical Methods in Fluids*, vol. 22, no. 7, pp. 673–688, 1996. cited By (since 1996)105.

[45] J.-L. Guermond and A. Salgado, "A splitting method for incompressible flows with variable density based on a pressure poisson equation," *Journal of Computational Physics*, vol. 228, no. 8, pp. 2834–2846, 2009.

[46] J. Guermond, P. Minev, and J. Shen, "An overview of projection methods for incompressible flows," *Computer methods in applied mechanics and engineering*, vol. 195, no. 44, pp. 6011–6045, 2006.

[47] A. Bonito, J.-L. Guermond, and S. Lee, "Modified pressure-correction projection methods: Open boundary and variable time stepping," in *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, pp. 623–631, Springer, 2015.

[48] R. Rannacher, *On Chorin's projection method for the incompressible Navier-Stokes equations*. Springer, 1992.

[49] J.-L. Guermond and L. Quartapelle, "On the approximation of the unsteady navier–stokes equations by finite element projection methods," *Numerische mathematik*,

vol. 80, no. 2, pp. 207–238, 1998.

[50] P. Bochev and R. B. Lehoucq, "On the finite element solution of the pure neumann problem," *SIAM review*, vol. 47, no. 1, pp. 50–66, 2005.

[51] J.-L. Guermond and A. J. Salgado, "Error analysis of a fractional time-stepping technique for incompressible flows with variable density," *SIAM Journal on Numerical Analysis*, vol. 49, no. 3, pp. 917–944, 2011.

[52] A. N. Brooks and T. J. Hughes, "Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations," *Computer methods in applied mechanics and engineering*, vol. 32, no. 1, pp. 199–259, 1982.

[53] T.-P. Fries and H. G. Matthies, "A review of petrov–galerkin stabilization approaches and an extension to meshfree methods," 2004.

[54] T. Coupez and E. Hachem, "Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing," *Computer Methods in Applied Mechanics and Engineering*, vol. 267, no. 0, pp. 65 – 85, 2013.

[55] T. E. Tezduyar and Y. Osawa, "Finite element stabilization parameters computed from element matrices and vectors," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 3âĂŞ4, pp. 411 – 430, 2000.

[56] G. C. Buscaglia and R. F. Ausas, "Variational formulations for surface tension, capillarity and wetting," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 45, pp. 3011–3025, 2011.

[57] M. C. Delfour and J.-P. Zolésio, *Shapes and geometries: metrics, analysis, differential calculus, and optimization*, vol. 22. Siam, 2011.

[58] J.-F. Gerbeau and T. Lelievre, "Generalized navier boundary condition and geometric conservation law for surface tension," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 5, pp. 644–656, 2009.

[59] E. Maitre, C. Misbah, P. Peyla, and A. Raoult, "Comparison between advected-field and level-set methods in the study of vesicle dynamics," *Physica D: Nonlinear Phenomena*, vol. 241, no. 13, pp. 1146–1157, 2012.

[60] V. Doyeux, V. Chabannes, C. Prud'Homme, and M. Ismail, "Simulation of vesicle using level set method solved by high order finite element," in *ESAIM: Proceedings*, vol. 38, pp. 335–347, EDP Sciences, 2012.

[61] V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'Homme, and M. Ismail, "Simulation of two-fluid flows using a finite element/level set method. application to bubbles and vesicle dynamics," *Journal of Computational and Applied Mathematics*, vol. 246, pp. 251–259, 2013.

[62] V. Doyeux, *Modeling and simulation of multi-fluid systems. Applications to blood flows*. PhD thesis, Université Grenoble Alpes, 2014.

[63] E. Bänsch, "Finite element discretization of the navier–stokes equations with a free capillary surface," *Numerische Mathematik*, vol. 88, no. 2, pp. 203–235, 2001.

[64] S.-R. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska, "Quantitative benchmark computations of two-dimensional bubble dynamics," *International Journal for Numerical Methods in Fluids*, vol. 60, no. 11, pp. 1259–1288, 2009.

[65] M. Deserno, "Fluid lipid membranes a primer," 2007.

