Title:
The Voronoi Implicit Interface Method with Applications to Multiphase Fluid Flow and Multiscale Modelling of Foam Dynamics

Author:
Saye, Robert Ian

Acceptance Date:
2013

Series:
UC Berkeley Electronic Theses and Dissertations

Degree:
Ph.D., MathematicsUC Berkeley

Advisor(s):
Sethian, James A

Committee:
Wilkening, Jon, Evans, Lawrence C, Szeri, Andrew J

Permalink:
http://escholarship.org/uc/item/9n91z6d6

Abstract:

The Voronoi Implicit Interface Method with Applications to Multiphase Fluid Flow and Multiscale Modelling of Foam Dynamics

by

Robert Ian Saye

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

in

Applied Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor James A. Sethian, Chair
Associate Professor Jon Wilkening
Professor Lawrence C. Evans
Professor Andrew J. Szeri

Spring 2013

The Voronoi Implicit Interface Method with Applications to Multiphase Fluid Flow and Multiscale Modelling of Foam Dynamics

Robert Ian Saye

Abstract

The Voronoi Implicit Interface Method with Applications to Multiphase Fluid Flow and Multiscale Modelling of Foam Dynamics

by

Robert Ian Saye

Doctor of Philosophy in Applied Mathematics

University of California, Berkeley

Professor James A. Sethian, Chair

In this thesis, the Voronoi Implicit Interface Method (VIIM) [1–3] is presented together with several applications in multiphase curvature flow, multiphase incompressible fluid flow, mesh generation for interconnected surfaces, and multiscale modelling of foam dynamics. The VIIM tracks the evolution of multiple interacting regions ("phases") whose motion may be determined by geometry, complex physics, intricate jump conditions, internal constraints, and boundary conditions. From a mathematical point of view, the method provides a theoretical framework to evolve interconnected interfaces with junctions. Discretising this theoretical framework leads to an efficient Eulerian-based numerical method that uses a single unsigned distance function, together with a region indicator function, to represent a multiphase system. The VIIM works in any number of spatial dimensions, accurately represents complex geometries involving triple and higher-order junctions, and automatically handles topological changes in the evolving interface, including creation and destruction of phases. Here, the central ideas behind the method are presented, implementation is discussed, and convergence tests are performed to illustrate the accuracy of the method. Several applications of the VIIM are shown, including in constant speed normal driven flow; multiphase curvature flow with constraints; and multiphase incompressible fluid flow in which density, viscosity, and surface tension can be defined on a per-phase basis and membranes can be permeable.

An efficient and robust mesh generation algorithm for interconnected surfaces [4] is also presented. The algorithm capitalises on a geometric construction used in the VIIM, known as the "Voronoi interface", to generate high-quality triangulated meshes that are topologically consistent, such that mesh elements meet precisely at junctions without gaps, overlaps, or hanging nodes. The generated meshes can be used in finite element methods for solving partial differential equations on a network of evolving interconnected curved surfaces.

Finally, a scale-separated, multiscale model for the dynamics of a soap bubble foam [5] is presented. The model leads to a computational framework for studying the interlinked effects of drainage, rupture, and rearrangement in a foam of bubbles, coupling microscale fluid flow in a network of thin-film membranes ("lamellae") and junctions ("Plateau borders") to macroscale gas dynamics driven by surface tension. Here, thin-film equations for fluid flow inside curved lamellae

and Plateau borders are derived, flux boundary conditions which conserve liquid mass are developed, and local conservation laws for transport of film thickness during rearrangement are designed. From a numerical perspective, several new numerical methods are developed, including Lagrangian-based schemes for conserving liquid in the membranes during rearrangement, finite element methods to solve fourth-order nonlinear partial different equations on curved surfaces, methods to accurately solve coupled flux boundary conditions at Plateau borders and quadruple points, and projection methods to couple gas dynamics to the VIIM. Convergence tests are performed to demonstrate the accuracy of the numerical methods, and results of the multiscale model are shown for a variety of problems, including collapsing foam clusters displaying thin-film interference effects.

Contents

Contents i

Acknowledgements iv

1 Introduction 1
- 1.1 Multiphase problems . . . 1
- 1.2 Tracking multiple interfaces . . . 2
  - 1.2.1 Mathematical considerations . . . 2
  - 1.2.2 Numerical goals . . . 3
- 1.3 Previous work . . . 4
- 1.4 New contributions . . . 7
- 1.5 Outline . . . 8

2 The Level Set Method 9
- 2.1 Introduction . . . 9
- 2.2 Mathematical properties . . . 12
- 2.3 Numerical methods . . . 14
- 2.4 Reinitialisation . . . 16
- 2.5 Extension velocities . . . 17

3 The Voronoi Implicit Interface Method 19
- 3.1 Central ideas . . . 19
- 3.2 The Voronoi interface . . . 20
- 3.3 Mathematical formulation . . . 21
  - 3.3.1 One-dimensional example . . . 21
  - 3.3.2 Two-dimensional example . . . 22
  - 3.3.3 Mathematical formulation in multiple dimensions . . . 23
  - 3.3.4 Additional comments . . . 23
- 3.4 Discrete approximations of the $\epsilon = 0^+$ limit . . . 24
  - 3.4.1 Spatial and temporal discretisations of the formal $\epsilon = 0^+$ limit . . . 24
  - 3.4.2 Key algorithmic steps . . . 26
  - 3.4.3 Reconstruction interval . . . 28

3.5 Adding physics and evaluation of the speed function . . . 28
3.6 Efficiency . . . 29
3.7 Creation and destruction of phases . . . 29
3.8 Interface extraction and visualisation . . . 29
3.9 Parallel implementation using MPI . . . 31
3.10 Convergence tests and verification . . . 31
3.10.1 Basic tests in two dimensions . . . 33
3.10.2 Triple points in two dimensions . . . 36
3.10.3 von Neumann-Mullins' law . . . 39
3.10.4 Triple lines in three dimensions . . . 44
3.10.5 Three-dimensional analogue of von Neumann-Mullins' law . . . 44
3.10.6 Summary of convergence results . . . 46
3.11 Concluding remarks . . . 46

4 Applications of the VIIM **48**
4.1 Geometric flows . . . 48
4.1.1 Constant speed normal driven flow . . . 48
4.1.2 Advection by an external velocity field . . . 49
4.1.3 Mean curvature flow with constraints . . . 51
4.2 Fluid flow with permeability . . . 54
4.2.1 Testing convergence during topological change . . . 58
4.2.2 Application to dry foams . . . 59
4.2.3 Phase-dependent fluid properties . . . 65
4.2.4 Additional fluid flow applications . . . 69
4.3 Implementing general domains using an exterior phase . . . 71

5 Mesh Generation for Interconnected Surfaces **73**
5.1 Previous work . . . 74
5.2 The Voronoi interface . . . 75
5.3 Mesh generation . . . 76
5.3.1 Creating the initial mesh . . . 77
5.3.2 Smoothing and projection . . . 80
5.3.3 Edge flipping . . . 86
5.3.4 Parallelisation . . . 86
5.4 Results . . . 88

6 Multiscale Modelling of Foam Dynamics **94**
6.1 Multiscale physics in foams . . . 94
6.2 Previous work . . . 96
6.3 Multiscale modelling using scale separation . . . 96
6.3.1 Rearrangement phase . . . 97
6.3.2 Drainage phase . . . 99

6.3.3 Rupture phase . . . 101
6.4 Summary . . . 101
6.5 Derivation of the thin-film equations . . . 101
6.5.1 Derivation of the lamella thin-film equation . . . 101
6.5.2 Derivation of the Plateau border thin-film equation . . . 107
6.5.3 Flux boundary condition . . . 109
6.5.4 Boundary layer scaling . . . 110

7 Numerical Methods for Foam Dynamics 111
7.1 Numerical methods for the rearrangement phase . . . 111
7.1.1 Interface tracking . . . 112
7.1.2 Solving the Navier-Stokes equations . . . 112
7.1.3 Local conservation law . . . 113
7.1.4 Topological changes . . . 117
7.1.5 Testing for macroscopic equilibrium . . . 118
7.1.6 Implementation and parallelisation . . . 119
7.2 Numerical methods for the drainage phase . . . 119
7.2.1 Mesh generation . . . 122
7.2.2 Plateau border boundary condition . . . 122
7.2.3 Flux boundary condition . . . 124
7.2.4 Lamella thin-film equation . . . 125
7.2.5 Plateau border thin-film equation . . . 128
7.2.6 Implementation and parallelisation . . . 129
7.2.7 Convergence tests . . . 130
7.3 Coupling drainage and rearrangement via rupture . . . 136
7.4 Results . . . 137
7.4.1 Rearrangement phase . . . 137
7.4.2 Drainage phase . . . 141
7.4.3 Rupture and redistribution of mass . . . 141
7.4.4 Coupled rearrangement, drainage, and rupture . . . 142
7.5 Concluding remarks . . . 144

8 Summary and Future Directions 147

References 151

Acknowledgements

First and foremost, I have the pleasure to thank my PhD advisor, colleague, and friend, James Sethian: for calling me while I was still living in Australia to entice me to come to Berkeley, for having lunch with me the day after I landed, for looking out for my health and well-being and helping me when I was homesick, for leading such an impressive applied math group at Berkeley Lab and quietly managing all of the bureaucracy behind the scenes, for the fond memories of our trip together to Washington DC to receive the Cozzarelli Prize, and for being such a great mathematician, mentor, and collaborator – thank you, Jamie.

A special "g'day" goes to Steve Roberts, my undergraduate advisor at the Australian National University in my home city, Canberra. Thank you for convincing me to come to the U.S. for graduate studies, and for continuing to support me in this adventure.

I have been fortunate to spend much of my time in the Mathematics Group at the Berkeley Lab. The group has a very stimulating, yet relaxing environment, with a wonderful mixture of smart and friendly people to interact with, all of which I have truly enjoyed. Credit must be given to Alexandre Chorin, the founder of the group, and to Jamie for continuing its traditions. To the Professors G. I. Barenblatt, Alexandre Chorin, Maria Garzon, Ole Hald, Per-Olof Persson, and Xuemin Tu: your unique and wise perspectives on math and life have been enlightening and utterly fascinating. To the younger professors and post-docs Ethan Atkins, August Johansson, Matti Morzfeld, Marcus Roper, Chris Rycroft, and Ben Stamm: it has been a true pleasure to know you and watch your careers flourish. Finally, to my fellow graduate students Jue Chen, Jeff Donatelli, Brad Froehle, Erica Isaacson, Michaeel Kazi, Jakub Kominiarczuk, Danielle Maddix, Trevor Potter, and Ben Preskill: thank you for all the good times! Lastly, I would like to give special mention to Valerie Heatlie, the retired administrative assistant and "mother" of the math group, one of the kindest people I know. Thank you to Paul Concus and Barbara Salisbury as well, for their dedication to the group.

To my thesis committee, consisting of Jamie, Jon Wilkening, Craig Evans, and Andrew Szeri, I am grateful for your time reading this thesis and for your wonderful feedback.

I would also like to thank the American Australian Association, who helped support my research with a Sir Keith Murdoch Fellowship during 2011–2012. The association promotes ties between Australia and America, and I look forward to creating more connections in applied mathematics across the Pacific Ocean. This research was also supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC02-05CH11231, by the Division of Mathematical Sciences of the National Science Foundation, while some computations used the resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

Finally, I have the pleasure of thanking my family – my dad, Rex, who sadly will not see me graduate, but was always proud of my work, my mum, Jane, and my sisters, Vickie and Katherine – you're awesome.

Chapter 1

Introduction

1.1 Multiphase problems

A myriad of science and engineering phenomena involve multiple interacting regions whose geometries change over time. The interfaces separating these different regions (or "phases") often meet at junctions and the physics taking place in the different regions can depend on a variety of factors, ranging from local geometry of the interface, to global properties transmitted instantaneously across the domain. A familiar example involves the dynamics of a soap bubble foam, in which gaseous bubbles are separated by a network of interconnected thin films of liquid, as shown in Figure 1.1(a). The physics of a foam are complex, and include gas dynamics, surface tension, and liquid dynamics in the network of thin-film membranes. The membranes may also be permeable, in which case bubble

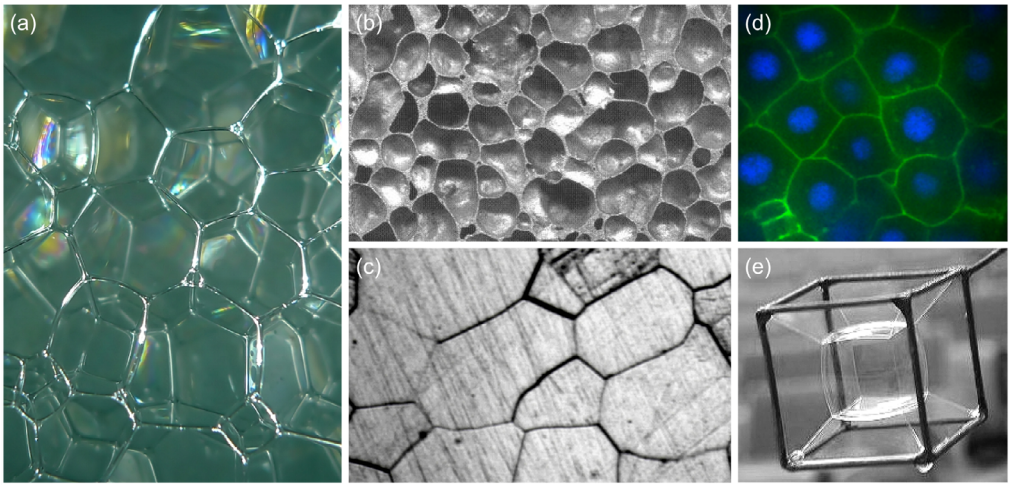

Figure 1.1. Examples of multi-region physics. (a) A soap bubble foam made with common washing detergent. (b) Metallic foam made out of aluminium (reproduced from [6]). (c) Grains in a polycrystalline metal (reproduced from [7]). (d) Cells in a zebrafish, stained so that the cell membranes are green and nuclei blue (reproduced with permission from [8]). (e) Minimal surfaces formed by steady-state soap films.

geometry can change over time as gas diffuses across the membranes, owing to pressure differences caused by surface tension. Highly specialised foams and foam-like materials, made with different gas and liquid species, as well as solidification processes, have a wide variety of applications in materials science, engineering, and industry. For example, metallic foams, such as the closed-cell aluminium foam shown in Figure 1.1(b), are used in crash-absorbent materials and in medical applications to fill voids in bones. Understanding the dynamics of foam evolution is a key step in controlling the shape, structure, and mechanical properties of these foam-like materials. Another example of a multiphase system is in grain growth. Many metal and ceramic materials exhibit grains, which are regions of different crystal orientation or alignment, see Figure 1.1(c). Based on chemistry and thermodynamics, grain boundaries move to reduce their interfacial energy, and predicting how these grains form and evolve is an important part of manufacturing special-purpose materials. Other multi-region problems include studying the formation of cellular tissues in biology, such as the ectoderm cells in zebrafish in Figure 1.1(d) [8]; multi-region shape optimisation problems, which have applications in chemistry, engineering, and high performance computing; and more mathematically-oriented problems, such as examining shapes of minimal surfaces, as illustrated in Figure 1.1(e).

1.2 Tracking multiple interfaces

Modelling these multiphase problems requires the ability to track the motion of evolving interconnected interfaces, coupled to the physics taking place in each phase. However, this is not always straightforward, as there are a number of associated mathematical and computational challenges. Often, the physics, chemistry, and mechanics that drive the interface motion are complex, requiring the solution of fluid mechanics equations with intricate jump conditions at the interface, elasticity solvers with different properties in each membrane, diffusion and transport effects of species both within and across the region boundaries, etc. It has been a challenge to construct well-posed mathematical models that treat the motion of the interface, especially at junctions, in a mathematically consistent fashion. At the same time, developing accurate and robust numerical methodologies for the interface motion can be difficult, due in part to the presence of junctions where interfaces meet and the vast number of different ways these interfaces can change topology.

1.2.1 Mathematical considerations

From a mathematical perspective, formulating a consistent model is a significant part of the challenge. In general, the motion of triple points in two dimensions and triple lines in three dimensions (as well as objects with higher degrees of connectedness) must be specified, either explicitly or implicitly through requirements on the curves (in two dimensions) and surfaces (in three dimensions) that connect them. Examples of how such motion may be specified for a purely geometric motion given by curvature flow in two dimensions include:

- Employing curvature flow on each interface, with the additional requirement that triple points remain fixed in time: in this case, the equilibrium solution is a network of straight lines.

- Another option might allow triple points to move in order to minimise the total length of the network of two-dimensional curves connecting the triple points: this motion would satisfy Young's law for triple point angles. (In the simplest scenario, Young's law states that triple points make 120° angles.) This case is of physical relevance, since various physical situations demonstrate triple point angle conditions, including grain growth and soap bubble foams. In this situation, generally speaking, quadruple points are unstable: they will quickly destabilise into two nearby triple points.
- A third option might be to disallow triple points and only allow quadruple points: this could lead to an equilibrium solution that resembles a quadrilateral mesh.

In comparison to corners that might exist in an otherwise smooth and orientable closed curve or surface, the junctions in a multiphase interface problem represent a stronger singularity. Corners in an evolving interface can be handled in a smooth fashion by relying on a well-posed notion of surface evolution that allows such singularities (such as in *level set methods*, as discussed in Chapter 2). The goal of a mathematical formulation of multiphase interface evolution is to smoothly handle the junctions, especially in the cases which apply to physical systems.

1.2.2 Numerical goals

From a computational perspective, a numerical method for tracking multiple evolving interfaces should have several features. Numerical methods should have the ability to:

- Represent the complex configurations and geometry of the junctions often present in multiphase systems. As shown in the examples in Figure 1.1, the geometry of multiphase systems is often complex, with a large number of ways that interfaces can meet at junctions.
- Robustly handle complex topological changes. When phases are created, change connectivity with other phases, or collapse and disappear, intricate topological changes take place in the interface, and these should be robustly handled, despite the vast number of ways topological changes can take place.
- Couple to physics in the most general fashion possible. It is important to include a physical notion of "time" with the flexibility of prescribing different types of possible motion.
- Accurately calculate geometric properties of the interface, such as the normal direction and components of curvature.
- Be applicable to both two- and three-dimensional settings. Ideally, the fundamental formulation should be independent of the spatial dimension.

Finally, numerical methods should be accurate in both space and time, as well as efficient.

1.3 Previous work

A variety of numerical methods have been proposed to track evolving interfaces in multiphase problems. Somewhat broadly, the major approaches can be divided into the following classes of methods:

- **Front tracking or marker-particle methods.** In a front tracking method, the interface is explicitly represented with a Lagrangian geometry, usually with connected line segments in two dimensions and triangular meshes in three dimensions. The positions of the elements making up the front are updated in time, based on local geometry of the interface and prescribed velocity fields. These methods have been adapted to multiphase systems by allowing junctions to share multiple interface elements. For example, a triple point in 2D is connected to three line segments, while a triple line segment in 3D is connected to three triangles. While conceptually simple, a drawback of this method is that whenever topological changes in the interface occur, some kind of explicit surgery is required to create the surface mesh topology. Surgery mechanisms are often ad-hoc, and are especially difficult to perform in three dimensions.

  One of the most widely used implementations of front tracking is in the *Surface Evolver* software [9]. In Surface Evolver, surfaces are moved in order to minimise a user-defined energy functional, using gradient descent methods, and topological changes in the surface are handled with explicit surgery techniques. While such an approach is predominantly aimed at finding steady-state configurations and cannot be directly coupled to physics, it has been used in a wide variety of contexts, e.g. in studying bubble size distribution in foams [10, 11], grain-growth [12], and multi-cellular tissues in biology [8, 13, 14].

  Another well-known method that uses explicit Lagrangian geometry is the Immersed Boundary Method [15], primarily aimed at fluid-structure interaction problems. This approach has recently been extended to study two-dimensional dry foam dynamics [16, 17], allowing shear dynamics to be studied in a full Navier-Stokes setting.

  Other front tracking approaches include two-dimensional multiphase curvature flow where Young's law is used as a boundary condition at triple points [18], and in modelling surfactant flow in a network of moving curves [19].

- **Volume-of-fluid methods.** In a volume-of-fluid method, each phase or region is described by a characteristic function, which has a value of one inside the phase and zero outside [20]. In discrete form, this leads to volume fractions in each grid cell which specify the fraction of volume occupied by each phase. In the multiphase case, junctions correspond to cells with more than two phases [21, 22]. To update these volume fractions in time, algorithms are used to locally reconstruct the interface by using neighbouring volume fraction values. However, reconstructing the interface at junctions can be difficult, especially in three dimensions. Volume-of-fluid methods have exact conservation of mass, but geometry of the interface, such as the normal vector and curvature, are difficult to evaluate accurately.

- **Methods using multiple level set functions.** Level set methods, introduced by Osher and Sethian [23], embed the interface as the zero level set of an implicitly defined signed distance function which evolves in time. As a result, they are primarily aimed at two-phase problems in which the interface separates one region from another. In the multiphase case, multiple level set functions are used, one for each each phase/region (or some other encoding scheme). Since the boundary of one phase coincides with the boundary of another phase, these level set functions must be coupled together correctly in order to avoid creating regions of overlap or voids. This is typically done through a "repair" procedure at the end of every time step to reconcile the level set functions, ensuring consistency.

  The first such method using this approach was introduced by Merriman *et al.* [24]. Later approaches include using a type of projection method that projects the values of the multiple level set functions to a specific manifold that does not allow vacuums or overlaps [25], and adapting particle level set methods to many regions, also using a type of projection [26]. However, it is often the case that the repair procedure can alter the dynamics of the multiphase motion. For example, in the works of [24–26], accurate evolution of triple junctions has not been rigorously demonstrated.

  In image segmentation problems, accurate time evolution is not needed, and different encoding schemes have been used. For example, $\lceil \log_2 n \rceil$ level set functions can be used to represent n phases, by using the sign of the level set functions in a base-2 encoding scheme to determine the phase [27]. This eliminates void regions/overlaps when the number of phases is exactly a power of two, however spurious phases can appear otherwise. Another image segmentation approach uses a single level set function which is piecewise constant, such that each phase has a different value [28].

- **Variational methods.** Variational methods derive their evolution from gradient descent on an energy functional, such that "time" measures the progress towards the desired minimisation configuration, similar to solving a parabolic equation to obtain the solution to an elliptic problem. These methods are often implemented in a level set framework for an evolving interface that moves to decrease the energy functional [29, 30]. Typical energy functionals take the form

$$E = \sum_{i,j} f_{ij} |\Gamma_{ij}| + \sum_i e_i |\Omega_i|,$$

  where $|\Gamma_{ij}|$ denotes the surface area of the interface between phase i and phase j, and $|\Omega_i|$ is the volume of phase i. Here, f_{ij} is a surface energy density and e_i is the bulk energy of phase i. Gradient descent on this energy functional leads to a normal speed v_{ij} of interface Γ_{ij} given by $v_{ij} = f_{ij}\kappa_{ij} + e_i - e_j$, where κ_{ij} is the mean curvature of Γ_{ij}. In a variational method, multiple level set functions are again employed, and, instead of repairing gaps or overlaps that may occur from the decoupled motion of individual phases, a penalty function is employed which inhibits gaps or overlaps from growing beyond some tolerance [29–31]. These variational methods are limited to geometric motion and are difficult to couple to time-resolved physics.

- **Phase-field models.** Reaction-diffusion or phase-field models model the interface as a thin internal boundary/transition layer of small width, the evolution of which is governed by potential energy functions that have special local minima, see [24, 32]. Since the interface motion and triple point angle conditions are sensitive to the choice of potential function, these methods are often special purpose. In addition, the interfacial layer must be resolved sufficiently well for numerical accuracy, which typically requires specialty techniques such as adaptive mesh refinement.

- **Diffusion-generated methods.** Using either a signed distance function for each phase, or the characteristic function for each phase, diffusion-generated methods evolve a multiphase interface by alternating between two steps: first, a diffusion step, corresponding to convolving the phase functions with a kernel, and second, a reconstruction step, in which the functions are rebuilt as signed distance or characteristic functions [24, 33, 34]. These methods are typically limited to mean curvature flow. In fact, it is possible to prove mathematically that a sequence of diffusion-followed-by-reconstruction steps converges to multiphase mean curvature flow in the limit of the step size going to zero [35, 36]. Numerically, these methods can be made unconditionally stable by using implicit schemes for the diffusion step, and, as such, can be used to study long-time behaviour of grain/foam coarsening under curvature flow. For example, in-depth statistics for grain coarsening have been performed for over 100,000 phases in two dimensions [37], and more recently, 130,000 phases in three dimensions [38]. This approach has also been generalised to allow almost arbitrary specification of the surface energy coefficients [39].

- **Other methods and mathematical results.** In the case of prescribed interface speeds, such that each type of interface moves with a constant speed in the normal direction, Taylor [40] has developed a mathematical approach in two dimensions based on a geometric Huygens' principle construction. Uniqueness and existence results are derived, however the method does not lend itself to numerical simulation and there appears to be no natural extension of the method to three dimensions. Under the same type of motion, an alternative theory has been developed, called the vanishing surface tension limit [41], which also establishes uniqueness and existence results.

  Other theoretical work includes finding and classifying all types of two-dimensional self-similar shrinking, expanding, and stationary curvature flow in multiphase systems [42]; theoretical results for three-dimensional self-similar grain growth [43]; and existence and uniqueness results for curvature driven motion with the phase-field model given by a Allen-Cahn system [44].

While these previous approaches have been successful in specific cases, none meet all of the above mathematical and numerical goals. For example, front tracking methods which use Lagrangian geometry, while efficient, require delicate dimension-dependent programming and ad-hoc surgery methods during topological change; the use of multiple level set functions offers robustness and the ability to automatically handle topological changes, but these methods have rarely been shown to

accurately evolve junctions in the correct way; and variational methods, phase-field models, and diffusion-generated methods are each limited to mostly geometric types of motions, making coupling to time-resolved physics difficult.

1.4 New contributions

One of the main contributions of this thesis, developed jointly with Sethian, is a general method for tracking the interface in multiphase problems, called the *Voronoi Implicit Interface Method* (VIIM) [1, 2]. It aims to meet all of the above mathematical and numerical goals. In particular, by using a geometric construction based on the Voronoi diagram, and the theory of implicit interface methods, the VIIM provides a mathematically consistent framework for defining the motion of a multiphase interface. Numerically, the VIIM accurately tracks the motion of multi-junction interfaces (in both space and time), automatically handles complex configurations and topological changes in the interface, accurately calculates geometric quantities, and can be coupled to time-resolved physics. Moreover, this formulation is independent of the number of spatial dimensions, allowing numerical methods to be developed for both two-dimensional and three-dimensional problems simultaneously. In this thesis, the central ideas behind the VIIM are presented, its implementation is discussed, and convergence tests are performed to illustrate its accuracy. Several example applications of the VIIM are also provided, including constant speed normal driven flow, advection, and multiphase curvature flow with constraints. In addition, new methods are developed for treating surface tension forces at junctions in multiphase fluid flow problems, and fluid solvers are developed in which density, viscosity, and surface tension can be specified on a per-phase basis.

The second main contribution of this thesis is a new multiscale model for the dynamics of a soap bubble foam. In this model, the idea of "scale separation" is used to couple macroscale gas dynamics and rearrangement of bubbles in a foam to film rupture and microscale fluid flow in the foam's network of thin-film membranes ("lamellae") and junctions ("Plateau borders"). New thin-film equations for curved lamellae and Plateau borders are derived, and these are coupled together with flux boundary conditions which conserve liquid mass. From a numerical perspective, several new numerical methods are developed to solve these coupled systems of partial differential equations (PDEs), including Lagrangian-based schemes for conservative transport of fluid during rearrangement, finite element methods for solving fourth-order nonlinear PDEs on curved surfaces and junctions, methods to couple discrete solutions of the thin-film equations at Plateau borders and quadruple points, and gas dynamics coupled to the VIIM for bubble rearrangement. Combined together, the multiscale model leads to a computational framework for studying the interlinked effects of drainage, rupture, and rearrangement in a foam of bubbles.

A third contribution of this work is the development of a new high-quality mesh generation algorithm for interconnected surfaces. Based on a geometric construction used in the VIIM, the algorithm can be used to automatically and efficiently generate high-quality triangulated meshes which are topologically consistent, guaranteeing that mesh elements at junctions meet precisely without any gaps or overlaps. In this approach, force-based iterative methods are adapted from *DistMesh* [45] and combined with a new locally-adaptive time stepping scheme for increased efficiency. This

algorithm has been used in the above multiscale model for foam dynamics, where it has been invoked tens of thousands of times to generate meshes of evolving interconnected surfaces.

1.5 Outline

This thesis is organised into three parts. In the first part, a review of level set methods is given in Chapter 2 and serves as a foundation for many of the ideas and numerical technologies used in the VIIM. The general VIIM methodology is then presented in Chapter 3, together with a discussion of its numerical implementation and convergence tests verifying the accuracy of the method. Several applications of the VIIM in geometric flow and multiphase fluid flow are given in Chapter 4. Chapters 3 and 4 have been adapted from the two papers

- R. I. Saye & J. A. Sethian, The Voronoi Implicit Interface Method for Computing Multiphase Physics, *PNAS*, **108**(49), 19498–19503 (2011),
- R. I. Saye & J. A. Sethian, Analysis and applications of the Voronoi Implicit Interface Method Method, *J. Comp. Phys*, **231**(18), 6051–6085 (2012),

and include expanded discussions on the motivation behind the VIIM as well as additional applications. In the second part, Chapter 5 presents the meshing algorithm for interconnected surfaces, adapted from [4] which is currently under review. In the last part, the multiscale model of foam dynamics is derived and discussed in Chapter 6. Numerical methods for the multiscale model and results are then presented in Chapter 7. Chapter 6 and some of the results of Chapter 7 have been adapted from the paper

- R. I. Saye & J. A. Sethian, Multiscale Modeling of Membrane Rearrangement, Drainage, and Rupture in Evolving Foams, *Science*, **340**(6133), 720–724 (2013).

Finally, Chapter 8 concludes with a summary of the VIIM and future directions of research in multiphase problems.

Chapter 2

The Level Set Method

As part of the development of the Voronoi Implicit Interface Method, several ideas from the theory and practice of level set methods are adopted. Here, we briefly review the level set method, its numerical implementation, and some of its mathematical theory. For more details, a general review of the level set method and associated numerical algorithms can be found in the books [46, 47].

2.1 Introduction

First introduced by Osher & Sethian [23], the level set method provides a robust and accurate technique for tracking interfaces that move under a variety of speed laws. By embedding the interface as an isosurface of a "level set function", level set methods recast the problem of moving the interface into a time-dependent PDE for evolving the level set function. In doing so, the method leads to many advantages, including the ability to automatically handle topological changes in the interface, as well as handling singularities in the interface by relying on viscosity solutions of the associated PDEs. The method works in any number of spatial dimensions, and can be coupled to speed laws derived from physics, such as two-phase fluid flow.

In more detail, let $\Gamma(t)$ denote the moving interface. Suppose that the interface separates one region (the "inside", denoted as Ω_+) from another region (the "outside", denoted as Ω_-). We thus assume that the interface is an orientable $n-1$ dimensional closed hypersurface in $\mathbb{R}^n$. At time $t = 0$, we embed the interface as the zero level set of the function $\phi : \mathbb{R}^n \times [0, \infty) \to \mathbb{R}$. Usually, this is done by using the signed distance to the interface, i.e.

$$\phi(x, t = 0) = \pm d\big(x, \Gamma(t = 0)\big),$$

where the sign is chosen such that ϕ is positive inside Ω_+ and negative inside Ω_-. In the level set method, we require that the evolving interface continues to be the zero level set of ϕ for all time. It follows that geometric quantities of the interface can be determined through operations acting on ϕ. For example, the unit normal vector of the interface, pointing from Ω_- into Ω_+, is $\hat{\mathbf{n}} = \nabla\phi/|\nabla\phi|$. The requirement that the zero level set of $\phi(\cdot, t)$ coincides with $\Gamma(t)$ leads to an evolutionary PDE for ϕ, as follows: suppose Γ is to move in the normal direction with a speed F, and that $x = x(t)$ is

the position of a particle on the interface. Then $\frac{d}{dt}x = F\hat{\mathbf{n}}$, while $\frac{d}{dt}\phi(x(t),t) = 0$, since the particle is to remain on the zero level set. Applying the chain rule, we thus find that

$$\phi_t + F|\nabla\phi| = 0. \tag{2.1}$$

This is the master evolution equation for the level set function ϕ, and is sometimes known as the *level set evolution equation*. We have thus transformed the problem of determining the motion of the interface into the problem of solving (2.1), since the evolving interface is implicitly embedded as the zero level set of ϕ for all time, i.e. $\Gamma(t) = \{x : \phi(x,t) = 0\}$. This evolution is quite general in the sense that the speed law F can take various forms. For example:

- If the interface is to move with constant speed $F = 1$ in the normal direction, then ϕ satisfies the first order Hamilton-Jacobi equation

$$\phi_t + |\nabla\phi| = 0.$$

- Alternatively, if the interface is to collapse under mean curvature flow, so that $F = -\kappa$, then ϕ is the solution of the second-order nonlinear parabolic PDE

$$\phi_t - \nabla\cdot\Big(\frac{\nabla\phi}{|\nabla\phi|}\Big)|\nabla\phi| = 0.$$

  Here, we have used the fact that the mean curvature of a level set of ϕ passing through a point x is given by $\kappa = \nabla\cdot\hat{\mathbf{n}}$ evaluated at x.

- Or, more simply, if the interface is advected by a given external velocity field $\mathbf{u}$, then ϕ satisfies the advection equation

$$\phi_t + \mathbf{u}\cdot\nabla\phi = 0.$$

More generally, the speed F may be any combination of the above types of flow, or determined by external factors. For example, in a coupled physics problem, the dynamics of the interface is determined by solving PDEs in all of the domain, which can depend on the position and geometry of the interface, jump boundary conditions across the interface, or material quantities on the interface itself. A specific example is given by a two-phase incompressible fluid flow problem with surface tension: the interface between two immiscible fluids is advected by the velocity field of the two fluids, which is governed by the Navier-Stokes equations with a surface tension forcing term that depends on the position and mean curvature of the interface.

Notwithstanding the generality of the level set method, it is worthwhile to consider a simple test case, as follows. This example serves to highlight some of the key mathematical and numerical properties of the level set method, many of which are capitalised upon in the development of the VIIM. Consider the two-dimensional problem of expanding two initially separated circles at unit speed. Eventually, the two circles will intersect and merge together into one. Applying the level set method to this problem, we define a level set function ϕ at time $t = 0$, which is the signed distance to the two-circle interface:

$$\phi(x,y,t=0) = \max\Big(r - \sqrt{(x-x_1)^2 + (y-y_1)^2},\ r - \sqrt{(x-x_2)^2 + (y-y_2)^2}\Big).$$

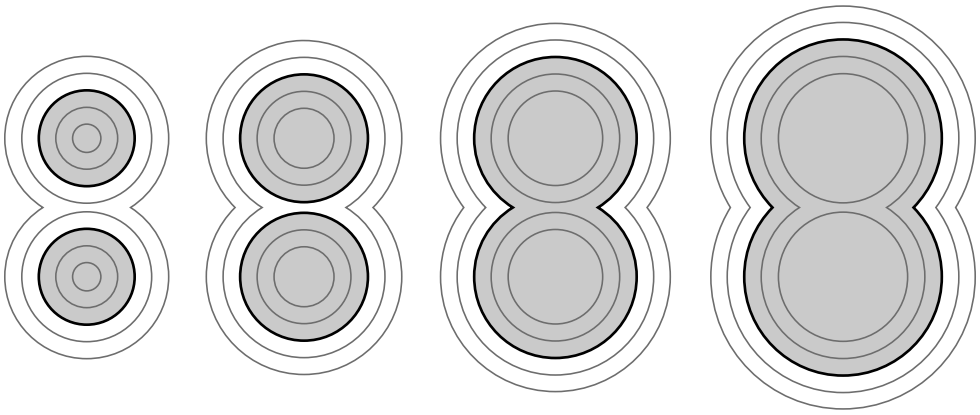

Figure 2.1. From left-to-right, evolution of an interface expanding with unit speed, computed using the level set method. Illustrated is a contour plot of the evolving level set function ϕ, with the zero level set identified by the thick black contour; two additional neighbouring level sets are shown as well, and the shaded region indicates Ω_+.

Here, (x_i, y_i) is the centre of circle i with radius r, and the interior of the two circles gives Ω_+. To determine the evolution, we thus solve the evolution PDE given by $\phi_t - |\nabla\phi| = 0$. Figure 2.1 shows the resulting evolution. As a function of time, a contour plot of ϕ is shown: the thick black curve is the zero level set of ϕ, and the shading illustrates the set of points where ϕ is positive, i.e. Ω_+. Several features in the ensuing evolution can be observed:

- The two circles, while separated at time $t = 0$, merge together into one region, i.e. the interface changes topology. This is handled automatically by the level set method: there is no explicit decision made at any point in time to determine how to merge the two circles. Instead, the only object being evolved in time is the level set function ϕ, and its evolution is determined uniquely by the Hamilton-Jacobi equation.
- In the level set method, the evolution of ϕ has no dependence on the fact that the zero level set is the object of interest – in fact, every level set of ϕ evolves according to the same speed law. We can see this in Figure 2.1: the zero level set, as well as neighbouring level sets, all expand with unit speed in the normal direction.
- In particular, and following the last point, we can see that the motion of the neighbouring level sets of ϕ bracket and follow the motion of the zero level set. Thus, if for whatever reason the location of the zero level set cannot be found, the location of nearby level sets can be used to approximately reconstruct the position and motion of the zero level set. This property is particularly pertinent in developing the VIIM, and will be commented on later.

An additional important property of the level set method is that it is essentially dimension-independent: once an $n-1$ dimensional hypersurface is embedded in the n-dimensional level set function ϕ, the evolution equation determines the motion, independent of explicit geometric considerations such as the representation of a curve as a polygon or a surface as a triangular mesh. Thus, independent

of the spatial dimension of the problem, the level set method automatically handles topological changes in the interface and is flexible in specifying different types of speed laws.

2.2 Mathematical properties

Before turning to numerical methods for the level set method, it is worthwhile to mention the mathematical theory of the method, particularly in connection with the general problem of surface[1] evolution. In one of its most general forms, the level set evolution equation (2.1) can be written as

$$\phi_t + F(x, t, D\phi, D^2\phi) = 0, \quad (x, t) \in \Omega \times (0, T). \tag{2.2}$$

Here[2] , $F : \Omega \times [0, T) \times \mathbb{R}^n \times S^n \to \mathbb{R}$ is a general "speed function" that may depend on position, time, and the gradient and second derivatives of ϕ (with respect to x). The function F encodes, in a very general form, the desired motion of the interface.[3] Often, F arises specifically from a geometric law of motion, as follows: suppose the surface is to move in the normal direction with a given speed function $V = V(x, t, \hat{\mathbf{n}}, \nabla\hat{\mathbf{n}})$, depending on space x, time t, the normal vector of the surface $\hat{\mathbf{n}}$, and curvature information embedded in $\nabla\hat{\mathbf{n}}$ (such as mean and Gaussian curvature). Then in a level set formulation, the normal vector is given by $\hat{\mathbf{n}} = \nabla\phi/|\nabla\phi|$, and $\nabla\hat{\mathbf{n}}$ can be calculated from a certain transformation acting on $D^2\phi$ (see [48] for details). Given this V, the resulting F used in (2.2) takes the following form: whenever $p \in \mathbb{R}^n \setminus \{0\}$,

$$F(x, t, p, X) = |p| V\Big(x, t, \tfrac{p}{|p|}, \tfrac{1}{|p|}(I - \tfrac{pp^T}{|p|^2})X(I - \tfrac{pp^T}{|p|^2})\Big). \tag{2.3}$$

For example, in the case of mean curvature flow, $V = -\kappa = -\nabla \cdot \hat{\mathbf{n}} = -\mathrm{tr}(\nabla\hat{\mathbf{n}})$, which leads to $F = -\mathrm{tr}\big((I - \frac{pp^T}{|p|^2})X\big)$. Any F arising from a speed V, as given by (2.3), automatically satisfies certain scaling properties relating to the fact that F is geometric in nature, and it is these properties which are important in the theory of level set equations as they apply to surface evolution. Specifically, for a general speed function F, we make the following definition.

Definition 1. *We say that the function F is* strongly geometric *if it satisfies two properties:*

- $F(x, t, \lambda p, \lambda X) = \lambda F(x, t, p, X)$*; and*
- $F(x, t, p, X + pq^T + qp^T) = F(x, t, p, X)$*,*

for all $\lambda > 0$, $p \in \mathbb{R}^n \setminus \{0\}$, $X \in S^n$, and $q \in \mathbb{R}^n$.

In order to develop existence and uniqueness results, a rich theory of viscosity solutions of (2.2) has been developed [48]. To do this, it is generally assumed that F also satisfies an ellipticity property.

[1] In this section, "surface" means an $n-1$ dimensional surface in $\mathbb{R}^n$.

[2] S^n denotes the space of real symmetric $n \times n$ matrices; $D\phi$ and $D^2\phi$ denotes the gradient and Hessian of ϕ with respect to x.

[3] The mathematical theory of the level set method is mainly concerned with speed functions that depend on ϕ and its derivatives only. When ϕ is coupled to external PDEs, as in coupled physics problems, theoretical results become more problem-specific.

Definition 2. *We say that F is* degenerate elliptic *if for each $x \in \mathbb{R}^n$, $t \in [0, \infty)$, $p \in \mathbb{R}^n \setminus \{0\}$, we have that*

$$F(x, t, p, X) \leq F(x, t, p, Y)$$

for all $X, Y \in S^n$ with $X \geq Y$. Here, $X \geq Y$ means $X - Y$ is a positive-semidefinite matrix. If F is degenerate elliptic, then we say that (2.2) *is* degenerate parabolic.

Roughly speaking, if F is degenerate elliptic, then existence and uniqueness of solutions to (2.2) are guaranteed; if F is also strongly geometric, then these solutions correspond to evolving surfaces. As examples, a surface moving in the normal direction with constant speed has $F(x, t, p, X) = |p|$, which is strongly geometric and trivially degenerate elliptic. For mean curvature flow, using the fact that $\mathrm{tr}(AB) \geq 0$ if $A \geq 0$ and $B \geq 0$, and that $I - \frac{pp^T}{|p|^2} \geq 0$, it follows that $F(x, t, p, X) = -\mathrm{tr}\big((I - \frac{pp^T}{|p|^2})X\big)$ is also degenerate elliptic, and it is easy to show that it is strongly geometric.

The theory of level set equations shows how, even in the cases that F involves singularities (as it does, for example, in mean curvature and Gaussian curvature flow), that the problem of evolving a surface via the level set equation (2.2) is well-posed. The general procedure is to (i) define an initial level set function ϕ_0, which is continuous in Ω, such that its zero level set corresponds to the initial position of the interface Γ; (ii) solve the Cauchy problem (2.2) with initial value $\phi(x, t = 0) = \phi_0$; and then (iii) *define* the solution to the surface evolution problem as $\Gamma(t) = \{x \in \Omega : \phi(x, t) = 0\}$. Showing that this procedure makes sense requires mathematical proof and is the subject of many papers. Here, the main results are summarised. Suppose that

- the initial condition ϕ_0 is uniformly continuous,
- the function F is strongly geometric, degenerate elliptic, and continuous on $\bar{\Omega} \times [0, T] \times (\mathbb{R}^n \setminus \{0\}) \times S^n$,

then the level set method has the following properties:

(i) A global solution to (2.2) exists and is unique, in the sense of the viscosity solution.

(ii) The evolution of the surface $\Gamma(t)$ is independent of the choice of the initial level set function: if ϕ_0^A and ϕ_0^B are continuous and their zero level sets coincide, then, while the solution to the Cauchy problem (2.2) with initial conditions ϕ_0^A and ϕ_0^B may be different, the evolving surface is identical: $\Gamma(t) = \{x \in \Omega : \phi^A(x, t) = 0\} = \{x \in \Omega : \phi^B(x, t) = 0\}$.

(iii) The motion of each level set is independent of all other level sets and the value of the level set. In particular, all level sets of the initial condition move under the *same* speed law.

For precise mathematical statements of these properties, as well as an in-depth study and review of the theory of surface evolution using level set methods, the reader is referred to [48].

2.3 Numerical methods

In the mathematical theory of the level set method, we saw that recasting the problem of moving an interface into the problem of evolving a level set function leads to many advantages, such as generality and well-posedness of interface evolution. These advantages carry over to a numerical setting. In particular, implementations of the level set method can take advantage of a wealth of well-established numerical methods for solving the associated Hamilton-Jacobi and parabolic PDEs. These methods rely in part on the links between interface propagation and hyperbolic conservation laws [49], and build appropriate schemes to perform shock capturing and upwinding.

A numerical implementation of the level set method involves discretising the level set equation on a background grid/mesh, together with standard methods to evolve in time, such as the forward Euler method. Although the method can work in a variety of discrete settings, a common approach is to use a regular Cartesian grid with cell size $\Delta x \times \Delta y \times \Delta z$ together with finite difference methods. Here, we state some of the simplest schemes for solving the level set equation. These are written in three dimensions, with a grid point indexed by ijk and time step labelled by superscript[4] n; adaptation to different spatial dimensions is straightforward.

- For the Hamilton-Jacobi equation $\phi_t + F|\nabla\phi| = 0$, where $F = F(x,t)$ does not depend on derivatives of ϕ, one of the simplest first-order schemes is

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} - \Delta t\big[\max(F_{ijk},0)\nabla^{+} + \min(F_{ijk},0)\nabla^{-}\big],$$

  where

$$\begin{aligned}\nabla^{\pm} = \big(&\max(D_{ijk}^{\mp x},0)^2 + \min(D_{ijk}^{\pm x},0)^2 \\ &+ \max(D_{ijk}^{\mp y},0)^2 + \min(D_{ijk}^{\pm y},0)^2 \\ &+ \max(D_{ijk}^{\mp z},0)^2 + \min(D_{ijk}^{\pm z},0)^2\big)^{1/2},\end{aligned}$$

  and $D_{ijk}^{\pm}$ is short-hand for the standard forward and backward finite differences, e.g. $D_{ijk}^{+x} = (\phi_{i+1,j,k} - \phi_{ijk})/\Delta x$, etc. For stability, the CFL condition for this scheme requires that $F_{ijk}\Delta t \leq \min(\Delta x, \Delta y, \Delta z)$ for all grid points ijk.

- A first-order upwinding scheme for the advection equation $\phi_t + \mathbf{u}\cdot\nabla\phi = 0$ with $\mathbf{u} = (u,v,w)$ is

$$\begin{aligned}\phi_{ijk}^{n+1} = \phi_{ijk}^{n} - \Delta t\Big(&\max(u_{ijk}^{n},0)D_{ijk}^{-x} + \min(u_{ijk}^{n},0)D_{ijk}^{+x}\Big) \\ &+ \max(v_{ijk}^{n},0)D_{ijk}^{-y} + \min(v_{ijk}^{n},0)D_{ijk}^{+y} \\ &+ \max(w_{ijk}^{n},0)D_{ijk}^{-z} + \min(w_{ijk}^{n},0)D_{ijk}^{+z}\Big).\end{aligned}$$

  A CFL condition for this scheme is to require $\max_{ijk}|\mathbf{u}_{ijk}|\Delta t \leq \min(\Delta x, \Delta y, \Delta z)$.

[4] The superscript n denoting the time step is also used to denote the spatial dimension; distinguishing these should be clear from context.

- For the second-order parabolic PDE for mean curvature flow, $\phi_t - \nabla \cdot (\frac{\nabla \phi}{|\nabla \phi|})|\nabla \phi| = 0$, no shock capturing schemes are required. Instead, the mean curvature can be calculated with central finite difference methods. One possibility for evaluating κ is by using expanded expressions, such as the two-dimensional formula

$$\kappa = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}},$$

  together with approximation of each of the terms using standard finite differences. An alternative method, which works in any number of spatial dimensions, is as follows. First, the unit normal $\hat{\mathbf{n}}$ is evaluated at the centre of each computational grid cell, by evaluating $\nabla \phi$ at $i + \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2}$ using standard central finite differences, and normalising to approximate $\hat{\mathbf{n}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$. Second, calculate $\nabla \cdot \hat{\mathbf{n}}$ at grid points, using standard central finite differences operating on $\hat{\mathbf{n}}_{i\pm\frac{1}{2},j\pm\frac{1}{2},k\pm\frac{1}{2}}$. This results in a finite difference scheme for evaluating $\kappa_{ijk} = \nabla \cdot (\frac{\nabla \phi}{|\nabla \phi|})$, and a forward Euler method yields

$$\phi_{ijk}^{n+1} = \phi_{ijk}^n + \Delta t\, \kappa_{ijk}^n |\nabla \phi|_{ijk}^n$$

  where $|\nabla \phi|_{ijk}^n$ can be calculated either by averaging the gradients found in the first part above, or by using central finite differences. For this forward Euler implementation, roughly speaking, the time step should satisfy $\Delta t < h^2/2N$ for stability, where h is the smallest grid cell size and N is the number of spatial dimensions.

These represent some of the simplest schemes for solving the level set equation. More involved schemes include: essentially non-oscillatory and weighted essentially non-oscillatory (ENO/WENO) schemes that yield high spatial accuracy whenever the solution is smooth with reduced numerical diffusion at shocks; high order total variation diminishing Runge-Kutta (TVD-RK) methods for increased accuracy in time; and implicit/semi-implicit schemes for increased stability. For more details of these methods, the reader is referred to [46, 47] and the references therein.

In a straightforward implementation of the level set method, the above level set updates are computed at every grid point in the domain. A useful technique for improving the efficiency of the level set method results from "narrow banding", first introduced in [50]. In this approach, the computation of the level set updates is confined to a narrow band surrounding the zero level set of ϕ; usually this narrow band has a fixed number of grid cells on either side of the interface (say, 10 grid cells). Depending on interface geometry, the number of grid points inside the narrow band can be significantly fewer than the total number of grid points in the computational domain, leading to considerable improvements in efficiency. However, narrow banding requires a more complicated implementation and needs some type of narrow band data structure. One must also be careful to ensure that numerical artifacts created at the boundary of the narrow band do not artificially affect the motion of the interface – avoiding this problem usually entails periodically reinitialising/reconstructing both the level set function and its narrow band. This can be done with a reinitialisation method, which is discussed now.

2.4 Reinitialisation

Although the mathematical theory of level set methods allows for quite general level set functions, in practice, it is often the case that using signed distance functions gives the best results. This is a natural choice, since the signed distance function is well-behaved, in the sense that level sets are equidistant, and knowing the distance to the interface can be advantageous in some problems. For example, in a two-phase fluid flow problem, surface tension can be written as a body force through the use of a Dirac delta function supported on the interface. In a numerical implementation of surface tension, the delta function is "smoothed" onto the background grid using the level set function ϕ; ensuring that the amount of smoothing is consistent across the entire domain requires that ϕ is approximately a signed distance function.

However, for many types of flow, it is rarely the case the level set function stays a signed distance function. One possible solution is to modify the speed function or velocity field, so that ϕ remains a distance function – this can be done through the use of "extension velocities", which are discussed in the next section. A second solution is to periodically *reinitialise* ϕ as a signed distance function. In a reinitialisation algorithm, the zero level set of ϕ, which is implicitly defined by some sort of interpolation of the values of ϕ at grid points, is located and used to calculate the signed distance to $\{\phi = 0\}$ on the background grid. The computed signed distance function is then used to replace ϕ.

Several possibilities exist for numerical algorithms to reinitialise a level set function as a signed distance function. These range from iterative procedures that solve an auxiliary PDE which attempts to freeze the location of the interface, and whose steady-state solution is a signed distance function, to highly accurate methods that use high-order polynomial interpolation to locate the interface. Here, a high-order and successful reinitialisation method developed by Chopp [51] is briefly described. It uses a combination of bicubic (in 2D) and tricubic (in 3D) interpolation, a Newton-based procedure for finding closest points, and the Fast Marching Method [52]. The method essentially consists of two steps: given a sufficiently smooth scalar function $\psi : \mathbb{R}^n \to \mathbb{R}$,

1. *Compute the signed distance to* $\{\psi = 0\}$ *in an initial band:* For each grid cell, the sign of ψ at the vertices of the grid cell are examined. If the signs are not all the same, then the grid cell is identified to contain the zero level set of ψ. In each such grid cell, ψ is interpolated using a C^1 bicubic interpolant (in 2D) or tricubic interpolant (in 3D), designed to interpolate ψ at the grid points of the reference cell, using a 4×4 stencil (in 2D) or $4 \times 4 \times 4$ stencil (in 3D). Next, at each of the vertices of the reference grid cell, the closest point to the interpolated interface is found, using a Newton-based iterative method. In more detail, denoting the interpolant by $p(x)$, if x_0 is the reference point (one of the vertices of the reference cell), then at each iteration, two step directions are calculated with

$$\delta_1 = -p(x_k)\frac{\nabla p(x_k)}{|\nabla p(x_k)|^2}, \text{ and}$$

$$\delta_2 = (x_0 - x_k) - (x_0 - x_k) \cdot \nabla p(x_k)\frac{\nabla p(x_k)}{|\nabla p(x_k)|^2},$$

and are used to update $x_{k+1} = x_k + \delta_1 + \delta_2$. Here, δ_1 is a step direction which moves the iterate towards the zero level set of p, while δ_2 is a step direction which moves the iterate in a tangential direction in order to find the closest point on the interface. The iterations continue until they converge, measured using some prescribed threshold. For sufficient accuracy, this typically requires as few as 1–5 iterations. Once converged, the closest point from x_0 to the zero level set of p has been found, from which the distance can easily be calculated. For more details on this Newton-based method, see [51].

2. *Execute the Fast Marching Method.* At the end of the first step, the signed distance to the interpolated zero level set of ψ is known in a small initial band of grid points on either side of the interface. This is then input into the Fast Marching Method [52], which is a Dijkstra-like ordered upwind finite difference scheme for efficiently solving the full Eikonal equation outside this initial band. A two-dimensional first order scheme solves

$$\max\left(-D_{ij}^{+x}\phi, D_{ij}^{-x}\phi, 0\right)^2 + \max\left(-D_{ij}^{+y}\phi, D_{ij}^{-y}\phi, 0\right)^2 = 1. \tag{2.4}$$

By capitalising on the flow of information in the Eikonal equation, the Fast Marching Method efficiently solves the upwinded discretisation of the Eikonal equation by marching outwards from the interface, solving (2.4) at each grid point in order of proximity to the interface. Using a min-heap data structure to determine the correct ordering, the computational complexity of the Fast Marching Method is $\mathcal{O}(N \log N)$, where N is the number of grid points in the domain. Furthermore, this approach can easily be used to restrict distance calculations to a narrow band, by stopping the marching once a band radius has been reached. For more details, including higher-order versions, see [46, 51].

In summary, the Chopp algorithm, in combination with a high-order Fast Marching Method, can compute a third-order[5] accurate signed distance function whenever the interface is smooth, and is robust and first-order in the case that the interface has sharp corners.

2.5 Extension velocities

In a level set method, the speed function F must be defined everywhere in the domain, or at least in a narrow band. However, in some moving interface problems, the speed function may only be known at the interface itself, in which case it is necessary to define and build a speed function away from the interface. Such an "extension velocity", first introduced in [53], can be beneficial in a numerical implementation of the level set method. For example, in some fluid flow problems, the velocity of the fluid away from the interface can be quite different to the velocity at the interface, which may cause irregularities in the evolving level set function. In this case, extending the fluid velocity off

[5] Distances calculated in the initial band are third-order accurate whenever the interface is smooth, and first-order accurate at corners. The accuracy of the Fast Marching Method determines the global accuracy of the signed distance function. In particular, if a second order Fast Marching Method is used to calculate the signed distance function in a narrow band of a fixed number of grid cells on either side of the interface, then the distance is in fact third-order in this narrow band (whenever the interface is smooth).

the interface can improve results, such as improved conservation of mass. Extension velocities can also be used to maintain the signed distance property of the level set function, leading to increased numerical accuracy.

The basic idea is as follows: given a grid point x, an extension speed F_{ext} is defined at x to be the value of the speed $F(\text{cp}(x))$ at the closest point $\text{cp}(x)$ on the interface. Thus, the extension speed is constant along characteristics of the Eikonal equation – these characteristics are straight lines emanating from the interface in the normal direction. If ϕ is a signed distance function, it follows that F_{ext} satisfies the equation

$$\nabla F_{\text{ext}} \cdot \nabla \phi = 0. \tag{2.5}$$

It is possible to show that if F_{ext} satisfies (2.5), and if $\phi_t + F_{\text{ext}}|\nabla\phi| = 0$, then $\frac{\partial}{\partial t}|\nabla\phi|^2 = 0$. Thus, if ϕ is initially a signed distance function, then it remains a signed distance function. An efficient method of constructing such an extension velocity, first developed in [54], is as follows. At each time step:

1. Determine the speed F at the interface via geometry, physics, fluid flow, etc.
2. Given a level set function ψ whose zero level set is the interface to be extended off, extend F by solving (2.5), where ϕ is the signed distance to the zero level set of ψ. This can be calculated efficiently using the same marching approach used in the Fast Marching Method, together with a closest point algorithm, such as the one used by the Chopp algorithm described in the previous section. In more detail:
   (i) At each grid point in a small initial band surrounding the zero level set of ψ, find the closest point to the zero level set of ψ. For example, we could use the bicubic/tricubic interpolation and Newton-based iteration scheme described in the previous section.
   (ii) Use these closest points to evaluate $F_{\text{ext}}(x) = F(\text{cp}(x))$ in the same initial band.
   (iii) Lastly, using the Fast Marching Method idea, march outwards from the initial band, such that each grid point is visited in order of proximity to the interface. At each grid point, (2.5) is solved by using the same upwinded finite difference stencil for the gradient which was found when solving (2.4).

By marching in this fashion, information is efficiently transported away from the interface to build the extension. In the original extension method introduced in [54], step 2(iii) above was performed simultaneously with reinitialising ψ as a signed distance function (even if the resulting signed distance function is discarded at the end of the procedure). An alternative method is to assume that ψ is already a signed distance function, in which case it is straightforward to order grid points by proximity to the interface. Finally, we note that the above procedure calculates an extension speed. Using the same algorithm, it is straightforward to calculate an extension velocity with the same properties. In other words, an extension velocity of $\mathbf{u} = (u_1, u_2, \ldots)$ can be constructed such that $\mathbf{u}_{\text{ext}}$ coincides with $\mathbf{u}$ on the interface, and solves $\nabla u_{\text{ext},i} \cdot \nabla \phi = 0$ at each time step. It is not hard to show that under such an extended velocity field, if ϕ is initially a signed distance function and solves the advection equation $\phi_t + \mathbf{u}_{\text{ext}} \cdot \nabla \phi = 0$, then ϕ remains a signed distance function. For more details on extension algorithms and their uses, see [46, 54, 55].

Chapter 3

The Voronoi Implicit Interface Method

In this chapter, the Voronoi Implicit Interface Method (VIIM) for tracking multiple interconnected interfaces is presented. Starting with the main ideas of the VIIM, the mathematical definition of the method is described, followed by details on its numerical implementation. Convergence is then tested with a number of problems, ranging from two-phase problems to check consistency with the level set method, to multi-junction problems in which Young's law and von Neumann's law are verified. The main features of the VIIM are then summarised, before several applications are given in the next chapter.

3.1 Central ideas

In the previous chapter, we reviewed the basic level set method for evolving an interface which separates two regions. In the level set method, the sign of the level set function determines which region a point x is in. However, in a multiphase problem, where three or more different regions can meet at a junction, it is no longer possible to characterise regions based on the sign of a function. Additional complexities with multiphase problems include:

- Junctions, such as triple points in 2D and triple lines and quadruple points in 3D, represent a strong singularity in the interface.
- Knowing precisely where these junctions move is not always straightforward. For example, in a curvature flow problem, the motion of triple points is intricately coupled to all phases.
- Topological changes in the interface, as phases collapse and change neighbourship with other phases, are difficult to characterise explicitly. For example, the number of different ways topological changes can occur in 3D is vast and difficult to enumerate; in four and higher-dimensions, topological changes are near impossible to describe explicitly.

On the other hand, from the level set method, it is clear that

- Evolving a surface which is orientable, i.e. one in which there is a notion of what is "inside" and "outside" the surface, leads to a well-posed problem.

- Implicitly representing surfaces as level sets of a higher-dimensional function yields many advantages, including the ability to automatically handle topological changes, as well as corners in the interface. Furthermore, by using the level set evolution equation, all level sets move according to the same speed law. This leads to the property that the motion of a level set is bracketed by the motion of the surrounding neighbouring level sets.

These observations are key to the development of the Voronoi Implicit Interface Method. The core idea in the VIIM is to define the motion of the interface in a multiphase system by using the motion of nearby hypersurfaces. These hypersurfaces are obtained as the set of points that are a small but fixed distance away from the interface, and form a collection of individual hypersurfaces that exist solely in one phase. While the physical interface in a multiphase system can have junctions, these nearby surfaces are remarkably well behaved as the system evolves. By using an *unsigned* distance function to the interface, denoted as ϕ, these nearby surfaces can be characterised as the ϵ-level sets of ϕ, where $\epsilon > 0$ is a small parameter. In the VIIM, these ϵ-level sets move, and the motion of the interface is reconstructed from the position of these nearby surfaces, using a simple geometric construction known as the Voronoi interface.

3.2 The Voronoi interface

Recall that the Voronoi diagram of a set of m nodes in $\mathbb{R}^n$ is a decomposition of $\mathbb{R}^n$ into m different cells with the property that all the points in a given cell are closer to one specific node than any other. The boundary between these cells is therefore the set of points that are equidistant to at least two nodes, and no closer to any other node. In more generality, instead of nodes, we may have a collection of non-overlapping regions in $\mathbb{R}^n$. We can still obtain a decomposition of $\mathbb{R}^n$ such that all the points in a particular cell are closer to one particular region than any other; see Figure 3.1 for a two-dimensional illustration. The *Voronoi interface* is defined to be the boundary of these cells, and is denoted by Γ_V.

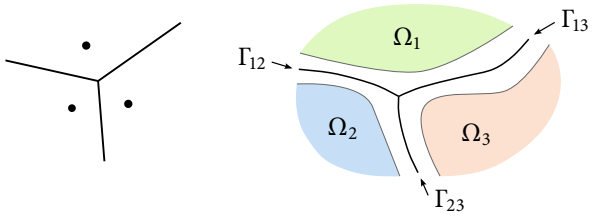

Figure 3.1. (Left) Voronoi diagram of three points in the plane. (Right) The Voronoi interface $\Gamma_V = \Gamma_{12} \cup \Gamma_{13} \cup \Gamma_{23}$ corresponding to three given regions Ω_1, Ω_2, and Ω_3.

3.3 Mathematical formulation

Using the geometric concept of the Voronoi interface and combining the main ideas of the VIIM, the basic approach is as follows. First, the interface of a multiphase system is embedded as the zero level set of the unsigned distance function ϕ to the interface. Thus

$$\phi(x, t = 0) = d\big(x, \Gamma(t = 0)\big).$$

In addition to specifying the distance, we also specify an indicator function χ that identifies which region a point x is in. For example, if x is in "phase one", then $\chi(x) = 1$. We also embed the speed function F that moves the interface with an "extension velocity" defined on and away from the interface. As per the main VIIM idea, nearby level sets of the interface are moved for a small amount of time Δt. This is achieved by solving the level set evolution equation

$$\phi_t + F|\nabla\phi| = 0$$

for a short time Δt. Under this motion, and depending on F and discretisation effects, the zero level set of ϕ will generally not remain a codimension-one surface. However, for a sufficiently small time, nearby level sets (with value $\epsilon > 0$, say) will remain codimension-one surfaces. One can then reconstruct the interface after time Δt as the Voronoi interface of these nearby level sets, and from this, reconstruct the unsigned distance function and indicator function χ. The method is summarised in Algorithm 1.

Algorithm 1 The Voronoi Implicit Interface Method

Given a multiphase system, calculate the initial unsigned distance function ϕ
and indicator function χ.
for time step $n = 0, 1, 2, \ldots$ **do**
Define a speed function F, which may depend on interface geometry, physics, etc.
Evolve the ϵ-level sets of ϕ by solving $\phi_t + F|\nabla\phi| = 0$ for one time step Δt.
Reconstruct ϕ and χ by using the Voronoi interface of the updated ϵ-level sets.

3.3.1 One-dimensional example

To illustrate the method, consider a one-dimensional example, as shown in Figure 3.2. Here, an interface, which is a single point located at $X(t)$ at time t, moves with speed $F(x, t)$, so that $\frac{d}{dt}X(t) = F(X(t), t)$. At $t = 0$, the zero level set of the unsigned distance function ϕ corresponds to the initial location of the point, and is at an extremum. However, at the nearby level sets with value ϵ, the distance function is smooth. We can thus update the distance function everywhere away from the zero level set in a straightforward manner, similar to a level set method. What remains is to obtain a suitable definition of the zero level set at time Δt, in such a manner that it corresponds to the location of the interface at this time. We can do so by *defining* the zero level set as the point equidistant from the two ϵ-level sets. This corresponds to constructing the Voronoi interface from the ϵ-level sets.

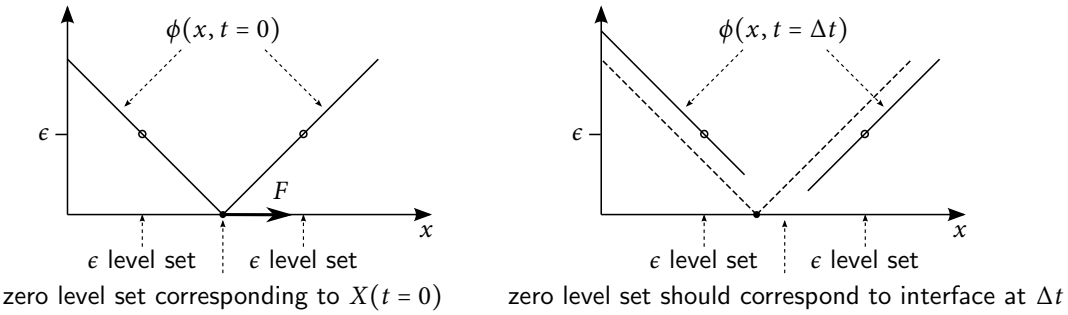

Figure 3.2. Evolution of an unsigned distance function ϕ for an interface in one dimension.

3.3.2 Two-dimensional example

The same Voronoi reconstruction procedure works in two and higher dimensions, and in the presence of junctions. To illustrate, a two-dimensional example is shown in Figure 3.3. In Figure 3.3(a), a three-phase system with a triple junction is shown. The unsigned distance function is then constructed, as illustrated in Figure 3.3(b), which shows a contour plot of ϕ. Nearby surfaces, as determined by the ϵ-level sets of ϕ, are isolated in Figure 3.3(c). Note that the ϵ-level sets are orientable and exist solely in one phase. In this two-dimensional example, curvature flow is used to illustrate motion – after a time Δt, the ϵ-level set belonging to phase two "rounds out" more than the other ϵ-level sets, since it initially had a sharper angle. After a time Δt, Figure 3.3(d) illustrates how the ϵ-level sets have moved. Finally, as per the VIIM idea, the interface is reconstructed from the ϵ-level

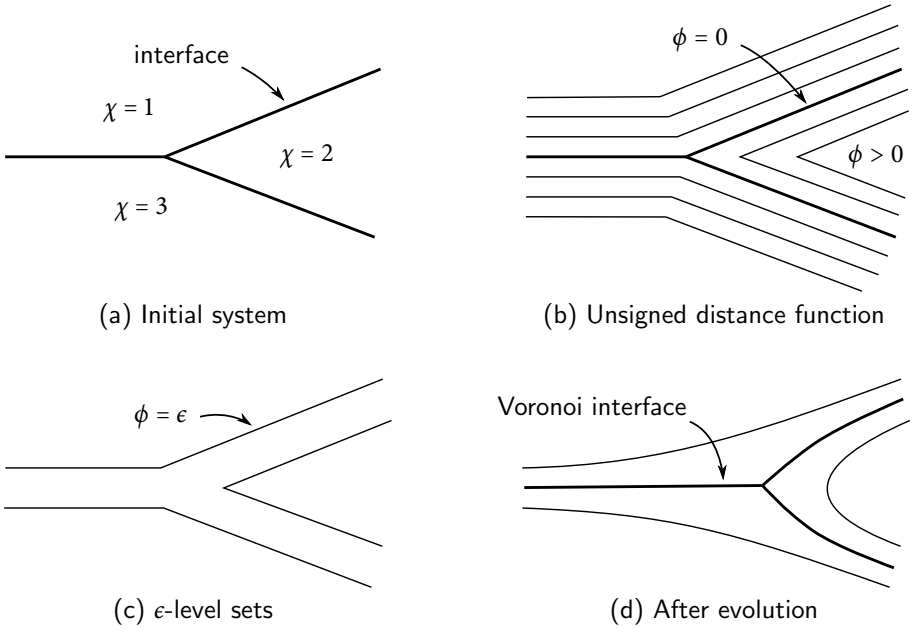

Figure 3.3. Evolution of a three-phase system for curvature flow in two dimensions.

sets using the Voronoi interface, as shown in Figure 3.3(d). Note that the Voronoi interface smoothly reconstructs the junction. Additionally, the triple junction has moved to the right.

3.3.3 Mathematical formulation in multiple dimensions

We can now define the evolution of a multiphase system in any number of dimensions. Let $V_\epsilon(\phi)$ be the operator that reconstructs the unsigned distance function from the ϵ-level sets of ϕ using the Voronoi interface. Let $E_{\Delta t}(\phi)$ be the evolution operator which evolves a given level set function ϕ for a time step Δt. Fix a particular final time $T > 0$ and let n be the number of time steps required to reach that time, so that $\Delta t = T/n$. For some $\epsilon > 0$, we apply n time steps, each step consisting of an evolution and a reconstruction. In the limit as n goes to infinity, this defines an *ϵ-smoothed solution* ϕ_ϵ, given by

$$\phi_\epsilon(t = T) = \lim_{\Delta t \to 0, n \to \infty} (V_\epsilon \circ E_{\Delta t})^n(\phi_0), \tag{3.1}$$

where ϕ_0 is the initial condition. Note that this construction defines ϕ_ϵ at intermediate times $0 \leq t \leq T$ as well as the final time T. In the VIIM, the ϵ-level sets are used as the nearby surfaces to determine the interface evolution. We take the limit of these ϵ-smoothed solutions as $\epsilon \to 0$ from above, to obtain a solution given by

$$\phi_{\epsilon=0^+} = \lim_{\epsilon \to 0^+} \phi_\epsilon. \tag{3.2}$$

The formal definition of multiphase interface evolution, as defined by the VIIM algorithm, is therefore taken to be the solution $\phi_{\epsilon=0^+}$ given in (3.2).

3.3.4 Additional comments

A few mathematical comments are in order at this point. First, the above formulation describes a sequence of problems, one for each $\epsilon > 0$, such that in the limit $\epsilon \to 0$, one obtains the mathematical definition of multiphase evolution. Second, while this formulation applies to multiphase problems with high order junctions (e.g. triple points, triple lines, etc.), it is close to the standard level set method in the case of only two phases.

It is suggested here that the above mathematical formulation leads to a well-posed definition of the solution to multiphase interface evolution. As shown in the results below, it does indeed provide a solution that is physically relevant. This formulation may serve as a possible way to analyse mathematically these flows, however the mathematical analysis is not straightforward. For example, in the theory of level set methods, the comparison principle is a fundamental result in proving various important statements about the well-posedness of surface evolution, such as the statement that if a level set is contained by another level set at time $t = 0$, then it remains contained for all time. For a multiphase problem, this statement does not have an immediate analogue, essentially because of the extra coupling existing at junctions. Developing mathematical theory to accompany the VIIM is the subject of current work.

3.4 Discrete approximations of the $\epsilon = 0^+$ limit

The goal now is to build numerical approximations to the formal definition given by the $\epsilon = 0^+$ limit in equations (3.1) and (3.2). In this section, different approaches of discretising space and time to find the limit are discussed, as well as the key algorithmic components needed to achieve this. Although quite general numerical schemes are possible, here we focus on an implementation that uses rectangular grids, together with standard finite difference approximations to spatial and temporal derivatives. Thus, many of the standard techniques used in level set methods can be adopted. Other approaches, such as finite elements, unstructured meshes, semi-Lagrangian methods, etc., are also possible.

3.4.1 Spatial and temporal discretisations of the formal $\epsilon = 0^+$ limit

The basic numerical approach for the VIIM is as follows. Given a rectangular grid with cell size h, the unsigned distance function ϕ and indicator function χ are defined and evaluated on the grid at time $t = 0$. Once the appropriate speed law is defined, we follow the natural procedure suggested by the VIIM: at each time step, the function ϕ is evolved for a small amount of time Δt, and is then reconstructed as the unsigned distance function corresponding to the Voronoi interface of the ϵ-level sets. In particular, we assume that the time step Δt is suitably linked to h, as is usually required for stability. A numerical implementation of the VIIM contains an additional parameter ϵ, corresponding to the choice of which neighbouring level sets to use to rebuild the unsigned distance function.

Given this discretisation with cell size h and time step Δt, we write our formal definition in more detail in two steps as

$$\phi_\epsilon = \lim_{n\to\infty, h\to 0} (V_\epsilon^h \circ E_{\Delta t}^h)^n(\phi_0), \tag{3.3}$$

followed by the limit

$$\phi_{\epsilon=0^+} = \lim_{\epsilon\to 0^+} \phi_\epsilon. \tag{3.4}$$

Here, V_ϵ^h is the operator that reconstructs the unsigned distance function on a grid with size h (using the Voronoi interface of the ϵ-level sets of ϕ), and $E_{\Delta t}^h$ is the evolution operator that evolves a given level set function on the grid for a time step Δt. The formal definition, given by $\phi_{\epsilon=0^+}$, is the limit as $\epsilon \to 0^+$ of the limit as the space and time discretisations Δt and h go to zero. Here, we discuss several different ways to approximate this limit $\phi_{\epsilon=0^+}$.

(i) *The formal limit of a limit.*

A discretised version of the formal limit given in (3.4) is to fix $\epsilon > 0$, compute a converged (in space and time) solution corresponding to the ϵ-smoothed solution given in (3.3), and then examine the limit as $\epsilon \to 0^+$ to compute $\phi_{\epsilon=0^+}$. More precisely:

- Each choice of ϵ, Δt, and h gives a solution $\phi_\epsilon^{\Delta t,h}$.
- For some fixed ϵ, we compute the converged limit $\phi_\epsilon = \lim_{(\Delta t,h)\to 0} \phi_\epsilon^{\Delta t,h}$.

- We then compute the limit of ϕ_ϵ as $\epsilon \to 0^+$ to obtain $\phi_{\epsilon=0^+}$.

Important aspects of this approach are:

- The physical region between the ϵ-level sets and the multiphase interface is resolved as the grid cell size goes to zero.
- If a narrow band is used to reduce the computational complexity, the size of the narrow band increases as the grid is resolved, since it must cover all of the domain between the ϵ-level sets and the interface.

In the convergence tests below, this approach is used to verify that ϕ_ϵ does indeed converge to a reasonable solution of multiphase interface evolution as $\epsilon \to 0^+$.

(ii) *The coupled limit: couple ϵ and h, such that $\epsilon > 2h$.*

In this approach, ϵ is coupled to h, such that ϵ is a constant multiple of h. This approach sends $\epsilon \to 0$ simultaneously with $\Delta t, h \to 0$, and hence differs from the formal "limit of a limit" in the strict mathematical definition. As a result, the same grid resolution between the ϵ-level sets and the interface is always used. This is, of course, much less laborious than looking at the sequence of converged ϵ-smoothed solutions in the approach (i) above. More precisely:

- Each choice of Δt and h gives a solution $\phi^{\Delta t,h}_{\epsilon(h)}$. Here, we write $\epsilon(h)$ to emphasise that the choice of h determines the value of ϵ.
- We then study convergence as $(\Delta t, h) \to 0$ to obtain $\phi_{\epsilon=0^+}$.

Important aspects of this method are:

- Setting ϵ to be a multiple of h means that the number of grid cells between the interface and the ϵ-level sets remains constant as $h \to 0$.
- Typically, ϵ is chosen to satisfy $\epsilon > 2h$, so that any finite difference stencils used to evolve the unsigned distance function stay completely in one phase and do not reach across the interface.

In the tests below, several test cases for this coupling are used to verify that coupling the two limits in this fashion does, in fact, obtain the same solution as the formal definition.

(iii) *Exchanging limits: find the limit $\phi^{\Delta t,h}_{\epsilon=0^+}$, and then construct the grid/time converged limit as $(\Delta t, h) \to 0$.*

In this approach, the limits in (3.3) and (3.4) are exchanged. More precisely,

- Compute the limit $\lim_{\epsilon\to 0^+} \phi^{\Delta t,h}_\epsilon$ to find $\phi^{\Delta t,h}_{\epsilon=0+}$.
- Now, take the limit as $(\Delta t, h) \to 0$ to produce $\phi_{\epsilon=0+} = \lim_{(\Delta t,h)\to 0} \phi^{\Delta t,h}_{\epsilon=0+}$.

Important aspects of this version are:

- This approach uses the motion of the ϵ-level sets, in the limit $\epsilon \to 0+$. This limit can be evaluated directly rather than through a limiting process.
- One approach is to use one-sided differences to make sure that finite difference stencils used in the level set update stay completely in one phase and do not reach across the interface.
- Alternatively, one can build suitable extensions of the unsigned distance function across the interface. In this fashion, finite difference stencils can reach across the interface without seeing discontinuities in the unsigned distance function. One of the simplest methods to perform this extension is to use the signed distance function of each phase. However, the resulting implementation essentially requires using data structures to represent multiple level set functions, each with their own narrow band; this requires additional programming complexity compared to methods (i), (ii) above, which can use a single unsigned distance function.

In the tests below, it is verified that this approach also obtains the same solution as the formal definition.

3.4.2 Key algorithmic steps

Here we discuss some of the main steps in finding the above discrete solutions. In each of the approaches presented, a key component is rebuilding the unsigned distance function from the Voronoi interface. This can be split into two steps: finding the Voronoi interface, and then rebuilding the unsigned distance function from it.

The Voronoi interface

Given a function ϕ and an indicator function χ, the first step is to characterise the Voronoi interface Γ_V from the ϵ-level sets.

- *Geometric construction*: Consider the hypersurfaces corresponding to the ϵ-level sets, that is

$$\Gamma_\epsilon = \left\{ x \in \Omega : \phi(x) = \epsilon \right\}.$$

These are curves in two dimensions and surfaces in three dimensions. We shall further classify these ϵ-level sets according to the phase in which they are located, and hence define

$$\Gamma_{\epsilon,i} = \left\{ x \in \Omega : \chi(x) = i \text{ and } \phi(x) = \epsilon \right\}.$$

Then, we can define the Voronoi interface Γ_V as the set of all points x that are equidistant to at least two different ϵ-level sets belonging to different phases, and no closer to any other ϵ-level sets. In other words, we define

$$\Gamma_V = \left\{ x \in \Omega : \exists\ i \neq j \text{ such that } d(x, \Gamma_{\epsilon,i}) = d(x, \Gamma_{\epsilon,j}) \leq d(x, \Gamma_{\epsilon,k}) \text{ for all } k \neq i, j \right\}. \tag{3.5}$$

- *Construction via solution of Eikonal equations*: A related formulation is to pose a boundary value Eikonal equation problem, with zero boundary values on the ϵ-level sets. This can be done in one of two ways: one can either pose a single Eikonal equation such that

$$|\nabla \psi| = 1, \quad \psi(x) = 0 \text{ if } x \in \Gamma_\epsilon.$$

  where ψ is negative inside Γ_ϵ (i.e., where $\phi > \epsilon$), and then find the Voronoi interface Γ_V as the maximal ridge of ψ between two different phases. Finding the ridge requires some care. A simpler alternative is to solve the Eikonal equation for each $\Gamma_{\epsilon,i}$,

$$|\nabla \phi_i| = 1, \quad \phi_i(x) = 0 \text{ if } x \in \Gamma_{\epsilon,i},$$

  choosing a solution such that ϕ_i is positive inside $\Gamma_{\epsilon,i}$, and then the Voronoi interface is given by

$$\Gamma_V = \left\{ x \in \Omega : \text{there exists } i \neq j \text{ such that } \phi_i(x) = \phi_j(x) \geq \max_{k \neq i,j} \phi_k(x) \right\}. \tag{3.6}$$

  In either case, the solution of the Eikonal equation(s) need only be computed in a small narrow band necessary to find the Voronoi interface. The size of this narrow band is directly related to the choice of ϵ.

Reconstructing the unsigned distance function

After finding the Voronoi interface, the next step is to reconstruct the unsigned distance function. Here, for every point in the domain, we need to find the distance to the Voronoi interface Γ_V. This is found by again solving an Eikonal equation, namely

$$|\nabla \phi| = 1, \quad \phi(x) = 0 \text{ if } x \in \Gamma_V.$$

Finally, we rebuild the indicator function χ which assigns the correct phase to each point by keeping track on which side of Γ_V the point belongs. This is done by utilising the information used to find the Voronoi interface: when using a single Eikonal solve, this is naturally done through a traceback procedure; when multiple Eikonal equations are used, as in (3.6), then the new indicator function is given by $\chi(x) = \arg\max_i \phi_i(x)$.

Choice of algorithm

Several possibilities exist for performing the above calculations. For example, the ϵ-level sets could be extracted explicitly as a polyhedral mesh using standard contouring algorithms, and then used in distance calculations. The Voronoi interface is most naturally defined in an implicit fashion, as in (3.5) or (3.6). An explicit method of finding Γ_V is to extract it as a polyhedral mesh using the algorithm given in §3.8 below, after which the distance to Γ_V may be calculated as the minimum distance to the mesh. While such explicit approaches are possible, it is often advantageous to use implicit approaches instead. Implicit methods do not require that ϵ-level sets or Γ_V be explicitly

constructed, can be programmed more easily in 2D, 3D, and higher dimensions, and they often lead to higher order accuracy.

An implicit approach for rebuilding the unsigned distance function is based on solving the Eikonal equation. Thus, the core tool is a robust, fast, and accurate reinitialisation algorithm, which solves the Eikonal equation on a fixed background grid. In this work, the reinitialisation algorithm developed by Chopp [51] is extensively used. This method utilises bicubic (in 2D) and tricubic (in 3D) interpolation to accurately initialise the Fast Marching Method [52]. More details are given in §2.4 of Chapter 2. This method is used in two steps for rebuilding the unsigned distance function. First, the Chopp algorithm is used to calculate the distance to the ϵ-level sets on the grid. From this, the formulation (3.6) is used to define a second function ψ whose zero level set corresponds to Γ_V, which is then input into the Chopp algorithm, to calculate the distance to Γ_V.

3.4.3 Reconstruction interval

In the mathematical definition of the VIIM, the unsigned distance function ϕ and indicator function χ are reconstructed at the end of each time step. In a numerical setting, it can be advantageous to reconstruct less frequently, say every ten or so time steps. There is a tradeoff between the temporal errors incurred in delaying reconstruction, compared to spatial errors incurred due to the sharp corners often present in the unsigned distance function of a multiphase system. The precise reconstruction interval that gives the best results depends on the exact application and the time step being used. For example, in a fluid flow problem in which the surface tension time step constraint required for stability leads to rather small time steps, it can be worthwhile to reconstruct every 20–30 time steps. More comments about the reconstruction interval are provided in the convergence tests.

3.5 Adding physics and evaluation of the speed function

Naturally, an appropriate speed function F must be specified as part of the multiphase interface evolution. In geometric types of flow, the speed can be calculated from position, normal, and curvature quantities evaluated from derivatives of the distance function ϕ, similar to level set methods. When coupled to physics, the speed function is determined by solving the relevant equations of motion, where the location and geometry of the Voronoi interface can provide jump conditions, source terms, etc. Typically, the solution of these equations requires solving PDEs throughout the entire domain. When explicit extraction of the interface is required in order to evaluate any of the quantities that serve as input to these PDEs, a natural meshing algorithm described below can be used. The final speed function F must be defined everywhere (or at least in a narrow band surrounding the interface); depending on the problem at hand, it can be advantageous to use extension velocities, as described in §2.5 of Chapter 2.

3.6 Efficiency

A natural method to improve the efficiency of the VIIM is to use the idea of narrow banding; see Chapter 2. In particular, the unsigned distance function ϕ is only defined and updated in an adaptive narrow band of size k grid cells on either side of the interface. Together with the Eikonal equation solver that uses the Fast Marching Method, the operation count for the numerical VIIM algorithm is $\mathcal{O}(kN \log N)$ per time step, where N is the number of grid cells containing the interface. In particular, using the narrow banding approach, the computational complexity depends only on the measure of the interface, and does not directly depend on the number of phases which define that interface.

3.7 Creation and destruction of phases

The VIIM provides a straightforward way to add or create new phases. Such a new phase may arise for several reasons, for example through spontaneous nucleation via a chemical reaction, solidification, or when parts of a liquid begin to boil. Suppose, at some time t in the evolving calculation, one wants to create a new phase in the multiphase system. The boundary of this new phase is then supplied as a boundary condition when the unsigned distance function is reconstructed; all grid points within the new phase are then assigned a new value of the indicator function.

An analogous process occurs for the destruction of phases. A common way that this occurs is due to collapse of the region itself, such as in curvature flow. This is naturally handled through the Voronoi reconstruction: as the region shrinks, its ϵ-level set ceases to exist at some point in time and then does not contribute to the Voronoi reconstruction. In other applications, a component of the network of interfaces may disappear spontaneously (such as the popping of a bubble). In this case, the boundary is removed and the two phases on either side of the removed interface are given the same value for the indicator function.

3.8 Interface extraction and visualisation

At times, it is important to explicitly extract the interface, for example, to calculate accurate jump conditions, physics, chemistry, etc. at the interface, or for visualisation to display the evolving structures. In the two-phase case, there are several standard visualisation algorithms which extract level sets of implicit functions in two and three dimensions, including marching cubes [56], marching tetrahedra [57–59], and their variants. In the multiphase case, we have a more complex interface structure.

Here, we can capitalise on the Voronoi interface to design a conceptually simple interface extraction algorithm. Using the approach of solving Eikonal equations to find the Voronoi interface, as given in Section 3.4.2, suppose we have signed distance functions ϕ_i, defined on a grid and which determine the distance to $\Gamma_{i,\epsilon}$. Then, the part of Γ_V which gives the interface Γ_{ij} between phase i and

phase $j \neq i$ is given implicitly by (3.6):

$$\Gamma_{ij} = \left\{ x \in \Omega : \phi_i(x) = \phi_j(x) \geq \phi_k(x) \text{ for all } k \neq i, j \right\}. \tag{3.7}$$

To extract an explicit representation of Γ_{ij}, there are three steps:

(a) Determine a continuous piecewise linear interpolation of every ϕ_i function (known only at grid points) to determine ϕ_i at arbitrary points.

(b) Extract the zero level set of $\phi_i - \phi_j$, thereby producing a collection of surface elements $\{E_\ell : 1 \leq \ell \leq n\}$, where each E_ℓ is a straight line segment in 2D or a triangle in 3D. Since the last condition in (3.7) is not necessarily satisfied on every surface element, we have that $\Gamma_{ij} \subseteq \bigcup_\ell E_\ell$.

(c) For each element E_ℓ, keep the set of points $x \in E_\ell$ that satisfy $\phi_i(x) = \phi_j(x) \geq \phi_k(x)$ for all $k \neq i, j$. This is achieved by a series of "chop" operations that takes E_ℓ and chops it into pieces (a set of line segments in 2D or triangles in 3D), using the zero level set of $\phi_i - \phi_k$ as the position of the cut, and keeping those pieces on which $\phi_i \geq \phi_k$. Pieces on which $\phi_i < \phi_k$ are discarded. The result is a collection of surface elements whose union is Γ_{ij}.

We note several features about the above procedure.

- The algorithm is *exact*, in the sense that the interface of the interpolated multiphase system is extracted exactly. As a result, different interfaces Γ_{ij} and Γ_{kl} which meet at a higher order junction do so without overlap or gaps.

- Piecewise linear interpolation is natural for visualisation purposes, since line segments and triangles are naturally displayed, and chopping such elements with other elements is straightforward. Furthermore, piecewise linear interpolation is second order accurate wherever the function being interpolated is smooth, which is accurate enough for many purposes. In this work, a marching tetrahedrons based interpolation is used, in which each grid cell is divided into two triangles (in 2D) or six tetrahedra (in 3D). This provides a well-defined, predictable piecewise linear interpolant.

- The algorithm can be made very efficient by noting that the interpolation of ϕ_i functions in step (a) above only needs to be performed in grid cells containing the interface. Lookup tables, similar to those found in Marching Cubes, yield efficient extraction of mesh elements and efficient chopping operations. The overall computational complexity is $\mathcal{O}(N)$, where N is the number of cells containing the Voronoi interface, independent of the number of phases, due to narrow banding.

An example mesh generated by this algorithm is shown in Figure 3.4. Although low in mesh quality, the meshes are perfectly adequate for visualisation purposes. In fact, such meshes have been extensively used in calculating accurate surface tension forces in multiphase fluid flow problems, as discussed in §4.2 of Chapter 4. Generating high-quality meshes is also possible: Chapter 5 presents a more advanced high-quality meshing algorithm, based in part on the approach presented here.

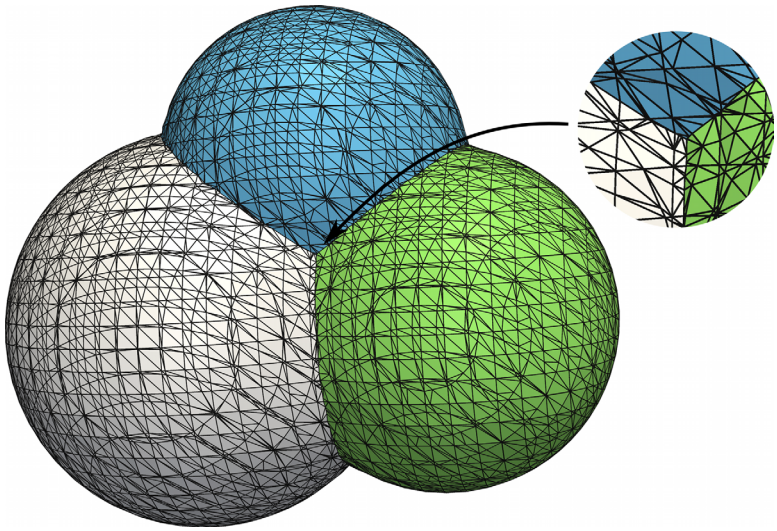

Figure 3.4. A mesh generated using piecewise linear interpolation and the Voronoi interface, for a four-phase system (white, blue, green, and exterior). The zoomed region highlights a quadruple point, where four triple lines meet (however only three are visible).

3.9 Parallel implementation using MPI

In much of this work, the VIIM and associated methods have been parallelised using MPI and a domain decomposition approach. This is useful for curvature flow calculations that involve many time steps, as well as complex three-dimensional fluid flow simulations. The domain is a simple rectangular Cartesian grid that is sub-divided into smaller grids, each assigned to an individual processor in the MPI implementation. Synchronisation of data on the subdomains is performed using ghost layers of sufficient size. Many algorithms can be parallelised easily with this approach, while other algorithms require more care, such as the solution of Eikonal equations.

To parallelise the Eikonal equation solver, we can exploit the fact that we only need data in a narrow band. Thus, each processor can solve the Eikonal equation locally, in an area which is the size of the subdomain assigned to that processor, plus a number of layers corresponding to the size of the narrow band. Once the data on this slightly larger grid is initialised with bicubic/tricubic interpolation, the normal Fast Marching Method can be used. In this fashion, each processor can solve the Eikonal equation locally and independently of the other processors. The parallel efficiency of this method depends on the geometry of the interface and the relative size of the subgrid compared to the narrow band: if the interface is sufficiently dense, as it often is in multiphase simulations, the method scales very well.

3.10 Convergence tests and verification

So far in this chapter, the main VIIM algorithm has been developed and its numerical implementation discussed. In this section, several convergence tests of the method are performed in two and

three dimensions, verifying the accuracy, robustness, and convergence properties under refinement of numerical parameters. Three different ways to test convergence are considered, corresponding to those outlined in §3.4.1:

(i) Fix ϵ independent of the grid size h and study convergence as $h \to 0$, followed by the limit $\epsilon \to 0$. Another way to say this is that we compute the converged solution to the problem for a fixed ϵ, i.e. the ϵ-smoothed solution, and then consider the limit of this ϵ-smoothed solution as ϵ goes to zero. This directly follows the mathematical definition of multiphase evolution, and produces a converged solution. In the following, this approach is denotes as the "ϵ fixed regime".

(ii) Couple ϵ with the grid size h, i.e. set $\epsilon = \alpha h$ where α is a fixed constant, and study convergence as $h \to 0$. This approach sends $\epsilon \to 0$ simultaneously with $h \to 0$, and hence differs from the formal "limit of a limit" in the strict mathematical definition. However, this method is more computationally practical, as the width of a narrow band implementation remains fixed, independent of the grid size. The results will show that coupling ϵ to h in this manner also produces the same, converged solution. This approach is denoted as the "ϵ coupled regime".

(iii) Exchange the limits, and first compute an inner limit with $\epsilon \to 0^+$, and then study convergence as $h \to 0$. In the following, the numerical solution obtained with this approach is labelled as "$\epsilon = 0^+$".

For some test problems, the solution is known and we can measure the error between the numerical results and the exact solution. In other cases, the solution is not known and grid convergence will be used. In both of these scenarios, the "error" measures the difference in interface position, defined by the Hausdorff metric d_H: given two interfaces Γ_1 and Γ_2 (each a surface of codimension-one), we let

$$d_H(\Gamma_1, \Gamma_2) = \max\Big(\sup_{x\in\Gamma_1} \inf_{y\in\Gamma_2} \|x - y\|_2, \sup_{x\in\Gamma_2} \inf_{y\in\Gamma_1} \|x - y\|_2 \Big).$$

Roughly speaking, the Hausdorff distance measures the maximum width of the region between Γ_1 and Γ_2. In the following convergence tests, convergence is measured in time, as well as in space. To do this, an averaged L^1 norm in time[1] is used, together with the Hausdorff metric in space, i.e. if $\Gamma_1(t)$ and $\Gamma_2(t)$ are interfaces evolving over a time interval $t \in [0, T]$, then we define

$$d(\Gamma_1, \Gamma_2) = \frac{1}{T} \int_0^T d_H\big(\Gamma_1(t), \Gamma_2(t)\big)\, dt. \tag{3.8}$$

To numerically evaluate (3.8) in practice, the interfaces Γ_i are reconstructed explicitly from the distance function ϕ as a mesh and the Hausdorff distance between two meshes is computed. This is a second order accurate approximation of $d_H(\cdot, \cdot)$ and is sufficient for the following tests.

[1]Tests were also performed using the maximum norm in time, e.g. $\max_{0\le t\le T} d_H(\cdot, \cdot)$. Convergence is still obtained under this norm as well, with the same overall convergence rate. However, between one grid size and the next, the convergence rate can be spurious, due to the sensitivity of the Hausdorff distance to the location of triple points in a grid cell. Instead, the L^1 norm is used in time, which for the tests performed here, do not weaken any conclusions made about convergence.

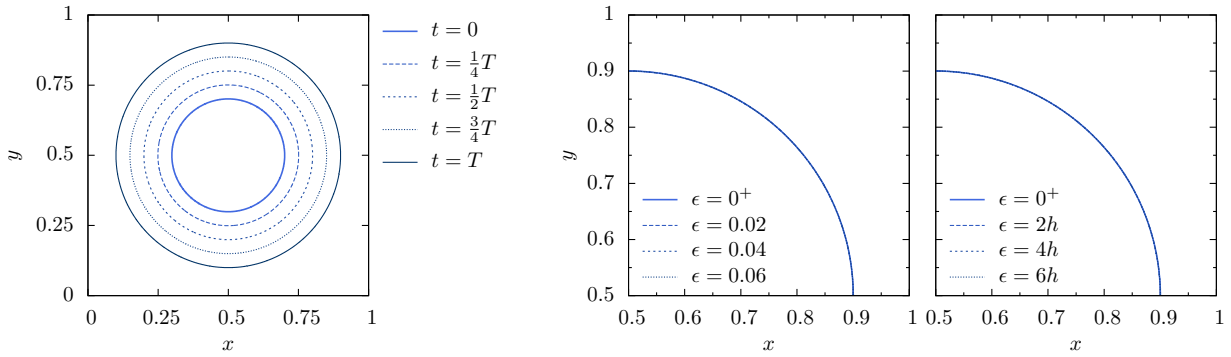

Figure 3.5. Evolution of a circle expanding with unit speed ($T = \frac{1}{5}$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

In all of the tests, the domain is a unit square in 2D, or unit cube in 3D, and a uniform Cartesian grid with cell size h is used. Standard first or second order finite difference schemes are used for the evolution (upwinding when necessary), and the time step constraint for forward Euler will be indicated. Convergence as $h \to 0$ is measured for a variety of ϵ values for the two regimes. We denote by $\Gamma_h^\epsilon = \Gamma_h^\epsilon(t)$ the time evolution of the interface obtained with a grid size h and parameter ϵ. When grid convergence is being used, we suppose that the leading order component of the error is Ch^p for some constants C and p and measure the difference of two solutions on two different grids, in this case using $d(\Gamma_h^\epsilon, \Gamma_{2h}^\epsilon)$, to determine the rate of convergence p. Specifically, we define $d_h^\epsilon := d(\Gamma_h^\epsilon, \Gamma_{2h}^\epsilon)$ and estimate $p \approx \log_2(d_{2h}^\epsilon / d_h^\epsilon)$. Finally, if a data point is missing in a convergence plot or table, then it is because the corresponding grid was too coarse to successfully determine the evolution (e.g. a phase became so small that its ϵ-level set vanished).

3.10.1 Basic tests in two dimensions

First, we verify that the VIIM produces correct results in the case of straightforward two-phase problems.

Circle expanding with unit speed

We first test the case of a circle expanding with unit speed $F = 1$, as illustrated in Figure 3.5 (left). The circle's initial radius is $\frac{1}{5}$ and is evolved over a time of $T = \frac{1}{5}$ so that its final radius is $\frac{2}{5}$. For this evolution, the time step constraint is $\Delta t < h$; here $\Delta t = \frac{1}{2}h$. For an unsigned distance function to a circle, the Voronoi reconstructed interface using any ϵ-level set is exact. This implies that for this test problem, errors in the numerical results of the VIIM will be largely independent of ϵ, and the results confirm this. Figure 3.5 shows the numerical results at final time $t = T$ obtained on the same grid size ($h = 1/512$) for different ϵ values; there are no observable differences. Figure 3.6 plots the error compared with the exact solution, $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$, and shows that first order accuracy in space and time is obtained, independent of the choice of ϵ.