[66] X. S. Li and J. W. Demmel, "SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems," *ACM Trans. Mathematical Software*,

vol. 29, pp. 110–140, June 2003.

[67] L. Grigori, J. W. Demmel, and X. S. Li, "Parallel symbolic factorization for sparse LU with static pivoting," *SIAM J. Scientific Computing*, vol. 29, no. 3, pp. 1289–1314, 2007.

[68] A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 345–363, 1973.

[69] J. R. Gilbert, "Some nested dissection order is nearly optimal," *Information Processing Letters*, vol. 26, no. 6, pp. 325–328, 1988.

[70] A. Laadhari, P. Saramito, and C. Misbah, "Computing the dynamics of biomembranes by combining conservative level set and adaptive finite element methods," *Journal of Computational Physics*, vol. 263, pp. 328–352, 2014.

[71] M. Droske and M. Rumpf, "A level set formulation for willmore flow," *Interfaces and free boundaries*, vol. 6, no. 3, pp. 361–378, 2004.

[72] L. Hsu, R. Kusner, and J. Sullivan, "Minimizing the squared mean curvature integral for surfaces in space forms," *Experimental Mathematics*, vol. 1, no. 3, pp. 191–207, 1992.

[73] U. Seifert, K. Berndl, and R. Lipowsky, "Shape transformations of vesicles: Phase diagram for spontaneous-curvature and bilayer-coupling models," *Physical Review A*, vol. 44, no. 2, p. 1182, 1991.

[74] U. Seifert and R. Lipowsky, "Morphology of vesicles," 1995.

[75] A. Kurganov and J. Rauch, "The order of accuracy of quadrature formulae for periodic functions," in *Advances in phase space analysis of partial differential equations*, pp. 155–159, Springer, 2009.

[76] J.-L. Guermond, "Un résultat de convergence d'ordre deux pour l'approximation des équations de navier-stokes par projection incrémentale," *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, vol. 325, no. 12, pp. 1329–1332, 1997.

[77] J. Guermond and J. Shen, "On the error estimates for the rotational pressure-correction projection methods," *Mathematics of Computation*, vol. 73, no. 248, pp. 1719–1737, 2004.

APPENDIX A

CONVERGENCE RATES OF DIRAC MEASURE WITH SIMPLER QUADRATURE RULES

It turns out that when we use a quadrature scheme on our Dirac integration tool of Chapter 3 that is exact for polynomials of degree less than 5, we observed that often we obtain the same convergence rates as are predicted with the higher quadrature scheme. This is merely a curiosity as we have a full proof above of the result with a quadrature scheme exact for polynomials of degree 5. To understand why we obtain these results in this less restrictive case, we focus on a one dimensional analysis. Recall that this is completely with regard to our using the dirac kernel

$$\phi(t) = \begin{cases} \frac{693}{512}(1-t^2)^5, & |t| < 1 \\ 0, & \text{otherwise} \end{cases}.$$

We consider Γ to be a single point in 1D, say $\Gamma = \{a\}$, for some $a \in \mathbb{R}$, Then we choose

$$d(x) = x - a$$

to be our signed distance function. In this case, we have

$$\delta_\varepsilon(d(\mathbf{x})) = \begin{cases} \frac{693}{512}\frac{1}{\varepsilon}\left(1 - \left(\frac{x-a}{\varepsilon}\right)^2\right)^5, & |x-a| < \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

and we want to estimate

$$\begin{aligned} E &= f(a) - Q(f(x)\delta_\varepsilon(d(x))) \\ &= \left(f(a) - \int_{a-\varepsilon}^{a+\varepsilon} f(x)\delta_w(x-a)dx \right) \\ &\quad + \sum_{m\in\mathbb{Z}} \left(\int_{mh}^{(m+1)h} f(x)\delta_w(x-a)dx - Q_{[mh,(m+1)h]}(f(x)\delta_\varepsilon(x-a)) \right) \\ &= E_{\text{analytic}} + E_{\text{quad}} \end{aligned}$$

for $Q_{[mh,(m+1)h]}(\cdot)$ a quadrature on cell $[mh,(m+1)h]$ which is exact for polynomials of degree less than 5.