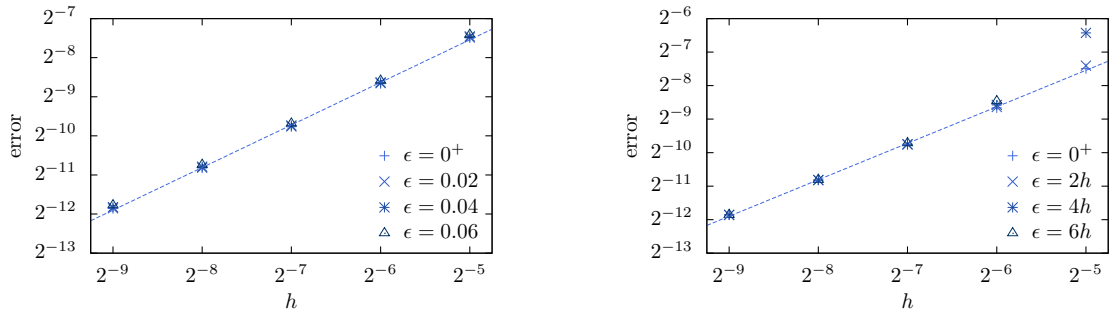

Figure 3.6. Corresponding to the test case in Figure 3.5, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 1.0.

Initially square interface expanding with unit speed

We now consider an initially square interface expanding with unit speed $F = 1$, as illustrated in Figure 3.7 (left) (recall that the viscosity solution of the corresponding Hamilton-Jacobi equation yields an expanding square with rounded corners). The square's initial width is $\frac{2}{5}$ and is evolved over a time of $T = \frac{1}{5}$ so that its final width is $\frac{4}{5}$. The time step constraint is $\Delta t < h$; here $\Delta t = \frac{1}{2}h$. In this test case, due to the corners present in the initial interface, the Voronoi reconstructed interface changes with the choice of ϵ. Figure 3.7 shows the numerical results at final time $t = T$ obtained on the same grid size ($h = 1/512$) for the two ϵ regimes. In particular, in Figure 3.7 (middle), we see that when $\epsilon > 0$ is fixed, the ϵ-smoothed solution is not the same as the exact solution, but converges to it as $\epsilon \to 0$. A convergence analysis of the results is presented in Table 3.1 and Figure 3.8. In Figure 3.8 (left) we see that if $\epsilon > 0$ is fixed, then as $h \to 0$, convergence to the exact solution is not obtained. Instead, the numerical results are converging to the ϵ-smoothed solution, as shown in Table 3.1. In the regime where ϵ is proportional to h, approximately first order convergence to the solution is obtained as $h \to 0$.

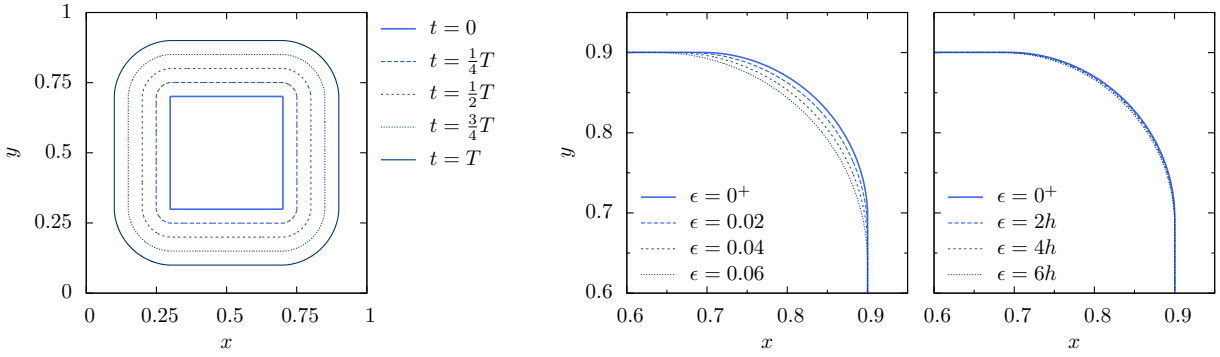

Figure 3.7. Evolution of an initially square interface expanding with unit speed ($T = \frac{1}{5}$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

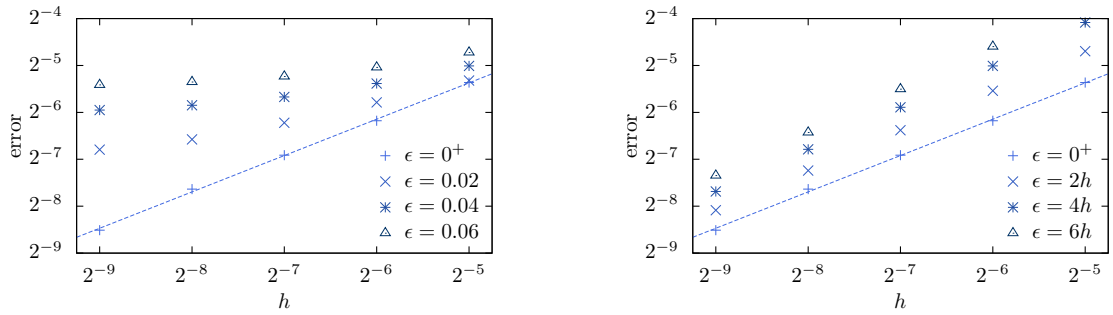

Figure 3.8. Corresponding to the test case in Figure 3.7, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 0.8.

	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
1/64	0.00924	–	0.00727	–	0.00739	–	0.00758	–
1/128	0.00445	1.1	0.00409	0.8	0.00390	0.9	0.00371	1.0
1/256	0.00262	0.8	0.00245	0.7	0.00218	0.8	0.00209	0.8
1/512	0.00174	0.6	0.00121	1.0	0.00108	1.0	0.00101	1.1

Table 3.1. Corresponding to the test case in Figure 3.7, convergence results for the fixed ϵ regime.

Square shrinking with unit speed

Now consider the case of a square shrinking with unit speed, as illustrated in Figure 3.9 (left). The square has an initial width $\frac{4}{5}$ and is evolved over a time $T = \frac{1}{5}$. Similar to before, the Voronoi reconstruction has an error depending on the choice of ϵ. Figure 3.9 shows the numerical results at final time obtained on the same grid size ($h = 1/512$) for different ϵ values. In particular, in Figure 3.9 (middle), we see that the ϵ-smoothed solution is not the same as the exact solution. However, as $\epsilon \to 0$, the ϵ-smoothed solution converge to the exact solution. A convergence analysis is presented in Table 3.2 and Figure 3.10, and shows similar convergence properties as in the previous test.

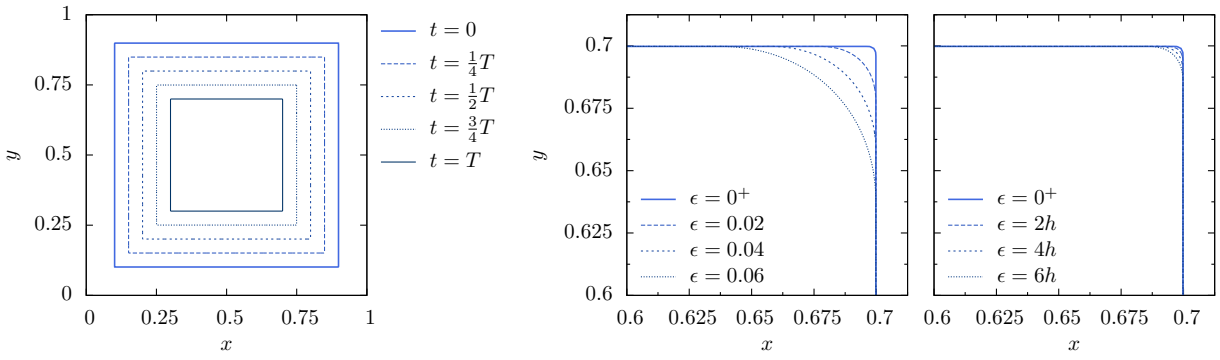

Figure 3.9. Evolution of a square shrinking with unit speed ($T = \frac{1}{5}$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

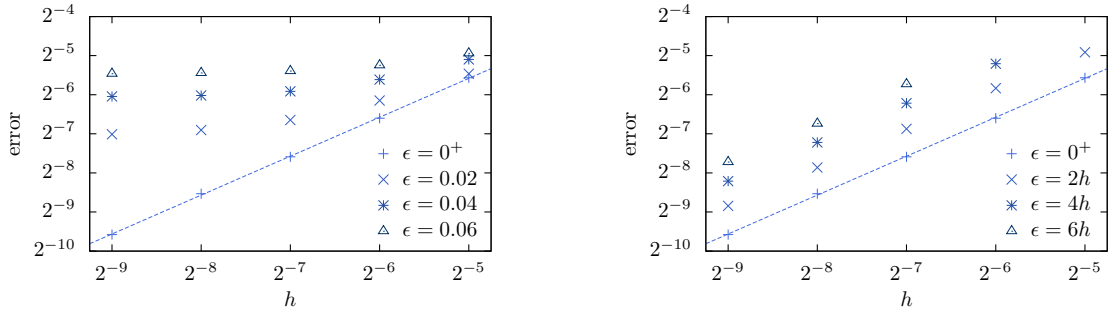

Figure 3.10. Corresponding to the test case in Figure 3.9, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 1.0.

	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
1/64	0.01655	–	0.01424	–	0.01475	–	0.01252	–
1/128	0.00855	1.0	0.00787	0.9	0.00738	1.0	0.00604	1.1
1/256	0.00415	1.0	0.00329	1.3	0.00296	1.3	0.00254	1.2
1/512	0.00225	0.9	0.00147	1.2	0.00118	1.3	0.00123	1.0

Table 3.2. Corresponding to the test case in Figure 3.9, convergence results for the fixed ϵ regime.

Curvature flow on a circle

We next consider curvature flow applied to a circle with speed $F = -\kappa$, as illustrated in Figure 3.11 (left). The circle has an initial radius of $r_0 = 0.375$ and is evolved over a time of $T = 0.04$; the exact solution has radius at time t given by $r(t) = \sqrt{r_0^2 - 2t}$. The time step constraint for this problem (using the fact that ϕ does not deviate from a distance function) is $\Delta t < \frac{1}{4}h^2$. At each grid point, standard finite difference stencils are used to approximate the curvature of the level set passing through that grid point. It follows that the ϵ-level sets inside and outside the circle move with different speeds, and this implies that the Voronoi reconstructed interface moves with a speed depending on ϵ. (In fact, it can be shown that this dependence on ϵ is of the form $F_{\text{actual}} = F_{\text{exact}} + \mathcal{O}(\epsilon^2)$.) Figure 3.11 shows the numerical results at final time $t = T$ obtained on the same grid size ($h = 1/512$) for different ϵ values. Once again, the limit of ϵ-smoothed solutions as $\epsilon \to 0$ yields the correct solution. This is made more precise in Figure 3.12 and Table 3.3. In the regime where ϵ is proportional to h, second order convergence to the solution as $h \to 0$ is obtained. In this case, second order convergence is obtained since second order finite difference methods are used together with a second order time step $\Delta t = \mathcal{O}(h^2)$.

3.10.2 Triple points in two dimensions

Next, we consider a single T junction that moves into a Y junction under curvature flow with $F = -\kappa$. We consider two different boundary conditions: zero Neumann and Dirichlet ("anchored"). In both cases, $\Delta t = \frac{1}{4}h^2$ and the interface is evolved over a time of $T = \frac{125}{512} \approx 0.244$ (corresponding to 1000 time steps on the coarsest grid). Figure 3.13 illustrates the evolution for both types of boundary

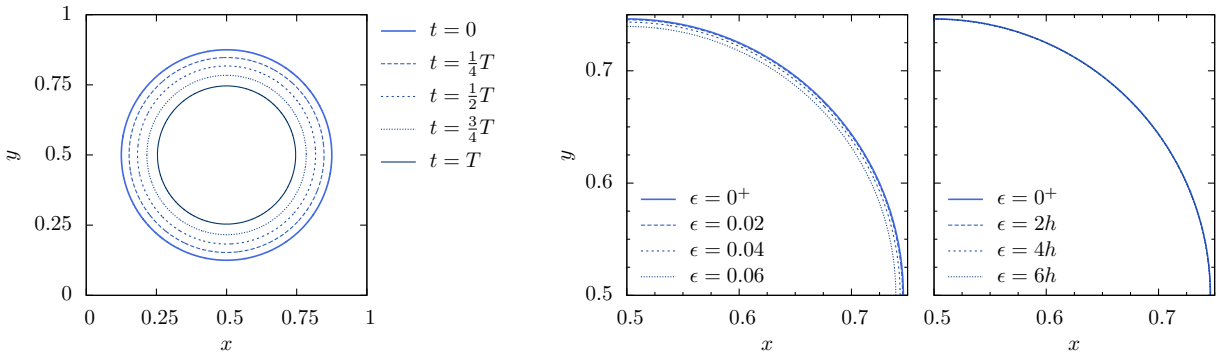

Figure 3.11. Evolution of a circle collapsing under curvature flow with speed $F = \kappa$ ($T = 0.04$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

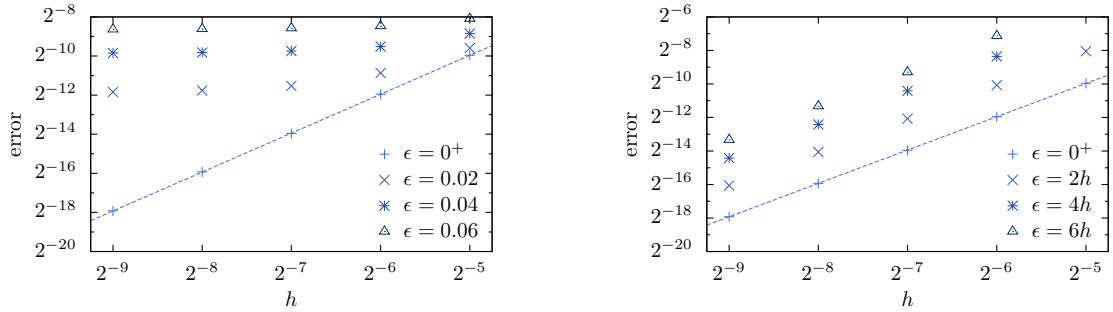

Figure 3.12. Corresponding to the test case in Figure 3.11, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 2.0.

	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
1/64	0.00123	–	0.00123	–	0.00127	–	0.00132	–
1/128	0.00030	2.0	0.00031	2.0	0.00030	2.1	0.00030	2.1
1/256	0.00008	1.9	0.00008	1.9	0.00009	1.7	0.00010	1.6
1/512	0.00002	2.0	0.00002	1.9	0.00003	1.7	0.00003	1.5

Table 3.3. Corresponding to the test case in Figure 3.11, convergence results for the fixed ϵ regime.

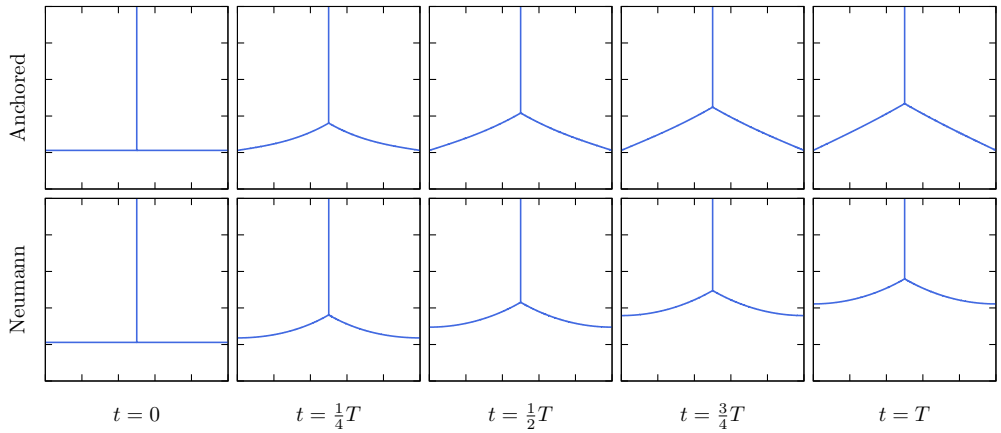

Figure 3.13. Evolution of curvature flow applied to an interface that is initially a T junction ($T = \frac{125}{512}$), with anchored boundary conditions (top) and Neumann boundary conditions (bottom).

conditions. We see that the motion defined by the VIIM is such that the T junction moves into a Y junction with 120° angles. In the case of anchored boundary conditions, the evolution ultimately converges to an equilibrium with straight lines.

Figure 3.14 shows the numerical results at final time $t = T$ obtained on the same grid size ($h = 1/512$) for different ϵ values. Once again, for the regime with fixed ϵ, we see that different ϵ-smoothed solutions are obtained for different fixed ϵ, but there is a well-defined limit as $\epsilon \to 0$. In this test, we rely on grid refinement to measure convergence. Table 3.4 (fixed ϵ regime) and Table 3.5 (ϵ coupled to h regime) contain the results. We see that in all cases, in both ϵ regimes, the VIIM converges in both space and time at first order. Although not shown here, it was also verified that the ϵ-smoothed solutions converge as $\epsilon \to 0$ to the same solution obtained in the ϵ coupled regime.

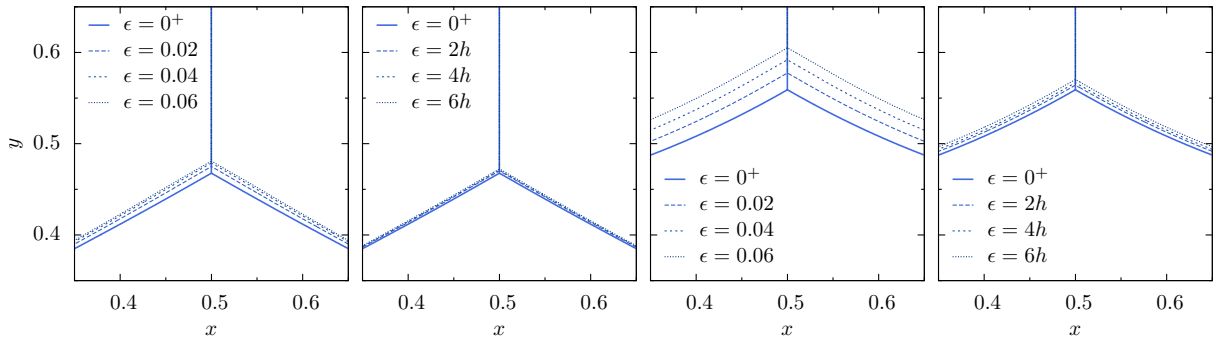

Figure 3.14. Corresponding to the test in Figure 3.13, the solution at time $t = T$ obtained on the same grid ($h = 1/512$) with different choices of ϵ for the two different ϵ regimes; anchored boundary conditions (left pair) and Neumann boundary conditions (right pair).

		$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
	h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
Anchored	1/64	0.00305	–	0.00269	–	0.00326	–	0.00393	–
	1/128	0.00164	0.9	0.00186	0.5	0.00229	0.5	0.00260	0.6
	1/256	0.00083	1.0	0.00118	0.7	0.00140	0.7	0.00175	0.6
	1/512	0.00044	0.9	0.00062	0.9	0.00076	0.9	0.00090	1.0
Neumann	1/64	0.00335	–	0.00290	–	0.00374	–	0.00448	–
	1/128	0.00183	0.9	0.00216	0.4	0.00261	0.5	0.00280	0.7
	1/256	0.00090	1.0	0.00129	0.7	0.00112	1.2	0.00182	0.6
	1/512	0.00048	0.9	0.00047	1.4	0.00062	0.9	0.00125	0.5

Table 3.4. Corresponding to the test in Figure 3.13, convergence results for the fixed ϵ regime.

		$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
	h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
Anchored	1/64	0.00305	–	0.01092	–	0.01626	–	0.01868	–
	1/128	0.00164	0.9	0.00631	0.8	0.01019	0.7	0.01322	0.5
	1/256	0.00083	1.0	0.00336	0.9	0.00549	0.9	0.00737	0.8
	1/512	0.00044	0.9	0.00174	1.0	0.00283	1.0	0.00383	0.9
Neumann	1/64	0.00335	–	0.01799	–	0.03571	–	–	–
	1/128	0.00183	0.9	0.00932	0.9	0.01695	1.1	0.02502	–
	1/256	0.00090	1.0	0.00475	1.0	0.00834	1.0	0.01190	1.1
	1/512	0.00048	0.9	0.00242	1.0	0.00415	1.0	0.00582	1.0

Table 3.5. Corresponding to the test in Figure 3.13, convergence results for regime in which ϵ is coupled to h.

3.10.3 von Neumann-Mullins' law

Verification of von Neumann-Mullins' law in 2D

In the last convergence test of a T junction moving into a Y junction, we saw that the VIIM applied to curvature flow on a triple point resulted in that triple point obtaining 120° angles. This is known as Young's law and naturally arises when viewing curvature flow as minimising perimeter (or surface area in 3D). We now study in more depth curvature flow on a random set of phases involving a connected network of interfaces with several triple points. Assuming Young's law holds, von Neumann [60] and Mullins [61] showed that if the speed of the interface is $F = -\gamma\kappa$ (where γ is a constant), then the rate of change of area of any phase is an affine function of the number of sides of the phase. The law can be derived from the relation $\frac{dA}{dt} = \int_\Gamma F$ (taking into account the angles at triple points), and states that

$$\frac{dA}{dt} = 2\pi\gamma\Big(\frac{n}{6} - 1\Big). \tag{3.9}$$

This is a remarkable result: the law states that the growth rate of a phase depends only on the number of sides it has, and not directly on the overall size or shape of the phase or the motion of any other phase. Consequently, as a multiphase system evolves under curvature flow, it can be expected that some phases shrink, others expand, and that various topological changes take place. Thus, according to the von Neumann-Mullins' law, the area of each phase as a function of time is piecewise linear.

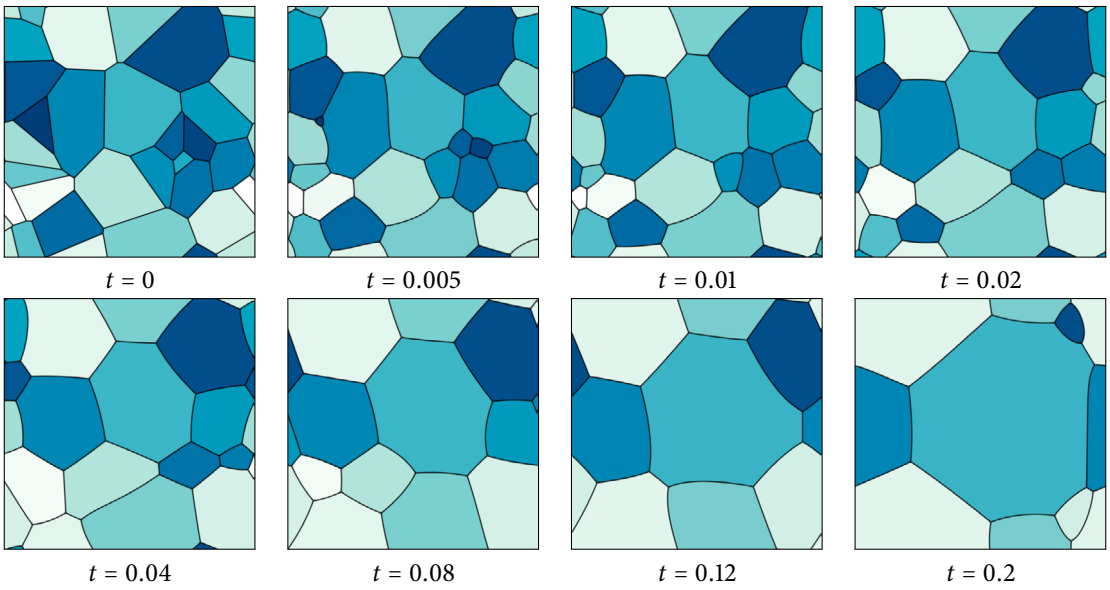

Figure 3.15. Curvature flow applied to an initial set of 25 randomly positioned phases. According to von Neumann-Mullins' law, phases with more than six sides grow, those with less than six shrink, and those with six sides conserve their area. This process leads to "coarsening" of the system.

To investigate this law in the context of the VIIM, Figure 3.15 shows the numerical results of curvature flow applied to a multiphase system which initially has 25 randomly positioned phases (with $\gamma = 1$), using periodic boundary conditions (grid size 256×256). We see that some phases expand, while others shrink and disappear. Moreover, there are a number of topological changes that take place, which are automatically handled by the VIIM. In Figure 3.16, the area as a function of time is shown for a selected set of nine phases. The results correctly match von Neumann-Mullins' law throughout the evolution, even as these intricate topological changes take place.

Using von Neumann-Mullins' law to check convergence

Using von Neumann-Mullins' law as information about the exact solution, we can also test the convergence of the VIIM applied to curvature flow as a function of grid refinement.

To test the numerical results of the VIIM against von Neumann-Mullins' law, we consider n sided shapes, for $3 \leq n \leq 8$, suitably connected to the boundary (see Figure 3.17). The initial shape is chosen to be circular with radius 0.3, (except if $n = 3$, in which case it has a radius 0.35), and is evolved over a time $T = 0.08$. The time step is again chosen to be $\Delta t = \frac{1}{4}h^2$. As part of the convergence tests, the role of the reconstruction interval, as discussed in §3.4.3, was tested. Denote by k the reconstruction interval, e.g. if $k = 1$ then the Voronoi interface is reconstructed every time step. For the time step used here, it was found that consistent convergence rates were obtained for approximately $k \geq 8$, and in the following, $k = 16$ has been used. To measure the error in this multiphase curvature evolution, we measure the difference between the computed value of $\frac{dA}{dt}$ (obtained via the slope of a linear regression of the shape's area over 100 equally spaced points in time) compared with that predicted

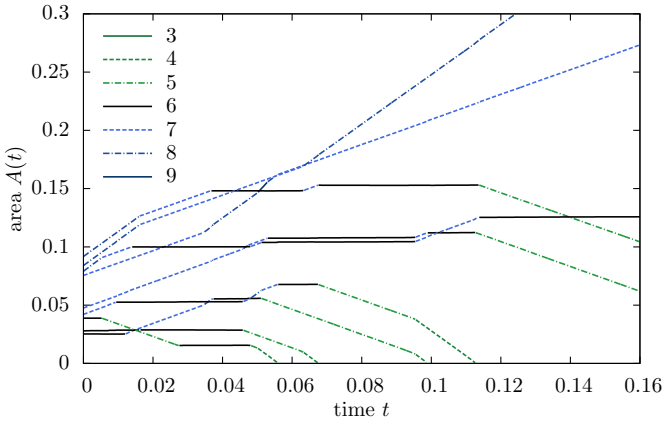

Figure 3.16. Area as a function of time for a selected set of nine phases corresponding to the evolution shown in Figure 3.15. According to von Neumann-Mullins' law, the area of a particular phase should be a piecewise linear function, with a derivative that is an affine function of the number of sides of the phase. We have therefore coloured each part of the trajectories by the number of sides the phase had at that particular time. The slopes of the trajectories matches with what is predicted by von Neumann-Mullins' law.

by von Neumann-Mullins' law. To simplify matters, convergence for the regime in which ϵ is fixed is skipped; from hereon, only the case when $\epsilon = \alpha h$ is coupled to the grid size h, as $h \to 0$, will be considered. For each n,

- a time sequence illustrating the evolution of the n-sided shape over time is shown in Figure 3.17;
- a plot of the error in $\frac{dA}{dt}$ as a function of h is shown in Figure 3.18; and
- a table with convergence rates obtained with grid refinement, measuring convergence of the interface evolution in space and time, is given in Table 3.6.

According to von Neumann-Mullins' law (3.9), the shapes decrease in area if $n < 6$, increase in area if $n > 6$, and have constant area if $n = 6$. This is demonstrated in the figures, as well as the fact that the evolution quickly become self-similar. In most cases well behaved convergence is obtained, whereby the VIIM converges at first order rate in both space and time. The exception is the case when $n = 6$ where the error is smaller than the other cases but does not exhibit consistent convergence rates. This is most likely due to the fact that the shape is essentially stationary, and so grid alignment errors dominate.

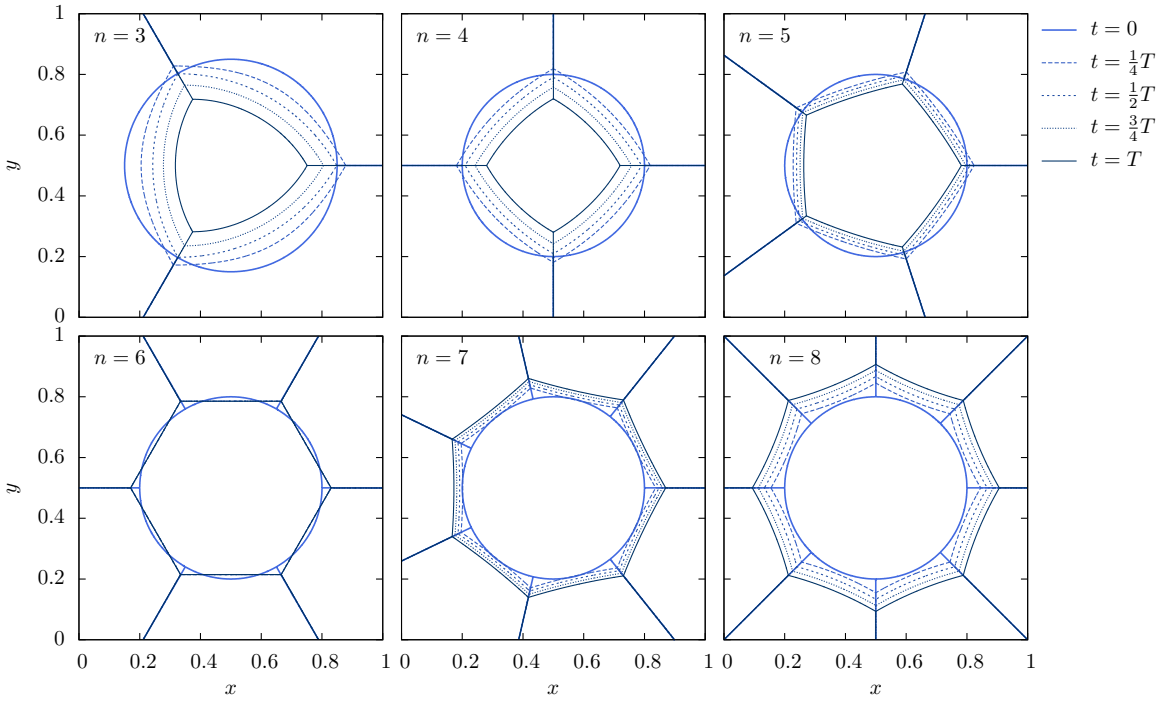

Figure 3.17. Evolution of initially circular n-sided shapes under curvature flow with $\gamma = 1$ ($T = 0.08$).

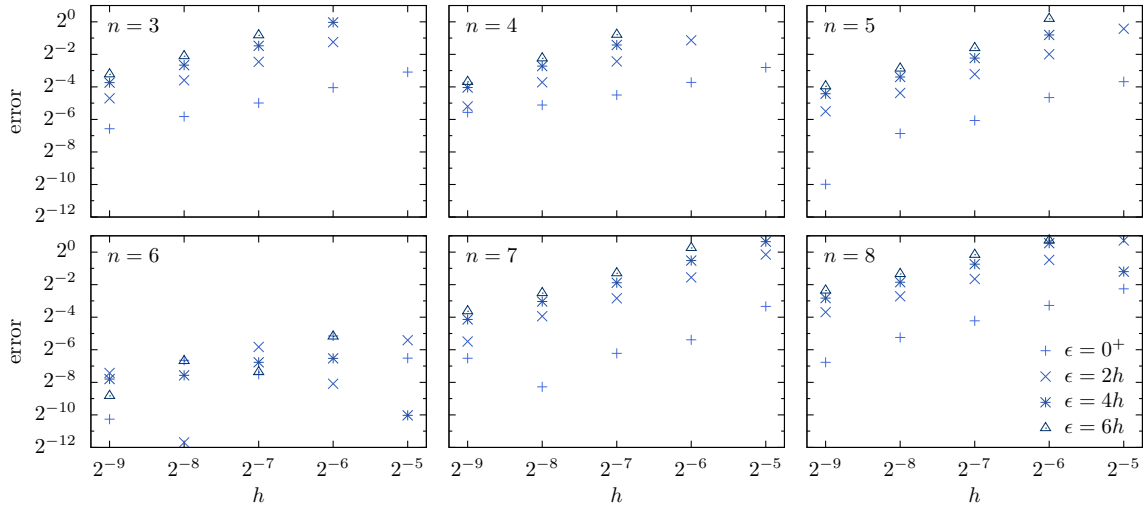

Figure 3.18. Plot of the error in the computed value of $\frac{dA}{dt}$ as a function of grid size h for the n-sided shapes of Figure 3.17.

		$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
n	h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
3	1/64	0.00254	–	0.01924	–	0.09240	–	0.13283	–
	1/128	0.00119	1.1	0.00593	1.7	0.01415	2.7	0.04314	1.6
	1/256	0.00061	1.0	0.00267	1.2	0.00507	1.5	0.00807	2.4
	1/512	0.00033	0.9	0.00130	1.0	0.00226	1.2	0.00329	1.3
4	1/64	0.00235	–	0.04159	–	0.11469	–	0.12696	–
	1/128	0.00104	1.2	0.00622	2.7	0.02101	2.4	0.06301	1.0
	1/256	0.00049	1.1	0.00249	1.3	0.00498	2.1	0.00823	2.9
	1/512	0.00024	1.0	0.00115	1.1	0.00207	1.3	0.00302	1.4
5	1/64	0.00236	–	0.01007	–	0.11946	–	0.20327	–
	1/128	0.00125	0.9	0.00279	1.9	0.00665	4.2	0.01501	3.8
	1/256	0.00064	1.0	0.00115	1.3	0.00204	1.7	0.00324	2.2
	1/512	0.00043	0.6	0.00064	0.8	0.00092	1.2	0.00131	1.3
6	1/64	0.00233	–	0.00307	–	0.00299	–	0.22553	–
	1/128	0.00252	-0.1	0.00169	0.9	0.00280	0.1	0.00370	5.9
	1/256	0.00116	1.1	0.00183	-0.1	0.00233	0.3	0.00276	0.4
	1/512	0.00082	0.5	0.00096	0.9	0.00121	1.0	0.00189	0.5
7	1/64	0.00302	–	0.01212	–	0.03456	–	0.04348	–
	1/128	0.00128	1.2	0.00550	1.1	0.01025	1.8	0.01756	1.3
	1/256	0.00117	0.1	0.00285	0.9	0.00493	1.1	0.00708	1.3
	1/512	0.00058	1.0	0.00177	0.7	0.00255	1.0	0.00352	1.0
8	1/64	0.00300	–	0.02056	–	0.03031	–	0.04964	–
	1/128	0.00149	1.0	0.00851	1.3	0.01793	0.8	0.02987	0.7
	1/256	0.00092	0.7	0.00422	1.0	0.00762	1.2	0.01141	1.4
	1/512	0.00056	0.7	0.00211	1.0	0.00368	1.1	0.00524	1.1

Table 3.6. Convergence results for the n-sided shape undergoing curvature flow in Figure 3.17.

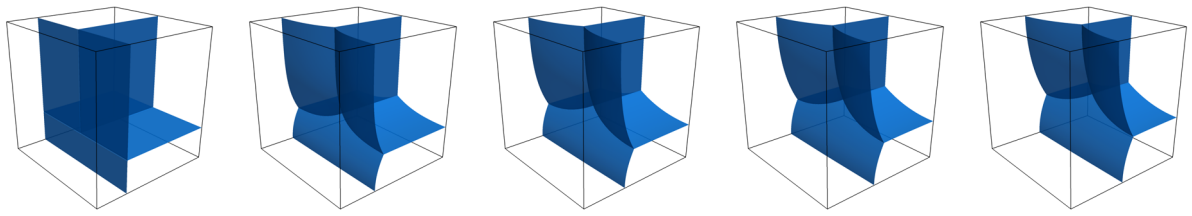

Figure 3.19. Evolution of a three-dimensional analogue of a T junction subject to curvature flow ($T = \frac{125}{1024}$, computed on a 128^3 grid).

	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
1/64	0.00529	–	0.02705	–	0.05866	–	0.16831	–
1/128	0.00279	0.9	0.01396	1.0	0.02469	1.2	0.03578	2.2
1/256	0.00147	0.9	0.00709	1.0	0.01225	1.0	0.01722	1.1

Table 3.7. Convergence results for the curvature flow illustrated in Figure 3.19.

3.10.4 Triple lines in three dimensions

The VIIM easily extends to three dimensions and correctly handles the three-dimensional analogues of triple points. Figure 3.19 shows the evolution of a "three-dimensional T junction", composed of four triple lines meeting at a single quadruple point, subject to curvature flow with Neumann boundary conditions. The shape is evolved over a time of $T = \frac{125}{1024} \approx 0.122$ (corresponding to 500 time steps on the coarsest grid). We observe qualitatively that the surfaces make 120° angles at triple lines, which is one of Plateau's laws on the shapes of soap bubbles in a foam. Using grid refinement to measure convergence, Table 3.7 shows that the VIIM has first order convergence in both space and time.

3.10.5 Three-dimensional analogue of von Neumann-Mullins' law

The two-dimensional von Neumann-Mullins' law was recently generalised to three or more dimensions and provides a formula for $\frac{dV}{dt}$ involving a "mean-width" [62], for multiphase mean curvature flow. In particular, the growth rate is no longer a function based purely on the topology (as it is in two dimensions) and can depend on the particular shape of the phase/region. In [62], a formula is given for calculating $\frac{dV}{dt}$ for a region approximated by a polyhedron, assuming that the polyhedron makes 120° angles at triple lines. It is reasonable to consider using this formula as verification in the same way that we did with the two-dimensional von Neumann-Mullins' law. However, the resulting formula for $\frac{dV}{dt}$ is sensitive to the angles that the polyhedron makes at triple lines, which makes prediction of $\frac{dV}{dt}$ using this method noisy.

Instead, we test the VIIM on a known solution for curvature flow that has a self-similar evolution, namely by using a shrinking Reuleaux tetrahedron, which is an analogue of the two-dimensional

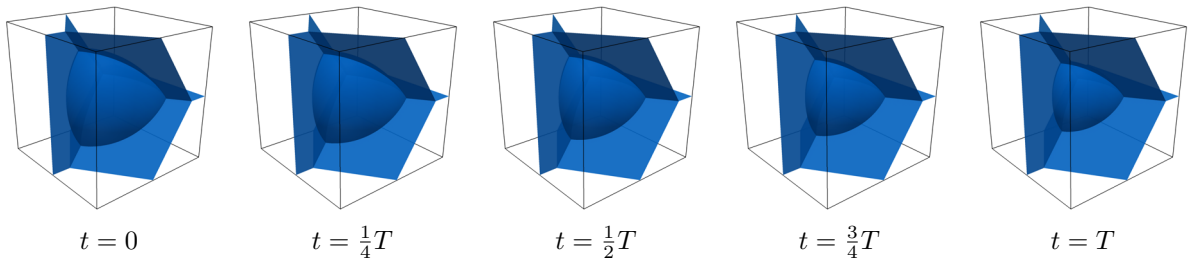

Figure 3.20. Evolution of a Reuleaux tetrahedron shrinking under multiphase curvature flow ($T = 0.025$, computed on a 128^3 grid).

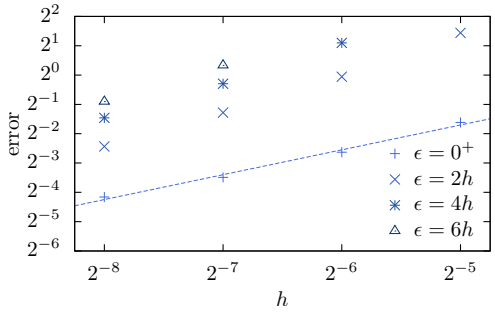

Figure 3.21. Plot of the error in the computed value of $\frac{d}{dt}V^{2/3}$ for the shrinking Reuleaux tetrahedron in Figure 3.20. The slope of the line is 0.85.

	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
h	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order	d_h^ϵ	order
1/64	0.00397	–	0.01774	–	0.09394	–	0.16874	–
1/128	0.00193	1.0	0.00838	1.1	0.01788	2.4	0.04128	2.0
1/256	0.00096	1.0	0.00415	1.0	0.00778	1.2	0.01187	1.8

Table 3.8. Convergence results for the curvature flow illustrated in Figure 3.20.

Reuleaux triangle. This is a tetrahedron with spherical sides that satisfies Young's law where its edges meet at planar surfaces. The shape and its evolution is shown in Figure 3.20. It satisfies $\frac{d}{dt}V^{2/3} = C$ where $C \approx 5.2885$ is a constant.[2] We compute the error between the exact value for $\frac{d}{dt}V^{2/3}$ and that of the simulation (obtained via the slope of a linear regression of $V^{2/3}$ over 50 equally spaced points in time). The results are shown in Figure 3.21 and Table 3.8 and shows that the VIIM once again obtains convergence as $h \to 0$ for every value of ϵ.

[2]The volume of a Reuleaux tetrahedron having spherical sides of radius r is $V(r) = C_V r^3$ where $C_V := \frac{8}{3}\pi - \frac{27}{4}\cos^{-1}\frac{1}{3} + \frac{\sqrt{2}}{4}$; its surface area is $S(r) = C_S r^2$ where $C_S := 4(2\pi - \frac{9}{2}\cos^{-1}\frac{1}{3})$. Since $\kappa = \frac{2}{r}$, we have that under self-similar curvature flow, $\frac{d}{dt}V = S\frac{2}{r}$. It follows that $\frac{d}{dt}V^{\frac{2}{3}} = \frac{2}{3}V^{-\frac{1}{3}}\frac{d}{dt}V = \frac{4}{3}C_V^{-1/3}C_S =: C \approx 5.28854633$.

3.10.6 Summary of convergence results

The preceding shows that the Voronoi Implicit Interface Method converges under several different tests. The convergence tests were in both time and space, verifying that the method correctly computes complex flows in a converged manner: first order convergence was obtained in almost all cases. The method yields the correct evolution in the case of two phase flow, yields a flow satisfying Young's law, and matches von Neumann-Mullins' law in two dimensions and can be used to accurately predict growth rates in three-dimensions. In these multiphase systems, using Eulerian based PDE methods, first order accuracy at triple junctions is probably all that can be expected in this framework. Further comments about the order of accuracy are discussed at the end of this thesis.

3.11 Concluding remarks

In summary, the VIIM provides a mathematically consistent framework for tracking the interface in multiphase problems. Its main features include:

- *Single representation.* In its mathematical form, only two functions are used for the entire multiphase system, the unsigned distance function, and indicator function. Numerical implementations can use this formulation, or, at the expense of implementing more complicated data structures, can use a multi-valued level set function for increased numerical accuracy.
- *Consistency.* By construction, regions of overlap or voids cannot occur. Additionally, the method is consistent with level set methods in the case of two-phase flow[3].
- *Accurate calculation of geometric quantities.* Geometric quantities, such as the normal and curvature of the interface, can be calculated accurately.
- *Dimension independent.* The formulation works in any number of spatial dimensions with no change to the fundamental algorithm.
- *Topological changes.* Complex configurations in the interface, as well as changes in topology, are automatically handled by the framework.
- *Accurate, efficient, and robust.* Convergence tests indicate the method is generally first order accurate in both space and time, while efficiency is gained through the use of narrow banding.
- *Ability to couple to physics.* Interfaces can evolve in such a way that time has physical meaning, and the method can be coupled to complex physics. This is demonstrated in the following chapter with several multiphase fluid flow applications.

[3]Except for pathological cases such as "interface fattening" in curvature flow; further comments are provided in Chapter 8.

The core idea of the VIIM is that the motion of surfaces nearby the interface determine the motion of the interface itself. Thus, the motion of the ϵ-level sets in some sense *regularises* the motion of the interface, even when the interface contains junctions. In particular, through the use of the Voronoi interface, the motion of the interface is essentially the average of the motion of neighbouring level sets. For example, in a curvature flow problem, triple points naturally move to satisfy certain angle conditions. As another example, if during one step, on one side of the interface the level sets move with speed $\gamma_1\kappa$ under curvature flow, and on the other side of the interface, they move with speed $\gamma_2\kappa$, then the net result is that the interface moves with speed $\frac{1}{2}(\gamma_1 + \gamma_2)\kappa$. This property is exploited in the next chapter in order to simulate curvature flow with variable coefficients of curvature.

Chapter 4

Applications of the VIIM

In the following, several example applications of the VIIM are presented. These include simple geometric and advection problems, as well as multiphase curvature flow with constraints in which curvature coefficients can be defined on a per-phase basis. A multiphase incompressible fluid flow solver is also developed, in which density, viscosity, and surface tension can be defined on a per-phase basis. In particular, convergence during topological change is demonstrated by analysing a "T1 event" driven by surface tension. Several applications of the fluid solver are shown, including the dynamics of cluster breakup in shear flow, bubble rising phenomena, and implementing general domains through the concept of an "exterior phase" in the VIIM.

4.1 Geometric flows

4.1.1 Constant speed normal driven flow

We first consider an example in which the interface moves in its normal direction with a constant speed that can change depending on the type of interface. A physical example corresponds to each phase being assigned a bulk energy density and the entire system evolving to minimise the total bulk energy. Under gradient descent, the interface between phase i and phase j moves with a speed proportional to the difference between bulk energy densities in phase i and j. In general, we may specify the speed F_{ij} of interface Γ_{ij} in the normal direction pointing from phase i into phase j. Thus we must require $F_{ji} = -F_{ij}$. The equation of evolution for ϕ is given by

$$\phi_t + F_{\text{ext}}|\nabla\phi| = 0,$$

where F_{ext} is an appropriate extension speed, computed by $F_{\text{ext}}(x) = F_{ij}$ where x is inside phase i, and j is chosen such that x is closest to Γ_{ij}. This uniquely defines j everywhere except on a set of measure zero, where one may choose any such j. This ambiguity does not affect the results of the VIIM, due to its underlying regularisation properties.

Figure 4.1 shows the results of two- and three-dimensional simulations of a system with three phases, composed of two circles (spheres in 3D), denoted by G=Green (left) and B=Blue (right), and

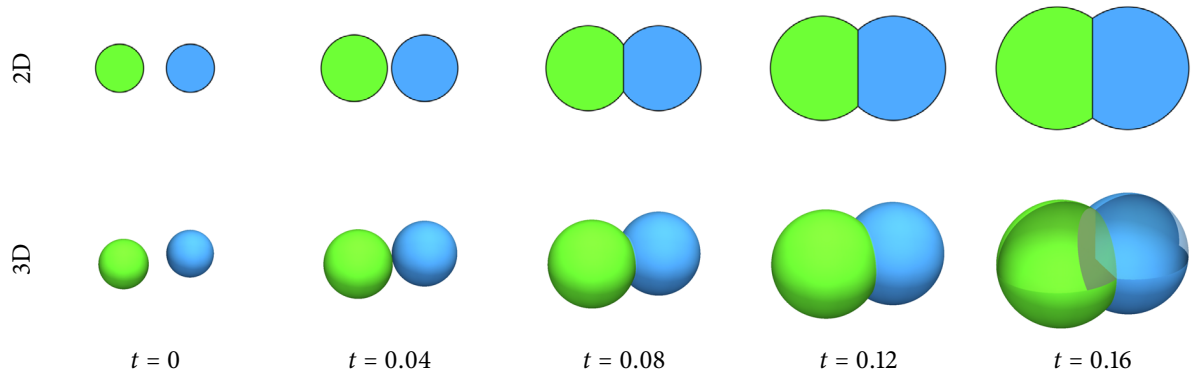

Figure 4.1. Results of a constant speed normal driven flow such that Green and Blue grow with unit speed into White and stop at each other: $F_{GW} = F_{BW} = 1$, $F_{GB} = 0$. The initial radii of the circles (spheres) is 0.1 with a separation of 0.3 between their centres.

an exterior phase (W=White), such that both Blue and Green expand into White with unit speed, $F_{BW} = F_{GW} = 1$, but neither Blue nor Green is allowed to advance on the other, $F_{BG} = F_{GB} = 0$. The resulting evolution is simply two circles (spheres) growing in size such that they do not overlap. In the last frame of the 3D simulation in Figure 4.1, a sector has been cut away to show that the interface Γ_{BG} is correctly obtained. In this and the next example, the frames are from a simulation computed on a 256×256 ($\times 256$) grid in the unit square (cube), with $\epsilon = 0^+$.

Figure 4.2 shows the results of a similar system but with $F_{WG} = 1$, $F_{BW} = 1$, $F_{GB} = 1$. Here White expands into Green, which in turn expands into Blue, which in turn expands into White. This type of flow generates two spiraling triple points in 2D, and a spiraling circular triple line in 3D, the positions of which do not change. A large number of turns is obtained, and the thickness of the spirals is limited by grid resolution. This example is discussed in [41], in which a mathematical framework, called the "vanishing surface tension" limit, is derived for studying two-dimensional multiphase systems whose evolution is governed by constant normal driven flow. The results in Figure 4.2 of the VIIM applied to this problem correspond to adding an artificial surface tension with strength proportional to the grid size h.

4.1.2 Advection by an external velocity field

In this example, the interface is advected by an external velocity field, such as one that might arise from rotation or translation. The equation of evolution is given by a pure advection equation, namely

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0.$$

In Figure 4.3, two- and three-dimensional simulations are shown using a constant velocity field that implements rotation. Here, a simple second order ENO upwinding scheme for the advection term and forward Euler in time was used. The simulation was computed on a 256×256 grid in 2D and a

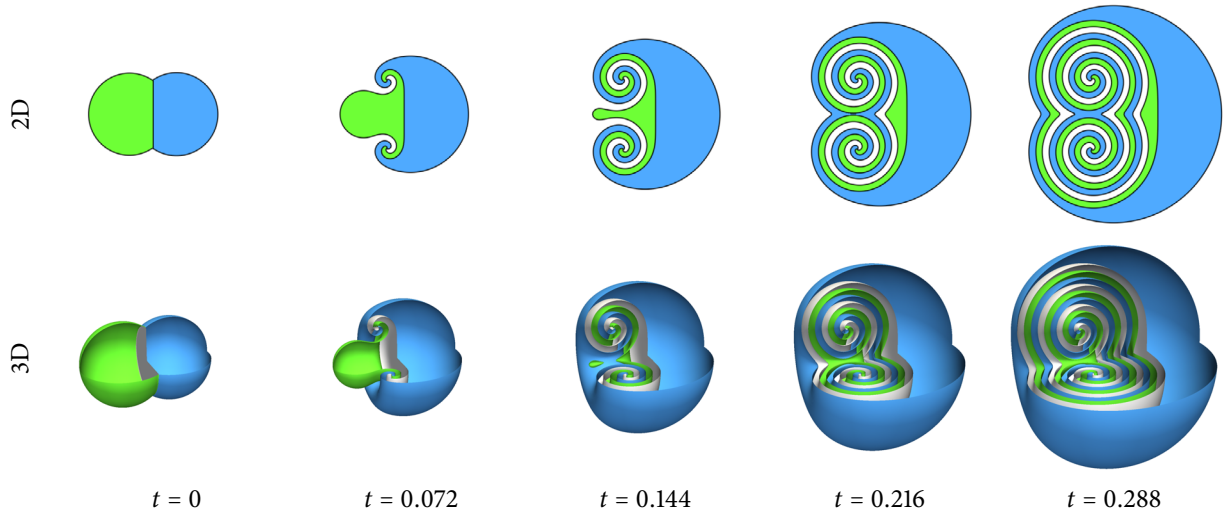

Figure 4.2. Results of a constant speed normal driven flow where White expands into Green, which in turn expands into Blue, which in turn expands into White, all with unit speed ($F_{WG} = 1$, $F_{BW} = 1$, $F_{GB} = 1$). This flow generates two spiraling triple points in 2D and a spiraling triple line in 3D that do not change position, where the thickness of the spirals is limited by grid resolution. The initial setup is two non-overlapping circles (spheres) with radii 0.175 and a separation of 0.2 between their centres. In the 3D case, a quadrant has been cut away to render the inside structure visible.

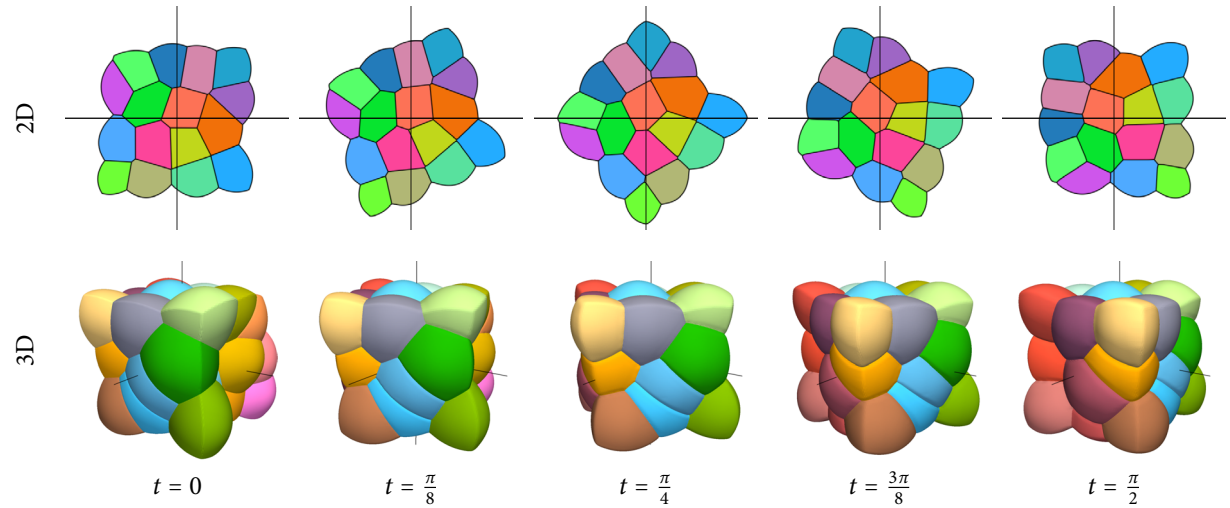

Figure 4.3. Advection by an external velocity field that implements counterclockwise rotation such that $t = 2\pi$ corresponds to one complete revolution about the indicated origin (axis of rotation pointing up in the case of 3D).

192^3 grid in 3D, applied to an initial set of 17 phases in 2D and 28 phases in 3D, with $\epsilon = 0^+$. Under this flow, the area (or volume) of each phase should be conserved, and we can perform a simple convergence test of this property. For a 2D case, let e be the error in conserving the area of each phase, measured by calculating the maximum deviation in area over the time interval $0 \le t \le T$ and summing this over each phase, i.e.

$$e = \sum_{i=1}^{N} \max_{t\in[0,T]} \left| A_i(t) - A_i(0) \right|, \tag{4.1}$$

where N is the number of phases and $A_i(t)$ is the area of phase i at time t. For the 2D simulation shown in Figure 4.3, the error e as a function of grid size is shown in Table 4.1. For this example and the associated numerical scheme, most of the numerical error in time and space is concentrated at the junctions. These occupy small regions of space and therefore contribute a small amount to the error in area conservation, and explains why the convergence rate observed in Table 4.1 is predominantly superlinear.

4.1.3 Mean curvature flow with constraints

In the previous chapter, Figure 3.15 showed an example in which the VIIM was applied to multiphase curvature flow. Here, the curvature flow equation is adapted to allow constraints.

Volume conservation

By adding a discontinuous source term to the right hand side of the mean curvature flow equation, the VIIM can simulate mean curvature flow with volume conservation. The modified forward Euler step is

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \gamma \kappa^n |\nabla \phi^n| + s^n |\nabla \phi^n|,$$

where

$$s^n(x) = \frac{V_i^0 - V_i^n}{A_i^n \Delta t} \quad \text{where } i \text{ is such that } x \in \Omega_i. \tag{4.2}$$

Here V_i^n denotes the volume (area in 2D) of phase i at time step n, V_i^0 is the initial volume (area) of phase i, and A_i^n is its surface area (perimeter in 2D) at time step n. In effect, the source term $s^n|\nabla \phi^n|$

n	error e	order
32	0.051121	–
64	0.016945	1.6
128	0.004958	1.8
256	0.001480	1.7
512	0.000535	1.5
1024	0.000316	0.8

Table 4.1. Convergence analysis for area conservation corresponding to the two-dimensional simulation of Figure 4.3. Here, the error e is measured using (4.1) with $N = 17$ phases and a grid size of $n \times n$.

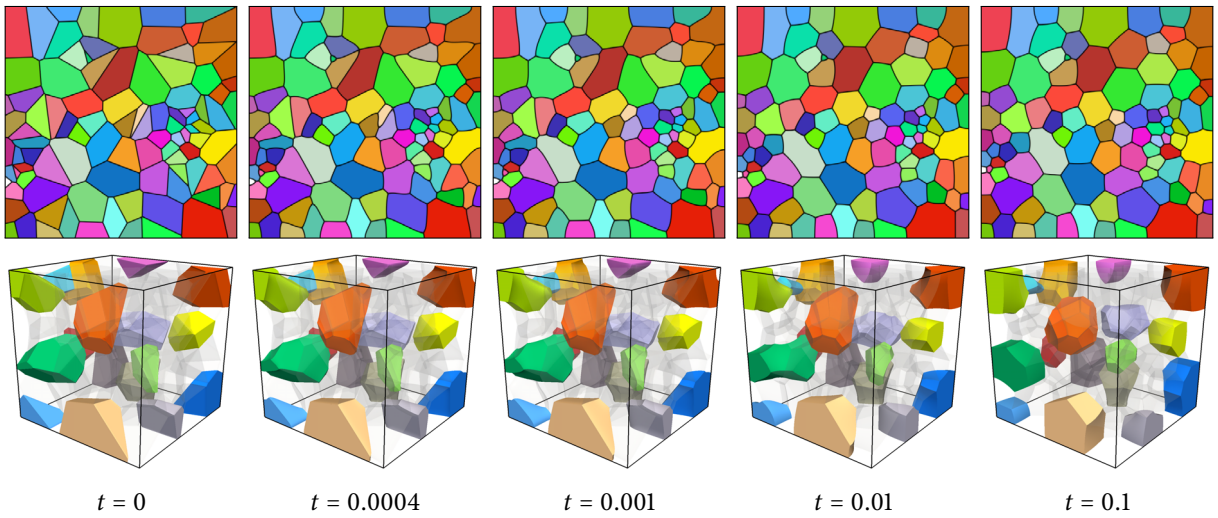

Figure 4.4. Mean curvature flow ($\gamma = 1$) with area/volume conservation on a set of 100 randomly created phases (zero Neumann boundary conditions). By the time $t = 0.1$, the solution has approximately attained equilibrium. (Top) Two-dimensional results, computed on a 256×256 grid on the domain $[0,1]^2$ (approximately $13,000$ time steps, taking about 20 minutes on a quad core laptop). (Bottom) Three-dimensional results, in which a selection of phases have been coloured solid, computed on a 128^3 grid in a unit cube (approximately $3,300$ time steps, taking about 150 minutes on an eight-core desktop).

grows or shrinks each phase equally around its boundary by an amount that corrects for any mass loss/gain. The volumes and surface areas of each phase at time step n are calculated by extracting the interface as a triangular mesh (polygon in 2D) using the method described in Section 3.8.

Despite each phase potentially growing or shrinking at different rates, the VIIM robustly and smoothly handles the discontinuity of the source term (4.2) and conserves volume almost exactly. In Figure 4.4, the method is demonstrated in two and three dimensions on a set of 100 initially random phases, using zero Neumann boundary conditions. Mean curvature flow minimises perimeter (surface area in 3D), which, subject to the constraint of area (volume) conservation, eventually attains an equilibrium. We see this in Figure 4.4, and in particular, one can observe various topological changes occur and at all times triple junctions have 120° angles.

Volume targeting

One can easily replace the constant V_i^0 in the source term in (4.2) with any constant. In particular, we can evolve a multiphase system under mean curvature flow to an equilibrium that has all phases with the same volume. Figure 4.5 shows the resulting evolution for the same set of phases used in Figure 4.4. In the two-dimensional case, the system ultimately attains a shape similar to a honeycomb. The best arrangement of tiles (in two dimensions) having the same area and minimal perimeter is known to be a honeycomb, but in the three-dimensional case the situation is less well-known.

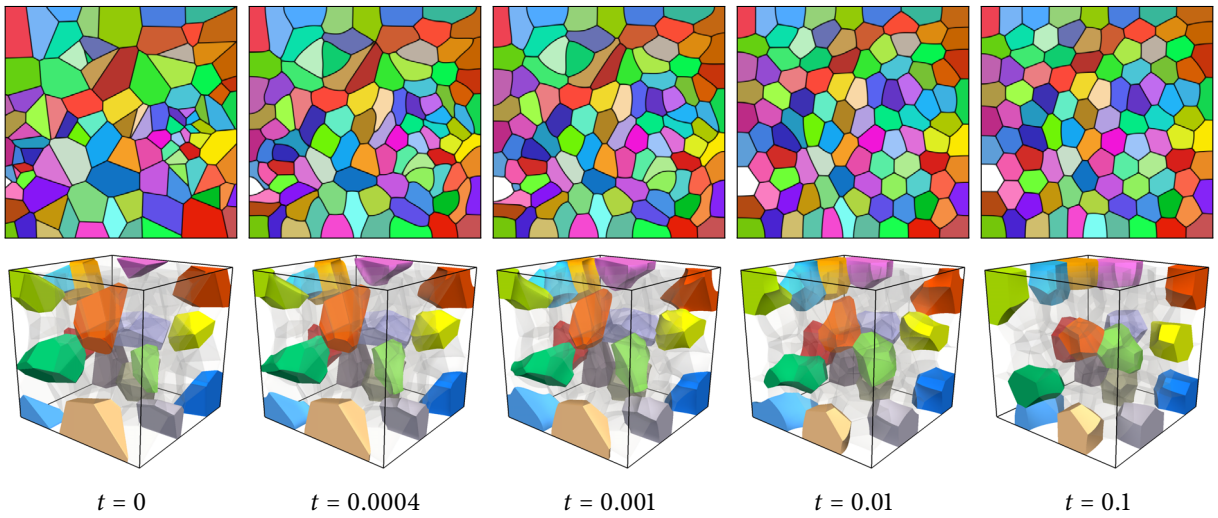

Figure 4.5. Mean curvature flow ($\gamma = 1$) with area/volume targeting, so that every phase ultimately has the same area/volume (zero Neumann boundary conditions). By the time $t = 0.1$, the solution has approximately attained equilibrium. (Top) Two-dimensional results, computed on a 256×256 grid on the domain $[0,1]^2$. (Bottom) Three-dimensional results in which a selection of phases have been coloured solid.

Different surface energy densities

One interpretation of curvature flow is that it corresponds to gradient descent on an energy functional measuring the surface energy of each interface. If all interfaces have the same surface energy density, then there is a global coefficient of curvature γ and the 120° angle Young's law holds. One can generalise this to the case where each interface Γ_{ij} has different surface energy densities, i.e. different coefficients of curvature. In this situation, a generalised Young's law states that the angles θ_i of a triple point satisfy

$$\frac{\sin\theta_i}{\gamma_{jk}} = \frac{\sin\theta_j}{\gamma_{ik}} = \frac{\sin\theta_k}{\gamma_{ij}}, \tag{4.3}$$

where θ_i is the angle in phase i and γ_{jk} is the coefficient of curvature of Γ_{jk}. Coefficients can be specified on a per-phase basis (rather than a per-interface basis), so that phase i has coefficient γ_i. For mean curvature flow, this naturally leads to interface Γ_{ij} having an effective coefficient of curvature of $\gamma_{ij} = \frac{1}{2}(\gamma_i + \gamma_j)$. Allowing different phases to have different coefficients of curvature can easily be implemented in the VIIM. The modified mean curvature flow equation is

$$\phi_t = \gamma\kappa|\nabla\phi|, \quad \text{where } \gamma(x) = \gamma_i \text{ and } i = \chi(x),$$

that is, the coefficient of curvature used in any finite difference updates is determined based on which phase occupies the grid point in question. The discontinuity in $\gamma(x)$ across the interface is easily handled by the VIIM, owing to the motion of the ϵ-level sets that exist solely in one phase. In Figure 4.6, the method is demonstrated by choosing a set of 100 random phases, with half being assigned a curvature coefficient of $\gamma_i = 0.1$ (white phases) and the other half a coefficient of $\gamma_i = 1$

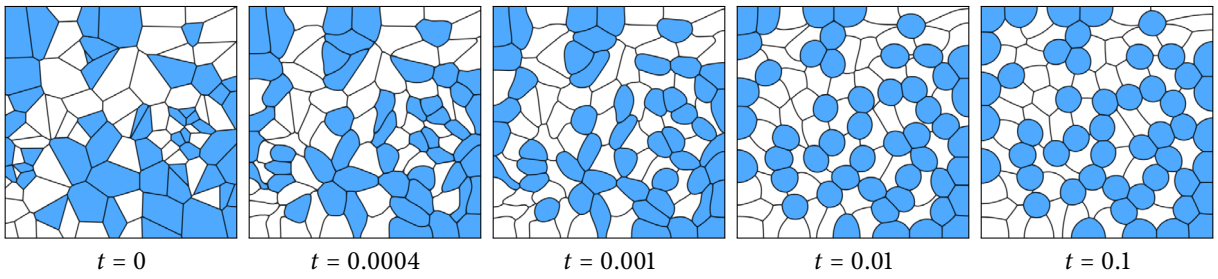

Figure 4.6. Mean curvature flow with area targeting, so that all phases ultimately have the same area, with variable coefficients of curvature: blue phases have $\gamma = 1$, while white phases have $\gamma = 0.1$. By the time $t = 0.1$, the solution has approximately attained equilibrium. Simulation computed on a 256×256 grid in the unit square, with zero Neumann boundary conditions.

(blue phases), moving under curvature flow with area targeting. According to Young's law (4.3), if three white phases or three blue phases meet at a triple point, then they do so at 120° angles. If two white phases and one blue phase meet at a triple point, the blue phase makes an angle of 170°, while the two white phases make 95° angles. If two blue phases and one white phase meet, the angles are 155.5° and 49°. This is observed in Figure 4.6, in which the blue phases surrounded by white phases are approximately circular. By using a suitably modified von Neumann-Mullins' law, it was confirmed that the VIIM correctly converges to a solution satisfying the generalised Young's law (4.3).

4.2 Fluid flow with permeability

Aside from geometric examples, the VIIM can also be applied to multiphase evolution determined by complex physics, and this is demonstrated here by coupling the method to multiphase incompressible fluid flow. As motivation, consider simulating the dynamics of a dry foam. A foam is a system of gas bubbles separated by a liquid component, and is considered "dry" when the fraction of volume occupied by the liquid is less than 10% [63]. In this case, the liquid forms a connected network of thin films separating the gas bubbles. These interfaces are flexible, exhibit surface tension, and meet at junctions (triple points in 2D and Plateau borders in 3D) with 120° angles. The membranes may also be permeable to the gas, so that gas can diffuse from one bubble to its neighbour, and this leads to "diffusive coarsening": bubbles with a large number of faces grow in size at the expense of bubbles with a small number of faces which shrink. As a foam evolves due to coarsening or any other dynamics, bubbles can change neighbourship with other bubbles, leading to complex topological changes, especially in three dimensions.

One can model the dynamics of a dry foam by considering the fluid mechanics of the gas phase coupled with membrane surface tension and permeability. For simplicity, here we consider the case where the membranes can be idealised as massless and infinitely thin, ignoring effects of liquid flow inside these membranes.[1] This system was studied by Kim *et al.* [16, 17], using an immersed

[1] A multiscale model that considers both macroscale gas dynamics and microscale liquid dynamics is developed in

boundary method. In that work, fluid mechanics of the gas were coupled to the evolution of the interface, allowing large shear forces to be applied to foams far from equilibrium. Two-dimensional motion was considered, and topological changes were not allowed.

In the following, we develop a multiphase fluid flow solver that works in both two and three dimensions and automatically handles topological changes. Suppose there are N phases, labelled $i = 1, \ldots, N$. Each phase satisfies the incompressible Navier-Stokes equations with density ρ_i and viscosity μ_i. Consider now an interface Γ_{ij} between phase i and phase j, having surface tension σ_{ij}. The interface provides a force of surface tension that induces a pressure jump across the interface of $[p] = \sigma_{ij}\kappa$ where κ is the mean curvature of the interface (measured with an orientation consistent with defining the pressure jump $[p]$). When there is no permeability, the interface is advected by the velocity $\mathbf{u}$ of the fluid, assumed to be continuous across the interface. When there is permeability, the rate of diffusion of gas/fluid is proportional to the pressure difference [63]. Here, the approach used in [16] is adopted, whereby permeability is modelled as a slip of the interface in the normal direction relative to $\mathbf{u}$. The velocity of the interface in this case is thus $\mathbf{u} - M\sigma_{ij}\kappa\mathbf{n}$, where $M \geq 0$ is a physical parameter denoting the amount of permeability, and $\mathbf{n}$ is the unit normal of the interface (with the same orientation as that used to calculate κ).

In practice, any discontinuities in density and viscosity of the fluid across the interface must be regularised. A standard approach used in two-phase fluid flow problems (see the review [64]), is to smooth the density and viscosity across the interface, through the use of a smoothed Heaviside function. In the multiphase case, we define

$$\rho(x) = \frac{\sum_{i=1}^{N} \rho_i H_\varsigma(\phi_i)}{\sum_{i=1}^{N} H_\varsigma(\phi_i)}, \qquad \mu(x) = \frac{\sum_{i=1}^{N} \mu_i H_\varsigma(\phi_i)}{\sum_{i=1}^{N} H_\varsigma(\phi_i)}, \tag{4.4}$$

where ϕ_i is a signed distance function to the boundary of phase i (positive inside phase i) and H_ς is a commonly used smoothed Heaviside function given by

$$H_\varsigma(x) = \begin{cases} 0 & \text{if } x \leq -\varsigma, \\ \frac{1}{2}(1 + \frac{x}{\varsigma} + \frac{1}{\pi}\sin\frac{\pi x}{\varsigma}) & \text{if } |x| \leq \varsigma, \\ 1 & \text{if } x \geq \varsigma. \end{cases} \tag{4.5}$$

Here, ς measures the width of the smoothing transition (and is usually denoted by ϵ, but to avoid confusion with the ϵ used in the VIIM, is denoted differently here). In the following simulations, the amount of smoothing was set to be $\varsigma = 2h$. Note that in (4.4), the sum of Heaviside functions has been used to normalise the quantities. This only matters at junctions, e.g. triple points, and is required to ensure $\min_i \rho_i \leq \rho(x) \leq \max_i \rho_i$ and $\min_i \mu_i \leq \mu(x) \leq \max_i \mu_i$.

Surface tension in the multiphase system is modelled as a body force through the use of a Dirac delta function [64–66]. In a two-phase fluid flow problem, this force takes the form $\mathbf{st} = -\sigma\kappa\delta(\phi)\nabla\phi$ where ϕ is a signed level set function for the interface. In the multiphase case, we have multiple phases which meet at triple junctions (and quad points in 3D), where "curvature" needs to be suitably defined. It is both physically and mathematically natural to take the same definition applied to each

Chapter 6.

phase separately, sum all of these, and normalise by a factor of two. This is physically consistent, since in the case of a dry foam, each bubble is separated from the others by a thin membrane, and it is mathematically natural because the resulting formula effectively enforces Young's law at triple points. To account for different interfaces having different surface tensions, we suppose that each phase has an effective "surface tension" σ_i such that $\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$. This is similar to the case of variable surface energy densities considered in Section 4.1.3. Using this formulation for surface tension for a multiphase system, we therefore define

$$\mathbf{st} = -\frac{1}{2}\sum_{i=1}^{N} \sigma_i \kappa(\phi_i)\delta(\phi_i)\nabla\phi_i, \tag{4.6}$$

where ϕ_i is a signed level set function for phase i. Note that this gives the correct surface tension of $\sigma_{ij} = \frac{1}{2}(\sigma_1 + \sigma_2)$ on the interface between phase i and j. We also note that the normal vector $\nabla\phi_i$ is not well-defined at corners, e.g. at a triple junction, but the "curvature times the normal", $\kappa(\phi_i)\nabla\phi_i$, is well-defined as a distribution, and is a Dirac delta function with magnitude related to the angle of the corner.

Of course, in practice we must smooth the surface tension term onto the grid. One possible approach is to use (4.6) directly but with smoothed Dirac delta functions, i.e.

$$\mathbf{st}_\varsigma = -\frac{1}{2}\sum_{i=1}^{N} \sigma_i \kappa(\phi_i)\delta_\varsigma(\phi_i)\nabla\phi_i,$$

where δ_ς is a smoothed one-dimensional delta function, e.g. the derivative of the smoothed Heaviside function in (4.5),

$$\delta_\varsigma(x) = \begin{cases} \frac{1}{2\varsigma}\left(1 + \cos\frac{\pi x}{\varsigma}\right) & \text{if } |x| < \varsigma, \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

However, this method can suffer from excessive noise at triple junctions, due to the singularity in the curvature and gradient calculations that are essentially decoupled. Better results are obtained by instead *mollifying* the surface tension to smooth it, i.e. defining $\mathbf{st}_\varsigma := \mathbf{st} * \delta_\varsigma^n$ where δ_ς^n is an n-dimensional mollifier. This method exhibits less noise because the calculation of interface curvature and normal are coupled together in a consistent fashion, using an explicitly reconstructed interface, as follows. We have

$$\begin{aligned} \mathbf{st}_\varsigma(x) &= (\mathbf{st} * \delta_\varsigma^n)(x) \\ &= \int_\Omega \mathbf{st}(y)\delta_\varsigma^n(x-y)\,dy \\ &= -\frac{1}{2}\sum_{i=1}^{N} \int_\Omega \delta(\phi_i)(y)\big(\sigma_i\kappa(\phi_i)\nabla\phi_i\big)(y)\delta_\varsigma^n(x-y)\,dy \\ &= -\frac{1}{2}\sum_{i=1}^{N} \int_{\Gamma_i} \sigma_i(\kappa\mathbf{n})(y)\,\delta_\varsigma^n(x-y)\,dS(y). \end{aligned} \tag{4.8}$$

We see that the n-dimensional convolution becomes a surface integral of $\kappa\mathbf{n}$ weighted by the mollifier. The formula further simplifies if we use a piecewise linear reconstruction of the interface $\Gamma = \bigcup_i \Gamma_i$,

i.e. a polygon in 2D or polyhedron in 3D using the procedure in Section 3.8, since then $\kappa\mathbf{n}$ is itself a sum of delta functions with support on the vertices of the polygon or edges of the polyhedron. The final resulting formula obtained with this approach is similar to that obtained in immersed boundary methods and better treats the singularity in surface tension calculations at triple junctions. In the following, the n-dimensional mollifier $\delta_\varsigma^n(x) = \prod_{i=1}^n \delta_\varsigma(x_i)$ has been used, where δ_ς is given in (4.7) and $x = (x_1, \ldots, x_n)$, and was found to give accurate pressure calculations.

The equations of motion for the entire multiphase system are therefore the incompressible Navier-Stokes equations with variable density, viscosity, and surface tension in the form of a body force, and are given by

$$\begin{aligned}\rho\big(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}\big) &= -\nabla p + \nabla\cdot\big(\mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\big) + \mathbf{st}_\varsigma(\phi) + \mathbf{F},\\ \nabla\cdot\mathbf{u} &= 0,\\ \phi_t + \mathbf{u}\cdot\nabla\phi &= M\sigma\kappa|\nabla\phi|,\end{aligned}$$

where ρ, μ are given by (4.4), $\mathbf{st}_\varsigma$ is given by (4.8), and $\mathbf{F}$ is any additional body forces (such as gravity). To solve the Navier-Stokes equations numerically, a first order in time, second order in space, variable density approximate projection method [67] is used, based on Chorin's projection method [68]. Briefly, the scheme is as follows: suppose $\mathbf{u}^n$ and ϕ^n are known at time step n, then compute

$$\phi^{n+1} = \phi^n - \Delta t(\mathbf{u}\cdot\nabla\phi)^n + \Delta t M\sigma(\kappa|\nabla\phi|)^n$$

using a second order ENO upwinding scheme for the advection term, standard central differences for the curvature term, and reconstructing ϕ when necessary as an unsigned distance function using the Voronoi interface. Using the Voronoi interface of ϕ, a mesh is extracted and used to evaluate the surface tension force, to produce an intermediate velocity field $\mathbf{u}^*$ satisfying

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u}\cdot\nabla\mathbf{u})^n = \frac{1}{\rho(\phi^{n+1})}\Big(\nabla\cdot\big(\mu(\phi^{n+1})(\nabla\mathbf{u} + \nabla\mathbf{u}^T)^{n+1/2}\big) + \mathbf{st}_\varsigma(\phi^{n+1}) + \mathbf{F}^n\Big).$$

Here, a second order upwinding ENO scheme for the advection term is used, and Crank-Nicholson is used for the diffusion term, leading to an implicit equation for $\mathbf{u}^*$. Note that there is a mixture of explicit and semi-implicit terms, and that the density/viscosity and forcing terms are evaluated at different points in time. Combined, these choices lead to a simple time-stepping method which is first order and relatively stable. The scheme calculates an intermediate velocity field $\mathbf{u}^*$ which is not necessarily divergence-free. In the projection step, we find the velocity at time step $n+1$ by solving

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p^n}{\rho(\phi^{n+1})}$$

where the pressure p is computed such that $\nabla\cdot\mathbf{u}^{n+1} = 0$. This leads to the variable coefficient Poisson equation

$$\nabla\cdot\Big(\frac{1}{\rho(\phi^{n+1})}\nabla p^n\Big) = \frac{1}{\Delta t}\nabla\cdot\mathbf{u}^*.$$

This in turn can be solved by using a variable coefficient Poisson solver, based on a finite element method, together with Neumann boundary conditions. For more details, see the schemes developed in [67]. In this work, the implicit equation for $\mathbf{u}^*$ has been solved using a simple Conjugate Gradient scheme, while the projection method was implemented with a variable coefficient multigrid solver. The entire multiphase fluid flow solver has also been parallelised with MPI.

4.2.1 Testing convergence during topological change

In a multiphase fluid flow calculation, accurately simulating topological changes represents a significant challenge. Here, convergence tests are performed to examine the ability of the VIIM and the Navier-Stokes solver to accurately calculate the evolution of a "T1" topological change in a two-dimensional foam. A T1 change (see [63]) is one in which two triple points come together and temporarily form a quadruple point, which then splits apart into two different triple points. During this process, an interface between two phases is destroyed and a new interface between the other two phases is created.

As a specific example, consider the situation shown in Figure 4.7, whereby two long and thin phases split apart from each other. The domain is $[0, \frac{1}{2}] \times [0, 1]$ and periodic boundary conditions are used, so that the velocity field is periodic on all four sides of the domain, and the interface is periodic on the bottom and top boundaries. In Figure 4.7, the flow of liquid is indicated by drawing representative streamlines, together with a density plot of the stream function ψ (satisfying $-\Delta\psi = -\partial_y u + \partial_x v$). During the evolution, the two long and thin phases (labelled Ω_1 and Ω_2) begin to retract so that their triple points merge. At time $t \approx 0.038$, two quadruple points instantaneously exist before splitting, so that phases Ω_1 and Ω_2 are no longer in contact, while Ω_3 and Ω_4 are now in

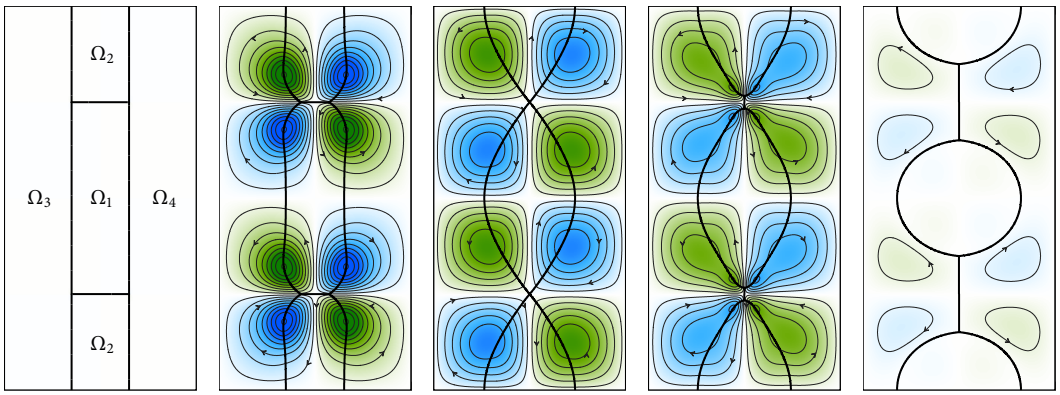

Figure 4.7. Evolution of a four-phase fluid system which has two T1 topological changes. Streamlines are shown, and the colours indicate the magnitude of the stream function ψ. At $t = 0$, a schematic is illustrated of the initial condition having periodic boundary conditions on the bottom and top (and the velocity field is periodic on all four sides). Due to local effects of surface tension, the triple points start to retract ($t = 0.008$), temporarily form two quadruple points ($t \approx 0.038$), which then split apart ($t = 0.042$), by which point Ω_1 and Ω_2 have detached from one another.

	Time of T1 event			Evolution	
h	$T_1(h)$	$T_1(2h) - T_1(h)$	order	d_h	order
1/64	0.050781	–	–	–	–
1/128	0.044470	0.006311	–	0.008390	–
1/256	0.040723	0.003748	0.8	0.007231	0.2
1/512	0.038704	0.002018	0.9	0.005036	0.5
1/1024	0.037512	0.001192	0.8	0.002720	0.9

Table 4.2. Convergence results for the evolution in Figure 4.7. Here, $T_1(h)$ is the time of the T1 event on grid with cell size h, and supposing that to leading order, $T_1(h) = t_0 + Ch^p$, the order column calculates p by using ratios of $T_1(2h) - T_1(h)$. In addition, $d_h = d(\Gamma_h, \Gamma_{2h})$ measures the difference in interface evolution, in space and time, over the time interval $0 \le t \le 0.1$.

contact. The system is near steady-state by the time $t = 0.1$. In this example, phases Ω_1 and Ω_2 have $\sigma_i = 10$, while phases Ω_3 and Ω_4 have $\sigma_i = 1$. These values of σ_i were chosen to make a T1 change favourable for the geometry considered here. In particular, the final state has much less surface energy than the initial configuration. To complete the specification of the physical parameters, the density and viscosity of all four phases have been set to $\rho = 1$ and $\mu = 0.1$.

With a grid size of $\frac{n}{2} \times n$ (so that $h = 1/n$), a convergence analysis is performed on the computed time of the T1 event, and grid convergence for the evolution of the interface is measured. Let $T_1(h)$ denote the time of the topological change computed on a grid with cell size h. This time is calculated by detecting the first instance of when an interface between phase Ω_3 and phase Ω_4 is born. Since the time step Δt is coupled to h, we may suppose that $T_1(h) = t_0 + \mathcal{O}(h^p)$ and measure the convergence rate p, where t_0 is the precise time of the $T1$ event. In addition, let $d_h := d(\Gamma_h, \Gamma_{2h})$ denote the difference in interface evolution for grid sizes h and $2h$, measured using the L^1 norm in time and the Hausdorff metric in space; see (3.8). This is the same method we used in §3.10 to measure convergence of the interface location in both space and time. Table 4.2 contains the results of these convergence measurements for the numerical method, computed on grid sizes of 32×64 to 512×1024. We see that the time of the T1 event is predicted with near first order accuracy, and that the VIIM successfully converges in both space and time, accurately predicting the evolution of multiphase fluid dynamics through a topological change.

4.2.2 Application to dry foams

To demonstrate the application of the VIIM to dry foams, we consider a foam that is being agitated by a strong external force, such that the foam is first spun counterclockwise, settles momentarily, and then is spun clockwise for the same amount of time. Results are shown in both two and three dimensions, for cases with and without permeability.

Two-dimensional results

The simulation parameters are as follows. The domain is a unit square $[0,1]^2$ with periodic boundary conditions, and all phases have the same density and viscosity, with $\rho = 1$, $\sigma = 1$, $\mu = 0.005$. These

parameters do not necessarily correspond to a specific physical problem, but are chosen so that effects of inertia, viscosity, and surface tension are of similar importance. To implement the agitator, an external force **F** in the Navier-Stokes momentum equations is used, given by

$$\mathbf{F}(x, y, t) = 15\big(\sin \pi x \sin 2\pi y, -\sin 2\pi x \sin \pi y\big) \sin \pi t, \quad 0 \le x, y \le 1,$$

which corresponds to a spinning force, in the counterclockwise direction about the centre point $x = y = \frac{1}{2}$ for $0 \le t \le 1$, and clockwise about the centre point for $1 \le t \le 2$. The factor of 15 in **F** has been chosen to give a relatively strong shearing/spinning force that dominates the stabilisation effects of surface tension.

A case with no permeability, i.e. $M = 0$, is shown in Figure 4.8. Starting with 25 random phases, the Navier-Stokes equations are evolved with the agitator forcing on a 256×256 grid, using the $\epsilon = 0^+$ method in the VIIM. The figure illustrates the results with plots of phase evolution, streamlines and stream function, and pressure fields, over five different points in time, $t \approx 0$, $0.5, 1, 1.5$ and 2, corresponding to extremums in the agitator forcing. We see that there is significant shearing, causing considerable rearrangement of the phases and several topological changes. The stream function plots show how the flow is strongly affected by the agitator forcing, but is also localised in nature to due to effects of surface tension. The pressure plots show how capillary waves are generated as the system evolves.

Next, a case with permeability is considered, with $M = 0.04$. This leads to significant coarsening over the time interval $0 \le t \le 2$. Starting with the same configuration[2] as in the previous example, Figure 4.9 illustrates the results. The simulation was performed on a 1024×1024 grid with periodic boundary conditions and $\epsilon = 2h$ in the VIIM. Despite strong shearing and complicated fluid-interface interaction, the fluid flow is incompressible and so von Neumann-Mullins' law describing the rate of change of area of each phase should nevertheless hold, and gives

$$\frac{dA}{dt} = 2\pi M \sigma\Big(\frac{n}{6} - 1\Big),$$

where n is the number of sides of a particular phase (or one of its connected components if it has more than one component). The area as a function of time for some of the phases in this simulation are plotted in Figure 4.10. Similar to the plot obtained in the convergence tests (i.e. Figure 3.16), we see that the area of each phase is a piecewise linear function of time, with slope a function of the number of sides only, in excellent agreement with von Neumann-Mullins' law, even in the presence of complicated fluid flow.

Three-dimensional results

We now consider a three-dimensional analogue of the agitator. The domain is a unit cube $[0, 1]^3$ and the external force is essentially the same but with no forcing in the z–direction, i.e.

$$\mathbf{F}(x, y, z, t) = 15\big(\sin \pi x \sin 2\pi y, -\sin 2\pi x \sin \pi y, 0\big) \sin \pi t, \quad 0 \le x, y, z \le 1,$$

[2]The factor in the external force **F** has been decreased to 10 in this example.

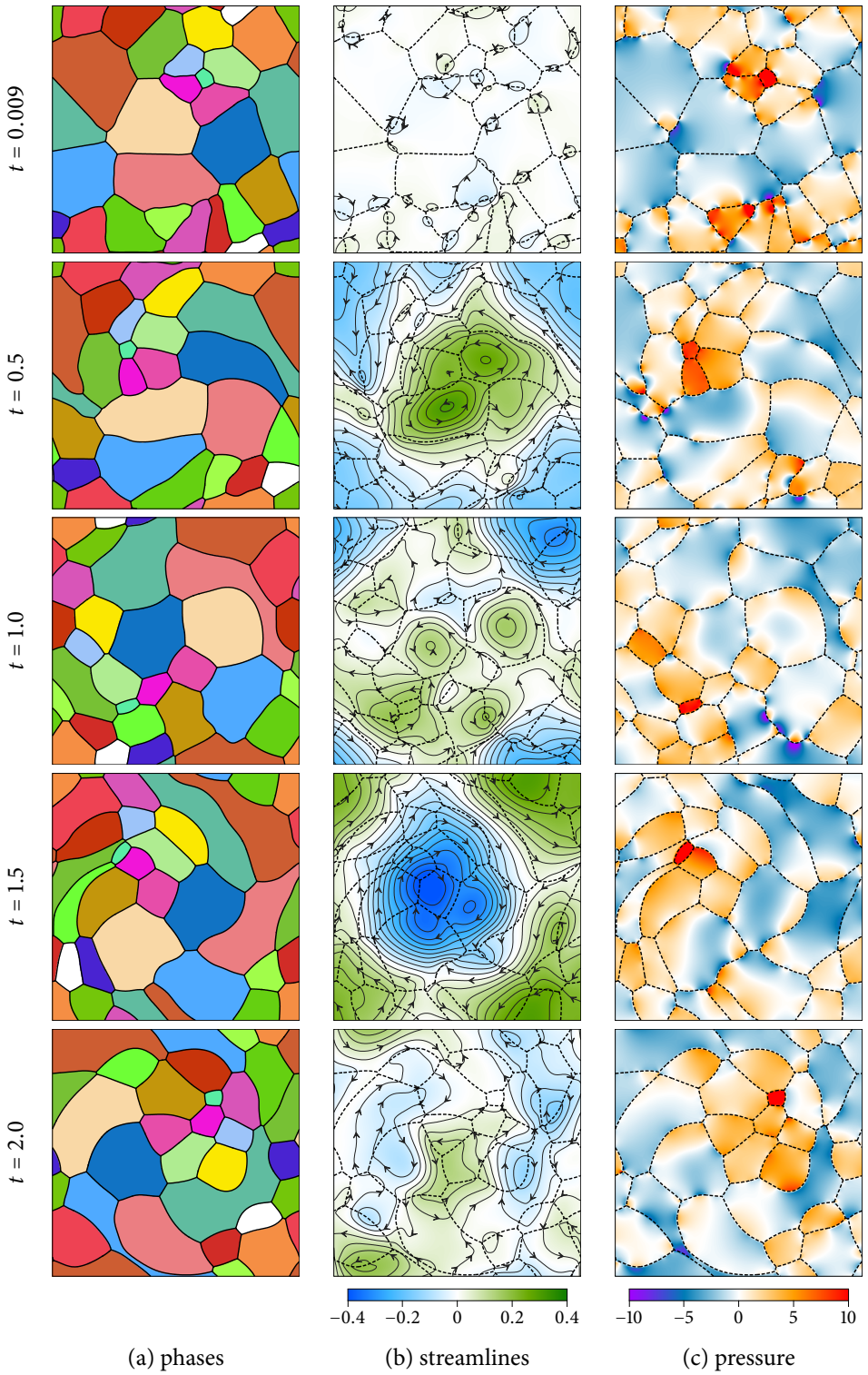

Figure 4.8. Results of a fluid flow simulation with an external agitator force, without permeability.

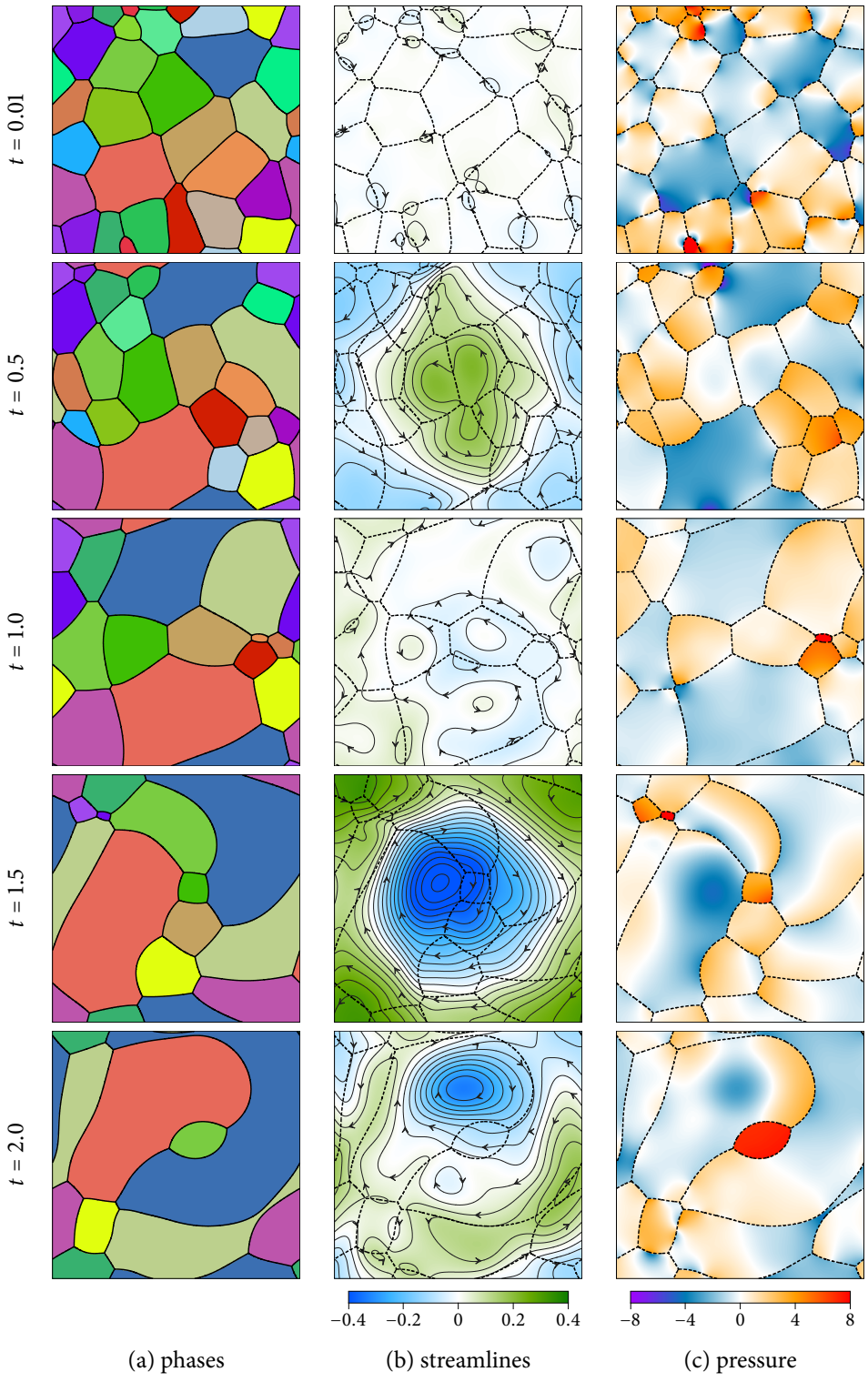

Figure 4.9. Results of a fluid flow simulation with an external agitator force and permeability.

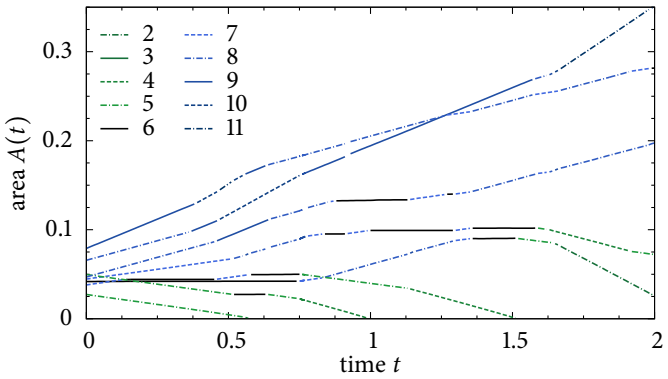

Figure 4.10. Area as a function of time for some of the phases in the simulation of Figure 4.9. Each line corresponds to a different phase, and the colour and style is determined by the number of sides that phase had at that particular instant.

which corresponds to a spinning force about a line with coordinates $x = y = \frac{1}{2}$. Other parameters are left unaltered, i.e. $\rho = 1$, $\sigma = 1$, $\mu = 0.005$ and periodic boundary conditions are used.

Two cases are considered: without permeability ($M = 0$) and with permeability ($M = 0.05$). We start with 125 random phases and evolve the Navier-Stokes equations with the agitator forcing; Figure 4.11 illustrates the results over five different points in time, $t \approx 0, 0.5, 1, 1.5$ and 2. The simulation was performed on a $128 \times 128 \times 128$ grid with periodic boundary conditions and $\epsilon = 0^+$. We have coloured solid 13 representative phases in order to visualise the bulk flow of the agitation. The velocity field is illustrated by freezing the velocity field and drawing streamlines seeded from a random set of points, for each time frame. The streamlines are coloured according to the azimuthal component of the velocity field, relative to the centre line of the agitation, computed by $\mathbf{u} \cdot \hat{\theta}$ where

$$\hat{\theta} = \frac{(-y + \frac{1}{2}, x - \frac{1}{2}, 0)}{\left[(-y + \frac{1}{2})^2 + (x - \frac{1}{2})^2\right]^{\frac{1}{2}}} \tag{4.9}$$

is the azimuthal tangent vector corresponding to a cylindrical coordinate system with centre line $x = y = \frac{1}{2}$. Thus, the colour of the streamlines give an indication of the direction the flow is spinning about the centre line, where green means counterclockwise (looking down from the tip of the indicated arrow), and blue means clockwise. When the forcing is at its maximum, i.e. when $t = 0.5$ and $t = 1.5$, the fluid flow is predominantly in the direction of forcing. At intermediate times, effects of surface tension are more apparent and the flow is more localised in nature. This is especially noticeable in the case of no permeability, in which the stabilisation effects of surface tension act more quickly to decrease the momentum gained from the agitator forcing. As in the two-dimensional case, there is significant rearrangement of phases, with several topological changes.