A.0.0.1 Convergence for trapezoidal quadrature for periodic functions in $W^{r,1}_{per}$

Let

$$E_N^{[a,b]}(f) := \int_a^b f(x)dx - \Delta x \sum_{n=0}^{N-1} f(x_n) = \int_a^b f(x)dx - T_N(f)$$

with $\Delta x = \frac{b-a}{N}$ and $x_n = n\Delta x$, be the error of an N point trapezoidal rule on a $(b-a)$-periodic function $f \in W^{r,1}_{\text{per}}([a,b])$. The following theorem from [75] gives an estimate for a 2π-periodic function.

Theorem A.0.1. *If $f \in W^{r,1}_{per}([0,2\pi])$ and $1 < r \in \mathbb{N}$ then the error of the trapezoidal rule satisfies*

$$\left| E_N^{[0,2\pi]}(f) \right| \leq \frac{C\|f^{(r)}\|_{L^1([0,2\pi])}}{N^r}, \qquad \textit{where } C = 2\sum_{k=1}^{\infty} \frac{1}{k^r}.$$

Now, for the sake of simplicity, we assume that h evenly divides ε ie. that there are $2\frac{\varepsilon}{h} =: N \in \mathbb{N}$ cells in $[-\varepsilon,\varepsilon]$ and that $\Gamma = \{0\}$. In this case, we can periodically extend the domain $[-\varepsilon,\varepsilon]$ and $\delta_\varepsilon(x)$ to get the periodic extension, $\delta_\varepsilon^{[-\varepsilon,\varepsilon]}(\mathbf{x})$. It is simple to compute that

$$\int_{-\varepsilon}^{\varepsilon} D^\alpha \delta_\varepsilon(t)dt = 0, \qquad \text{for } \alpha = 1\ldots 5$$

which is equivalent to

$$D^\beta \delta_\varepsilon(-\varepsilon) = D^\beta \delta_\varepsilon(\varepsilon), \qquad \text{for } \beta = 0 \dots 4.$$

Thus, our periodic extension has 5 integrable derivatives, ie

$$\delta_\varepsilon^{[-\varepsilon,\varepsilon]} \in W^{5,1}_{\text{per}}([-\varepsilon,\varepsilon]).$$

We can transform the periodic extensions from $[-\varepsilon,\varepsilon]$ to $[0,2\pi]$ and then apply Theorem (A.0.1) to obtain

$$E_N^{[-\varepsilon,\varepsilon]}\left(\delta_\varepsilon^{[-\varepsilon,\varepsilon]}\right) = \frac{1}{\pi} E_N^{[0,2\pi]}\left(\phi^{[0,2\pi]}\right) \leq \frac{C\left\|\frac{d^r}{dy^r}\phi^{[0,2\pi]}(y)\right\|_{L^1([0,2\pi])}}{N^5} \leq c\left(\frac{h^5}{\varepsilon^5}\right)$$

where $\phi^{[0,2\pi]}$ is the 2π-periodic extension of the translated and stretched ($y = \pi t + \pi$) Dirac kernel,

$$\phi(y) := \frac{693}{512}\left(1 - \left(\frac{y}{\pi} - 1\right)^2\right)^5, \qquad \text{for } y \in [0,2\pi].$$

This is a polynomial so the $L^1([0,2\pi])$ norm of the 5-th derivative is a constant. Thus our estimate for the trapezoidal rule is at least $\mathcal{O}\left((h/\varepsilon)^5 = h^{5/4}\right)$. This is close to the observed error rates of $\mathcal{O}\left((h/\varepsilon)^6 = h^{3/2}\right)$, seen for instance in Figure 3.5 where we used a Gauss Quadrature rule exact for cubics. We hypothesize that it is because our function is also even, that we get the 6th power instead of the 5th.