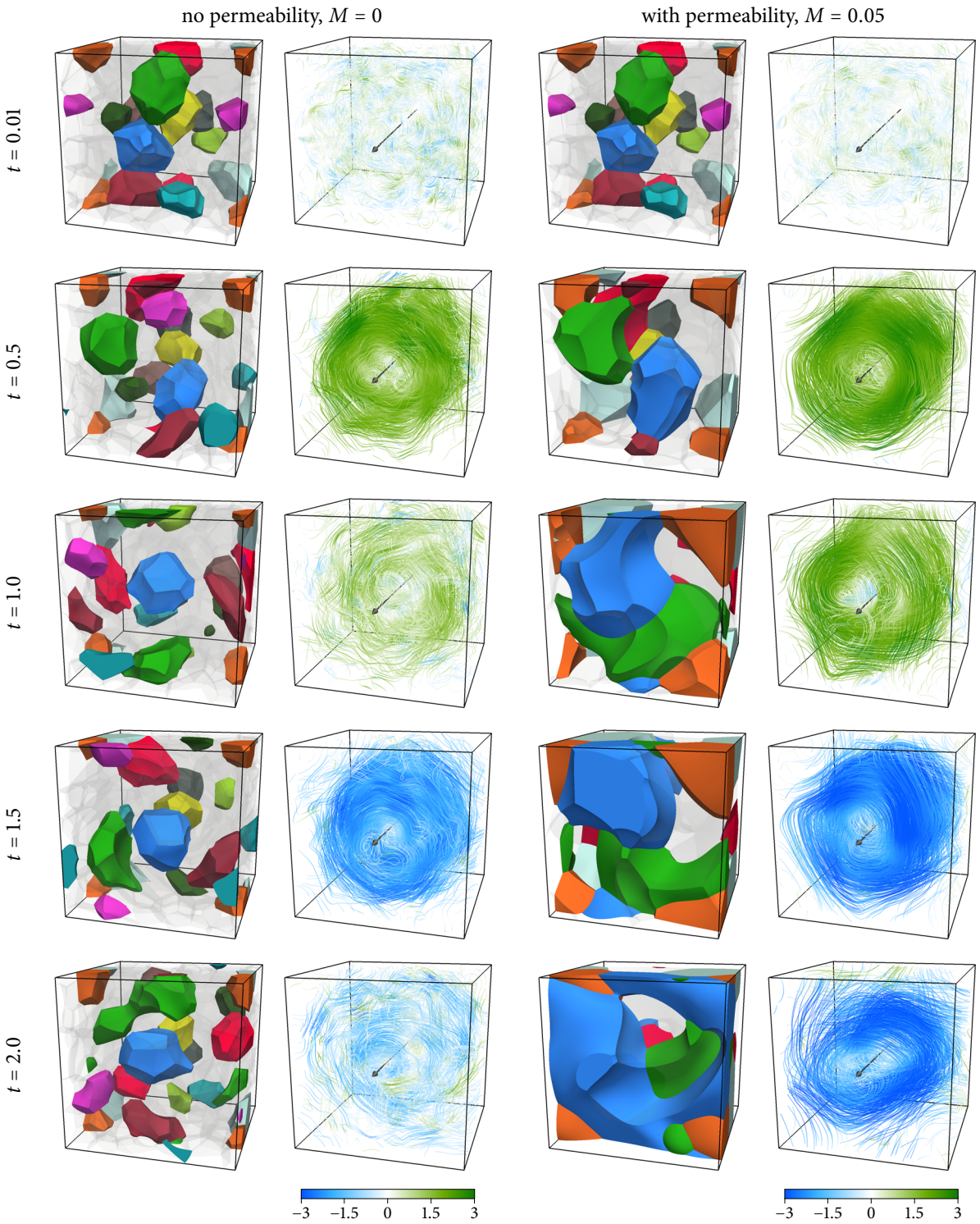

Figure 4.11. Results of a fluid flow simulation with an external agitator force in three dimensions, with and without and permeability. Streamlines are coloured by the azimuthal component of the velocity field, computed by $\mathbf{u} \cdot \hat{\theta}$, where $\hat{\theta}$ is defined in (4.9).

4.2.3 Phase-dependent fluid properties

In the next application, the VIIM is coupled to multiphase fluid flow with variable density and viscosity. This is demonstrated with a foam that has one phase more dense and more viscous than the other phases. In addition, this phase initially has several components dispersed throughout the foam. Under the action of gravity, the heavier phase sinks and the surrounding less dense phases rise. This process depends on the competing effects of surface tension at triple junctions and gravity: regions of denser liquid need to be sufficiently large in size for the force of gravity to dominate the stabilisation effects of surface tension. As the system evolves, and the components of the heavier phase sink, they merge together, forming a pool of liquid at the bottom. In these examples, the effect of different interfaces having different surface tensions is also demonstrated, giving rise to triple junctions with different angle configurations.

The parameters of the simulation are as follows. The domain Ω is the unit square (unit cube in 3D), and slip boundary conditions on the velocity field are employed, given by

$$\mathbf{u} \cdot \mathbf{n} = 0, \quad \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \tau = 0, \quad \text{on} \quad \partial\Omega,$$

where $\mathbf{n}$ is the normal to the boundary and τ is any tangent vector to the boundary. Identifying the heavy phase with the label $\mathcal{H}$, the density, viscosity, and surface tension for each phase i is set as

$$(\rho_i, \mu_i, \sigma_i) = \begin{cases} (1, 0.005, 0.1) & \text{if } i = \mathcal{H}, \\ (0.1, 0.00005, 0.01) & \text{otherwise.} \end{cases}$$

The system therefore has density ratios of ten and viscosity ratios of 100, in such a way that there is a Reynolds number ratio of ten (using the same velocity and length scales independent of the phase). Due to the different surface tensions, there are also different triple junction angles. According to the generalised Young's law (4.3), the angles $(\theta_i, \theta_j, \theta_k)$ made by phases i, j, k at a triple point are

$$(\theta_i, \theta_j, \theta_k) = \begin{cases} (120^\circ, 120^\circ, 120^\circ) & \text{if none of } i, j, k \text{ equal } \mathcal{H}, \\ (170^\circ, 95^\circ, 95^\circ) & \text{if } i = \mathcal{H} \text{ and } j, k \neq \mathcal{H}. \end{cases} \tag{4.10}$$

Permeability is set to zero, $M = 0$, and the external force $\mathbf{F}$ in the Navier-Stokes equations is given by gravity, $\mathbf{F} = \rho g \hat{\mathbf{g}}$, where $g = 5$ and $\hat{\mathbf{g}}$ is a unit vector pointing down. Once again, while these choices of parameters may not necessarily correspond to a particular physical situation, they have been chosen to illustrate the various effects of variable density, viscosity, and surface tension.

A 90° contact angle model has been implemented, whereby the interface meets the boundary of the domain at 90° angles. Since the unsigned distance function ϕ in the VIIM is advected only by $\mathbf{u}$, and $\mathbf{u}$ satisfies slip boundary conditions, it is inappropriate to specify boundary conditions on ϕ. Instead, the contact angle model is implemented through the surface tension force in the Navier-Stokes equations: the force drives the velocity field which in turn restores 90° angles in the interface. This is implemented in the same way that would occur if the contact point was considered to be an imaginary triple point that was allowed to move only tangentially along the boundary.

Finally, some remarks about connected components and merging and breaking: just as in the standard level set method, the VIIM easily and automatically handles merging and breaking, which

means the number of connected components of any particular phase can change over time, depending on the dynamics driving the evolution of the interface. In the following simulation, we make the analogy that the heavier phase $\mathcal{H}$ is a liquid embedded in a foam of gas bubbles, and then these connected components are used in two distinct ways:

- The heavy phase $\mathcal{H}$ will initially have several connected components of various different sizes. Due to the external force of gravity and the higher density of phase $\mathcal{H}$, these components will tend to sink to the bottom and merge together, forming a pool of liquid at the bottom of the domain. Thus, we rely on the VIIM to handle the merging of the different components of phase $\mathcal{H}$ into fewer connected components. All these components have the same identifier in the indicator function χ.
- On the other hand, the less dense phases are thought of as making a "gaseous foam", in which merging of different bubbles of gas is not allowed. Thus, here we *choose* to not allow any of the less dense phases to have more than one connected component. This means that, as the simulation evolves, if one of these phases splits into two components, then each component is separately given a new unique identifier, thereby preventing re-merging at a later point in time. As a result, at the end of the simulation we often have more than the initial number of phases. This choice is a simple model of the complex multiscale phenomena of foam production, whereby microscale dynamics of thin-film membranes, surfactant flow, and surface tension determine the macroscopic creation and evolution of foams.

Two-dimensional results

We begin with a two-dimensional simulation of the above variable density fluid flow problem. Figure 4.12 illustrates the results, starting with the configuration shown at time $t = 0$, and is computed on a 256×256 grid with slip boundary conditions, with $\epsilon = 0^+$. Here, the heavier phase, initially having nine separate components of circular shape, is coloured orange. The other phases, of which there are initially approximately 35, are coloured shades of blue and green. Figure 4.12 shows snapshots of the simulation at different times of note, showing plots of phase evolution, streamlines and stream function, and pressure fields.

We note several features of the results. Most of the components of the heavy phase $\mathcal{H}$ (shown in orange) sink to the bottom. In particular, the component initially attached to the top, first falls down, leaving behind it a trailing tail. It then detaches from the top boundary, forming a jet that quickly retracts (as seen at $t = 1.32$). On the other hand, the two smallest components of the heavy phase do not sink, and remain embedded in the foam at time $t = 1.81$. Here, the local forces of surface tension, particularly at the triple points, dominate the force of gravity and prevent them from falling. This is similar to an air bubble at the surface of water: depending on its diameter, the bubble can range from being almost fully submersed and spherical in shape to entirely on the surface with a hemispherical shape.

By the time the component of $\mathcal{H}$ initially at the top merges with the other components at the bottom, it has gained a large amount of momentum. This causes some less dense phases to break off from the bulk and be submerged (see, e.g., time $t = 0.92$). These two bubbles, still submerged

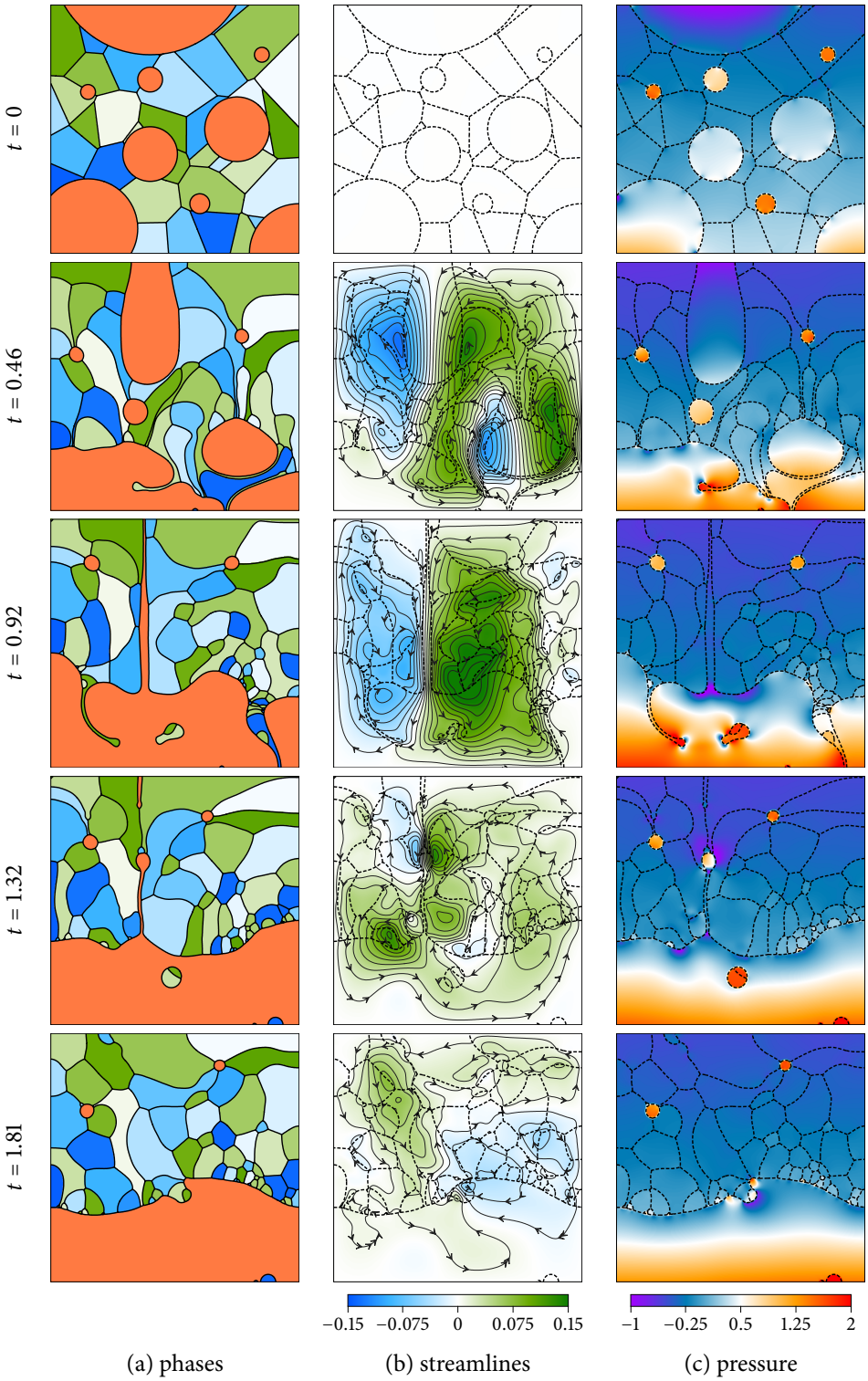

Figure 4.12. Results of a fluid flow simulation with gravity, in which the orange coloured phase is more viscous and more dense than the other phases.

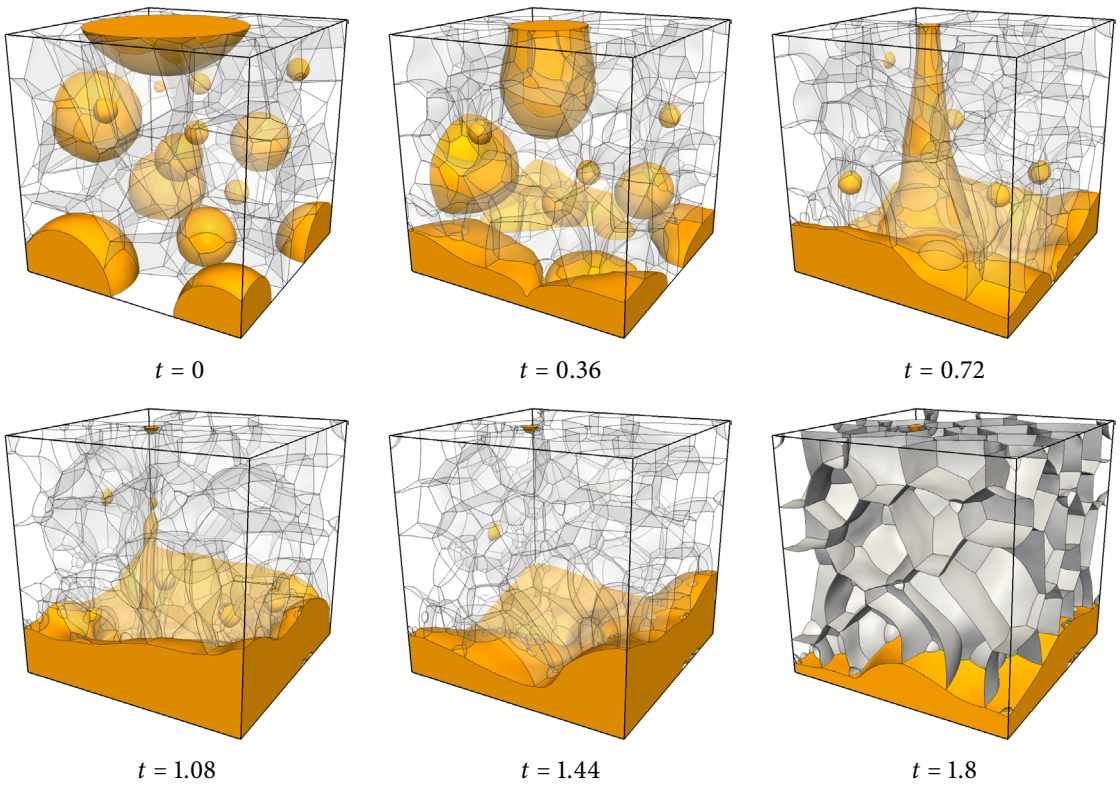

Figure 4.13. Results of a fluid flow simulation in three dimensions with gravity, in which the orange coloured phase is more viscous and more dense than the other phases. The bulk foam is rendered mostly transparent except for the last frame, where it is rendered opaque to make the structure more prominent.

at $t = 1.32$, eventually rise due to buoyancy, and burst at the surface of $\mathcal{H}$, at $t = 1.81$. On the other hand, near the bottom right corner, some less dense phases break off and remain attached to the bottom of the square. Here, the local effects of surface tension implementing a 90° contact angle model dominates the force of buoyancy, and the bubbles remain attached to the bottom.

From the streamlines, we can observe that the flow inside the heavier phase is more viscous, since the streamlines there are more regular. One can also see that the pressure has larger gradients inside the heavier phase, consistent with the liquid having a higher density there. Finally, we note that the angles in the triple points are consistent with those predicted by Young's law (4.10): the heavier phase is nearly locally flat at triple points, while the less dense phases meet the heavier phase at nearly 90° angles, and have 120° angles elsewhere in the foam.

Three-dimensional results

Figure 4.13 illustrates the results for an analogous, three-dimensional simulation, computed on a 128^3 grid with slip boundary conditions, using $\epsilon = 0^+$. The simulation starts with 15 components of $\mathcal{H}$, and there are approximately 100 of the less dense phases. For all but the last snapshot in Figure 4.13, the heavier phase $\mathcal{H}$ is coloured solid orange, while the other phases have been rendered mostly

transparent, together with the triple line junctions as a network of curves. In the last snapshot, at time $t = 1.8$, the bulk foam has been rendered opaque, to make the structure of the foam more obvious.

Various phenomena similar to the 2D case is observed. The large component of $\mathcal{H}$ initially attached to the top falls down under the action of gravity, leaving behind it a tail that eventually detaches. The tail splits into three components, much like a mean curvature flow on a dumbbell splits into two components. The bottom two components continue travelling to the pool of liquid at the bottom, while the top component remains attached to the ceiling of the domain, and stays there due to the local effects of the 90° contact angle boundary condition dominating gravity. Meanwhile, one other component (seen on the back left-facing wall) stays attached to the wall, unable to fully sink, and this is again due to local effects of surface tension dominating effects of buoyancy. Finally, we note that the heavier phase is locally flat at triple lines, while triple lines elsewhere in the foam have 120° angles, consistent with the 3D analogy of the generalised Young's law.

4.2.4 Additional fluid flow applications

In the last set of example applications of the multiphase fluid flow framework, some preliminary results for two fluid flow problems are presented, in cluster breakup in shear flow, and in foam generation.

Cluster breakup

In this example, a cluster of cells is subjected to shear flow in an enclosed square channel. Since surface tension acting at junctions provides a type of "adhesion" between cells, a cluster made out of cells with lower surface tension is more likely to break apart compared to one with high surface tension. This is demonstrated in Figure 4.14: a cluster of cells is suspended in a fluid of the same density in a rectangular channel (square channel in three dimensions), and an external force pointing to the right is applied, similar to the pressure gradient that arises in Poiseuille flow. Over time, this force builds and eventually balances with viscous effects together with no-slip boundary conditions, leading to a shear flow such that the fluid velocity at the centre of the channel is highest. The resulting shearing eventually breaks apart the cluster. In Figure 4.14 a moving reference frame is used that follows the bulk of the cluster. The left and right pair shows two- and three-dimensional results, and in each pair, the left uses a lower value of surface tension compared to the right. We can observe that breakup occurs more quickly in the case with lower surface tension.

Foam generation

In the VIIM, it is straightforward to create phases during a simulation, as discussed in §3.7 of Chapter 3. This is demonstrated here with a foam generation problem, whereby gas bubbles are introduced at the bottom of a container filled with liquid, as shown in Figure 4.15. In particular, small gas bubbles with diameter of about a few grid cells are instantaneously created at a random location at the bottom of the chamber. These bubbles are then artificially grown to slowly increase their volume, as they rise

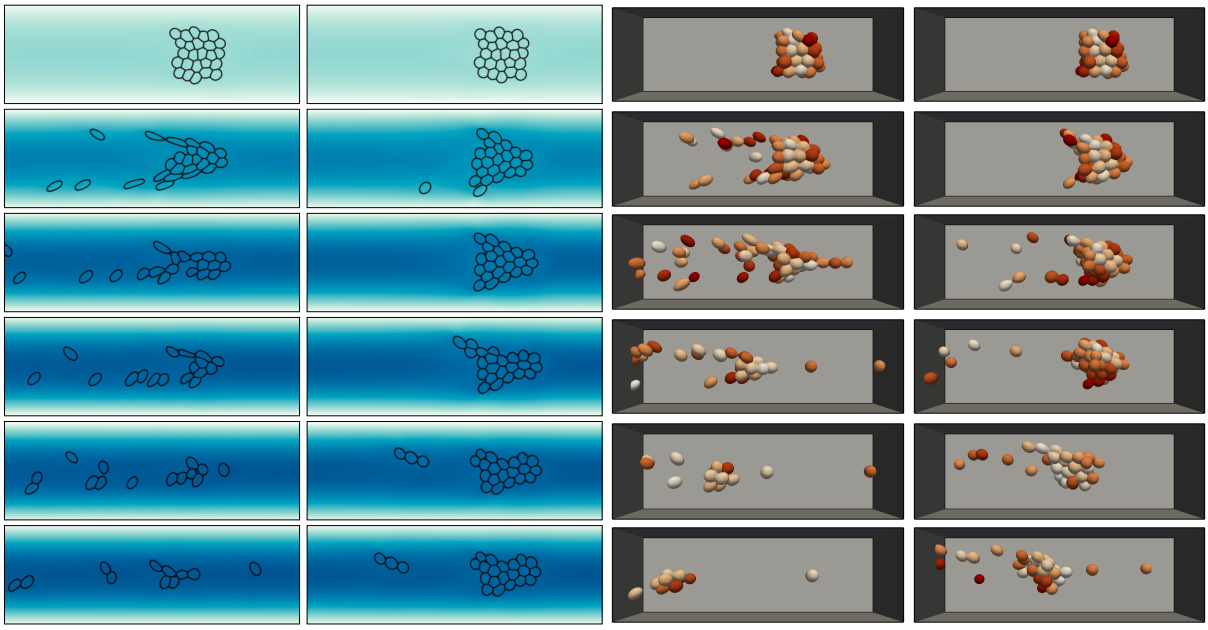

Figure 4.14. Cluster breakup in shear flow; time progresses from top to bottom. Left pair: two-dimensional example where the blue colour indicates the speed of the fluid velocity. Right pair: three-dimensional results. In each pair, the left column has a lower value of surface tension than the right column.

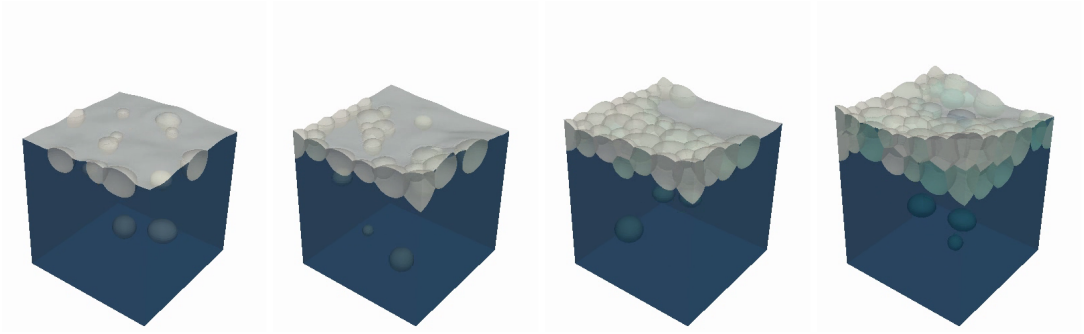

Figure 4.15. Generation of gas bubbles which rise to the surface; time progresses from left-to-right.

to the surface due to buoyancy effects, foaming a foam at the top. While this is not representative of the complex chemistry and physics taking place in, say, boiling or nucleation of gas bubbles in carbonated liquid, this example is included here to demonstrate the ability of the VIIM to handle creation of phases.

4.3 Implementing general domains using an exterior phase

By freezing ϵ-level sets in the VIIM, it is possible to create an "exterior phase" which does not move and can be used to represent domains with a general shape. This is demonstrated here with two applications in an annulus-shaped domain (Figures 4.16 and 4.18) and in a torus (Figure 4.17). To implement this, a phase representing the exterior of the annulus/torus is created and frozen in time. Thus, the ϵ-level set of the exterior phase essentially becomes a boundary condition in the Voronoi interface reconstruction. In Figure 4.16, this idea is used to compute curvature flow with area targeting so that every phase finishes with the same area. Note that 90° boundary conditions naturally arise at the boundary of the annulus. In Figure 4.17 an analogous flow is shown in three dimensions for mean curvature flow with volume targeting so that every phase finishes with the same volume. Figure 4.18 shows a fluid flow problem, whereby the inner ring boundary of the annulus rotates clockwise, and the outer ring rotates counterclockwise. These boundary conditions are supplied to the Navier-Stokes solver by fixing the velocity field at each grid point exterior to the annulus at the end of each projection step in the projection method. In the figure, we can see how the resulting shearing effects causes the multiphase system to mix.

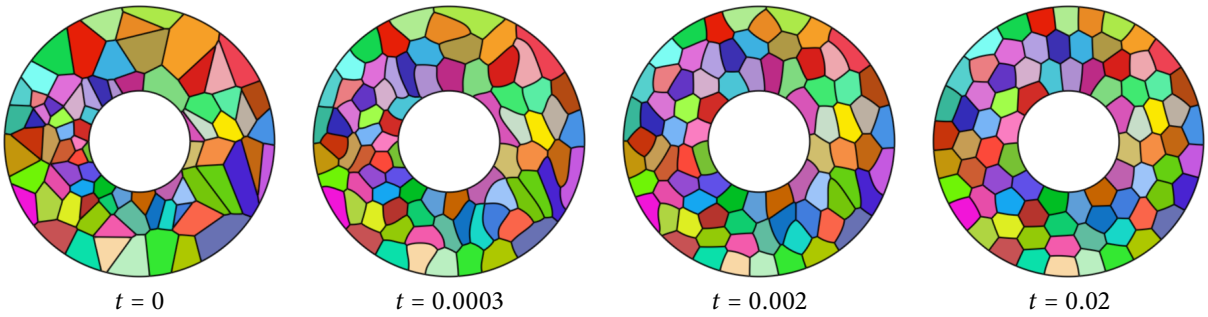

Figure 4.16. Curvature flow with area targeting in an annulus.

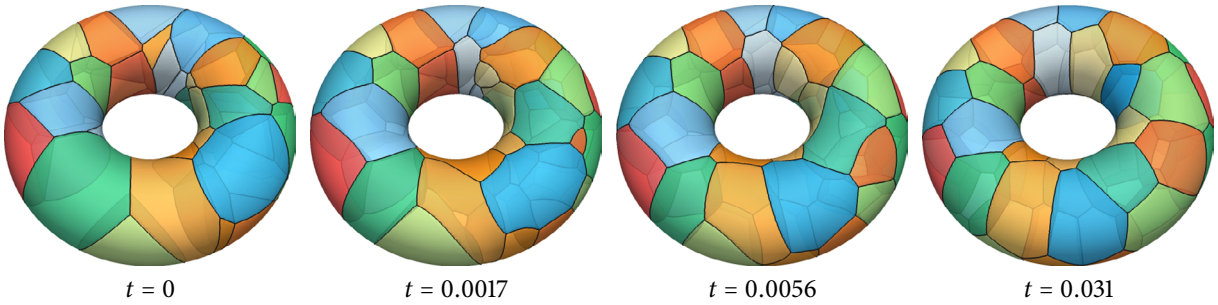

Figure 4.17. Mean curvature flow with volume targeting in a torus.

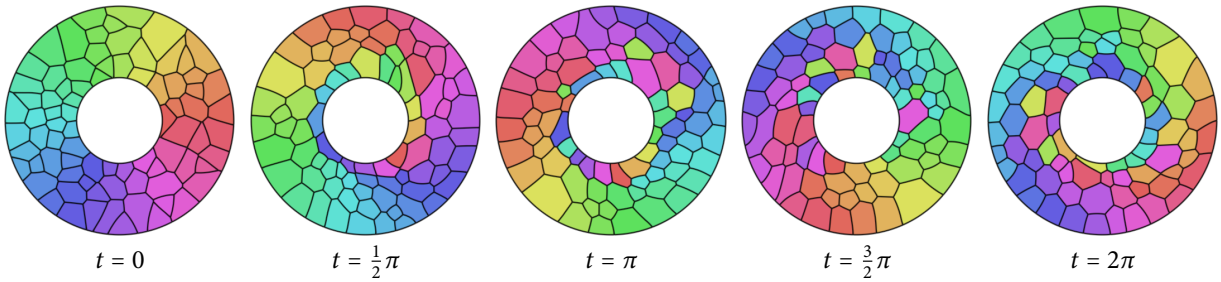

Figure 4.18. Shear flow with inner ring rotating clockwise, outer ring rotating counterclockwise; phases are coloured according to their initial position.

Chapter 5

Mesh Generation for Interconnected Surfaces

In this chapter, a high-quality mesh generation algorithm for a network of interconnected surfaces is presented. The development of the algorithm was motivated by the multiscale model for foam dynamics presented in the following two chapters: in that model, certain PDEs are solved numerically, using the finite element method, on a network of curved thin-film membranes which evolve under gas dynamics driven by surface tension. A mesh generation algorithm was therefore needed to automatically generate high-quality meshes for a set of interconnected surfaces. The algorithm should be able to reliably handle a wide variety of topologically complex situations, and mesh elements should meet consistently at junctions, so that there are no gaps, overlaps or other artifacts. Furthermore, in order to implement coupled boundary conditions at these junctions, it is advantageous for the mesh to have the property that if two mesh elements meet at a junction, then they do so by sharing a common edge, so that there are no hanging vertices. As shown here, all of these goals can be met by capitalising on mathematical aspects of the Voronoi interface.

The mesh generation algorithm has essentially two stages: in the first stage, a topologically consistent mesh is created that has no gaps, overlaps, or other artifacts at junctions. This step involves no heuristic or complex decisions about surface topology; instead properties of the Voronoi interface are used to guarantee consistency. The resulting mesh is suitable for many purposes, such as visualisation, but consists of many low-quality elements, so in the second stage of the algorithm, a short sequence of force-based smoothing, projection, and edge flipping iterations is applied. Here, the ideas of the DistMesh [45] algorithm are extended to the case of multiple surfaces – mesh vertices are moved according to forces, based on the current topology of the mesh, which together with edge flipping, improve mesh quality. By using a new adaptive time stepping strategy, convergence to a high-quality mesh can be obtained within as few as 10-20 iterations, taking less than a second on a typical desktop computer for a mesh with 10,000 elements.

Since the concept of the Voronoi interface makes generating a topologically consistent mesh straightforward, it may be advantageous to convert other representations (such as voxel-based data or point clouds) into this form. For example, CT imaging and MRI produce voxel data wherein each voxel identifies different regions or types of tissue, and the boundary between the different regions defines the surfaces. It is straightforward to convert this into a Voronoi interface by using, for example, smoothed indicator functions for the different regions. Another possibility is using

three-dimensional scattered point clouds, and this is demonstrated in the results.

The outline of the chapter is as follows. First, some previous work on meshing multiple surfaces is reviewed and compared with the approach presented here. In the next section, the idea of the Voronoi interface is recalled and discussed in the context of mesh generation, and then the main meshing algorithm is presented. Lastly, results and mesh quality analyses are shown using a variety of applications arising from multiphase curvature flow, geometrically-defined objects, and surface reconstruction from point clouds.

5.1 Previous work

Much of the previous work on meshing interconnected surfaces has focused on multi-material data sets, wherein each voxel is assigned a different label identifying different regions or materials. A variety of algorithms have been developed, which, somewhat broadly, are based on:

- *Lookup tables:* Here, a marching cubes [56] or marching tetrahedra [57–59]-style algorithm is extended to handle the case of multi-material data sets, leading to lookup tables that take into account material labels. In general, the lookup tables rapidly increase in size with the material count, and produce a mesh that has many bad quality elements (e.g. slivers). In some cases, see e.g. [69–71], they are created through a series of heuristic decisions that decide a plausible topology, making it unclear if the resulting meshes are guaranteed to be free of artifacts. In another approach, in [72] a subdivision algorithm is used together with trilinear interpolation that leads to a mesh guaranteed to be topologically consistent, however the subdivision comes at the price of generating very large lookup tables or a large number of tiny mesh elements. In [73], a lookup table is used in the case of three materials, and in the general case, triangle removal algorithms are used, allowing meshes with boundaries and holes to be created.

- *Delaunay refinement:* These methods are an extension of typical Delaunay refinement-based algorithms to the case of multiple surfaces [74–76]. A mesh is iteratively created by adding vertices and updating mesh topology until a quality criteria is reached that terminates the algorithm. They generally involve many subtle pre-processing steps, including the need to identify and extract the triple junctions as a network of curves. The created meshes are topologically consistent, having no gaps or overlaps, and in addition, mesh elements that meet at junctions do so by sharing a common edge. Parallel implementations of these algorithms can be difficult, however the approach has the advantage of allowing quality criteria to be defined on per-surface basis, allowing different materials to have different mesh resolutions, and volume tetrahedra meshes (consistent with the interface topology) are often produced at no extra cost.

- *Particles:* A different approach involves first placing particles (or spheres) on the set of surfaces, optimising the distribution of the particles, and then creating mesh topology through a constrained Delaunay triangulation/tetrahedrisation. A very early method using this idea was the *bubble mesh* [77]: spheres of an adaptive size are “packed” onto explicitly parameterised

curves, surfaces, and volumes, in that order. In a relaxation step, the spheres are then dynamically moved according to force-based laws to optimise their distribution, before invoking a constrained Delaunay algorithm to obtain the mesh topology. More recently, in [78] an algorithm is proposed in which the particles, once seeded on the implicitly defined surfaces, are dynamically moved to optimise an energy functional that measures surface features, allowing particles to concentrate near regions of high curvature. The resulting meshes are of very high quality, but this comes at the price of a computationally expensive optimisation process; some of the meshes presented in the latter work took several hours to generate.

- *Other approaches:* In [79], a straightforward subdivision strategy is used that subdivides the unit cube into sufficiently many smaller sub-cubes, and assigns different materials to each sub-cube. An interface between different materials is then extracted directly from the subdivided cube, creating a topologically consistent mesh, but which has a staircase shape that then needs to be smoothed, affecting the accuracy of the surface representation. More recently, in [80], an octree-based approach that uses a dual contouring method is presented, also based on a subdivision algorithm that instead uses trilinear interpolation to resolve topological ambiguities. Another octree-based algorithm is developed in [81], allowing both triangular and quadrilateral surfaces meshes (as well as hexahedral volume meshes) to be created of good quality. A different approach is presented in [82], wherein surfaces are meshed away from the junctions, leaving behind void regions which are then meshed using the previously-created surface meshes as constraints.

  Finally, when the surfaces are represented via other means, different approaches are available. For example, in [83], point clouds of surfaces are transformed into implicit functions via partitions of unity, before executing a marching tetrahedra variant to extract a mesh. Lastly, in the volume-of-fluid method for tracking interfaces in multiphase fluid flow, the volume fraction of every fluid in each grid cell is tracked, and in order to evolve the fluids, the interface must be reconstructed from the volume fractions, see e.g. [84].

In comparison, for the mesh generation algorithm presented here, an initial mesh is constructed using a simple marching tetrahedra-style algorithm that does not require special material-dependent lookup tables. Instead, the Voronoi interface, together with properties of piecewise linear interpolation, guarantee topological consistency of the mesh (provided the surfaces are sufficiently resolved). In this fashion, heuristic or complex decisions about surface topology are avoided. In the second stage, vertices of the mesh are dynamically moved to improve mesh quality, in-part based on the ideas presented in the *DistMesh* algorithm [45]. Here, a specially designed "clamping" function is used to automatically prevent triangle inversion, while allowing low quality elements to quickly change shape into good quality elements.

5.2 The Voronoi interface

Recall the definition of the Voronoi interface given in §3.2: given a set of non-overlapping regions Ω_i, the Voronoi interface is the set of points that are equidistant to two regions and no closer to

any other region. For the purposes of mesh generation, it is advantageous to use multiple level set functions ϕ_i in order to define the Voronoi interface, as was done in §3.4.2. Specifically, suppose that $\phi_i : \mathbb{R}^n \to \mathbb{R}$ is the signed distance function for region Ω_i, such that ϕ_i is positive inside Ω_i. Then the Voronoi interface Γ_V inside a domain Ω is given by

$$\Gamma_V = \left\{ x \in \Omega : \exists\ i \neq j \text{ such that } \phi_i(x) = \phi_j(x) \geq \max_{k \neq i,j} \phi_k(x) \right\}. \tag{5.1}$$

In particular, Γ_V can be separated into a union of individual surfaces separating pairs of different regions, such that $\Gamma_V = \bigcup_{i \neq j} \Gamma_{ij}$, where

$$\Gamma_{ij} = \left\{ x \in \Omega : \phi_i(x) = \phi_j(x) \geq \max_{k \neq i,j} \phi_k(x) \right\}. \tag{5.2}$$

Although distances were used as motivation, note however that (5.1) and (5.2) can be used to define an interface, even when the ϕ_i functions are not necessarily signed distance functions. All that is required is that the functions ϕ_i are continuous, and that the individual sets Γ_{ij} are codimension-one surfaces. Therefore, it is useful to make a generalisation: for any set of functions $\phi_i : \mathbb{R}^n \to \mathbb{R}$, the *Voronoi interface of the functions* ϕ_i is defined to be Γ_V given by (5.1).

Defining the interface in this manner, i.e. implicitly rather than explicitly, provides many virtues. In relation to the meshing problem, as shown below, this particular implicit representation makes it straightforward to extract a topologically consistent mesh with no artifacts at junctions. Assuming continuity of the functions ϕ_i, an equivalent characterisation of the interface is that a point x is inside phase/material i if and only if $\phi_i(x) > \max_{j \neq i} \phi_j(x)$. This characterisation was used in some previous works on meshing multiple surfaces (wherein the functions were smoothed characteristic/indicator functions); here, (5.1) is used directly.

In practice, there are many possible methods for defining the functions ϕ_i. The meshing algorithm does not depend heavily on the particular method of determining ϕ_i; in the following, all that is assumed is that the ϕ_i functions are defined on a background grid (such as a regular Cartesian grid). This naturally occurs in the context of the VIIM, where reinitialisation methods are used to calculate signed distance functions to ϵ-level sets. In other cases, the functions could be derived from a single label/indicator function $\chi : \mathbb{R}^3 \to \mathbb{N}$ that divides the domain into different regions. For example, one could define ϕ_i as a (possibly smoothed) per-phase indicator function such that $\phi_i(x) = 1$ inside region i and $\phi_i(x) = 0$ outside. However, it is important to note that in the meshing algorithm it is not necessary to define every ϕ_i function everywhere in the entire domain: it is only necessary to know the values of ϕ_i in a small narrow band surrounding the boundary of phase i. This can dramatically improve efficiency, and is a common technique used, for example, in narrow band level set methods [50].

5.3 Mesh generation

Given N functions ϕ_i defined on some three-dimensional grid, the goal is to extract an explicit representation of the Voronoi interface, defined by (5.1), as a high-quality triangulated mesh. This is

accomplished by first creating an initial mesh, which although will generally be of low quality, it will nevertheless be topologically consistent in the sense that triangles meet at junctions without overlap or gaps. In the second stage of the algorithm, the mesh is iteratively improved using a sequence of force-based smoothing, projection, and edge flipping steps. The method is summarised in Algorithm 2, and, in the next set of sections, the implementation of the individual steps is described.

Algorithm 2 General algorithm for mesh generation

Create initial mesh.
repeat
    Apply forces and project vertices.
    Edge flip triangles.
until desired mesh quality is achieved.

5.3.1 Creating the initial mesh

To create an initial mesh of the network of interfaces, the Voronoi interface of the multiphase system is extracted using a marching tetrahedra-style algorithm that involves creation and "chopping" of mesh elements. The approach is based on the procedure described in §3.8 of Chapter 3. For a particular surface Γ_{ij}, defined by (5.2), the mathematical procedure consists of three steps:

(i) First, establish a continuous piecewise linear interpolation of every ϕ_i function, to determine ϕ_i at arbitrary points. For example, if the background grid is a regular Cartesian grid, then each cell can be divided into six tetrahedra, as shown in Figure 5.1 (left), and in each tetrahedron one can use the obvious linear interpolant of ϕ_i. This is precisely the interpolant that is implicitly used by the marching tetrahedra algorithm. Alternatively, if the background grid is already a tetrahedral mesh, one can directly use the canonical linear interpolant of ϕ_i.

(ii) Next, extract the zero level set of $\phi_i - \phi_j$ as a collection of planar polygon surface elements $\{E_\ell\}_{i=1}^n$.[1] Since the last condition in (5.2) is not necessarily satisfied on every surface element, it follows that $\Gamma_{ij} \subseteq \bigcup_\ell E_\ell$.

(iii) For each element E_ℓ, keep the set of points $x \in E_\ell$ that satisfy $\phi_i(x) = \phi_j(x) \geq \phi_k(x)$ for all $k \neq i, j$. This is achieved by a series of "chop" operations that takes E_ℓ and chops it using the zero level set of $\phi_i - \phi_k$ as the position of the cut, and keeping the piece on which $\phi_i \geq \phi_k$. The piece on which $\phi_i < \phi_k$ is thrown away. Such chopping is made possible by the fact that every level set function ϕ_i is piecewise linear, hence particular level sets as well as intersections of level sets are always linear. In particular, it can be shown that throughout the chopping process, the element is always a convex planar polygon. After chopping is complete, the polygon E_ℓ is then converted into a collection of triangles.

[1] Note that $\phi_i - \phi_j$ is a continuous piecewise linear function defined on Ω, and so its zero level set must necessarily be piecewise planar. In particular, because $\phi_i - \phi_j$ is linear on each tetrahedron, it follows that its zero level set in each tetrahedron, if it exists, is either a triangle or a planar quadrilateral.

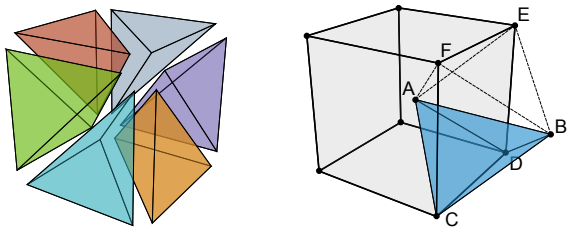

Figure 5.1. (Left) Dividing a cube into six tetrahedra. (Right) A body centred cubic lattice tessellates space into identical tetrahedra; each tetrahedra has two vertices at the centroid of neighbouring cells, and the other two vertices on an edge of a cell. In the figure, the blue tetrahedron ABCD is one of four sharing a face; the other three are ABCF, ABDE, ABEF.

The result of the above procedure is a collection of triangles whose union is Γ_{ij}. Note that the algorithm is *exact*, in the sense that the Voronoi interface of the interpolated multiphase system is extracted exactly. As a result, different interfaces Γ_{ij} and Γ_{kl} which meet at higher order junctions do so without any overlap or gaps. In particular, it follows that triangles which meet at junctions do so by sharing edges and vertices along those junctions.

In the work presented here, the functions ϕ_i are defined on a regular Cartesian grid. If each grid cell is divided into six tetrahedra, see Figure 5.1 (left), as per a common variant of the marching tetrahedra algorithm, it is almost always the case that the resulting triangulation is extremely poor: many triangles are almost degenerate (small in diameter or slivers), and there is often many more triangles than necessary to capture the features of the interface, as demonstrated in Figure 5.2 (left). To improve this, two complementary ideas are used:

- Instead of dividing each grid cell into six tetrahedra, a "body centred cubic lattice" [59] is used, as shown in Figure 5.1 (right). The lattice tessellates space into equal shaped tetrahedra which are more symmetric and more evenly distributed. As noted in [59], the resulting triangulation of the interface has far fewer triangles, while maintaining the same feature resolution.

- In addition, a vertex snapping procedure is used to eliminate sliver triangles. For a single scalar function ϕ defined on the background grid's vertices, assuming the zero level set is sought, vertex snapping slightly perturbs ϕ by setting the value of ϕ to be zero at any vertices where it is approximately zero:

$$\tilde{\phi}(x) = \begin{cases} 0 & \text{if } |\phi(x)| < \epsilon, \\ \phi(x) & \text{otherwise.} \end{cases}$$

  This effectively snaps mesh vertices that are approximately near background grid points to be precisely on those grid points, without introducing any holes or artifacts in the mesh. In practice, even for considerable reduction of sliver elements, the threshold ϵ can be very small. Here, the mesh created by this algorithm is used as the input to an iterative smoothing and projection procedure (see next section), and therefore does not need to be entirely accurate. As a result, large tolerances can be used, resulting in a significant reduction of unnecessary triangles in the mesh, while maintaining the same feature resolution. When the scalar func-

tions are approximate distance functions, it was found that setting $\epsilon \approx 0.2h$, where h is the Cartesian grid cell size gave good results.

For the multiphase system, vertices can be snapped by perturbing the level set function values in a pairwise fashion:

for each background mesh point x **do**
  Define $\tilde{\phi}_i(x) := \phi_i(x)$ for all i.
  for $i = 1, \ldots, N$ **do**
    for $j = i+1, \ldots, N$ **do**
      if $|\tilde{\phi}_i(x) - \tilde{\phi}_j(x)| < \epsilon$ **then**
        $\tilde{\phi}_j(x) \leftarrow \tilde{\phi}_i(x)$

Snapping in a pairwise fashion like this ensures that previous alterations do not get overridden by later alterations, which in turn means the topology of the multiphase interface is not affected (provided it was already sufficiently resolved).

Combining these ideas, we are lead to Algorithm 3 for creating an approximation of the Voronoi interface. On line 3, of course it is only necessary to loop over the pairs of phases defined in the particular tetrahedron being considered. On lines 4 and 5, a lookup table, similar to those used in a standard marching tetrahedra algorithm, can be used to determine how to extract the polygon (as either a triangle or quadrilateral). On line 7, a simple method to evaluate ψ at arbitrary locations is to use pre-computed Lagrange basis functions, while on line 8, it is a simple exercise to design an algorithm to cut a convex planar polygon by the zero level set of a linear function defined on its vertices. Polygons are then dissected into triangles and added to the overall collection of triangles. These steps are straightforward if one is not concerned about duplicating vertices, i.e. triangles are represented as 3-tuples of vectors in $\mathbb{R}^3$. Clearly, this is not optimal, since representing a triangle with a 3-tuple of vertex indices is more efficient. So, on line 11, the collection of vertices are uniquified, by identifying vertices as equal if they are within a small amount of machine precision round off error.[2] Alternatively, one may uniquify vertices simultaneously with building the collection of triangles.

The result of the mesh creation algorithm on an example multiphase system is shown in Figure 5.2. Here, a five-phase system is defined by setting $\{\phi_i\}_{i=1}^4$ to be signed distance functions for four differently positioned spheres, i.e. $\phi_i(x) = r_i - \|x - x_i\|$, and then defining

$$\phi_5 = \min(-\phi_1, -\phi_2, -\phi_3, -\phi_4)$$

to be the exterior phase. This system has a total of five phases, ten individual surfaces Γ_{ij} separating pairs of phases, ten distinct triple line junctions, and five quadruple point junctions. (Only some of these surfaces/junctions are visible in the figures; see also Figure 5.5 for a partial cutaway view.)

[2] With mild assumptions on the functions ϕ_i, it is possible to show that the vertex snapping procedure guarantees that if two vertices on a surface are within a distance ϵ from each other, such that $\epsilon \ll h$ (where h is the tetrahedron length), then they are in fact the *same* vertex. In this work, a tolerance of $\epsilon = 10^{-14}$ was used (corresponding to double precision and unit-length domains). Extensive tests analysing separation distance of vertices found that this tolerance correctly uniquified vertices in all cases.

Algorithm 3 Creating the initial Voronoi interface mesh

Snap vertex values.
for each tetrahedron $\mathcal{T}$ **do**
    for each subset $\{i, j\} \subseteq \{1, \ldots, N\}$ **do**
        Extract the zero level set of $\phi_i - \phi_j$ inside $\mathcal{T}$ using linear interpolation.
        Define P to be the resulting convex planar polygon;
          (if it does not exist, **continue** to next $\{i, j\}$ subset).
        for each $k \neq i, j$ **do**
            Evaluate $\psi := \phi_i - \phi_k$ at each vertex of P.
            Cut the polygon at the zero level set of ψ and keep the part on which $\psi \geq 0$.
            if P is now empty, **continue** to next $\{i, j\}$ subset.
        Divide P into triangles and add to the collection Γ_{ij}.
If necessary, uniquify vertices.

Figure 5.2 (left) shows the mesh obtained by using the standard decomposition of each Cartesian grid cell into six tetrahedra, without vertex snapping. In comparison, Figure 5.2 (right) shows the result using the body centred cubic lattice, with vertex snapping. We can see that there is a significant reduction in sliver triangles, as well as the overall triangle count.

5.3.2 Smoothing and projection

Once an initial mesh of the Voronoi interface of the multiphase system is created, a high-quality mesh can be obtained by using force-based smoothing, projection, and edge flipping. Since this

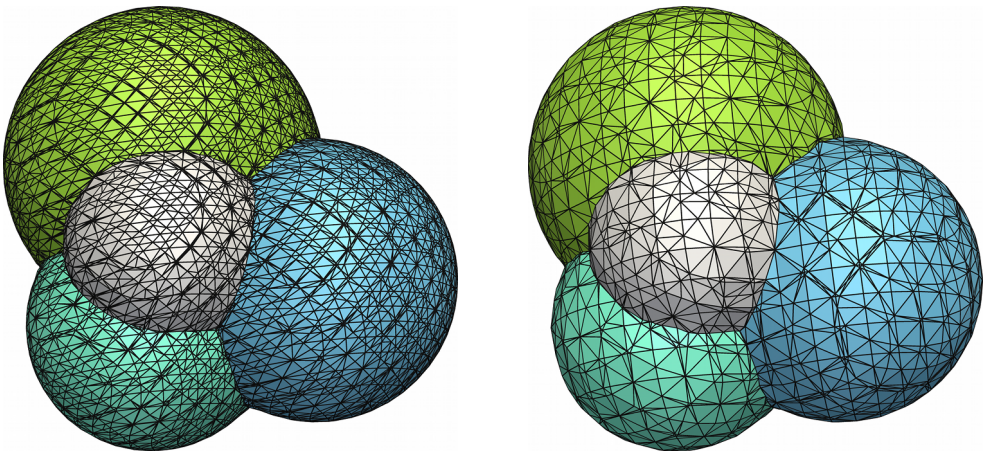

Figure 5.2. A multiphase mesh of a five-phase system of four spheres, using the polygon chopping method (Algorithm 3). Four pairwise surfaces and three quadruple point junctions are visible. (Left) Using the standard decomposition of dividing a grid cell into six tetrahedra, without vertex snapping. (Right) Using the body centred cubic lattice, with vertex snapping.

process predominantly involves calculations involving vertices and their edges, it is advantageous to use a vertex-and-edges data structure. Hence, let $x_i \in \mathbb{R}^3$, $i = 1, \ldots, n$, denote the mesh vertices and $\mathcal{E}_i \subset \mathbb{N}$ denote the set of neighbours of vertex i, so that (x_i, x_j) is an edge for each $j \in \mathcal{E}_i$. From the creation of the initial mesh, we also known which surfaces each vertex is situated on. For each vertex x_i, let $\chi_i \subset \mathbb{N}$ denote the set of phases for which x_i is on the boundary. Thus, if χ_i has two, three, or four elements, then x_i is on a surface, a triple junction, or a quadruple point, respectively. At times, it is also necessary to determine the set of triangles connected to a particular vertex; denote this set as $\mathcal{T}_i$. Determining $\mathcal{T}_i$ can be done on-the-fly by manipulating the edge data structure.

To iteratively improve the triangulated mesh, the ideas underlying the DistMesh algorithm [45] are extended and adapted to the case of interconnected surfaces. In the DistMesh approach, force-based smoothing, together with mesh topology updates and projection of vertices onto constraining surfaces, enables fast convergence to a high quality mesh. In particular, the vertices of the mesh are moved according to "forces" exerted on them by the edges in the mesh; in the original DistMesh algorithm, the edges are analogous to springs that resist compression when shorter than a certain rest-length, but do not otherwise resist expansion. It was observed that this repulsive force combines exceptionally well with edge flipping (or regular Delaunay triangulation) to quickly generate meshes with very good connectivity properties. Conventional Laplacian smoothing [85] may also be viewed as force-based, in which case the mesh edges are attractive rather than repulsive. In the general case, each edge (i, j) exerts a force $f_{ij} = -f_{ji}$ on vertex i. The goal is to find an equilibrium such that at each vertex, the net force is zero. Solving such a nonlinear system is a relatively difficult task, so instead the vertices are moved iteratively, using a simple forward Euler analogy:

$$x_i^{n+1} = x_i^n + \Delta t \sum_{j \in \mathcal{E}_i} f_{ij}.$$

Here, Δt is an artificial time step that controls the progress of the mesh towards the final desired state. Note however that if Δt is too large, mesh elements can invert, and this can cause the iterative process to become wildly unstable. On the other hand, if Δt is too small, then efficiency can be sacrificed since too little progress may be made. A solution to this problem is presented later, using a method that essentially locally computes the time step according to a basic stability condition that prevents inversion of triangles, and leads to rapid convergence.

Force functions

In [45], the force function represented edges as springs that resist compression when shorter than a certain rest-length, and do not resist expansion when longer. Despite being a simple approach, it was shown that such a force quickly leads to a mesh with uniformly high quality elements. For an edge (x, y), the force exerted on x is defined by

$$f_D(x, y, \ell_0) = \frac{x - y}{|x - y|} \max(\ell_0 - |x - y|, 0),$$

where ℓ_0 is a rest-length that is related to the desired average edge length of the final mesh. In fact, it is possible to allow ℓ_0 to vary spatially, allowing the mesh to automatically refine where necessary.

However, in this work, a mesh which has uniform triangle sizes throughout is sought, and so ℓ_0 will be spatially uniform. As noted in [45], it is important for vertices of the mesh to spread out across the whole geometry, which means that the rest-length ℓ_0 should be slightly larger than the actual desired edge length in the mesh. This is achieved with a simple scaling: at the beginning of each iteration, the average edge length (measured in a L^2 norm) is computed and scaled by a factor[3] of 1.2 to define

$$\ell_0 := 1.2 \left(\frac{\sum_{i=1}^{n} \sum_{j \in \mathcal{E}_i} \|x_i - x_j\|^2}{\sum_{i=1}^{n} |\mathcal{E}_i|} \right)^{\frac{1}{2}}. \tag{5.3}$$

This repulsive force ties in well with edge flipping and quickly leads to high quality elements. However, when edge flipping is not possible, as is the case on triple junctions, the repulsive force can lead to situations in which vertices attempt to exit their constraining surfaces, causing mesh elements to invert. Instead, it was observed that Laplacian smoothing works very well on, and near, junctions. In this case, an attractive force f_L is used, where

$$f_L(x, y) = \tfrac{1}{2}(y - x).$$

Correctly normalised by the number of edges, the attractive force is equivalent to a half-step of Laplacian smoothing, which acts to move vertices to the average location of its neighbours. This force is used for all junction vertices, as well as surface vertices which have at least one neighbour on a junction.

Projection

A side-effect of the force-based smoothing is that over time, mesh vertices can deviate from the surfaces on which they are meant to be constrained, i.e. the mesh vertices can easily stray from the Voronoi interface. To fix this, one can *project* the vertices back onto the surfaces on which they belong, by moving them in a direction orthogonal to the constraining surface. Such a scheme was used in [45, 86] for vertices constrained to live on codimension-one surfaces, and a different scheme was used in [78] for triple junctions. Note that this procedure also fixes the small aberrations caused by the vertex snapping used in creating the initial mesh.[4] For a single function, a vertex x close to the zero level set of ϕ is (approximately) projected onto the zero level set with the update

$$x \leftarrow x - \frac{\phi(x) \nabla \phi(x)}{|\nabla \phi(x)|^2}.$$

The update can be viewed as moving x to its closest point on the zero level set of the linear approximation of ϕ at x, given by $\phi(x + \delta) \approx \phi(x) + \delta \cdot \nabla \phi(x)$. In the case of a vertex belonging to a triple junction, or a higher order constraint set, one would like to project x to the corresponding

[3] The scaling factor of 1.2 was determined empirically, and is the same as that used in [45]. It forces mesh vertices to spread apart across the whole surface and leads to a smoothing behaviour that performs consistently well.

[4] In the projection step, the original, unperturbed functions ϕ_i are used, i.e. they have not been altered by the vertex snapping procedure used in creating the initial mesh.

multi-junction interface. To do this, the method for a single level set function can be generalised to multiple level set functions by considering pairwise combinations: for a vertex x belonging to the boundary of phases $\chi \subseteq \{1 \ldots, N\}$, define

$$p(x, \chi) = -\frac{1}{|\chi| - 1} \sum_{i \in \chi} \sum_{\substack{j \in \chi \\ j > i}} \frac{\big(\phi_i(x) - \phi_j(x)\big)\big(\nabla\phi_i(x) - \nabla\phi_j(x)\big)}{|\nabla\phi_i(x) - \nabla\phi_j(x)|^2}. \tag{5.4}$$

In essence, the function accumulates the shifts arising from projection onto individual surfaces Γ_{ij}, such that one iteration is given by $x \leftarrow x + p(x, \chi)$. Note that (5.4) reduces to the case of a single surface when $|\chi| = 2$, and also that $p(x, \chi) = 0$ whenever x is already on the multi-junction interface (i.e. $\phi_i(x) = \phi_j(x)$ for all $i, j \in \chi$). The normalisation factor in (5.4) is designed to give efficient convergence without causing oscillations (i.e. by ensuring the steps are not too large). Finally, it should be noted that if the denominators in (5.4) are close to zero and cause issues, then it is likely that the functions ϕ_i are poorly defined; in this case, one could reinitialise the functions as signed distance functions using the Voronoi interface.

Note that the gradient of the functions ϕ_i (as well as the function values themselves) must be somehow calculated or approximated. If they are given by closed-form mathematical expressions, then one could calculate the gradients using these expressions. Alternatively, if the functions are defined on a background Cartesian grid, then one could use, for example, standard second order finite differences to calculate their gradients at grid points, followed by trilinear interpolation to evaluate the gradients and function values anywhere within the domain. This latter method was adopted in much of the work presented here.

Adaptive time stepping

The net result of the force-based smoothing and projection leads to displacements of mesh vertices in the form

$$x_i \leftarrow x_i + \delta_i.$$

Here, δ_i is a displacement vector that is the sum of the forces, together with the projection. If this displacement is too large, mesh elements can invert, causing the smoothing process to become unstable. To solve this, a simple clamping algorithm can be used, wherein δ_i is made sufficiently small in such a way that no triangle inverts. The algorithm is designed to allow vertex x_i to move "as far as it can" in the direction δ_i, so that a small edge or triangle can quickly expand if the forces want it to. The clamping is performed for each vertex independently of all other vertices, and effectively establishes an easy-to-implement time stepping that is locally adaptive. Specifically, for a particular vertex i, each of the triangles connected to i are used to clamp the necessary amount, to form the replacement

$$\delta_i \leftarrow \delta_i \times \min\Big(1, \min_{t \in \mathcal{T}_i} \tfrac{1}{2}\mathsf{clamp}(\delta_i, x_i, t)\Big). \tag{5.5}$$

Here, clamp is a function that returns the maximum amount by which x_i can be moved in the direction δ_i without inverting the triangle t, independently of how the other vertices of the triangle

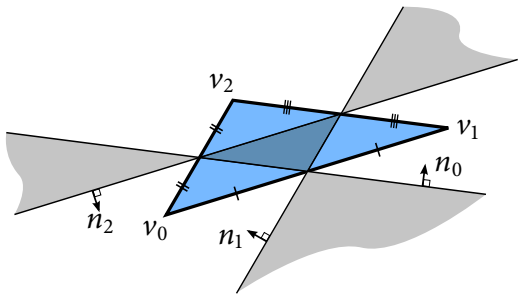

Figure 5.3. Lines passing through the midpoints of a triangle (with vertices v_0, v_1, v_2) divide the plane into the shaded region (having four separate components) and non-shaded region (having three separate components), such that the line opposite vertex v_i has normal n_i. In the clamping procedure, each of the vertices v_i is allowed to move anywhere inside its own particular component of the non-shaded region.

move. The function is implemented as follows. Consider Figure 5.3, which shows a triangle having vertices v_0, v_1, v_2. The indicated lines are constructed by connecting midpoints of the three edges. The lines divide the plane into three disjoint regions that are separated by the shaded region indicated in Figure 5.3. If it can be guaranteed that the vertices v_0, v_1, v_2 move only in their respective regions, without crossing the shaded region, then the triangle will not invert. In full three-dimensional space, these constraint regions are convex, bounded by the planes with the indicated normal vectors, which in turn are parallel to the plane of the triangle.

Consider then a particular vertex, v_0, say, with a corresponding desired displacement vector δ. One computes the minimum $s > 0$ (if it exists) such that $v_0 + s\delta$ is on the boundary of its constraint region. For a particular plane with normal n containing the point a, it follows that $(x + s\delta - a) \cdot n = 0$, giving $s = (a - x) \cdot n / \delta \cdot n$. By doing this for each appropriate plane, this leads to Algorithm 4 to compute the clamp function for a particular vertex v_0 of a specified triangle, with the specified displacement δ.

Algorithm 4 Calculating $\mathrm{clamp}(\delta, v_0, t)$ where t is the triangle with vertices v_0, v_1, v_2

Compute the three normal directions

$$n_0 = v_2 - v_0 - \frac{(v_2 - v_0)\cdot(v_2 - v_1)}{\|v_2 - v_1\|^2}(v_2 - v_1),$$
$$n_1 = v_0 - v_1 - \frac{(v_0 - v_1)\cdot(v_0 - v_2)}{\|v_0 - v_2\|^2}(v_0 - v_2),$$
$$n_2 = v_1 - v_2 - \frac{(v_1 - v_2)\cdot(v_1 - v_0)}{\|v_1 - v_0\|^2}(v_1 - v_0).$$

return s calculated by

$$s = \min\left(\frac{n_0 \cdot (\frac{1}{2}(v_0 + v_1) - v_0)}{\max(n_0 \cdot \delta, 0^+)}, \frac{n_1 \cdot (\frac{1}{2}(v_0 + v_1) - v_0)}{\max(n_1 \cdot \delta, 0^+)}, \frac{n_2 \cdot (\frac{1}{2}(v_2 + v_0) - v_0)}{\max(n_2 \cdot \delta, 0^+)}\right),$$

where it is understood that if one of the max statements evaluates to 0^+, then the corresponding fraction is $+\infty$.

Note that in (5.5), the clamping is reduced by a factor of two, thereby preventing triangle inversion entirely. With this approach, triangles may become nearly degenerate only after an accumulation of displacements over several iterations. In this scenario, the clamping factor and local "time step" would reduce to zero, and convergence would halt. However, this event never occurs in practice, precisely because edge flipping changes the topology of the mesh so that any nearly degenerate triangles are removed.

Combining smoothing and projection

Putting all of the above together, Algorithm 5 performs one iteration of force-based smoothing and projection, with the adaptive time stepping. On line 3, in the second item of the case statement, only the neighbours which belong to the same type of surface (via the conditional $\chi_i \subseteq \chi_j$) are considered. This ensures that the force law involves only edges belonging to the same type of surface (or triple junction). On line 5, the force computed from the edges is enforced to be tangential to the surface, by removing the component of the vector orthogonal to the surface. (Here, n_{χ_i} is a normal vector to the surface $\chi_i = \{j, k\}$, i.e. is proportional to $\nabla\phi_j - \nabla\phi_k$, and can be calculated with finite differences.) Without this, it was observed that the non-tangential components of the force can start to compete with projection, causing oscillation. Hence, the force-based smoothing is restricted to tangential forces.

Algorithm 5 A single smooth and projection step

1: Calculate the rest length ℓ_0 using (5.3).
2: **for** $i = 1, \ldots, n$ **do**
3: Calculate the force on vertex i due to its neighbours $\mathcal{E}_i$:

$$\delta_i = \begin{cases} 0 & \text{if vertex } i \text{ is a high-order junction } (|\chi_i| > 3), \\ \Big(\sum\limits_{\substack{j \in \mathcal{E}_j \\ \chi_i \subseteq \chi_j}} 1 \Big)^{-1} \sum\limits_{\substack{j \in \mathcal{E}_j \\ \chi_i \subseteq \chi_j}} f_L(x_i, x_j) & \text{else if vertex } i \text{ is on a triple junction } (|\chi_i| = 3), \\ & \text{or has a neighbour on a triple junction,} \\ \dfrac{1}{|\mathcal{E}_j|} \sum\limits_{j \in \mathcal{E}_j} f_D(x_i, x_j) & \text{otherwise.} \end{cases}$$

4: **if** vertex i is on a surface ($|\chi_i| = 2$) **then**
5: Remove the component of δ_i orthogonal to the surface: $\delta_i \leftarrow \delta_i - (\delta_i \cdot n_{\chi_i}) n_{\chi_i}$.
6: Add to δ_i the projection displacement: $\delta_i \leftarrow \delta_i + p(x_i, \chi_i)$ using (5.4).
7: Clamp δ_i to prevent inversion using (5.5) and Algorithm 4.
8: **for** $i = 1, \ldots, n$ **do**
9: $x_i \leftarrow x_i + \delta_i$

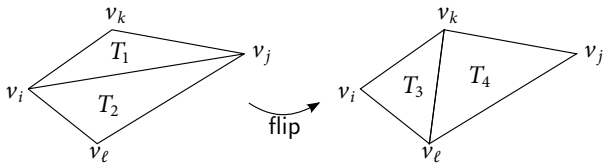

Figure 5.4. Edge flipping exchanges an edge shared by two triangles T_1 and T_2 (left) with the edge formed by opposing vertices, giving two new triangles T_3 and T_4 (right).

5.3.3 Edge flipping

In the edge flipping step, every pair of triangles sharing a common edge is considered. If flipping the shared edge (see Figure 5.4) improves the quality (defined below) of the pair of triangles, then this is done and the connectivity of the mesh is updated, before visiting other edges. In the plane, and with a quality measure that is based on the standard Delaunay in-circle condition, this iterative procedure is a well-known method that converges to a Delaunay triangulation [87]. A simple measurement of triangle quality that is suitable for triangles in $\mathbb{R}^3$ is to define

$$\begin{aligned} q = q(v_0, v_1, v_2) &= 4\sqrt{3}\frac{\text{triangle area}}{\text{sum of squared edge lengths}} \\ &= 2\sqrt{3}\frac{\|(v_1 - v_0) \times (v_2 - v_0)\|}{\|v_1 - v_0\|^2 + \|v_2 - v_1\|^2 + \|v_0 - v_2\|^2}, \end{aligned}$$

where v_i are the vertices of the triangle. Here, q is normalised so that $0 \leq q \leq 1$: an equilateral triangle has $q = 1$ and a degenerate triangle has $q = 0$. (There are also many other types of measures of element quality, see the review [88].) Even though the triangles generally lie on a curved surface, the edge flipping algorithm is essentially equivalent to the logic used in the plane; the algorithm is summarised in Algorithm 6. Note that on line 2 of Algorithm 6, the potential flip is required to only involve triangles belonging to the same surface. This ensures that edges on mesh junctions do not change connectivity. In addition, line 7 checks to see if the triangle pair is already of good quality; if they are, then any possible edge flipping is essentially inconsequential, since any increases in quality would be minor. This simple check leads to markedly better efficiency in the algorithm.

5.3.4 Parallelisation

Combining the above steps, i.e. Algorithm 2, yields the basic algorithm to generate a triangular mesh of the Voronoi interface. In some applications, the Voronoi interface is determined by the evolution of complex physics on high resolution three-dimensional grids. For example, in the multiscale model of foam dynamics presented in the following chapters, computations were often performed on hundreds of processors. In particular, the computational domain was divided into subdomains, which were then assigned to individual processors in an MPI implementation. In such cases, the functions which define the Voronoi interface are split across several subdomains, and it follows that the mesh generation algorithm must also be parallelised to maintain some level of computational efficiency. This can be done with a few modifications to the individual steps of the algorithm, mainly

Algorithm 6 One iteration of edge flipping

for each edge (i, j) in the mesh **do**
  if the edge is shared by two triangles that belong to the same surface **then**
    Let v_i, v_j, v_k and v_i, v_ℓ, v_j be the vertices of the two triangles (see Figure 5.4).
  else
    continue to next edge.
  Compute $q_1 = q(v_i, v_j, v_k)$ and $q_2 = q(v_i, v_\ell, v_j)$.
  if $\min(q_1, q_2) \geq 0.9$ **then**
    continue to next edge *(since the triangles are already of good quality)*.
  Compute $q_3 = q(v_i, v_\ell, v_k)$ and $q_4 = q(v_\ell, v_j, v_k)$.
  if $\min(q_3, q_4) \leq \min(q_1, q_2)$ **then**
    continue to next edge *(since edge flipping will not improve quality)*.
  Let n_3 and n_4 be proportional to the normal of the two new triangles:
    $n_3 = (v_\ell - v_i) \times (v_k - v_i)$,
    $n_4 = (v_k - v_j) \times (v_\ell - v_j)$.
  if $n_3 \cdot n_4 \leq 0$ **then**
    continue to next edge *(since edge flipping will invert the triangles)*.
  Proceed with edge flip: remove edge (i, j) and add edge (k, ℓ) by altering $\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k$, and $\mathcal{E}_\ell$.

in relation to synchronisation of mesh connectivity information across processor boundaries, as follows.

In the first step, an initial mesh approximating the Voronoi interface is found. This step is essentially unchanged: each processor can create a set of triangles from the functions defined on its subdomain. It is only necessary to ensure elements are not duplicated across boundaries shared between subdomains. In a synchronisation step, the processors then communicate and assign unique identifiers to the mesh vertices found in the first step.

At this stage, only mesh smoothing and edge flipping remain, with many possible methods of parallelisation. In this work, the implementation has been simplified by keeping the same domain decomposition and processor layout. In other words, each vertex of the mesh is assigned ownership to a particular processor, according to the same subdomain decomposition. With this simple approach, the smoothing and edge flipping can be parallelised by

(i) re-assigning ownership of vertices whenever they move across subdomain boundaries; and

(ii) maintaining a "ghost layer" of mesh connectivity information, to allow each processor to perform edge-based calculations. That is, each processor maintains a data structure for the vertices it owns, plus any neighbours of those vertices.

Together, these allow each processor to perform one step of force-based smoothing and projection, after which a synchronisation step is needed to update information near the subdomain boundaries.

The only subtle difficulty with this approach is the need to edge flip across subdomain boundaries. This can be achieved by using two ghost layers, i.e. each processor knows the connectivity and position of each neighbouring vertex of each neighbouring vertex of each vertex the processor owns. Processors then mutually agree before the edge flipping step, as to exactly which processor performs the edge flipping on triangles spanning a subdomain boundary. By taking this in turns, one can ensure that every pair of triangles in the overall mesh will be subject to edge flipping.

Note however that this simple domain decomposition approach may not necessarily scale with the number of processors. It is entirely possible that the Voronoi interface exists in some parts of the domain, and not in others, so that some processors are assigned no mesh vertices, while other processors are assigned many. It follows that the scaling of this approach depends highly on the geometry of the interface. Therefore, different parallelisation methods may be necessary if scaling is crucial to performance; in the case of the foam application, this simple approach was sufficient to make the computational cost of mesh generation only a small fraction of the overall cost.

5.4 Results

Consider the example shown in Figure 5.2 of four intersecting spheres. Using the same initial mesh (i.e. Figure 5.2 (right)), Figure 5.5 shows successive iterations of smoothing and edge flipping. One can see that after just one or two steps, many of the poorly shaped triangles have significantly improved in quality. After ten steps, much of the mesh has been "cleaned up". At this point, the majority of changes in the mesh after more smoothing steps is in edge connectivity alone, rather than in improving element quality. Convergence to an equilibrium is essentially attained after 50 iterations. These observations can be made more quantitative by examining various measures of mesh quality and geometry as a function of iteration count, as shown in Figure 5.6. Here, histograms as a function of iteration count are shown, for triangle inradius, circumradius, edge length, quality, angles, and vertex degree. The plots show that poor quality elements are quickly replaced, and that the distribution of inradii, circumradii, edge lengths, and triangle quality converge favourably so that the vast majority of elements have approximately the same geometry. The exception is in the distribution of triangle angles, which has higher variance. In addition, one can see that after 20–30 iterations, the measures of mesh geometry have approximately reached an equilibrium. Essentially identical behaviour was seen in every other problem the mesh generator was tested on.

In the next example, the Voronoi Implicit Interface Method has been used to evolve seven randomly created phases under the action of multiphase curvature flow using periodic boundary conditions. Figure 5.7 shows the result of the meshing algorithm applied to the multiphase system at a particular instant in time. One can see that the meshing algorithm is able to handle complicated geometry, especially so in this case, due to the proximity of some junctions to other junctions, as shown in the magnified portion of Figure 5.7.

This particular example of a multiphase interface is used to demonstrate the efficiency of the algorithm. A parallelised implementation of the algorithm was used, on a mainstream desktop computer using eight cores (by dividing the cube into $2 \times 2 \times 2$ subdomains). Creating the initial mesh (as described in §5.3.1) took 10ms of time. Vertices of the initial mesh were then given unique identifiers

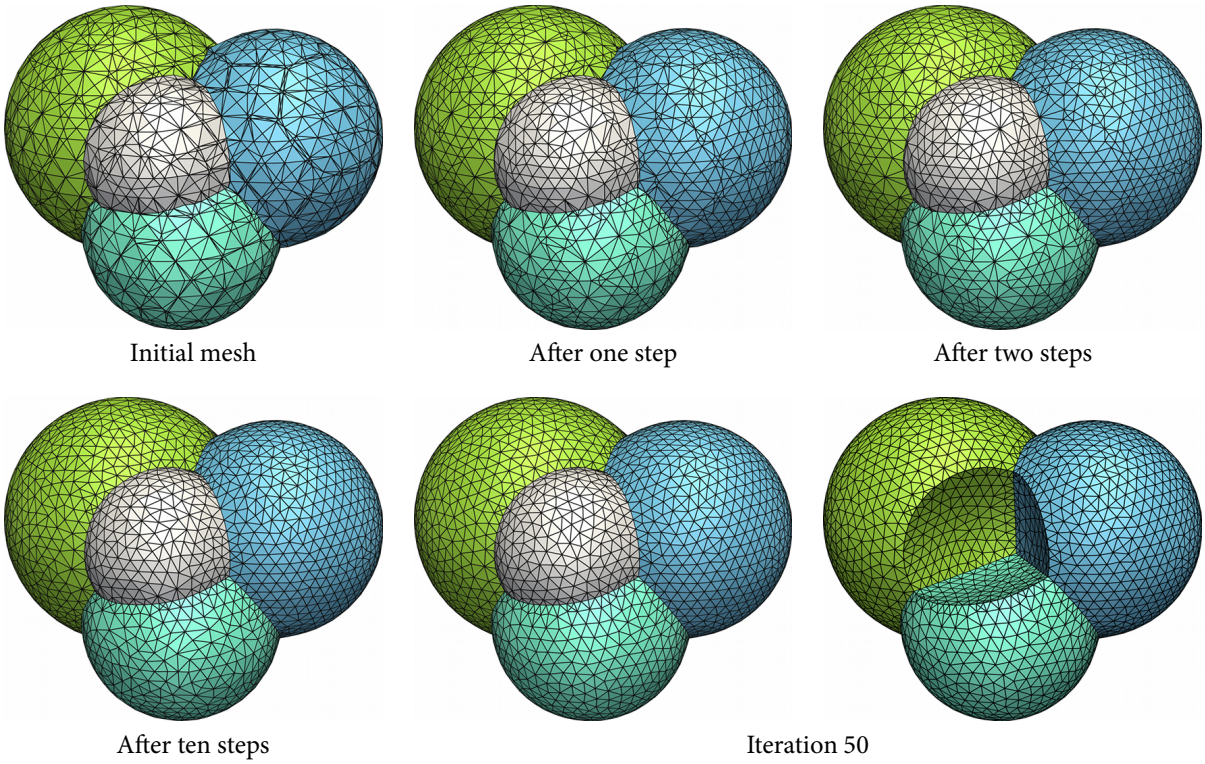

Figure 5.5. Using the same initial mesh as in Figure 5.2 (bottom), the mesh is shown after one, two, ten, and 50 steps of smoothing and edge flipping. The bottom-right two frames correspond to the same mesh, except in one the middle sphere has been cut away in order to reveal the interior mesh.

across all eight processors, and this synchronisation step took 0.25s, though the implementation of this step could be made faster. For the example shown in Figure 5.7, there were 16,592 triangles in total, and 50 iterations of smoothing were used, taking about 1.5s in total. Thus, each iteration took approximately 30ms time, and it was observed that the individual components contributed: 35% for force-based smoothing and projection; 20% for edge flipping; and 40% for synchronisation among processors. In general, the cost of the mesh generation algorithm has two components: in the first stage of creating the initial mesh, the cost is linear in the number of tetrahedra visited; in the second stage, each smoothing iteration has a cost linear in the number of mesh vertices.

To conclude this chapter on mesh generation, three more applications are shown.

Figure 5.8 shows a case that involves defining objects as intersections or unions of others, in a fashion that capitalises on the implicit representation of an interface. A sphere is divided into two halves, with the dividing surface defined implicitly via the zero level set of

$$f(x, y, z) = \cos x \sin y + \cos y \sin z + \cos z \sin x. \tag{5.6}$$

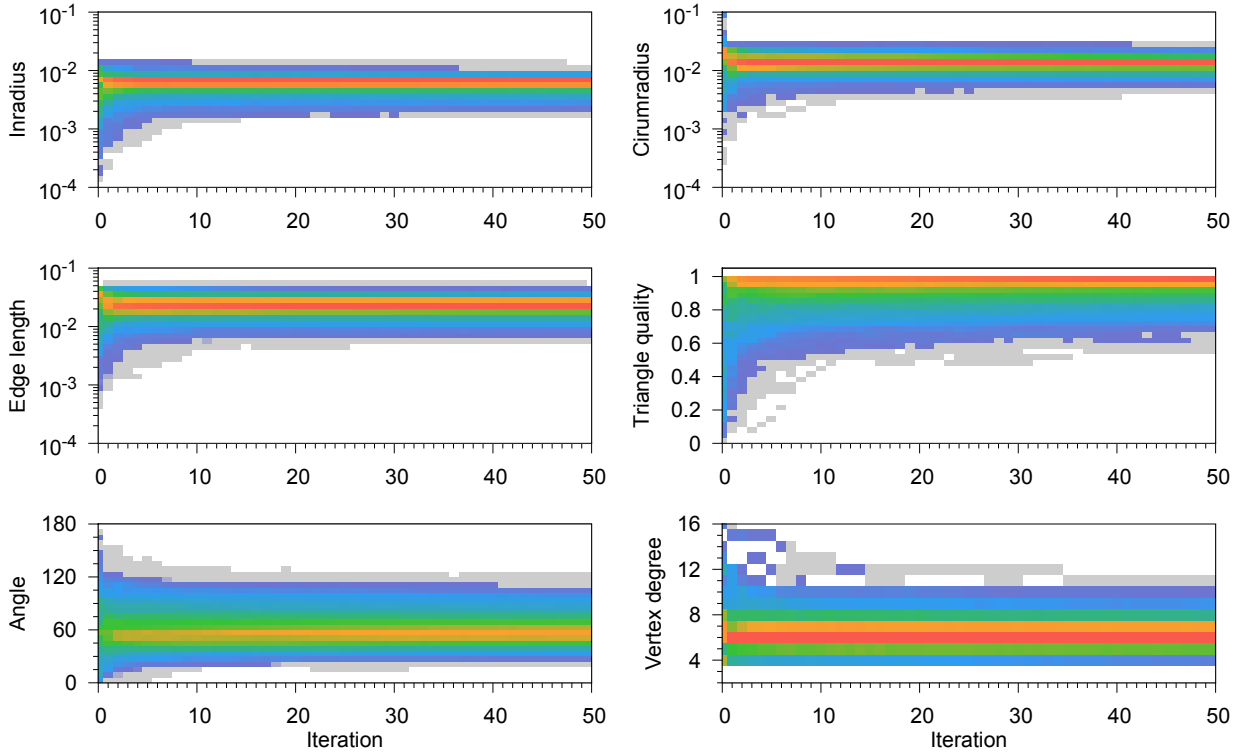

Figure 5.6. Measures of mesh quality and geometry as a function of the number of iterations, corresponding to the example shown in Figure 5.5. After a specific number of iterations, the corresponding vertical slice in each graph is a histogram indicating the percentage of features with the associated measurement: indicates less than 0.1%, indicates 0.1%, 1%, 10%, 20%, and indicates at least 50%.

The three functions are defined as

$$\phi_0(x,y,z) = \sqrt{x^2+y^2+z^2} - r + |f(x,y,z)|^3,$$
$$\phi_1(x,y,z) = f(x,y,z)^3,$$
$$\phi_2(x,y,z) = -f(x,y,z)^3,$$

where $r = 3\pi/2$ is the radius of the sphere in this example. As shown in Figure 5.8, the algorithm is able to smoothly divide the sphere into two halves, correctly reproducing sharp features.

In the next application, the Voronoi interface is used to reconstruct surfaces from volumetric point cloud data, i.e. a set of scattered points in 3D, such that the points are labelled according to which region they occupy. Point cloud data may arise in various applications, such as in experiments that use tracer particles to study fluid flow patterns, Lagrangian-based multiphase fluid flow simulations, or imaging devices that probe the interior of an object. In this particular example, a cloud of randomly generated points has been created, as shown in Figure 5.9 (left). Individual particles have been assigned to one of four regions: red, green, blue, or grey. Let y_i, $i = 1, \ldots, m$, denote the position of the particles and let χ_i denote what region they are in. For each region, define the

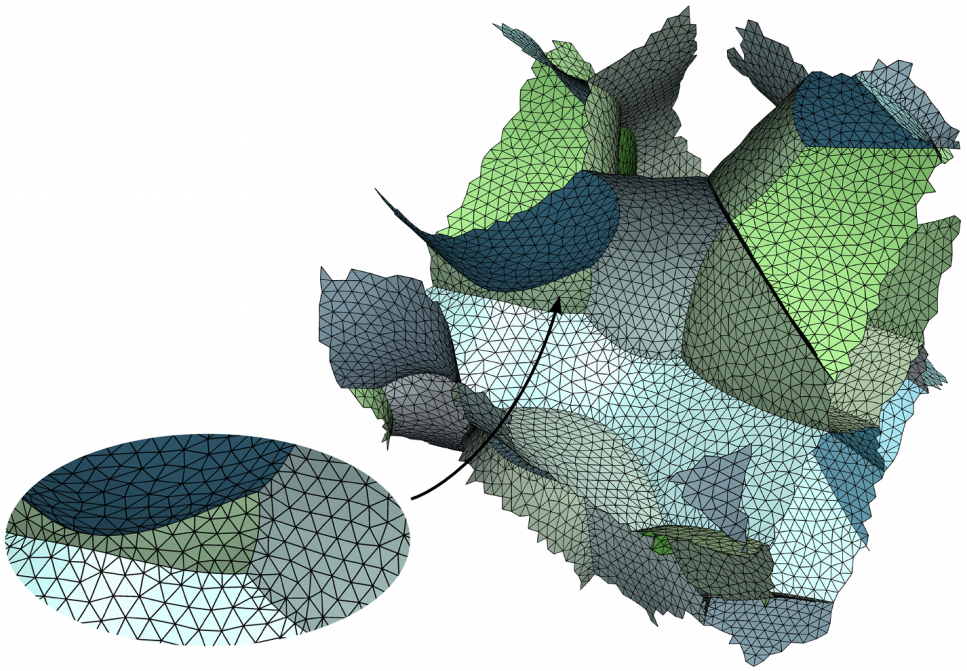

Figure 5.7. An instant in time of a multiphase system undergoing curvature flow with volume conservation. The jagged boundary is due only to the periodic boundary conditions.

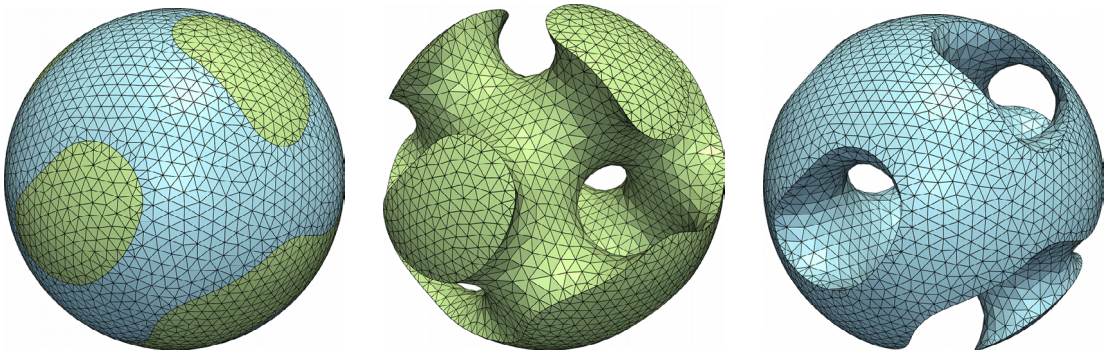

Figure 5.8. Dividing a sphere into two parts, with the dividing surface given implicitly by the zero level set of (5.6). The left figure shows the sphere as a whole, and the right two figures show the individual halves.

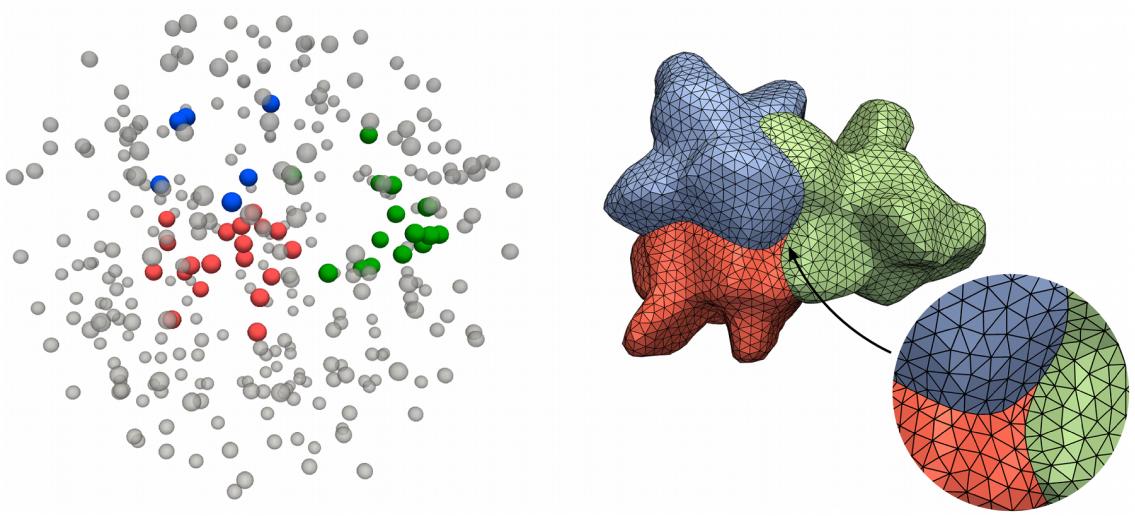

Figure 5.9. (Left) A cloud of scattered points in 3D. Points belong to one of four different regions: red, green, blue, and grey. (Right) The reconstructed surface obtained from the Voronoi interface of the functions defined in (5.7) and the mesh generation algorithm.

function ϕ_i which measures the minimum distance to the cloud of region i, i.e.

$$\phi_i(x) := \min_{\substack{1 \le j \le m \\ \chi_j = i}} \|x - x_j\|. \tag{5.7}$$

With this definition, it follows that the Voronoi interface of the functions ϕ_{red}, ϕ_{green}, etc. separates one region from another by a surface going through the middle of the space between them. By computing ϕ_i on a background Cartesian grid, the mesh generation algorithm can be used to automatically reconstruct these surfaces. For a background grid of size $32 \times 32 \times 32$, Figure 5.9 shows the reconstructed surfaces, showing nontrivial geometry at the junctions. This type of surface reconstruction from volumetric point cloud data could be especially useful when there is a large number of points since very efficient algorithms can be used to evaluate the functions ϕ_i.

Finally, in Figure 5.10, an example is taken from the multiscale model of foam dynamics that is presented in the next two chapters. In this application, it is necessary to rely on a robust mesh generation algorithm to automatically create meshes that are used in a finite element method. In a typical bubble simulation, tens of thousands of meshes are generated automatically. Figure 5.10 shows a specific example in which capillary waves have given rise to circular ripples in the membrane surfaces, seen near the top of the cluster.

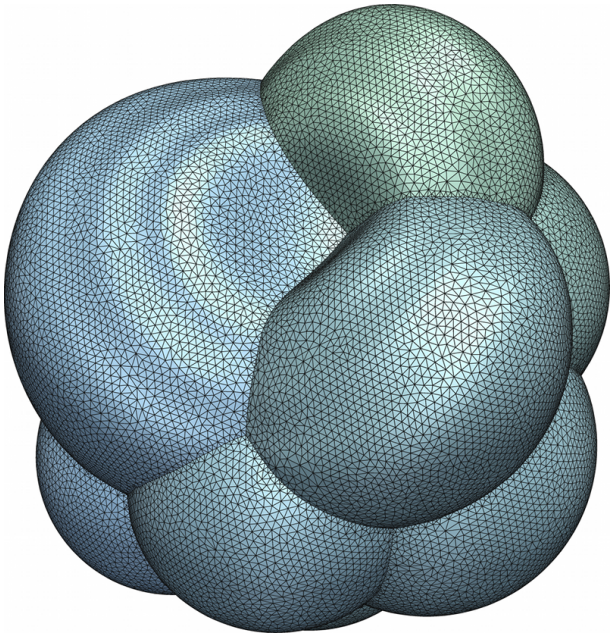

Figure 5.10. A mesh of a cluster of soap bubbles, taken from a simulation of the multiscale dynamics of soap bubble foams (see Figure 7.16 in Chapter 7). The circular ripples seen near the top are capillary waves caused by the bubbles moving under effects of surface tension. In this example, 20 iterations of force-based smoothing was used to generate the mesh.

Chapter 6

Multiscale Modelling of Foam Dynamics

Foams represent a familiar physical system in which multiple interconnected surfaces evolve over time. Liquid foams, characterised by fluid-filled membranes separating gaseous regions, include soap detergents, substances to separate out hydrophobic molecules, and even the head on a beer. Solid foams, formed by solidifying liquid foams, include lightweight materials such as plastic foams and crash-absorbent metallic foams. Understanding the dynamics of foam evolution is a key step in controlling the structure and properties of foam-like materials. However, deriving computational models to quantitatively predict foam evolution is challenging, since the underlying physics takes place over vastly different time and space scales.

We saw in Chapter 4 an application of the VIIM in modelling dry foam dynamics with permeability. While useful, this model did not include important microscale effects which are crucial in determining how foams collapse due to membrane rupture. In an attempt to include the microscale physics, one approach might be to develop computational methods that fully resolve the extreme scales of the films, which are typically only tens or hundreds of nanometres thick. However, this would require such a fine resolution that there is no practical hope of following a calculation long enough to observe the macroscale behaviour, even with advanced techniques such as adaptive mesh refinement. Thus, a multiscale model which "separates scales" must be developed, in such a way that the smallest scales are not fully intertwined with the largest scales. In this chapter, we use this idea to design a multiscale model for the dynamics of a foam. Together with the numerical methods developed in the following chapter, this leads to a computational methodology that can be used to quantitatively study collapsing foams.

6.1 Multiscale physics in foams

Consider a foam made up of common soap bubbles. A single bubble consists of a thin membrane containing soapy fluid, known as the lamella, which separates the inside gas from the outside. In a cluster of such bubbles, such as the one in Figure 6.1(a), multiple lamellae meet at junctions known as Plateau borders, forming a network of interconnected thin-film membranes. The dynamics of this foam cluster are intricate [63, 89], and depend on complex interactions between microscale fluid flow

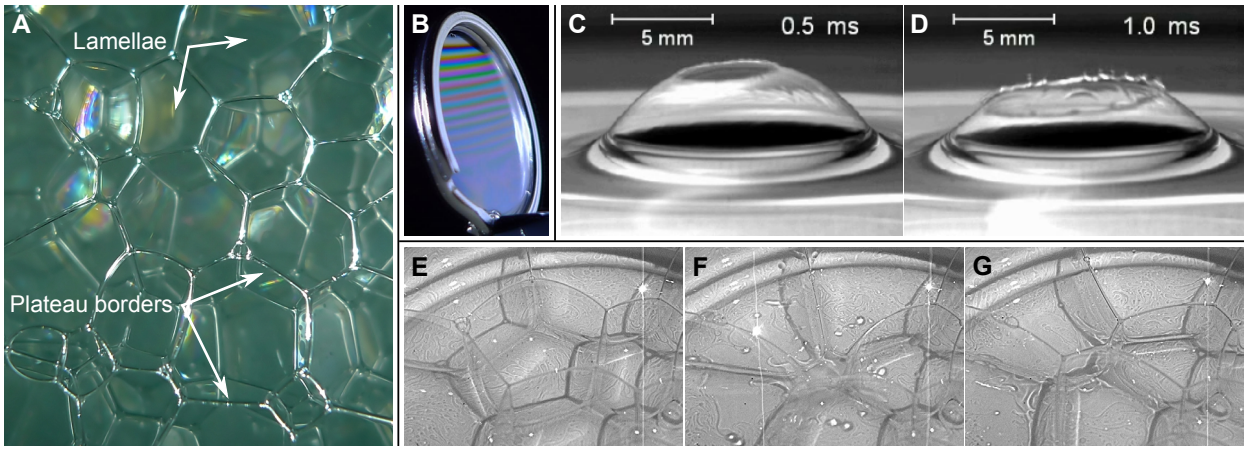

Figure 6.1. Physics of foam drainage. (a) A foam of soap bubbles made with common washing detergent. (b) Drainage and thin-film interference – a keyring suspended in soap solution makes a film, which then drains due to gravity. The subsequent variations in film thickness create interference patterns when lit with white light. This image was captured with a consumer digital camera. (c,d) Rupture of a lamella; reproduced from [90] by permission from Macmillan Publishers Ltd, Nature, copyright 2010. (e,f,g) Rearrangement – a lamella (centre of (**E**)) bursts, leading to macroscopic rearrangement of a foam. The frames are from a movie captured using a CCD black and white camera.

inside the lamellae and Plateau borders, as well as macroscale motion of the gas inside the bubbles. To illustrate, consider a foam whose macroscopic configuration appears to be in equilibrium, such as the foam in Figure 6.1(a). While seemingly stable, liquid inside the films drains over time, due to effects of gravity and surfactant. Additionally, due to the reduced pressure of the liquid inside the Plateau borders, liquid is "sucked" from the lamellae into the Plateau borders through a process called marginal regeneration. When one of the membranes becomes too thin, it ruptures and its liquid contents is redistributed, destroying the macroscopic equilibrium of the remaining membranes. Driven by macroscale gas dynamics and surface tension, these other membranes, as well as their film thicknesses, further change as they bend, stretch, and settle into a new equilibrium satisfying Plateau's laws.

These processes take place over six orders of magnitude in space and time. Bubbles can range from millimetres to centimetres in size, while typical film thickness in the lamellae range from tens of nanometres to micrometres, and a typical cross-sectional width of a Plateau border is tens to hundreds of micrometres. Liquid inside the thin films drains over seconds to tens of seconds, leading to variations in film thicknesses; these can be observed from thin-film interference effects, as for example shown in Figure 6.1(b). When a membrane ruptures, it bursts and the film retracts at hundreds of centimetres per second [90], as shown in Figure 6.1(c,d). Finally, macroscopic rearrangement of bubble topology through surface and fluid forces occurs over less than a second, as demonstrated in Figure 6.1(e,f,g). These disparate time scales suggest that it may be possible to separate foam rearrangement from liquid drainage, and this is the key idea used in the multiscale model below.

6.2 Previous work

One of the earliest and well-known studies of foams was by Plateau in the 19th century [91]. His descriptions of the geometry of a stable foam are now known as Plateau's laws: these state, among other properties, that lamellae meeting at Plateau borders make 120° angles, and that lamellae have constant mean curvature. Another well-known result is due to von Neumann and Mullins, who independently derived equations describing the growth rate of cells in a two-dimensional foam. A substantial amount of work relating to foams and foam dynamics now exists in the literature. Typically, these works focus on only one aspect of foam physics, with methods of study ranging from mathematical analyses, to numerical and experimental studies. Soap bubbles, as well as simple multi-membrane structures, provide one of the best known examples of minimal surfaces having constant mean curvature. Minimal surfaces have received a great deal of mathematical attention, and computational methods that find minimal surfaces, and thus steady-state shapes of foams, include the methods of Chopp [92] and Polthier [93]. Significant contributions have also been made by the Surface Evolver software [9], which can be used to compute minimal energy states of complex configurations. Mathematical theory for small-scale capillary-generated oscillations of soap bubbles have been developed [94], and these have been compared to experiment using high-speed cameras [95]. Three-dimensional versions of von Neumann-Mullins' law have recently been discovered [62], while statistical variants [96] can be used to study three-dimensional foam coarsening. Rearrangement in two-dimensional foams have been studied in many works, including [97] that considers in more detail one of the main types of topological change in a 2D foam, and in works such as [98–100] which gather experimental statistics about the frequency and distribution of rupture events taking place in 2D and 3D foams. Computational tools aimed at specific aspects of macroscopic rearrangement include numerical studies of foam studies based on two-dimensional hydrodynamics [16, 17]. Meanwhile, liquid drainage in stationary, isolated, and planar lamellae have received a great deal of attention, and often lead to lubrication style thin-film equations. For a review of thin-film equations, see [101, 102]; thin-film equations have also been derived on moving curved surfaces [103]. There have also been studies of liquid drainage in stationary Plateau borders [104–106]. Finally, the physical processes governing liquid suction between lamellae and Plateau borders has also been studied with asymptotic methods [107–109].