APPENDIX B

CONVERGENCE THEORY FOR NAVIER-STOKES PROJECTION ALGORITHMS

We give a review of the main general theorems in the literature on expected convergence rates for the various forms of the projection schemes before stating the expected convergence results for the above algorithm. While the convergence rates for the above algorithm have not been fully proved, it appears to behave qualitatively like the algorithms for which the results have been proven and so the conjectures at the end summarize what we expect. The theorems are stated without proof although reference is given to where they come from which often includes the proof or sketches of proof.

B.1 Constant density, incompressible Navier-Stokes

The following are some theorems related to the convergence rates for approximations of the constant density incompressible Navier-Stokes equation through the use of a pressure correction splitting in various forms: standard and rotational.

B.1.1 Algorithm in standard form

The numerical solution $(\mathbf{u}^k, \tilde{\mathbf{u}}^k, p^k)$ of the constant density incompressible Navier-Stokes algorithm in standard form using BDF2 time derivative has the following error estimates hold, Theorems (3.1)-(3.5) of [76].

Theorem B.1.1. *If* $\boldsymbol{u}(t,x) \in W^{1,\infty}\left(0,T;\ H_0^1(\Lambda)^d \cap H^{\ell+1}(\Lambda)^d\right) \cap H^2\left(0,T;\ H^1(\Lambda)^d\right)$ *and* $p \in W^{1,\infty}\left(0,T;\ H^\ell(\Lambda)\right) \cap H^2\left(0,T;\ L^2(\Lambda)\right)$*, then*

$$\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^\infty\left(L^2(\Lambda)^d\right)} + \|(\boldsymbol{u}-\tilde{\mathbf{u}}_h)_\tau\|_{\ell^\infty\left(L^2(\Lambda)^d\right)} \lesssim \tau + h^{\ell+1} \tag{B.1}$$

Theorem B.1.2. *If* $\boldsymbol{u}(t,x) \in W^{2,\infty}\left(0,T;\ H_0^1(\Lambda)^d \cap H^{\ell+1}(\Lambda)^d\right) \cap H^3\left(0,T;\ H^1(\Lambda)^d\right)$ *and*

$p \in W^{2,\infty}\left(0,T;\ H^{\ell}(\Lambda)\right) \cap H^{3}\left(0,T;\ L^{2}(\Lambda)\right)$*, then there exist* $c_s > 0$ *and* $h_s > 0$ *such that for* $h \in (0, h_s]$ *and* $\tau \leq c_s / \left(1 + |\log(h^{-1})|\right)^{1/2}$ *in 2 dimensions or* $\tau \leq c_s h^{1/2}$ *in 3 dimensions we have*

$$\|(\boldsymbol{u} - \tilde{\mathbf{u}}_h)_\tau\|_{\ell^\infty\left(H^1(\Lambda)^d\right)} + \|(p - p_h)_\tau\|_{\ell^\infty(L^2(\Lambda))} \lesssim \tau + h^{\ell} \tag{B.2}$$

Theorem B.1.3. *Under the regularity hypothesis of Theorem B.1.1 and its restrictions on* τ *and* h*, the error bounds also hold*

$$\|(\boldsymbol{u} - \boldsymbol{u}_h)_\tau\|_{\ell^2\left(L^2(\Lambda)^d\right)} + \|(\boldsymbol{u} - \tilde{\mathbf{u}}_h)_\tau\|_{\ell^2\left(L^2(\Lambda)^d\right)} \lesssim \tau^2 + h^{\ell+1} \tag{B.3}$$

If we want improved error bounds for $\ell^\infty\left(L^2(\Lambda)^d\right)$ norm, we must add more regularity as follows:

Theorem B.1.4. *If* $\boldsymbol{u}(t,x) \in W^{3,\infty}\left(0,T;\ H_0^1(\Lambda)^d \cap H^{\ell+1}(\Lambda)^d\right) \cap H^4\left(0,T;\ H^1(\Lambda)^d\right)$ *and* $p \in W^{2,\infty}\left(0,T;\ H^{\ell}(\Lambda)\right) \cap H^4\left(0,T;\ L^2(\Lambda)\right)$*, with the restrictions on* τ *and* h *of Theorem B.1.1, then*

$$\|(\boldsymbol{u} - \boldsymbol{u}_h)_\tau\|_{\ell^\infty\left(L^2(\Lambda)^d\right)} + \|(\boldsymbol{u} - \tilde{\mathbf{u}}_h)_\tau\|_{\ell^\infty\left(L^2(\Lambda)^d\right)} \lesssim \tau^{7/4} + \tau^{3/4} h^{\ell} + h^{\ell+1} \tag{B.4}$$