6.3 Multiscale modelling using scale separation

We now develop a multiscale model using the idea of scale separation. This model includes the most fundamentally important physical phenomena occurring in foam dynamics. It is important to note however that there are additional forces, scales, and regimes, beyond those included below, which can play an important role in foam dynamics, although they could be added to this framework. For example, diffusive coarsening, which results from gas exchange between bubbles separated by permeable membranes, is important over very long time scales (minutes to hours) [63]; although the macroscale Navier-Stokes fluid solver easily allows such permeability effects (as was shown in §4.2 of Chapter 4), the thin-film equations derived below assume a static equilibrium, and thus coarsening

effects are not included. Evaporation of liquid from the lamellae, which can be important for exceptionally stable foams and also occurs over a very long time scale, is not included here. Additionally, Marangoni forces at the liquid-gas interface, which act to equilibriate surfactant concentration, are assumed to occur quickly enough to produce a uniformly constant surface tension. In addition, the liquid-gas interface is assumed to satisfy a no-slip boundary condition. While this is a reasonable approximation to make for some types of surfactant solution, in other cases, mobile boundary conditions, as well as other effects of surface rheology, can be important. It is also assumed that that the gas and liquid flow can be taken as incompressible, which is a reasonable approximation to make under the time and space scales under consideration. Finally, dry foams, i.e. foams with liquid occupying less than approximately 10% of the total volume [63], are the focus of this model, though the methodology below has extensions to wet foam modelling as well. Further remarks about how the model can be generalised to include additional physics are provided later.

To derive a multiscale model for foam dynamics taking into account rearrangement as well as film drainage, the idea of "scale separation" is employed. In this approach, different models and equations at different scales are derived to compute physics using different resolutions, in such a way that essential information is communicated across the scales.

Here, the dynamics of a foam are separated into a cycle of three distinct stages that couple different scales across space and time. These stages are: (i) a rearrangement phase, in which a foam out of macroscopic balance undergoes rearrangement due to surface tension and gas dynamics, leading to an equilibrium; (ii) a liquid drainage phase, in which the foam is essentially in macroscopic equilibrium, and the microscopic flow of liquid is modelled until a lamella becomes too thin; and then (iii) a rupture phase, in which a lamella ruptures, sending the foam out of macroscopic balance, after which step (i) is invoked and the process repeated. Together, the dynamics of each phase affects the next, leading to a multiscale model which captures the key effects of foam rearrangement, liquid drainage, and rupture.

To summarise, in addition to modelling the gas dynamics and determining the motion of the network of thin film membranes, we also determine film thicknesses of each lamellae and Plateau border. These film thicknesses are allowed to vary in space and time, and are affected by gravity and pressure gradients inside the films, as well as membrane stretching and compression when the films move.

6.3.1 Rearrangement phase

In the rearrangement phase, the foam structure is out of macroscopic equilibrium. Surface tension at the liquid-gas interface influences the gas dynamics, which in turn evolve the network of lamellae and Plateau borders, rearranging the system of bubbles. Liquid contained in the thin films and Plateau borders is conserved and transported during this readjustment. Since macroscopic fluid mechanics dominate the dynamics, the membranes are idealised as massless and vanishingly thin, so that their inertial effects are assumed negligible. Mathematically, this leads to the incompressible Navier-Stokes equations for the gas phase, with continuity of the velocity field across the liquid-gas interface Γ, and an effective surface tension of 2σ (i.e. twice the coefficient of a single liquid-gas

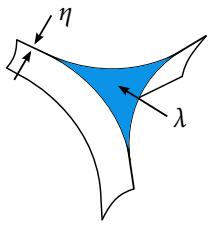

Figure 6.2. Cross-section of a Plateau border connecting to three lamellae. Here, η denotes the half-thickness of a lamella, and λ denotes the cross-sectional area of a Plateau border.

interface). The interface is thus advected by the velocity field $\mathbf{u}$ of the gas, satisfying

$$\rho_g(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla p + \mu_g\Delta\mathbf{u} - 2\sigma\kappa\mathbf{n}\delta(\Gamma),$$

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

where μ_g is the viscosity of the gas and ρ_g is its density. Here, the surface tension force, existing only at the interface Γ, has been written as a body force through the use of a Dirac delta function with support concentrated at the interface [65]. As shown in §4.2 of Chapter 4, the resulting dynamics naturally enforce 120° angle conditions at Plateau borders obeyed by dry foams and allow the interface to change topology.

During rearrangement, the liquid in the lamellae and Plateau borders is transported by the motion of the interface in such a way that the amount of liquid is locally conserved. We exploit the thinness of the lamellae and Plateau borders by describing their "thickness" with a single scalar function, allowed to vary in space and time, by using certain symmetry assumptions which are described further in §6.5 below. For the lamella, its half-thickness is defined as η, and for the Plateau border, we define λ as the cross-sectional area at any particular location in space; see Figure 6.2. For liquid contained in the lamellae, conservative transport is modelled by requiring that

$$\frac{d}{dt}\int_{S(t)}\eta = 0 \tag{6.1}$$

where $S(t)$ is any surface patch on $\Gamma(t)$ passively advected by the velocity field $\mathbf{u}$. In particular, (6.1) states that as a surface patch stretches and deforms over time, the amount of liquid inside the patch is conserved.[1] Furthermore, (6.1) allows surface currents at the interface to move the liquid tangentially. Liquid in the Plateau borders is conserved with an analogous conservation law: if $L(t)$ is a line segment on a Plateau border which is passively advected by $\mathbf{u}$, then

$$\frac{d}{dt}\int_{L(t)}\lambda = 0. \tag{6.2}$$

[1]Using the same symmetry assumptions that are used to define η and λ, as described in detail in §6.5, it follows that the mass of liquid in a lamella surface patch is proportional to $\int_S \eta$, and the mass of liquid in a Plateau border line segment is proportional to $\int_L \lambda$.

Thus, in the rearrangement phase, the main equations of motion are the incompressible Navier-Stokes equations with surface tension, together with the local conservation laws (6.1) and (6.2). These are solved until the network of interfaces is in equilibrium – numerical methods for solving these equations, as well as detecting equilibrium, are described in the next chapter.

6.3.2 Drainage phase

During the liquid drainage phase, the foam is essentially in macroscopic equilibrium, which means that the dynamics of the gas phase may be taken as negligible. In addition, due to macroscopic effects of surface tension, it follows that the surface area of the lamellae has been locally minimised, hence individual lamellae have constant mean curvature. We thus require a model for liquid drainage in the (fixed) network of lamellae and Plateau borders. By capitalising on the inherent scales involved, and following the philosophy of "thin-film approximations" [101, 102] which describe the evolving membrane thickness in a single lamella, thin-film approximations are derived for drainage in the curved lamellae, as well as the Plateau borders, together with interrelated boundary conditions which couple the regions together.

The general idea in deriving these thin-film equations is to rewrite the equations of motion, which in this case are the incompressible Navier-Stokes equations for liquid flow inside the lamellae and Plateau borders, in a non-dimensionalised form, taking into account the scales of the geometry. In an asymptotic limit in which the films are considered to be much thinner than their extent, leading order flow equations are extracted from the non-dimensionalised equations. The method leads to PDEs that determine how the thickness of the films change as a function of time, depending on surface curvature, surface tension, and gravity. Details on the derivation of these equations are given in §6.5.

For a lamella of constant mean curvature, the thin-film equation is

$$\eta_t + \frac{1}{3\mu}\nabla_s \cdot \left(\sigma\eta^3\nabla_s\left((k_1^2 + k_2^2)\eta + \Delta_s\eta\right) + \rho\mathbf{g}_s\,\eta^3\right) = 0, \tag{6.3}$$

where μ is the viscosity of the liquid, ρ is its density, and $\mathbf{g}_s$ is the component of gravity tangential to the surface. Here, ∇_s is the surface gradient, $\nabla_s\cdot$ is the surface divergence and Δ_s is the surface Laplacian on the curved surface of the lamella, while k_1 and k_2 are its principal curvatures. This is a fourth order PDE and needs two boundary conditions on the boundary of the lamella. One condition is chosen to be that of zero Neumann: $\partial\eta/\partial\nu = 0$, where ν is tangent to the lamella and orthogonal to its boundary. The other is provided by a flux boundary condition [107, 108] that implements suction of liquid into the Plateau borders at its boundary. This is specified shortly.

A similar PDE is derived (see §6.5.2) for the Plateau border: in this case, the cross-sectional area λ of the Plateau border satisfies the equation

$$\lambda_t + \frac{C_\Delta}{\mu}\frac{\partial}{\partial\ell}\left(-\tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{1/2}\sigma\lambda^{1/2}\partial_\ell\lambda + \lambda^2\rho g_\tau\right) = S, \tag{6.4}$$

where C_Δ is a constant associated with the cross-sectional shape of the Plateau border, g_τ is the tangential component of gravity, and S is a source term representing the incoming flux of liquid

from the three lamellae connected to the Plateau border. This equation requires boundary conditions where Plateau borders meet at quadruple junctions, and these are provided by conservation of liquid mass and quasi-static pressure balance. In particular, under quasi-static Stokes flow in the network of Plateau borders, it follows that the pressure of the liquid is continuous at quadruple junctions, which together with the Young-Laplace equation, implies that the Plateau borders meeting at a quadruple junction have the same local thickness. Thus, if λ_i, $i = 1, \dots, 4$ denotes the four Plateau border thickness functions meeting at a quadruple point, then

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4, \quad \text{and} \\ \sum_{i=1}^{4} \frac{1}{2}(\sqrt{3} - \frac{\pi}{2})^{\frac{1}{2}} \sigma \lambda_i^{\frac{1}{2}} \partial_\ell \lambda_i - \lambda_i^2 \rho g_{\tau_i} = 0. \end{cases} \tag{6.5}$$

For each quadruple point, this is a nonlinear system of five equations in five unknowns (the common value of λ and the four fluxes).

To determine the flux boundary condition coupling the flow in the lamellae to the flow in the Plateau borders, a local Stokes flow argument is used to derive an equation for the flux given the local thicknesses of adjoining lamellae and Plateau borders. The derivation is given in §6.5.3, and states that the flux Q is of the form

$$Q(\eta, \lambda) = \tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{\frac{3}{4}} \frac{\sigma \eta^{5/2}}{\mu \lambda^{3/4}}.$$

This is used as a flux boundary condition in (6.3), as follows: if x is a point on the boundary of a lamella, with unit outwards pointing normal ν tangent to the lamella's surface, and if λ denotes the thickness of the Plateau border at the boundary of the lamella at the point x, then

$$\frac{1}{3\mu} \left(\sigma \eta^3 \nabla_s \big((k_1^2 + k_2^2)\eta + \Delta_s \eta \big) + \rho \mathbf{g}_s \eta^3 \right) \cdot \nu = \tfrac{1}{2} Q(\eta, \lambda). \tag{6.6}$$

This specifies an outflow of liquid along the boundary of the lamella, which coincides with the source term of incoming liquid for the Plateau border. Thus, the source term in S in (6.4) is given by

$$S = \sum_{i=1}^{3} Q(\eta_i, \lambda), \tag{6.7}$$

where η_i denotes the value of the lamellae thickness functions at a point x on the Plateau border, for the three lamellae connected to the Plateau border.

In summary, the drainage phase gives rise to a system of PDEs to solve for the thickness functions for each lamella and each Plateau border. Each lamella solves the thin-film equation in (6.3), together with a Neumann boundary condition and the flux boundary condition in (6.6), while each Plateau border solves the thin-film equation (6.4), with a source term given by (6.7), and coupled boundary conditions at each quadruple point given by (6.5). This system of PDEs is solved until such time that a lamella reaches a critically-thin threshold, in which case the rupture phase is executed.

6.3.3 Rupture phase

The rupture phase occurs when a lamella becomes critically thin due to drainage. A small tear appears, and the hole in this curved two-dimensional sheet rapidly expands as surface tension causes the membrane to retract; an example was shown in Figure 6.1(c,d). For the bubble sizes considered here, liquid in the membrane retreats to the Plateau borders [90], and this occurs over a time scale that is just a small fraction of the total time it takes for bubbles to rearrange. While this rupture could itself be treated as an evolving interface, here we simplify and assume that rupture, once initiated through a prescribed threshold, is instantaneous, and that liquid in a ruptured lamella is uniformly distributed to the neighbouring Plateau borders. This thresholding is a simple approximation of otherwise complex rupture dynamics in thin liquid films, which include among other effects, van der Waals forces and short-range surfactant destabilisation effects. Further comments are provided in the next chapter.

After rupturing the film, the foam, now out of equilibrium, undergoes rearrangement by invoking the rearrangement phase, thus completing the cycle in the multiscale model.

6.4 Summary

In summary, a multiscale model of the interplay between gas, liquid, and interface forces has been developed for a dry foam. The essential idea behind the model is the use of *scale separation* to separate the fundamental time scales, simultaneously with separating space scales. In the rearrangement phase, macroscale dynamics are modelled over a short time scale, which subsequently transports film thicknesses in the network of membranes. In the drainage phase, which uses these updated film thicknesses, liquid drainage inside the network of membranes is modelled over a long time scale, until a membrane ruptures. These two phases are coupled via the rupture phase, which also transports liquid mass. In principle, additional microscopic and macroscopic forces could be incorporated into this model, such as disjoining pressures, van der Waals forces, diffusive coarsening, etc. Generalising the model and future work are discussed in the conclusions of the next chapter, after numerical methods and results for the above multiscale model are presented.

6.5 Derivation of the thin-film equations

6.5.1 Derivation of the lamella thin-film equation

In the following, a brief derivation of the lamella thin-film equation is given. As in other thin-film equations, the derivation is based on applying a lubrication limit to obtain leading order equations of motion from the underlying Navier-Stokes equations governing the flow of liquid inside the lamella. This flow is bounded above and below by two free surfaces that correspond to the gas-liquid interface, and the derivation leads to an evolution equation for thickness between the two free surfaces. For more detailed considerations, including different types of thin-film equations on flat manifolds, see the reviews [101, 102], and for curved substrates, see [103, 110, 111].

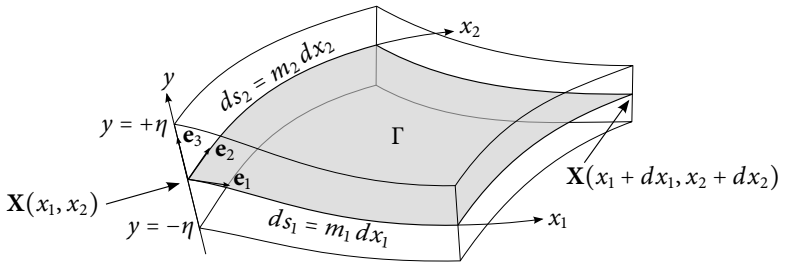

Figure 6.3. The surface Γ is parameterised with coordinates x_1, x_2 following the orthogonal lines of curvature on Γ. Together with y this forms a three dimensional curvilinear coordinate system with axes $\mathbf{e}_1, \mathbf{e}_2$, and $\mathbf{e}_3$. The upper and lower free surfaces are related to the thickness function via $y = \pm\eta$.

The flow of liquid in a lamella is attached to a macroscopic surface, which is a codimension-one smooth surface denoted by Γ. A typical simplification made in free surface thin-film flows, especially over long time scales, is to assume the film thickness is symmetric about this surface [101, 112, 113]. During the drainage phase, the dynamics of the gas phase are assumed to play a negligible role, i.e. there are no currents in the gas that cause significant drag on the liquid in the lamella. Together with the no-slip assumption, this implies that an "immobile interface" boundary condition is appropriate, wherein the tangential components of the velocity field at the two free surfaces are zero. An additional assumption is that Γ has constant mean curvature; ideally, this is the case when the rearrangement phase terminates and the bubbles are in equilibrium, so that surface area has been locally minimised. Numerically, an approximation to this state is made; however, our experiments indicate that small grid-dependent deviations from constant mean curvature have negligible effect on the flow.

Coordinate system and notation

In order to derive the thin-film equation, a coordinate system must be used.[2] A convenient curvilinear coordinate system parameterises Γ by $(x_1, x_2) \mapsto \mathbf{X}(x_1, x_2) \in \Gamma$, such that the curves $x_1 =$ constant and $x_2 =$ constant follow the orthogonal lines of curvature on Γ [110]; see Figure 6.3. (In the following, curvilinear coordinate expressions for various differentials are frequently used, e.g. the surface gradient operator and surface Laplacian, as detailed in [114].) In this coordinate system, the curvature tensor is diagonal and the diagonal components are the principal curvatures of Γ, denoted by k_1 and k_2. Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be the unit orthogonal vectors tangent to the curves of constant x_2 and x_1, respectively. They are related to the metric coefficients m_i via $\partial\mathbf{X}/\partial x_i = m_i\mathbf{e}_i$, and the arc length element ds satisfies $ds^2 = (m_1\,dx_1)^2 + (m_2\,dx_2)^2$. Given this coordinate system on Γ, it is extended off the surface into full three-dimensional space in the normal direction $\mathbf{e}_3 := \mathbf{e}_1 \times \mathbf{e}_2$. This 3D coordinate system is given by $(x_1, x_2, y) \mapsto \mathbf{X}(x_1, x_2) + y\,\mathbf{e}_3(x_1, x_2)$. Such a coordinate system is well-defined provided that y and the curvatures k_1, k_2 allow a one-to-one map. The metric coefficients h_1, h_2, h_3

[2] Importantly, however, the final evolution equations will be a coordinate-free expression involving surface divergence and gradient operators.

in these coordinates are

$$h_i = m_i(1 - k_i y), \quad h_3 = 1.$$

Notice that the metric changes in the normal direction, according to how much curvature the surface exhibits.

Let $\eta = \eta(x_1, x_2, t)$ be the half-thickness of the film. Then the upper $(+)$ and lower $(-)$ free surfaces are given by $y = \pm\eta(x_1, x_2, t)$ and the liquid occupies the region $|y| \le \eta(x_1, x_2, t)$. At the upper free surface, the unnormalised tangent vectors are

$$\begin{aligned}\mathbf{t}_i &= \frac{\partial}{\partial x_i}\Big(\mathbf{X}(x_1, x_2) + \eta(x_1, x_2, t)\mathbf{e}_3(x_1, x_2)\Big) \\ &= m_i\mathbf{e}_i + (\partial_{x_i}\eta)\mathbf{e}_3 + \eta\,\partial_{x_i}\mathbf{e}_3 \\ &= h_i\mathbf{e}_i + (\partial_{x_i}\eta)\mathbf{e}_3 \quad \text{at} \quad y = \eta.\end{aligned}$$

The normal vector of the upper free surface is thus proportional[3] to

$$\mathbf{n} \propto \mathbf{t}_1 \times \mathbf{t}_2 = (-h_2\partial_{x_1}\eta, -h_1\partial_{x_2}\eta, h_1h_2) \quad \text{at} \quad y = \eta.$$

Expressions for the normal and tangent vectors at the lower free surface are the same except for some minus signs.

Kinematic condition

The kinematic condition states that fluid particles on the free surface remain on the surface, and leads to an equation relating η_t to the velocity (u_1, u_2, v) of the fluid evaluated at the free surface. To derive the formula, let $\mathbf{x}(t) = (x_1(t), x_2(t), y(t))$ be the position of a particle on the upper free surface. Its instantaneous velocity is (u_1, u_2, v) which implies $\dot{\mathbf{x}} = (u_1/h_1, u_2/h_2, v)$. Since $\eta(x_1, x_2, t) - y = 0$, differentiating with respect to t, one obtains

$$\eta_t = v - \frac{u_1}{h_1}\partial_{x_1}\eta - \frac{u_2}{h_2}\partial_{x_2}\eta \quad \text{at} \quad y = \eta. \tag{6.8}$$

A similar equation holds for the lower free surface.

Conservation of mass

Combined with the kinematic condition, conservation of mass yields a conservation law for the thickness function η. For an arbitrary divergence-free velocity field (u_1, u_2, v), the conservation law is

$$2(1 + k_1k_2\eta^2)\eta_t + \nabla_s \cdot \mathbf{Q} = 0, \tag{6.9}$$

where $\nabla_s\cdot$ is the surface divergence operator, and $\mathbf{Q} = \mathbf{Q}(x_1, x_2)$ is an effective total flux that integrates the tangential components of the velocity field in the normal direction, and is given by

$$\mathbf{Q} = \int_{-\eta}^{\eta} u_1(1 - k_2 y)\mathbf{e}_1 + u_2(1 - k_1 y)\mathbf{e}_2\, dy. \tag{6.10}$$

[3] A three-tuple (f_1, f_2, f_3) indicates the vector $f_1\mathbf{e}_1 + f_2\mathbf{e}_2 + f_3\mathbf{e}_3$.

To derive (6.9), consider a small test volume bounded by the instantaneous free surfaces and coordinate surfaces $x_1, x_1 + dx_1, x_2, x_2 + dx_2$, as depicted in Figure 6.3. The rate at which fluid leaves this volume (expressed to first order of infinitesimal quantities dx_1 and dx_2) is the sum of four terms: first

$$\int_{-\eta(x_1+dx_1,x_2)}^{\eta(x_1+dx_1,x_2)} (u_1 h_2 dx_2)\big|_{x_1+dx_1} dy - \int_{-\eta(x_1,x_2)}^{\eta(x_1,x_2)} (u_1 h_2 dx_2)\big|_{x_1} dy$$

for the surfaces x_1 and $x_1 + dx_1$; second

$$\int_{-\eta(x_1,x_2+dx_2)}^{\eta(x_1,x_2+dx_2)} (u_2 h_1 dx_1)\big|_{x_2+dx_2} dy - \int_{-\eta(x_1,x_2)}^{\eta(x_1,x_2)} (u_2 h_1 dx_1)\big|_{x_2} dy$$

for the surfaces x_2 and $x_2 + dx_2$. The remaining two terms are for the upper and lower free surface, requiring the calculation of $\int_{y=\pm\eta} \mathbf{u}\cdot\mathbf{n}\, dS$. Since $\mathbf{n} = (\mathbf{t}_1 \times \mathbf{t}_2)/|\mathbf{t}_1 \times \mathbf{t}_2|$ and $dS = |\mathbf{t}_1 \times \mathbf{t}_2| dx_1 dx_2$, by utilising the kinematic condition (6.8), it follows that

$$\int_{y=\eta} \mathbf{u}\cdot\mathbf{n}\, dS = (u_1, u_2, v)\cdot(-h_2\partial_{x_1}\eta, -h_1\partial_{x_2}\eta, h_1 h_2)\, dx_1 dx_2 = h_1 h_2 \eta_t\, dx_1 dx_2 \quad \text{at} \quad y = \eta.$$

A similar equation holds for the lower free surface. Incompressibility of the flow field requires that the net flux of fluid out of this volume is zero. Adding up the four contributions, equating to zero, and then dividing by $dx_1 dx_2$ gives

$$\begin{aligned}(h_1 h_2|_{y=\eta} + h_1 h_2|_{y=-\eta})\eta_t = &-\frac{1}{dx_1}\left[\int_{-\eta(x_1+dx_1,x_2)}^{\eta(x_1+dx_1,x_2)} (u_1 h_2)\big|_{x_1+dx_1} dy - \int_{-\eta(x_1,x_2)}^{\eta(x_1,x_2)} (u_1 h_2)\big|_{x_1} dy\right]\\ &-\frac{1}{dx_2}\left[\int_{-\eta(x_1,x_2+dx_2)}^{\eta(x_1,x_2+dx_2)} (u_2 h_1)\big|_{x_2+dx_2} dy - \int_{-\eta(x_1,x_2)}^{\eta(x_1,x_2)} (u_2 h_1)\big|_{x_2} dy\right]\\ \to &-\frac{\partial}{\partial x_1}\left[\int_{-\eta}^{\eta} u_1 h_2\, dy\right] - \frac{\partial}{\partial x_2}\left[\int_{-\eta}^{\eta} u_2 h_1\, dy\right],\end{aligned}$$

where on the last line the limit $dx_1, dx_2 \to 0$ has been taken. Now note that for a vector-valued function $\mathbf{f} = (f_1, f_2)$ defined on the surface Γ, the surface divergence of $\mathbf{f}$ is given by the expression $\nabla_s \cdot \mathbf{f} = \frac{1}{m_1 m_2}\big[\partial_{x_1}(m_2 f_1) + \partial_{x_2}(m_1 f_2)\big]$. Hence, upon defining the total flux $\mathbf{Q}$ as in (6.10), one finds that

$$\begin{aligned}2(1 + k_1 k_2 \eta^2)\eta_t &= \big[(1 - k_1\eta)(1 - k_2\eta) + (1 + k_1\eta)(1 + k_2\eta)\big]\eta_t\\ &= \frac{1}{m_1 m_2}(h_1 h_2|_{y=\eta} + h_1 h_2|_{y=-\eta})\eta_t = -\nabla_s \cdot \mathbf{Q}.\end{aligned}$$

Scaling and non-dimensionalisation

In order to construct a leading order asymptotic solution of the governing Navier-Stokes equations in the lamella, appropriate scales must be specified. These, in turn, are used to non-dimensionalise the equations. In the following, typical scaling arguments are used (see [101, 102] for a general review of such methods). Let L be a tangential reference scale. This is the same as the typical radius of

curvature of the surface Γ; it is also the scale on which film thickness varies. Let H denote the typical film thickness. The so-called "lubrication limit" for which $\epsilon := H/L \ll 1$ is considered, where one seeks leading order solutions of the Navier-Stokes equations as $\epsilon \to 0$.

The coordinate system is non-dimensionalised by using the scaled coordinates $Y := y/H$ and $X_i := x_i/L = \epsilon x_i/H$. A velocity scale of U_0 is chosen for the tangential direction, so that the non-dimensionalised velocity components in the tangential direction are $U_i := u_i/U_0$. The natural time scale is $L/U_0 = H/(\epsilon U_0)$, which gives $T := \epsilon U_0 t/H$. Meanwhile, the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ implies that the normal component of the velocity field is scaled according to $V := v/(\epsilon U_0)$, and the natural scaling for pressure is $P := \epsilon H p/(\mu U_0)$. The principal curvatures of the surface Γ are scaled relative to $1/L$ giving $K_i := H k_i/\epsilon$. Finally, the film thickness function η is non-dimensionalised with $\zeta := \eta/H$.

Leading order flow equations

By using the curvilinear coordinate expressions for the gradient, Laplacian, etc. in the (x_1, x_2, y) coordinate system (see [114]), one can write the Navier-Stokes momentum equations

$$\rho(\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \mu \Delta \mathbf{u} + \rho g \hat{\mathbf{g}} \tag{6.11}$$

in the new scaled variables. The result is a complicated expression involving different powers of ϵ, but keeping only the lowest order terms, one obtains[4] for the $i = 1, 2$ component of (6.11)

$$\mathcal{O}(\epsilon \,\mathsf{Re}) = -\frac{1}{h_i}\partial_{X_i} P + \frac{1}{h_{i'}}\frac{\partial}{\partial Y}\frac{h_{i'}}{h_i}\frac{\partial (h_i U_i)}{\partial Y} + \mathsf{G}\,\hat{g}_i + \mathcal{O}(\epsilon),$$

while the $i = 3$ component leads to

$$\mathcal{O}(\epsilon^2 \,\mathsf{Re}) = -\partial_Y P + \epsilon\,\mathsf{G}\,\hat{g}_3 + \mathcal{O}(\epsilon^2).$$

Here, $\mathsf{Re} = \rho U_0 H/\mu$ is the Reynolds number relative to the length scale H, $\mathsf{G} = \rho g H^2/(\mu U_0)$ is a unit order gravity number, and $\hat{\mathbf{g}} = (\hat{g}_1, \hat{g}_2, \hat{g}_3)$ is a unit vector pointing in the direction of gravity. In the lubrication limit, it is typically assumed that the Reynolds number is $\mathcal{O}(1)$ as $\epsilon \to 0$. In addition, it is assumed that $\mathsf{G} = \mathcal{O}(1)$, so that the flow is not strongly dominated by gravitational effects. The asymptotic approximation for the leading order flow comes from assuming that higher order terms are negligible, and so only the leading order terms are kept. In this limit, the above set of equations reduces to

$$\begin{cases} \partial_{YY} U_i = \frac{1}{m_i}\partial_{X_i} P - \mathsf{G}\,\hat{g}_i, \quad i = 1, 2, \\ \partial_Y P = 0. \end{cases}$$

In the original unscaled and dimensional quantities, they are

$$\begin{cases} \mu \partial_{yy} u_i = \frac{1}{m_i}\partial_{x_i} p - \rho g \hat{g}_i, \quad i = 1, 2, \\ \partial_y p = 0. \end{cases} \tag{6.12}$$

[4] A primed index denotes the dual tangential component, i.e. if $i \in \{1, 2\}$ then $i' = 3 - i$.

Boundary conditions

The no-slip boundary conditions requires that

$$u_1 = u_2 = 0 \quad \text{at} \quad y = \pm\eta.$$

Surface tension on the upper and lower free surface provides the remaining boundary condition, via the Young-Laplace equation describing the pressure jump across the free surfaces, i.e. $[p] = \sigma\kappa$, where $[\cdot]$ denotes the jump in value, σ is the surface tension, and κ is the mean curvature of the interface.[5] Since the formula to calculate the mean curvature of the free surface is a complicated nonlinear expression, one can instead approximate it through an asymptotic expansion in ϵ. To do so, one approach is to observe that the (upper) free surface is the zero level set of the function $f(x_1, x_2, y) = y - \eta(x_1, x_2)$, and so $\kappa = \nabla \cdot (\nabla f/|\nabla f|)$. Using the curvilinear coordinate expressions for vector differentials and the appropriate scales, a tedious calculation shows that

$$K = -K_1 - K_2 + \epsilon(K_1^2 + K_2^2)\zeta + \epsilon\Delta_s\zeta + \mathcal{O}(\epsilon^2)$$

where $(K, K_1, K_2) = L(\kappa, k_1, k_2)$ are non-dimensionalised curvature quantities and ζ is the scaled film-thickness. Keeping the leading order terms, and in the original variables,

$$\kappa = -k_1 - k_2 + (k_1^2 + k_2^2)\eta + \Delta_s\eta.$$

Recall that the lamella is assumed to have constant mean curvature, so that $k_1 + k_2$ is constant. It follows from the Young-Laplace equation that this term can be effectively absorbed into the background reference pressure. Hence, after correctly accounting for orientation, the pressure at the free surface satisfies the relation

$$p = -\sigma\big((k_1^2 + k_2^2)\eta + \Delta_s\eta\big) \quad \text{at} \quad y = \eta. \tag{6.13}$$

Solution

The solution of the leading order flow equations with the given boundary conditions is straightforward. From (6.12), the pressure p is constant in the normal direction, with value determined by (6.13). Meanwhile, the tangential velocity attains a parabolic profile, and with no-slip boundary conditions, has the solution[6]

$$(u_1, u_2) = \frac{1}{2\mu}\big(\nabla_s p - \rho g\hat{\mathbf{g}}_s\big)\big(y^2 - \eta^2\big).$$

[5] Strictly speaking, the Young-Laplace equation only applies to static interfaces between two fluids. In full generality, one may consider the general stress balance equation for a free capillary surface, which requires that $\mathbf{n} \cdot \mathbf{T} \cdot \mathbf{n} = \sigma\kappa$ where $\mathbf{T} = -pI + \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ is the stress tensor for an incompressible Newtonian fluid. By writing this equation in scaled, non-dimensionalised quantities, one can construct an asymptotic approximation of stress balance for the free surface on the lamella. The leading order approximation of the resulting expression as $\epsilon \to 0$ is appropriate, and this reduces to the Young-Laplace equation.

[6] Here, a subtle assumption has been made that the gravitational term has negligible variation in the normal direction. This is mainly required by the condition that the thin film remains symmetric about the surface Γ.

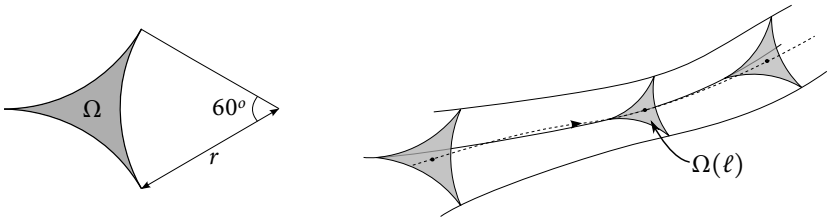

Figure 6.4. (Left) Cross-section of a Plateau border – a "rounded triangle" in which each side is an arc of a circle with radius r. (Right) In the model, a Plateau border may have variable thickness along its extent, however the cross-sectional area $\Omega(\ell)$ is always a rounded triangle as shown on the left.

Here, curvilinear expression for the surface gradient have been used, and $\hat{\mathbf{g}}_s$ denotes the tangential component of $\hat{\mathbf{g}}$. Finally, using the conservation of mass formula (6.9) leads to an evolution equation for η: a simple integral calculation reveals that

$$\mathbf{Q} = -\frac{2}{3\mu}\eta^3\Big(-\sigma\nabla_s\big((k_1^2+k_2^2)\eta+\Delta_s\eta\big)-\rho g\hat{\mathbf{g}}_s\Big),$$

and so[7]

$$\eta_t+\frac{1}{3\mu}\nabla_s\cdot\Big(\sigma\eta^3\nabla_s\big((k_1^2+k_2^2)\eta+\Delta_s\eta\big)+\rho g\hat{\mathbf{g}}_s\,\eta^3\Big)=0.$$

6.5.2 Derivation of the Plateau border thin-film equation

In this section, the "thin-film equation" for an idealised Plateau border is derived. The idea and method of the derivation is similar to the case of a lamella: a Stokes flow approximation is considered together with the Young-Laplace equation for surface tension. The Plateau border's trajectory is parameterised with arc length ℓ and its thickness is allowed to vary along its extent. To extract leading order flow equations, the thickness of the Plateau border is assumed to be much smaller than the radius of curvature seen in its macroscopic trajectory (the "line curvature"). In addition, the cross-sectional shape is idealised as a rounded triangle as shown in Figure 6.4, as satisfied by Plateau borders in dry foams [63]. This shape arises because the thickness of the lamellae connected to the three arms is much smaller than the thickness of the Plateau border, while balance of surface tension for steady flow implies that the arcs have constant radius of curvature. This cross-section at position ℓ is denoted by $\Omega(\ell)$. The radius of curvature at the boundary of this triangle is r, and this is related to the area $\lambda = |\Omega(\ell)|$ of the triangle by

$$\lambda = \Big(\sqrt{3}-\tfrac{\pi}{2}\Big)r^2. \tag{6.14}$$

Given the scale assumptions already made, it can be assumed that any line curvature of the Plateau border is negligible. It follows that the amount of fluid in a line element $d\ell$ is $\rho\,\lambda\,d\ell$. Conservation

[7] Note that the term $k_1k_2\eta^2$ in the conservation of mass equation is an $\mathcal{O}(\epsilon^2)$ correction, and thus can be neglected.

of mass leads to the evolution equation[8]

$$\lambda_t + \frac{\partial}{\partial \ell} \int_{\Omega(\ell)} u = S, \tag{6.15}$$

where u denotes the tangential component of the fluid velocity, integrated over the cross-section $\Omega(\ell)$. Here, S is a source term which is needed to account for the flux boundary condition, such that ρS has units of mass per length per time.

Stokes flow with the no-slip boundary condition yields a flow of the "parabolic" type. Indeed, let ϵ be the ratio of the typical diameter of the Plateau border to its length. Then $\epsilon \ll 1$, and as in the case of the lamella, one can extract the first order asymptotic solution of the governing equations of flow as $\epsilon \to 0$. Similar to (6.12) in the case of the lamella, this leads to

$$\mu \Delta_\Omega u = \partial_\ell p - \rho g \hat{g}_\tau \quad \text{in} \quad \Omega(\ell),$$

where Δ_Ω denotes the Laplacian on the two-dimensional Euclidean surface $\Omega(\ell)$ and $\hat{g}_\tau$ is the tangential component of gravity. Meanwhile, the Young-Laplace equation determines the pressure on the cross-section as $p = \sigma\kappa$ where κ is the mean curvature at the surface. Utilising the scales once again, the curvature can be approximated as $\kappa = 1/r$, since the principal curvatures of the Plateau border are essentially the curvature seen in its cross-section and that seen along its trajectory. Equation (6.14) thus gives $\kappa = -(\sqrt{3} - \frac{1}{2}\pi)^{\frac{1}{2}} \lambda^{-\frac{1}{2}}$ (the minus sign correctly accounts for orientation). Solving these set of equations, it follows that

$$u = \frac{1}{\mu}\Big(\rho g \hat{g}_\tau + \sigma(\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}} \partial_\ell \lambda^{-\frac{1}{2}}\Big)\, w$$

where w is the solution of

$$\begin{cases} \Delta_\Omega w = -1 & \text{in } \Omega(\ell), \\ w = 0 & \text{on } \partial\Omega(\ell). \end{cases} \tag{6.16}$$

Figure 6.5 illustrates the profile of the solution w. Straightforward scaling arguments show that there is a constant $C_\Delta > 0$, depending only on the shape of $\Omega(\ell)$ but not its size, such that

$$\int_{\Omega(\ell)} w = C_\Delta |\Omega(\ell)|^2 = C_\Delta \lambda^2.$$

By making use of the conservation of mass equation (6.15), it follows that

$$\lambda_t + \frac{C_\Delta}{\mu} \frac{\partial}{\partial \ell}\Big(-\tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}} \sigma \lambda^{\frac{1}{2}} \partial_\ell \lambda + \lambda^2 \rho g \hat{g}_\tau\Big) = S.$$

[8] Consider a small length element $[\ell, \ell + d\ell]$ on the Plateau border. The total mass inside the element is $\rho \int_\ell^{\ell+d\ell} \lambda \, ds$. In a time interval dt, a mass of $dt\, \rho \int_{\Omega(\ell)} u \, dS$ enters from the left, a mass $dt\, \rho \int_{\Omega(\ell+d\ell)} u \, dS$ exits at the right and mass of $dt \; d\ell \, \rho \, S$ arises from the source term. Letting $dt \to 0$, one obtains

$$\rho \frac{d}{dt} \int_\ell^{\ell+d\ell} \lambda \, ds = \rho\Big[\int_{\Omega(\ell)} u \, dS - \int_{\Omega(\ell+d\ell)} u \, dS\Big] + \rho S \, d\ell,$$

which leads to (6.15).

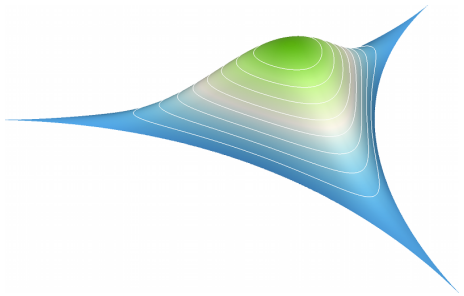

Figure 6.5. Solution of $\Delta w = -1$ with zero Dirichlet boundary conditions in a cross-section of the Plateau border, represented as a height function.

Finally, to determine the precise value of C_Δ, a finite element method was used to solve (6.16); convergence studies were performed that determined $C_\Delta \approx 0.0201$, accurate to three significant digits.

6.5.3 Flux boundary condition

The process by which fluid drains from the lamella into the Plateau borders on its boundary (due to the reduced pressure in the Plateau border), called "marginal regeneration", suggests that there is a flux-type boundary condition between the two domains. One treatment of this subject is in [107], wherein the authors consider regimes in which the concentration of surfactant can change over time, and mainly consider liquids that have slip at the liquid-gas interface. In [108], stress free boundary conditions are considered, showing that in this case, suction of liquid at Plateau borders is instantaneously transmitted throughout the film. However, in this work, the case in which the liquid films exhibit immobile boundary conditions is considered. In this case, similar arguments can be used to derive a flux boundary condition, as follows.

Suppose there is a transition region between the lamella and the Plateau border, in which the flow transitions from the Stokes-type flow in the lamella, to the Stokes flow in the cross-section of the Plateau border. On one side of the transition region, the thickness of the liquid equals that in the lamella, η. On the other side, the curvature of the free surface equals that of the Plateau border, $1/r$. Assume, as in [107], that throughout this transition region, the fluid thickness remains on the same scale as the lamella. This imposes a constraint on its width w satisfying $w = \mathcal{O}(\sqrt{\eta r})$. Now, consider the pressure. The Young-Laplace equation implies that the pressure difference between the two ends of the transition region is approximately σ/r, since the curvature at the Plateau border dominates that at the lamella. Hence the pressure gradient p_x, as measured in the outwards pointing direction orthogonal to the boundary of the lamella, is approximately $\sigma/(rw)$. Stokes flow (and the no-slip boundary condition) in the transition region then shows that the total flux, measured in the outwards pointing direction and near the side connected to the lamella, is

$$Q = \frac{\eta^3}{\mu} p_x = \frac{\eta^3 \sigma}{\mu r} \mathcal{O}((\eta r)^{-\frac{1}{2}}) = C \frac{\sigma \eta^{5/2}}{\mu r^{3/2}}.$$

This simple derivation does not determine what value $C = \mathcal{O}(1)$ carries. To determine that would require more careful matched asymptotics, which is not considered here. The flux boundary condition is implemented as follows: using (6.14) to relate λ to r, define

$$Q(\eta, \lambda) := C(\sqrt{3} - \tfrac{\pi}{2})^{\frac{3}{4}} \frac{\sigma \eta^{5/2}}{\mu \lambda^{3/4}}. \tag{6.17}$$

This relation is used in equations (7.8) and (7.10) in the next chapter to implement the flux boundary condition. In the simulations presented here, the value of the constant was set to $C = \frac{1}{2}$. This lead to qualitative agreement with some experimental results, however a more careful justification of this flux boundary condition would be beneficial.

6.5.4 Boundary layer scaling

Finally, in this section, a scaling argument is used to determine the width of certain boundary layers which develop in the draining of a lamella. This scaling is used as part of the discussion of the results in the following chapter. Consider a lamella which starts with a uniform thickness η_0. As the lamella drains into nearby Plateau borders, a boundary layer develops (see, for example, Figure 7.13), the width of which depends on η_0 and the thickness λ of the connecting Plateau borders. To investigate the scaling, consider a simplification of the governing thin-film equation in one-dimension, on the half-line $[0, \infty)$, such that $\eta = \eta(x)$ satisfies the conservation equation $\eta_t + \partial_x Q = 0$, where

$$Q = \frac{1}{3\mu}\big(\sigma \eta^3 \partial_x (k^2 \eta + \eta_{xx})\big). \tag{6.18}$$

Here, the effect of gravity has been assumed negligible. The flux boundary condition yields the condition that

$$Q = -C \frac{\sigma \eta^{5/2}}{\mu \lambda^{3/4}} \quad \text{at} \quad x = 0, \tag{6.19}$$

where λ is the thickness of an idealised Plateau border situated at $x = 0$. Non-dimensionalising (6.18) and (6.19) with a reference length scale of L yields

$$\frac{\eta_0^4}{3\mu L^3}\big(\sigma \tilde{\eta}^3 \partial_{\tilde{x}} (\tilde{k}^2 \tilde{\eta} + \tilde{\eta}_{\tilde{x}\tilde{x}})\big) = -C \frac{\eta_0^{5/2}}{\lambda_0^{3/4}} \frac{\sigma \tilde{\eta}^{5/2}}{\mu \tilde{\lambda}^{3/4}}, \tag{6.20}$$

where λ_0 is a typical Plateau border thickness, and where the tilde denotes non-dimensional quantities. To find the scaling of the boundary layer, one expects the length scale L should balance the flux with the flux boundary condition. Rearranging (6.20), this yields $L = \mathcal{O}(\eta_0^{1/2} \lambda_0^{1/4})$ as the scaling of the boundary layer width. These simple arguments do not reveal how the width depends on the time of drainage – this could be determined by finding solutions of the governing PDE, for example with a self-similarity reduction to a nonlinear ordinary differential equation. Instead, additional numerical experiments were performed, using the methods in the next chapter, and suggested that the relation $L \sim A \eta_0^{1/2} \lambda_0^{1/4}$ gave a good indication of the boundary layer width, where the value of A is order one, with a weak dependence on time, for the typical drainage time scales considered.

Chapter 7

Numerical Methods for Foam Dynamics

In the previous chapter, a multiscale model of foam dynamics was developed. Recall that the model separates foam dynamics into three coupled phases: rearrangement, drainage, and rupture. In order to implement the associated computational framework, several new numerical methods have been developed for each of these phases. In the next set of sections, these algorithms are presented and their convergence is tested. The entire system is then coupled and results are shown for a variety of problems in foam dynamics in the last section.

7.1 Numerical methods for the rearrangement phase

In the rearrangement phase, the incompressible Navier-Stokes equations for the gas phase are solved, together with the equations for local conservation of mass for the liquid contained in the lamellae and Plateau borders. Briefly, the Navier-Stokes equations are solved on a fixed Cartesian grid using a second order approximate projection method. To track the interface, the Voronoi Implicit Interface Method is used, and for the local conservation of the thickness functions λ and η, a Lagrangian based scheme is developed. The main algorithm for the evolution during the macroscopic rearrangement phase is:

1. Initialise the VIIM unsigned distance function ϕ^0 and indicator function χ^0 at time step zero, together with λ^0 and η^0. Initialise the gas velocity field with $\mathbf{u}^0 = 0$.
2. Enter time step loop: for $n = 0, 1, \ldots,$
   (i) Advance the interface and velocity field in time to find ϕ^{n+1} and $\mathbf{u}^{n+1}$ at time step t^{n+1} (§7.1.1 and §7.1.2).
   (ii) Update the thickness functions to find λ^{n+1} and η^{n+1} (§7.1.3).
   (iii) Check for equilibrium (§7.1.5) and if it has been reached, execute the drainage phase (§7.2). Otherwise, continue time stepping.

7.1.1 Interface tracking

The VIIM is used to track the motion of the network of interfaces (i.e. the membranes) in the foam. Here, the bubbles and the exterior region are considered separate regions ("phases" in the terminology of Chapter 3) in the multiphase interface evolution. A shown in §4.2 of Chapter 3, the VIIM provides an ideal framework to accurately calculate effects of surface tension, not only on membrane surfaces, but at junctions as well. In particular, in this application of Navier-Stokes driven interface evolution, the interface is simply advected by the velocity field $\mathbf{u}$ of the gas. In this work, the numerical method identified by the $\epsilon = 0^+$ limit (see §3.4.1) is extensively used, and the implementation essentially used multiple narrow-banded level set functions.

Since the viscosity of a typical gas species is relatively low, the velocity field computed by Navier-Stokes is often not particularly smooth. As a result, one may expect that the use of extension velocities could provide better numerical results, in terms of reduction of numerical diffusion and increased conservation of mass. This was indeed the case – extension velocities significantly reduced the effect of currents that exist nearby the interface. The method to calculate the extension velocity is relatively straightforward: at each time step, the Voronoi interface is reconstructed as a mesh, by using the same mesh that is employed in surface tension calculations. For each grid point in a small initial band surrounding the Voronoi interface, an extension velocity is calculated at the grid point by interpolating the velocity field at the corresponding closest point on the mesh (using trilinear interpolation). The initial band is then used to calculate the extension velocity in the entire narrow band of ϕ, using the pre-sorted fast marching strategy described in §2.5. Once found, the extension velocity is used together with a standard second order ENO upwinding scheme to advect the level set functions in the VIIM.

7.1.2 Solving the Navier-Stokes equations

The Navier-Stokes solver used in the foam model is similar to the method developed for multiphase fluid flow problems in §4.2. Except for the fact that the interface evolution in the VIIM is first order, the Navier-Stokes solver implemented here is second order (in space and time). Specifically, a second order approximate projection method is used, together with a Godunov scheme for the advection term, based on the work developed in [115] . These improvements, compared to the solver used in Chapter 4, were made in order to better treat the low-viscosity gas dynamics. In more detail, the second order approximate projection method finds an intermediate velocity $\mathbf{u}^*$ such that

$$\rho_g\Big(\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u}\cdot\nabla\mathbf{u})^{n+\frac{1}{2}}\Big) = -\nabla p^{n-\frac{1}{2}} + \mu_g\frac{\Delta\mathbf{u}^* + \Delta\mathbf{u}^n}{2} + \mathbf{st}_\varsigma. \tag{7.1}$$

Here, the advection term $(\mathbf{u}\cdot\nabla\mathbf{u})^{n+\frac{1}{2}}$ is implemented with a second order unsplit Godunov scheme (see [115]), while the diffusion term is implemented with a standard Crank-Nicholson scheme. Surface tension is computed as a body force with a regularised Dirac delta function, such that

$$\mathbf{st}(x) = \sigma\sum_i\int_{\Gamma_i}(\kappa\mathbf{n})(y)\delta_\varsigma(x-y)\,dS,$$

using the method described in §4.2. This method of calculating surface tension is physically consistent, since in the case of a dry foam, each bubble is separated from the others by a thin membrane. Moreover, the method automatically enforces Plateau's laws, i.e. lamellae make 120° angles at Plateau borders and quadruple points have symmetric angle conditions. To complete the projection step, once the intermediate velocity in (7.1) is found, the velocity at the next time step is determined by performing a projection

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathcal{P}\Big(\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t}\Big). \tag{7.2}$$

Here, $\mathcal{P}$ is a projection operator which takes a velocity field and projects it onto the space of divergence free velocity fields, and in so doing, determines the pressure $p^{n+\frac{1}{2}}$ at the next time interval. The projection operator is essentially equivalent to the projection method described in §4.2; for more details, including a discussion on the merits of projecting $\mathbf{u}_t$ (as is done in (7.2)) rather than projecting $\mathbf{u}^*$, the reader is referred to [115] and the references therein.