The estimates on the pressure norm of Theorem B.1.2 can be improved by introducing a discrete norm as follows. Let $(v_h, q_h) \in \mathbb{X}_h \times \mathbb{M}_h$ and define $B_h \in \mathcal{L}(\mathbb{X}_h, \mathbb{M}_h)$ be the discrete divergence operator $(B_h v_h, q_h) = (v_h, B_h^T q_h) = -(\nabla \cdot v_h, q_h)$. Then define the discrete norm for $v_h \in \mathbb{X}_h$ as $\|v_h\|_{\mathcal{A}_h} = \sup_{w_h \in \mathbb{X}_h} (\nabla v_h, \nabla w_h)/\|w_h\|_{L^2(\Lambda)}$ and for $q \in L^2(\Lambda)$, let $\|q\|_{B_h, \mathcal{A}_h} = \sup_{v_h \in \mathbb{X}_h} (q, \nabla \cdot v_h)/\|v_h\|_{\mathcal{A}_h}$.

Theorem B.1.5. *Under the regularity hypothesis of Theorem B.1.4 and its restrictions on*

τ and h, the error bounds also hold

$$\|(p-p_h)_\tau\|_{\ell^2\left(\|\cdot\|_{B_h,\mathcal{A}_h}\right)} \lesssim \tau^{3/2} + h^\ell \tag{B.5}$$

B.1.2 Algorithm in rotational form

Theorem B.1.6. *Given that the true solution is smooth enough, the numerical solution, $(\boldsymbol{u}^k, \tilde{\mathbf{u}}^k, p^k)$, of the constant density incompressible Navier-Stokes algorithm using the second order approximation in time, pressure correction scheme in rotational form gives the following convergence estimates:*

$$\begin{aligned}
\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^2\left([L^2(\Lambda)]^d\right)} + \|(\boldsymbol{u}-\tilde{\mathbf{u}}_h)_\tau\|_{\ell^2\left([L^2(\Lambda)]^d\right)} &\lesssim \tau^2 \\
\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^2\left([H^1(\Lambda)]^d\right)} + \|(\boldsymbol{u}-\tilde{\mathbf{u}}_h)_\tau\|_{\ell^2\left([H^1(\Lambda)]^d\right)} &\lesssim \tau^{3/2} \\
\|(p-p_h)_\tau\|_{\ell^2\left([L^2(\Lambda)]^d\right)} &\lesssim \tau^{3/2}
\end{aligned}$$

Proof. See Theorem 4.1 of [77] for details □

B.2 Variable density incompressible Navier-Stokes

B.2.1 Algorithm in standard form

Theorem B.2.1. *If $\rho \in W^{1,\infty}(0,T;\ W^{1,\infty}(\Lambda))$, $\boldsymbol{u} \in W^{1,\infty}\left(0,T;\ H_0^1(\Lambda)^d \cap H^{\ell+1}(\Lambda^d)\right)$ and $p \in W^{1,\infty}\left(H^\ell(\Lambda)\right)$ and there exist $\chi, \eta > 0$ such that for all k, $\chi \leq \min_{\boldsymbol{x}} \rho^k(\boldsymbol{x})$ and $\sup_{\boldsymbol{x}} \rho^k(\boldsymbol{x}) \leq \eta$. Assume (to decouple the density from velocity) that we can construct a*

sequence $(\rho^k)_k$ from a sequence $(\boldsymbol{u}^k)_k$ of velocities in such a way that

$$\|(\rho-\rho^k)_\tau\|^2_{\ell^\infty(H^1)} + \left\|\left(\rho_t - \frac{\rho^k-\rho^{k-1}}{\tau}\right)_\tau\right\|^2_{\ell^\infty(L^2)} \tag{B.6}$$

$$\leq c(\lambda)(\tau+h^{\ell+1})^2 + \lambda\left\|\Pi_h\boldsymbol{u}(t^k)-\boldsymbol{u}^k\right\|^2_{H^1} + c(\lambda)\left\|\sqrt{\rho^k}\left(\Pi_h\boldsymbol{u}(t^k)-\boldsymbol{u}^k\right)\right\|^2_{L^2} \tag{B.7}$$

where $\lambda > 0$ can be chose as small as is needed. And finally, we assume that the initial approximation of data

$$\|\rho(t^0)-\rho^0\|_{L^\infty} + \|\boldsymbol{u}(t^0)-\boldsymbol{u}^0\|_{L^2} + h\|\boldsymbol{u}(t^0)-\boldsymbol{u}^0\| + h\|p(t^0)-p^0\| \lesssim h^{\ell+1} \tag{B.8}$$

holds. Then the incremental BDF1 variable density algorithm in standard form has the following error estimates

$$\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^\infty(L^2(\Lambda)^d)} \lesssim \tau + h^{\ell+1} \tag{B.9}$$

$$\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^2(H^1(\Lambda)^d)} \lesssim \tau + h^{\ell} \tag{B.10}$$

$$\|(p-p_h)_\tau\|_{\ell^2(L^2(\Lambda))} \lesssim \tau + h^{\ell} \textit{ (not stated or proven but should be true)} \tag{B.11}$$

Proof. See proof of Theorem 4.2 of [51]. □

Conjecture B.2.2. *Under the assumptions of Theorem B.2.1, the BDF2 standard form algorithm with variable density has the following error estimates:*

$$\left\|(\sqrt{\rho}\boldsymbol{u}-\sqrt{\rho_h}\boldsymbol{u}_h)_\tau\right\|_{\ell^\infty(L^2(\Lambda)^d)} \lesssim \tau^2 + h^{\ell+1} \tag{B.12}$$

$$\|(\boldsymbol{u}-\boldsymbol{u}_h)_\tau\|_{\ell^2(H^1(\Lambda)^d)} \lesssim \tau + h^{\ell} \tag{B.13}$$

$$\|(p-p_h)_\tau\|_{\ell^2(L^2(\Lambda))} \lesssim \tau + h^{\ell} \textit{ (not stated or proven but should be true)} \tag{B.14}$$

B.2.2 Algorithm in rotational form

Conjecture B.2.3. *Under the assumptions of Theorem B.2.1, the BDF2 rotational form algorithm with variable density has the following error estimates:*

$$\left\|\left(\sqrt{\rho}\boldsymbol{u}-\sqrt{\rho_h}\boldsymbol{u}_h\right)_\tau\right\|_{\ell^\infty(L^2(\Lambda)^d)} \lesssim \tau^2+h^{\ell+1} \tag{B.15}$$

$$\left\|\left(\boldsymbol{u}-\boldsymbol{u}_h\right)_\tau\right\|_{\ell^2(H^1(\Lambda)^d)} \lesssim \tau^{3/2}+h^{\ell} \tag{B.16}$$

$$\left\|\left(p-p_h\right)_\tau\right\|_{\ell^2(L^2(\Lambda))} \lesssim \tau^{3/2}+h^{\ell} \textit{ (not stated or proven but should be true)} \tag{B.17}$$

This last conjecture is more of an observation of what we expect we should observe under smooth conditions.

Conjecture B.2.4. *If Λ is smooth, for instance a circular domain, and the exact solution is smooth, then we should see the $\ell^\infty(E)$ norms behaving as well as the $\ell^2(E)$ norms in every instance. In many cases, we cannot state a good result for the ℓ^∞ in time norms because the theory is tricky or unclear, but we should still observe them to be as good as the ℓ^2 in time norms. Additionally, the general theory says that $L^2(\Lambda)$ norm of pressure and the $H^1(\Lambda)$ norm of velocity should have $\tau^{1/2}$ power worse rates than the $L^2(\Lambda)$ norm of velocity but in smooth domains, they are often observed to all have the higher rates. It is only when corners appear in the domain or the exact solution has less regularity that the loss appears.*