7.1.3 Local conservation law

Recall the local conservation equation for evolving the lamella film thickness η, which states that

$$\frac{d}{dt}\int_{S(t)} \eta \, dS = 0 \tag{7.3}$$

for any surface patch $S(t)$ on a lamella Γ that is passively advected by the velocity field $\mathbf{u}$. One approach for solving this conservation law is developed in [116, 117], wherein η is extended off the interface Γ and the conservation law is rewritten in strong form as a PDE for the extension function satisfying[1]

$$\eta_t + \mathbf{u}\cdot\nabla\eta = \Big(\mathbf{n}\cdot\frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \nabla\cdot\mathbf{u}\Big)\eta. \tag{7.4}$$

Here, $\mathbf{n}$ is the normal vector field for the interface Γ, while the term in brackets measures the local stretching and compression of the interface in the directions tangential to the surface. In [116, 117], this PDE is solved via standard techniques on a background grid, leading to an Eulerian approach that offers certain advantages especially when the interface Γ is a sufficiently smooth closed surface. However, in this application, we have an interconnected network of lamellae with individually-defined η functions, possibly being created and destroyed as the foam undergoes topological change. In this setting, a Lagrangian particle-based approach offers some advantages, and has been adopted here.

The Lagrangian approach is based on a simple method of characteristics corresponding to the PDE (7.4). Consider a specific lamella with surface Γ. On Γ, a set of particles is uniformly seeded with positions x_i, $i = 1, \ldots, N$, and each particle carries a thickness η_i. A specific particle's coordinates

[1] The formulas derived and presented in [116] do not appear as succinctly as written here, but it is a simple exercise to show they are equivalent; see also the derivation presented in [118].

and thickness values are evolved in time by solving the ODE system

$$\begin{aligned}\tfrac{d}{dt}x_i &= \mathbf{u},\\ \tfrac{d}{dt}\eta_i &= \Big(\mathbf{n}\cdot\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - \nabla\cdot\mathbf{u}\Big)\eta_i.\end{aligned}$$

In fact, one can simplify this further by noting that $\nabla\cdot\mathbf{u} = 0$, due to the incompressibility constraint. To solve this ODE for each particle, a simple forward Euler scheme together with trilinear interpolation of the velocity field is used, as follows:

$$\begin{aligned}\frac{x_i^{n+1}-x_i^n}{\Delta t} &= \mathbf{u}^n(x_i^n)\\ \frac{\eta_i^{n+1}-\eta_i^n}{\Delta t} &= \hat{n}\cdot\frac{\mathbf{u}^n\big(x_i^n+h\hat{n}\big)-\mathbf{u}^n\big(x_i^n-h\hat{n}\big)}{2h}.\end{aligned}$$

Here, $\mathbf{u}^n$ is evaluated from a trilinear interpolation of the values defined on the grid, and the normal of the interface at x_i^n, denoted by $\hat{n}$ in the above, is evaluated by a finite difference approximation based on the level set functions used in the VIIM.

A similar algorithm is used to track the thickness of the Plateau borders. In this case, the local conservation law states that

$$\frac{d}{dt}\int_{L(t)}\lambda\,ds = 0$$

for any line segment $L(t)$ on the Plateau border that is passively advected by the velocity field. Particles on the Plateau border are seeded with positions x_i and carry thickness values λ_i. By considering infinitesimal line segments, it is straightforward to derive the corresponding differential equations for the particles, which are given by

$$\begin{aligned}\tfrac{d}{dt}x_i &= \mathbf{u},\\ \tfrac{d}{dt}\lambda_i &= -\Big(\tau\cdot\frac{\partial\mathbf{u}}{\partial\tau}\Big)\lambda_i.\end{aligned}$$

Here, $\tau = \tau(x_i)$ is a unit tangent vector to the Plateau border, and in analogy with the case of the lamella, the term in parentheses measures the local stretching and compression of the Plateau border. A similar forward Euler and finite difference scheme is used to update the λ thickness values for particles on the Plateau border.

One side effect of this otherwise simple Lagrangian approach is that, over time, the particles on the surface of the lamellae can easily become dispersed in areas and highly concentrated in others. It therefore becomes necessary to intermittently reseed the particles by creating and destroying particles with the goal of establishing a more uniform concentration. To reseed particles, and also to seed particles at the beginning of the simulation, the entire multiphase interface is extracted as a triangulated mesh, using the algorithm described in §3.8. This produces a collection of triangles for each lamella and a set of line segments for each Plateau border. One η particle is created for each mesh triangle and one λ particle is created for each Plateau border mesh segment. When particles

are created, their initial thickness values are determined by interpolating the thickness values of old particles – in other words, a scattered data interpolation problem must be solved. To do this, a simple algorithm has been designed, based on the idea of finding a suitable set of particles which are as close as possible to the interpolation point x and which can be used to interpolate with. In particular, given a point x, the closest three points to x are found that have the property that the projection of x onto the triangle formed by the three points is in the interior of the triangle. The method is given in detail in Algorithm 7.

Algorithm 7 Interpolation of scattered particles

Suppose $\{x_i\}_{i=1}^N$ is a collection of distinct points located on a sufficiently smooth surface, together with function values $\{f_i\}_{i=1}^N$. Given an interpolation point x on or near the same surface:

1: Order the points by proximity to x, such that $d(x_1, x) \leq d(x_2, x) \leq d(x_3, x) \leq \cdots \leq d(x_N, x)$.
2: **for** $i = 2, \ldots, N$ **do for** $j = 1, \ldots, i-1$ **do for** $k = 0, \ldots, j-1$ **do**
3: Let x_p be the projection of x onto the plane formed by points x_i, x_j, and x_k.
4: **if** x_p is inside the triangle formed by x_i, x_j and x_k **then**
5: Determine the unique function p such that p is a linear polynomial on the plane formed by x_i, x_j, x_k and which interpolates f_ℓ at x_ℓ (for $\ell = i, j, k$).
6: Return the value of p at the projection point x_p and exit.
7: If no appropriate interpolation triangle was found, return f_1.

This algorithm provides a general purpose method of accurately reseeding the Lagrangian particles used to track the lamellae and Plateau border film thickness functions. Some comments are necessary:

- For the general particle reseeding problem, the existing particles can be binned into grid cells, and this information can be used to make the search and proximity ordering part of the interpolation algorithm very efficient. In particular, in line 1 above, only the particles located within one grid cell on either side of the interpolation point x are considered. Even with such few particles in consideration, it was found that in practice, line 7 was rarely executed. In other words, a successful triad of particles (x_i, x_j, x_k) (in close proximity to x and on which the projection of x is in the interior) was almost always found.
- For the case of the Plateau border, a similar algorithm is used: the closest pair of points (x_i, x_j) is found with the property that the projection of x onto the line formed by (x_i, x_j) is inside the segment, and then linear interpolation of the two (f_i, f_j) values are used.
- If the function f being interpolated is smooth, the interpolation algorithm is second order accurate in h, where h is the approximate particle separation.

As mentioned above, reseeding the Lagrangian particles must be performed intermittently in order to ensure they do not become too dispersed. In principle, this only has to be done a fixed number of times in the course of a simulation, even as $\Delta t \to 0$. In this work however, for reasons of simplicity (and in connection with topological changes, see §7.1.4), the reseeding process was executed every fixed number of time steps. Because the interpolation algorithm is second order accurate, but the advection and evolution of the particles is only first order accurate, it is expected that the error in

evolving the particles dominates the error contributed by interpolation. This is confirmed in the following convergence tests.

Convergence tests

Here, some tests are performed to confirm the numerical properties of the above Lagrangian particle-based scheme. Consider a sphere with radius $r = \frac{1}{4}$, located at the origin in the domain $\Omega = [-\frac{1}{2}, \frac{1}{2}]^3$, initialised with a uniform thickness of $\eta \equiv 1$. It is advected by the velocity field $\mathbf{u} = (u, v, w)$ given by

$$\begin{aligned} u(x, y, z, t) &:= \sin \pi x \cos 2\pi t, \\ v(x, y, z, t) &:= -\tfrac{1}{3} \tfrac{\partial u}{\partial x} y, \\ w(x, y, z, t) &:= -\tfrac{2}{3} \tfrac{\partial u}{\partial x} z. \end{aligned} \tag{7.5}$$

This velocity field is designed to be a nontrivial velocity field satisfying $\nabla \cdot \mathbf{u} = 0$, and for which pathlines can be computed directly with a closed-form expression. Specifically, a particle with initial position (x_0, y_0, z_0) has a pathline $\mathbf{x}(t) = (x(t), y(t), z(t))$ satisfying $\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), t)$, and this has the solution

$$\begin{aligned} x(t) &= \tfrac{2}{\pi} \arctan\Big(e^{\frac{1}{2}\sin 2\pi t} \tan \tfrac{\pi x_0}{2}\Big), \\ y(t) &= y_0 \left(e^{-\frac{1}{2}\sin 2\pi t}(1 + e^{\sin 2\pi t} \tan^2 \tfrac{\pi x_0}{2})\Big(1 + \tan^2 \tfrac{\pi x_0}{2}\Big)^{-1}\right)^{1/3}, \\ z(t) &= z_0 \left(e^{-\frac{1}{2}\sin 2\pi t}(1 + e^{\sin 2\pi t} \tan^2 \tfrac{\pi x_0}{2})\Big(1 + \tan^2 \tfrac{\pi x_0}{2}\Big)^{-1}\right)^{2/3}. \end{aligned}$$

Over the time interval $0 \le t \le 1$, the sphere stretches and compresses, returning to its original shape at $t = \frac{1}{2}$ and $t = 1$. Let $P(t) : \Omega \to \Omega$ denote the pathline mapping, i.e. $P(t)(x_0, y_0, z_0) = (x(t), y(t), z(t))$. Then $P(t)$ is invertible and its inverse is simply $P(-t)$. Given a point x on the interface at time t, the exact thickness value can be computed with the formula

$$\eta(x, t) = \lim_{\epsilon \to 0} \frac{|(\epsilon\tau_1) \times (\epsilon\tau_2)|}{|(P(t)(y + \epsilon\tau_1) - x) \times (P(t)(y + \epsilon\tau_2) - x)|}, \quad \text{where } y = P(-t)(x), \tag{7.6}$$

where τ_1 and τ_2 are orthogonal tangent vectors to the sphere at the point y. Equation (7.6) is derived directly from (7.3), and states that an infinitesimal surface element on the sphere at time $t = 0$ is mapped to a different infinitesimal surface element under $P(t)$, and the ratio of their areas gives the change in thickness of η. Formula (7.6) could further be simplified by computing the limit in terms of the derivatives of P, however it is sufficient to evaluate the right hand side with a fixed and small value of ϵ (in effect, performing numerical differentiation); in the following, a value of $\epsilon = 10^{-6}$ is used, which is enough to determine the solution with approximately six digits of accuracy. In Figure 7.1, the evolution of the sphere and the resulting thickness function is shown.

Using the VIIM to advect the sphere by the velocity field (7.5) (forward Euler in time and a second order ENO algorithm for the advection term), the thickness function η is evolved by using the particle scheme described above. On a grid of $n \times n \times n$ cells (corresponding to $h = 1/n$), the

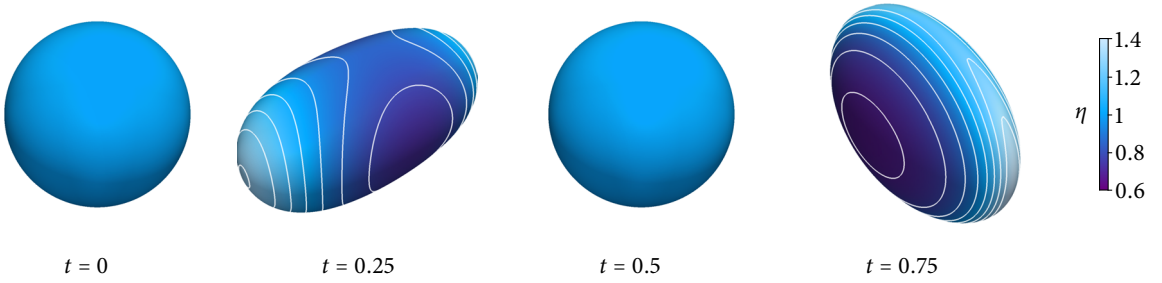

Figure 7.1. Evolution of the lamella thickness function for a sphere advected by the velocity field defined in (7.5). Solution computed on a $128 \times 128 \times 128$ grid.

seeding algorithm described above is used to generate $N = \mathcal{O}(n^2)$ particles $\{x_i\}_{i=1}^N$ on the sphere, initialised with thickness $\eta_i = 1$. Two measures of the total error, based on an L^2 and L^∞ norm in space, are defined with

$$e_2 = \max_{0\le t\le \frac{3}{4}} \left(\frac{1}{N} \sum_{i=1}^{N} |\eta_i - \eta(x_i, t)|^2 \right)^{1/2}, \quad e_\infty = \max_{0\le t\le \frac{3}{4}} \max_{1\le i\le N} |\eta_i - \eta(x_i, t)|.$$

In Table 7.1, the results are presented for two time stepping schemes. In the first, $\Delta t = h/6$ and the particles are reseeded every 16 time steps. In the second, $\Delta t = \frac{16}{3}h^2$ and there is no reseeding. The results in Table 7.1 agree with the expectation that the scheme is first order accurate in time and second order accurate in space. The reseeding/reconstruction procedure itself is second order in space, and this does not interfere with the overall error provided the total accumulated error from reconstruction is less than other accumulated truncation errors.

7.1.4 Topological changes

During the rearrangement phase, topological changes often occur, whereby bubbles change neighbourship with other bubbles. When such a change takes place, individual lamellae and Plateau borders disappear, and new interfaces are born after the topological change. With regards to tracking the interface as a whole (i.e. the entire network of lamellae), these topological changes are automatically handled by the VIIM framework. However, the particle based approach for tracking film thicknesses needs more explicit treatment, and this is discussed here.

	$\Delta t \propto h$				$\Delta t \propto h^2$			
h	e_∞	order	e_2	order	e_∞	order	e_2	order
1/32	0.03928	–	0.01991	–	0.01964	–	0.01181	–
1/64	0.02060	0.9	0.01182	0.8	0.00524	1.9	0.00286	2.0
1/128	0.01067	0.9	0.00651	0.9	0.00158	1.7	0.00071	2.0
1/256	0.005820	0.9	0.00341	0.9	0.00044	1.8	0.00018	2.0

Table 7.1. Corresponding to the evolution shown in Figure 7.1, convergence results for the thickness function η for different grid sizes.

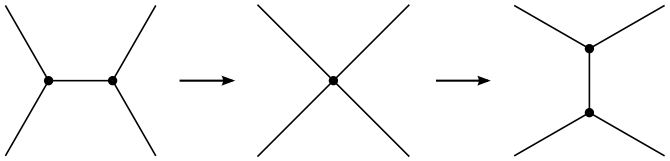

Figure 7.2. A T1 topological change in two dimensions. An initial configuration (left) evolves into an unstable configuration with a quadruple point (middle), soon after splitting and forming a new film (right).

Consider a two-dimensional case, more specifically, a T1 topological change – see Figure 7.2. Here, the horizontal interface in the middle shrinks and disappears, and a new vertical interface is born. At the same time, the two triple points in Figure 7.2 (left) come closer, merge into a single quadruple point, and split apart into two new triple points in Figure 7.2 (right). It is difficult to precisely track the film thicknesses of the fluid in these interfaces throughout this process, primarily because a topological change involves a complicated interaction of macroscopic and microscopic fluid dynamics. For example, Durand and Stone [97] have examined what role various physical parameters of the fluid play (such as surface viscoelasticity) in the time it takes for a T1 event to occur. The situation in three dimensions is likely to be even more involved.

Instead of resorting to a fully three-dimensional simulation, operating under the extreme scales of the films, a simple procedure is used to provide a mechanism for redistributing liquid mass. After a topological change occurs, liquid mass from destroyed lamellae and Plateau borders is equally divided amongst the newly created lamellae and Plateau borders. This method is based on conservation of liquid mass, and the idea that during the topological change, Plateau border and lamella liquid dynamics occur on the same spatial scale and contribute equally to the destruction and creation of interfaces. In the numerical algorithm, this reassignment of liquid mass takes place when the Lagrangian particles are reseeded: the mass associated with particles for destroyed interfaces is collected into a reserve. This reserve is evenly distributed amongst new interfaces to create particles with thickness values based on the measure of the interfaces. Importantly, this reassignment of mass occurs only locally, so that if multiple topological changes have occurred since the last reseed event, mass is not unphysically transferred to other parts of the system. To ensure this redistribution of mass is as localised an event as possible, it is necessary to detect for such topological changes frequently, and in this work, this is done every 20–60 time steps of the Navier-Stokes solver.

7.1.5 Testing for macroscopic equilibrium

The Navier-Stokes equations, interface evolution, and local conservation laws are solved until the collection of bubbles reaches a macroscopic equilibrium. A simple method is employed to determine when equilibrium is reached: snapshots of the interface configuration are taken at regular time intervals, and if the "difference" between one snapshot and the next falls below some threshold, then the interface is declared to be stationary. This difference is calculated by simply counting the number of grid points for which the indicator function χ used in the VIIM is different. Suitably weighted by the surface area of each phase, this is a simple estimate of the volume occupied by the exclusive-or boolean operation applied to the current and past interface configuration. The snapshot interval

must be long enough to eliminate false positives, while the tolerance must be chosen so that grid-size oscillations (occurring even when the system as a whole is stationary) are not scrutinised too heavily. The interval and threshold can easily be determined empirically, although this could be automated by using the characteristic time of rearrangement (often dictated by the Navier-Stokes time step CFL condition) and a threshold based on distances in units of grid points. Despite the simplicity of the method, investigations have shown that it works well and robustly detects equilibrium.

7.1.6 Implementation and parallelisation

All of the above has been parallelised using a simple domain decomposition approach, wherein the background rectangular Cartesian grid is subdivided into smaller grids that are assigned to individual processor in an MPI implementation. Synchronisation of grid-based data on the subdomains is performed using ghost layers of sufficient size, while the Lagrangian particles are assigned ownership to individual processors, and their ownership is transferred whenever particles cross processor boundaries. The Crank-Nicholson step in the Navier-Stokes solver has been implemented with a simple parallelised Conjugate Gradient algorithm, while the pressure Poisson problem (for the projection step) is solved with a parallelised multigrid algorithm. It was found that this approach scales well up to 1000s of processors.

7.2 Numerical methods for the drainage phase

In the drainage phase, it is assumed that the macroscopic rearrangement of bubbles has reached an equilibrium, and thus the network of membranes is stationary. The goal is to solve the coupled nonlinear system of thin-film equations for the thickness functions η and λ, until such time that a lamella becomes critically thin and ruptures. In summary, the system of equations is:

- For each lamella with surface Γ, the lamella thin-film equation is

$$\begin{gathered} \eta_t + \nabla_s \cdot \mathbf{Q} = 0 \text{ on } \Gamma, \text{ where} \\ \mathbf{Q} = \frac{1}{3\mu}\left(\sigma\eta^3\nabla_s\left((k_1^2 + k_2^2)\eta + \Delta_s\eta\right) + \rho\mathbf{g}_s\,\eta^3\right), \end{gathered} \tag{7.7}$$

  subject to the boundary conditions

$$\frac{\partial\eta}{\partial\nu} = 0 \quad \text{and} \quad \mathbf{Q}\cdot\nu = \tfrac{1}{2}Q(\eta,\lambda) \quad \text{on } \partial\Gamma, \tag{7.8}$$

  where ν is the outwards-pointing unit vector orthogonal to $\partial\Gamma$ and tangential to Γ. Here, $\mathbf{Q}$ is the vector flux of liquid in the lamella while Q is the function determining the flux where a lamella connects with Plateau border, as defined in (6.17). Thus, the value of λ in (7.8) changes along each piece and between pieces on the boundary of Γ.

- For each Plateau border, parameterised by arc-length ℓ, the evolution equation is

$$\lambda_t + \frac{C_\Delta}{\mu}\frac{\partial}{\partial \ell}\Big(-\tfrac{1}{2}(\sqrt{3}-\tfrac{\pi}{2})^{1/2}\sigma\lambda^{1/2}\partial_\ell\lambda + \lambda^2\rho g_\tau\Big) = S. \tag{7.9}$$

  Here, $C_\Delta \approx 0.0201$ is a constant associated with the cross-sectional shape of the Plateau border (see §6.5.2), and S is a source term determined by the sum of fluxes of incoming liquid of the three lamellae connected to the Plateau border, and is given by

$$S = \sum_{i=1}^{3} Q(\eta_i, \lambda), \tag{7.10}$$

  where η_i is the thickness of lamella i. Equation (7.9) is supplemented by a quadruple point boundary condition where four Plateau borders meet: let λ_i, $i = 1, \ldots, 4$, denote the four Plateau border thickness functions, then at each quadruple point,

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4, \quad \text{and} \\ \sum_{i=1}^{4} \frac{1}{2}(\sqrt{3}-\frac{\pi}{2})^{\frac{1}{2}}\sigma\lambda_i^{\frac{1}{2}}\partial_\ell\lambda_i - \lambda_i^2\rho g_{\tau_i} = 0. \end{cases} \tag{7.11}$$

  For each quadruple point, this is a nonlinear system of five equations in five unknowns (the common value of λ and the four fluxes).

In this section, a collection of numerical methods for solving this system of PDEs is developed. These methods are designed to effectively handle the high order nonlinear terms in the lamella and Plateau border thin-film equations, as well as provide a straightforward method for implementing the coupled boundary conditions.

To motivate the choice of numerical method, consider first the case of a single lamella. There are a variety of approaches for solving degenerate fourth order nonlinear equations, particularly those that describe thin-film liquid dynamics whose solutions are positivity-preserving. Generally speaking, they are split into two categories: thin-film equations on a flat Euclidean domain, and thin-film equations on a curved surface. In the case of the flat Euclidean domain, more is known about the convergence of numerical schemes, see for instance [119–121] that describe finite element, finite volume and finite difference based schemes. These schemes satisfy discrete analogues of certain energy integrals satisfied by the exact solution in the continuous setting, and these properties are used to prove convergence of the numerical method [122]. Because these equations are fourth order, special attention must be given to the time-stepping scheme, since an explicit scheme usually entails a very restrictive time step condition required for stability. In [123], semi-implicit methods and convexity-splitting schemes that alleviate this problem are discussed. In particular, convergence is proved for a semi-implicit biharmonic-modified forward time stepping scheme, based on the Laplace-modified forward time stepping scheme introduced in [124]. For a specific class of fourth-order nonlinear parabolic PDEs, it is possible to prove that this biharmonic-modified time-stepping scheme is unconditionally stable. The method also happens to be straightforward to implement, and has been adopted here to time-step the lamella thin-film equation. Different time integrations can also be obtained with dynamical systems approaches [125].

For the case of solving PDEs on a curved surface, there are two main approaches: either explicitly representing the surface or implicitly representing the surface, each with their advantages and disadvantages. In the first case, a typical method is to mesh the surface (e.g. as a set of triangles) and use the finite element (see, e.g. [126]). Once a high quality mesh is generated, this approach is straightforward to implement, but the disadvantage is that generating such a mesh can sometimes be difficult to do automatically. In the second case, the surface in $\mathbb{R}^3$ is implicitly represented as a fixed level set of a function defined in $\mathbb{R}^3$, such as the signed distance function, and the solution of the PDE (e.g. η) is extended off the surface into $\mathbb{R}^3$ (see, e.g. [116, 117, 127, 128]). This implicit approach transforms the two-dimensional PDE on the surface to a full three-dimensional PDE, so that methods based on regular Cartesian grids can be used, such as finite differences. However, one must carefully consider how to solve the higher-dimensional PDE and assess the need for (possibly frequent) reinitialisation/re-extension of the embedding function. Another influencing factor is that the implicit approach is easiest when the surface is smooth and closed, whereas the explicit approach easily allows for surfaces with boundaries.

In the case of multiply connected lamellae and Plateau borders, a finite element based method appears to be the most natural approach. Once a high quality mesh is generated, flux boundary conditions for the lamellae and Plateau borders can be incorporated more easily. However, the coupling imposed by the boundary conditions must be considered. One approach is to implement the flux boundary condition implicitly, but this requires solving a nonlinear system of equations for each lamella and Plateau border at every time step. Instead, it was found that implementing the flux boundary condition in an essentially explicit fashion, at the beginning of each time step, leads to a much simpler implementation. Despite the mixture of semi-implicit methods (for the lamellae and Plateau borders) and explicit methods (for the boundary conditions), it was found that the overall scheme was in fact quite stable, and allows for relatively large time steps.

The overall method is as follows. Given the mesh, discretised film thicknesses for η and λ are defined on mesh vertices. In particular, Plateau borders, which correspond to the junctions on the mesh where three lamellae meet, have multiple functions defined at the corresponding mesh vertices: at least three η function values (for the three lamellae) and at least one λ value. At a quadruple point, there are six lamellae η function values and four λ functions defined. These collocated function values share information via the flux boundary conditions and quadruple point boundary conditions. With this general picture in mind, the main algorithm for the drainage phase is:

1. Generate a triangular mesh of the set of interfaces (§7.2.1).
2. Interpolate λ and η from the rearrangement phase to initialise λ and η at mesh vertices at time step zero.
3. Enter time step loop: for $n = 0, 1, \ldots$
   (i) Solve the Plateau border boundary condition at each quadruple point to obtain Neumann boundary conditions for each Plateau border (§7.2.2).
   (ii) Calculate the flux Q, used as a boundary condition for the lamellae and as a source for the Plateau borders (§7.2.3).

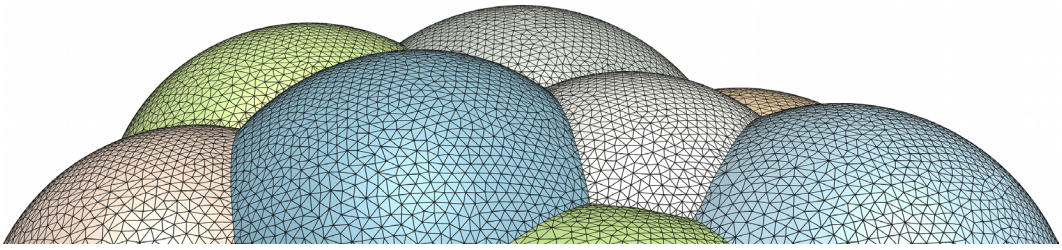

Figure 7.3. An example of a triangular mesh generated automatically from the network of connected lamellae. Individual lamellae are coloured differently and triangles meet consistently at junctions to form segmented line curves at Plateau borders.

(iii) Solve for η^{n+1} and λ^{n+1} for each lamella and Plateau border (§7.2.4 and §7.2.5).

(iv) If a lamella becomes too thin, simulate rupture (§7.3) and terminate loop, otherwise continue time stepping.

4. With the final thickness functions of the remaining lamella and Plateau borders, reinitialise the rearrangement phase (§7.3).

In the next set of sections, more details are given for these components and how they couple with each other. Some remarks are then given on implementation and parallelisation, before turning to some convergence tests which confirm the accuracy of the proposed algorithms.

7.2.1 Mesh generation

To automatically generate a mesh of the set of lamellae, the algorithm in Chapter 5 is used. The result is a high-quality mesh such that triangles meeting at junctions (i.e. the Plateau borders) do so by sharing common edges. In other words, each Plateau border is a set of connected line segments, where each line segment is shared with three triangles from different lamellae. Figure 7.3 illustrates a typical mesh obtained by this algorithm. In practice, it was observed that the generated meshes were always of good quality, with reasonable lower and upper bounds on the angles of the triangles.

7.2.2 Plateau border boundary condition

The boundary conditions at quadruple points (7.11) are a set of nonlinear simultaneous equations combining Dirichlet and Neumann boundary condition types. This boundary condition has been implemented in an explicit fashion: viewed as a type of domain decomposition, the approach decouples the Plateau borders for a short amount of time at each time step. It was found that it is possible to maintain first order accuracy (in time), as well as conservation of mass, by combining both Dirichlet and Neumann boundary conditions in an alternating fashion, as follows.

Consider a specific quadruple point. On the mesh, the quadruple point is a single vertex which has emanating from it four Plateau borders, which are four curves in space made up of line segments; let $i = 1, 2, 3, 4$ index these Plateau borders. Let $\lambda_{i,0}$ denote the value of the thickness function for

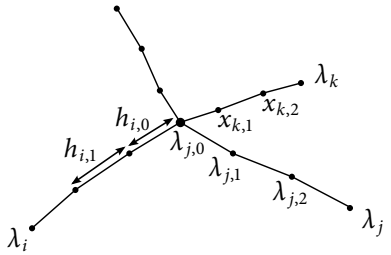

Figure 7.4. Schematic of a quadruple point and the four Plateau borders emanating from it. Here, $h_{i,0}$ and $h_{i,1}$ are edge lengths and $\lambda_{i,j}$ are thickness values at mesh vertices $x_{i,j}$.

Plateau border i at the quadruple point, let $\lambda_{i,1}$ denote its value at the position at precisely one edge length of size $h_{i,0}$ away from the quadruple point, and let $\lambda_{i,2}$ denote its value two edge lengths away, the second edge having length $h_{i,1}$; see Figure 7.4. Also let $x_{i,j}$ denote the position of the vertices, so that $x_{1,0} = \cdots = x_{4,0} =: x_0$. The strategy is to:

1. Use the values of $\lambda_{i,1}$ and $\lambda_{i,2}$, $i = 1, \ldots, 4$, together with the quadruple point boundary condition, to obtain a common value of λ_0 at the quadruple point;
2. Use λ_0 to calculate fluxes λ_i', used as Neumann boundary conditions for the Plateau border time stepping scheme;
3. Time step each Plateau border according to the finite element method described below (§7.2.5) – in general, due to the decoupling, this yields different values of $\lambda_{i,0}$ at the quadruple point at the next time step;
4. Project these values onto a common value, in such a way that the total mass is conserved.

In essence, this strategy uses the quadruple point boundary condition to obtain flux boundary conditions for each Plateau border in such a way to conserve the total mass in the system, at the expense of obtaining discontinuities in λ at each quadruple point after each time step. At the end of each time step, this discrepancy is resolved via a projection procedure that conserves mass. Altogether, the procedure leads to a first order accurate time stepping scheme that conserves the total mass of liquid in the network of Plateau borders.

In more detail, by approximating the derivatives in (7.11) with second order finite differences, the quadruple point boundary condition, with common value λ_0, leads to

$$\sum_{i=1}^{4} \tfrac{1}{2}\big(\sqrt{3} - \tfrac{\pi}{2}\big)^{\frac{1}{2}} \sigma \lambda_0^{\frac{1}{2}} \left[\frac{\lambda_{i,1} - \lambda_0}{h_{i,0}} + \frac{h_{i,1}(\lambda_{i,1} - \lambda_0) + h_{i,0}(\lambda_{i,1} - \lambda_{i,2})}{h_{i,1}(h_{i,0} + h_{i,1})} \right] - \lambda_0^2 \rho g_i = 0. \tag{7.12}$$

Here, g_i is a second order approximation of the tangential component of gravity evaluated at the quadruple point for Plateau border i, computed as[2]

$$g_i = g\hat{\mathbf{g}} \cdot \Big(\frac{x_{i,1} - x_0}{h_{i,0}} + \frac{h_{i,1}(x_{i,1} - x_{i,0}) + h_{i,0}(x_{i,1} - x_{i,2})}{h_{i,1}(h_{i,0} + h_{i,1})} \Big).$$

[2]The equation for g_i may be viewed as calculating $\hat{\mathbf{g}} \cdot \tau = \hat{\mathbf{g}} \cdot \frac{d}{d\ell}\mathbf{x}(\ell)$ where ℓ is the arc length and $\mathbf{x}(\ell)$ parameterises the Plateau border.

Equation (7.12) reduces to a cubic equation, as follows. We seek a physical solution and so may assume that $\lambda_0 > 0$. Letting $z := \sqrt{\lambda_0}$, the summation in (7.12) leads to the equation

$$C(a - bz^2) - dz^3 = 0, \tag{7.13}$$

where

$$\begin{aligned} C &= \tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}}\sigma, \\ a &= \sum_{i=1}^{4} \frac{\lambda_{i,1}}{h_{i,0}} + \frac{h_{i,1}\lambda_{i,1} + h_{i,0}(\lambda_{i,1} - \lambda_{i,2})}{h_{i,1}(h_{i,0} + h_{i,1})}, \\ b &= \sum_{i=1}^{4} \frac{1}{h_{i,0}} + \frac{1}{h_{i,0} + h_{i,1}}, \\ d &= \sum_{i=1}^{4} \rho g_i. \end{aligned}$$

If $d = 0$, there is exactly one positive solution of (7.13). If $d \neq 0$, it can be shown that if $h_{i,j} = \mathcal{O}(h)$ and h is sufficiently small, then there is exactly one positive solution of the same order as $\lambda_{i,1}^{1/2}$. In practice, it was observed that an appropriate solution of the cubic equation could always be found, afterwhich the sought after solution to the quadruple point boundary condition is simply $\lambda_0 = z^2$.

Equipped with this value of λ_0, flux values for each Plateau border can be calculated as the summands in (7.12):

$$W_i := \tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}}\sigma\lambda_0^{\frac{1}{2}}\left[\frac{\lambda_{i,1} - \lambda_0}{h_{i,0}} + \frac{h_{i,1}(\lambda_{i,1} - \lambda_0) + h_{i,0}(\lambda_{i,1} - \lambda_{i,2})}{h_{i,1}(h_{i,0} + h_{i,1})}\right] - \lambda_0^2\rho g_i. \tag{7.14}$$

These values are used in the finite element method for advancing in time each Plateau border (see below, §7.2.5), and since they satisfy $\sum_i W_i = 0$, it follows that the (discrete) mass in the network of Plateau borders is conserved (ignoring the source term S). After one time step of the finite element method for each Plateau border, four not necessarily equal thickness values $\tilde{\lambda}_{i,0}^{n+1}$ are obtained at the quadruple point at time step $n + 1$. They are replaced via a simple projection:

$$\lambda_{i,0}^{n+1} = \frac{\sum_{j=1}^{4} \tilde{\lambda}_{j,0}^{n+1} h_{j,0}}{\sum_{j=1}^{4} h_{j,0}}.$$

This projection conserves mass since $\sum_i \lambda_{i,0}^{n+1} h_{i,0} = \sum_i \tilde{\lambda}_{i,0}^{n+1} h_{i,0}$.

7.2.3 Flux boundary condition

The flux boundary condition is calculated at the beginning of each time step, and used as a boundary condition for the lamella thin-film equation, and as a source term for the Plateau border thin-film equation. Consider a specific Plateau border, and let $i = 1, 2, 3$ index the three lamellae connected to the Plateau border. At a mesh vertex x_j belonging to the Plateau border, let $\eta_{i,j}$ and λ_j be the value

of the thickness functions at that particular mesh vertex. Given these film thicknesses, a per-lamella flux value is calculated with

$$Q_{i,j} = \tfrac{1}{2}\left(\sqrt{3} - \tfrac{\pi}{2}\right)^{\frac{3}{4}} \frac{\sigma \eta_{i,j}^{5/2}}{\mu \lambda_j^{3/4}}. \tag{7.15}$$

The sum of the three flux values defines the source term for the Plateau border:

$$S_j = \sum_{i=1}^{3} Q_{i,j}. \tag{7.16}$$

Once calculated, the three flux values $Q_{i,j}$, $i = 1, 2, 3$, and the source term S_j are used in the finite element methods below.

7.2.4 Lamella thin-film equation

Let Γ denote the surface of a specific lamella. To describe the finite element method for solving the thin-film equation, the time discretisation is first considered, followed by defining a suitable variational form of the governing PDE and its discrete counterpart.

Time discretisation

Since (7.7) is a fourth order, nonlinear parabolic PDE, any explicit time stepping scheme, such as forward Euler, is likely to have a severe time step constraint required for stability, such as $\Delta t = \mathcal{O}(h^4)$. This would make the computation prohibitively expensive, and so implicit schemes are necessary. The backward Euler method is not an appealing option either, since it leads to a fully nonlinear system of equations to solve, the solution of which requires good initial guesses and/or many iterations, and this again essentially requires small time steps. One might then consider a mixture, and apply an implicit method only to the highest order term (i.e. $\nabla_s \cdot ((\eta^n)^3 \nabla_s \Delta_s \eta^{n+1})$) and keep all other terms explicit, including the nonlinear terms. This method leads to a linear symmetric positive definite system to invert at each time step, the matrix of which is a function of the nonlinearity. However, in practice, this particular method was observed to be numerically unstable for large time steps.

A solution is to use a biharmonic-modified forward time stepping scheme. The scheme is derived by applying forward Euler and adding a term of the form $\alpha \Delta_s^2(\eta^{n+1} - \eta^n)$, where α is a constant, and for (7.7), corresponds to the scheme

$$\frac{\eta^{n+1} - \eta^n}{\Delta t} + \alpha \frac{\sigma}{3\mu} \Delta_s^2(\eta^{n+1} - \eta^n) + \frac{1}{3\mu} \nabla_s \cdot \left(\sigma (\eta^n)^3 \nabla_s \left((k_1^2 + k_2^2)\eta^n + \Delta_s \eta^n \right) + \rho \mathbf{g}_s (\eta^n)^3 \right) = 0. \tag{7.17}$$

Since $\Delta_s^2(\eta^{n+1} - \eta^n)$ is $\mathcal{O}(\Delta t)$ in magnitude, the addition of this term does not alter the convergence rate of the forward Euler scheme – it will still be first order, and the error depends on the size of α. For any $\alpha > 0$, the scheme becomes implicit and leads to a symmetric positive definite system of equations to solve at each time step, the matrix of which is a discretisation of $I + \Delta t \alpha \frac{\sigma}{3\mu} \Delta_s^2$. By choosing α large enough, it is possible to prove mathematically that the resulting scheme is stable – see [123] for a proof for a similar PDE. For the scheme (7.17), α needs to bound η^3; in this work,

at each time step α is set to be $\alpha = 2\max_\Gamma \eta^3$. In addition, (7.17) is supplemented by boundary conditions as given by (7.8), such that $\nabla_s \eta^{n+1} \cdot \nu = 0$ and

$$\frac{2}{3\mu}\Big(\alpha\sigma\nabla_s\Delta_s(\eta^{n+1}-\eta^n) + \sigma(\eta^n)^3\nabla_s\big((k_1^2+k_2^2)\eta^n + \Delta_s\eta^n\big) + \rho\,\mathbf{g}_s\,(\eta^n)^3\Big)\cdot\nu = Q(\eta^n,\lambda^n) \quad \text{on} \quad \partial\Gamma,$$

where ν is the outwards-pointing unit vector orthogonal to $\partial\Gamma$ and tangential to Γ. Notice that the flux boundary condition is a mixture of implicit and explicit terms that takes the same form as the biharmonic-modified time stepping scheme; for the most part the flux boundary condition is evaluated explicitly (at the current time step) with the α modification making it semi-implicit. This form of the boundary condition is suitable for the finite element formulation used below.

Finite element method

The finite element method used here is based on the formulation and theory presented in [126] for parabolic PDEs on surfaces.

Weak form. On a smooth surface, there exist analogues of the usual integration-by-parts formulas, including

$$\int_\Gamma \nabla_s f\cdot\nabla_s g = -\int_\Gamma f\,\Delta_s g + \int_{\partial\Gamma} f\,\nabla_s g\cdot\nu,$$

and, if $\mathbf{f}$ is everywhere tangential to the surface,

$$\int_\Gamma g\,\nabla_s\cdot\mathbf{f} = -\int_\Gamma \mathbf{f}\cdot\nabla_s g + \int_{\partial\Gamma} g\,\mathbf{f}\cdot\nu.$$

By making use of the integration-by-parts formulas on the manifold, a suitable variational form of the governing PDE (7.7) is to find two functions η and p satisfying

$$\int_\Gamma \eta_t\phi = \frac{1}{3\mu}\int_\Gamma\Big(\sigma\eta^3\nabla_s\big((k_1^2+k_2^2)\eta+p\big)+\rho\mathbf{g}_s\,\eta^3\Big)\cdot\nabla_s\phi \;-\int_{\partial\Gamma}\phi\,\mathbf{Q}\cdot\nu, \text{ and}$$

$$\int_\Gamma p\,\psi = -\int_\Gamma \nabla_s\eta\cdot\nabla_s\psi,$$

for all test functions $\phi\in C([0,T];H^1(\Gamma))$, $\psi\in H^1(\Gamma)$. This formulation introduces a second function p satisfying $p=\Delta_s\eta$ in the weak sense and creates a coupled system of equations to solve for both η and p. This weak form requires less regularity on η, and this carries over to the discrete setting, since then linear finite elements can be used for both η and p.

Discrete variational form. A finite element scheme using linear elements is as follows. The smooth surface of the lamella is approximated by a triangulation $\mathcal{T}_h$ such that $\Gamma_h = \cup_{t\in\mathcal{T}_h} t$. The vertices $(x_i)_{i=1}^n$ are assumed to lie on Γ so that Γ_h is a Lipschitz continuous surface interpolating Γ. Here, h denotes the maximum circumdiameter of the triangles in $\mathcal{T}_h$. The error due to approximating the surface Γ by the triangulation Γ_h introduces an error that is no worse than second order as $h\to 0$ [126]. Let

V_h be the linear finite element space consisting of all continuous functions defined on Γ_h that are linear when restricted to an element of $\mathcal{T}_h$. Let $\{\chi_j\}$ be the standard Lagrange basis functions so that if $v \in V_h$, then $v = \sum_i v(x_i)\chi_i$.

Using this finite element, together with the forward time stepping scheme, the discrete variational form is as follows. Given $\eta^n, p^n \in V_h$ at the current time step, find $\eta^{n+1}, p^{n+1} \in V_h$ at the next time step, such that

$$\begin{aligned}\int_{\Gamma_h} \frac{\eta^{n+1}-\eta^n}{\Delta t}\phi = \alpha\frac{\sigma}{3\mu}\int_{\Gamma_h} \nabla_s(p^{n+1}-p^n)\cdot\nabla_s\phi \\ + \frac{1}{3\mu}\int_{\Gamma_h}\left[\sigma(\eta^n)^3\nabla_s\left(k\eta^n+p^n\right)+\rho\mathbf{g}_s(\eta^n)^3\right]\cdot\nabla_s\phi \\ - \frac{1}{2}\int_{\partial\Gamma_h} Q(\eta^n,\lambda^n)\phi,\end{aligned} \tag{7.18}$$

for all $\phi \in V_h$, and

$$\int_{\Gamma_h} p^{n+1}\psi = -\int_{\Gamma_h}\nabla_s\eta^{n+1}\cdot\nabla_s\psi,$$

for all $\psi \in V_h$. Here, $\mathbf{g}_s$ and Q are defined to be the piecewise linear interpolant of their values defined on mesh vertices (in the case of Q, they are given by (7.15)), while k is defined to be the piecewise linear function whose value on mesh vertices is $k_1^2+k_2^2$; see later for additional remarks. This choice of interpolation allows the integral quantities to be calculated with straightforward quadrature schemes. In particular, many of the integrals reduce to the form of $\int_T f^3 g$ where T is a triangle and f and g are linear functions on T. This in turn reduces to a function of the three values of f and g on the vertices of the triangle, a closed form expression of which can be found by, e.g., symbolic computer algebra.

The variational form leads to a linear solve for η^{n+1} in the usual way, as follows. Let η_i^{n+1} denote the value of η^{n+1} at vertex i, let M be the mass matrix defined by $M_{ij} = \int_{\Gamma_h}\chi_i\chi_j$ and let $K = \int_{\Gamma_h}\nabla\chi_i\cdot\nabla\chi_j$ be the usual stiffness matrix. Abusing notation and considering η^{n+1} and p^{n+1} as a vector in $\mathbb{R}^n$, we thus require that

$$\begin{cases} \frac{M\eta^{n+1}-M\eta^n}{\Delta t} = \alpha\frac{\sigma}{3\mu}Kp^{n+1}+f, & \text{and} \\ Mp^{n+1} = -K\eta^{n+1}, \end{cases} \tag{7.19}$$

where $f = f(\eta^n)$ contains the explicit terms, i.e.

$$f_i = -\alpha\frac{\sigma}{3\mu}\int_{\Gamma_h}\nabla_s p^n\cdot\nabla_s\phi_i + \frac{1}{3\mu}\int_{\Gamma_h}\left[\sigma(\eta^n)^3\nabla_s\left(k\eta^n+p^n\right)+\rho\mathbf{g}_s(\eta^n)^3\right]\cdot\nabla_s\phi_i - \frac{1}{2}\int_{\partial\Gamma_h}Q(\eta^n,\lambda^n)\phi_i.$$

According to (7.19), it follows that $p^{n+1} = -M^{-1}K\eta^{n+1}$, however this requires inversion of the full mass matrix. It is simpler and more efficient to make an approximation using instead the lumped mass matrix, i.e. $\tilde{M} = \mathrm{diag}(M_{ii})$, and write $p^{n+1} = -\tilde{M}^{-1}K\eta^{n+1}$. Experiments indicated this approximation did not alter the overall convergence rate of the finite element scheme. Eliminating p^{n+1} from (7.19), we obtain

$$\left(M+\alpha\Delta t\tfrac{\sigma}{3\mu}K\tilde{M}^{-1}K\right)\eta^{n+1} = M\eta^n + \Delta t f(\eta^n).$$

This is a symmetric positive definite system for the unknown η^{n+1}, whose matrix (left-multiplied by M^{-1}) approximates the operator $I + \alpha\Delta t\frac{\sigma}{3\mu}\Delta_s^2$. To solve this equation, a simple Conjugate Gradient method is used, although more sophisticated methods such as multigrid would lead to more efficient solvers.

Additional comments

The curvature function k containing the principal curvatures k_1 and k_2 in (7.18) is defined to interpolate $k_1^2 + k_2^2$ at mesh vertices. To calculate the vertex values, we must calculate the principal curvatures of the interface Γ given implicitly by the function ϕ in the VIIM. By utilising the indicator function χ in the VIIM, it is simple to construct a signed level set function ψ whose zero level set contains Γ. Using this function, calculating k can be done with simple finite differences, as follows. Note that $\mathbf{n} = \nabla\psi/|\nabla\psi|$ is a normal vector field of Γ. The gradient of $\mathbf{n}$ has eigenvalues $\{0, k_1, k_2\}$ and is related to the shape operator on Γ. The mean curvature of Γ can be calculated with $\kappa = k_1 + k_2 = \nabla\cdot\mathbf{n} = \mathrm{tr}(\nabla\mathbf{n})$. A similar relation can be used to calculate k, specifically that $k = k_1^2 + k_2^2 = \mathrm{tr}\big((\nabla\mathbf{n})^2\big)$. This quantity is calculated at grid points using standard second order finite differences, after which the grid point values are trilinearly interpolated onto the vertices of the triangulation Γ_h, thereby yielding k at mesh vertices.

Similarly, the tangential component of gravity $\mathbf{g}_s$ in (7.18) is also defined to be the piecewise linear interpolant of $\mathbf{g}_s$ defined at mesh vertices. This in turn is calculated by using the same normal vector field $\mathbf{n}$ that was used to calculate k above: specifically, using the same signed level set function ψ, the normal $\mathbf{n} = \nabla\psi/|\nabla\psi|$ is evaluated at grid points using standard second order finite differences. One can then calculate $\mathbf{g}_s = \mathbf{g} - (\mathbf{g}\cdot\mathbf{n})\mathbf{n}$ at grid points, and then use trilinear interpolation to define $\mathbf{g}_s$ on mesh vertices.

7.2.5 Plateau border thin-film equation

The numerical method for solving the Plateau border thin-film equation (7.9) is similar to that used for the lamella. An analogue of the biharmonic-modified forward time stepping scheme is adopted, except in this case, it is forward Euler modified by a term of the form $\alpha\partial_{\ell\ell}$. Provided α is large enough, this time stepping scheme is expected to be a stable, first order accurate time stepping scheme for the second order nonlinear parabolic PDE. In addition, a finite element-based spatial discretisation has been used, but since there is only one spatial dimension, this can also be viewed as a conservative finite difference approximation.

Numerical discretisation

Let Γ denote the curve of a Plateau border. In the triangulated mesh, Γ is approximated by a set of connected line segments $\mathcal{S}_h$ such that $\Gamma_h = \cup_{e\in\mathcal{S}_h} e$. The vertices $(x_i)_{i=1}^n$ are assumed to lie on Γ so that Γ_h is a Lipschitz continuous curve interpolating Γ. Let V_h be the linear finite element space consisting of all continuous functions defined on Γ_h that are linear when restricted to an element of $\mathcal{S}_h$. Let $\{\chi_j\}$ be the standard Lagrange basis functions so that if $v \in V_h$, then $v = \sum_i v(x_i)\chi_i$.

Using the analogue of a Laplace-modified forward time stepping scheme, the finite element method is as follows. Given $\lambda^n \in V_h$ at the current time step, find $\lambda^{n+1} \in V_h$ at the next time step, such that

$$\int_{\Gamma_h} \frac{\lambda^{n+1} - \lambda^n}{\Delta t}\phi = \frac{C_\Delta}{\mu}\int_{\Gamma_h}\Big[-\tfrac{1}{2}\sigma(\sqrt{3}-\tfrac{\pi}{2})^{\frac{1}{2}}\big(\sqrt{\lambda^n}\partial_\ell\lambda^n + \alpha\partial_\ell(\lambda^{n+1}-\lambda^n)\big) + (\lambda^n)^2\rho g_\tau\Big]\partial_\ell\phi + \int_{\partial\Gamma_h} W\phi + \int_{\Gamma_h} S^n\phi, \tag{7.20}$$

for all $\phi \in V_h$. Here, W is the Plateau border flux boundary condition that is derived from the quadruple point boundary condition (as given by (7.14)), while S^n is defined to be the piecewise linear interpolant of the source term defined at the vertices given by (7.16). The tangential component of gravity is calculated on a per-edge basis, such that $g_\tau = \mathbf{g}\cdot\tau$ is piecewise constant. Finally, the stabilisation factor used in the Laplace-modified forward time stepping scheme has been set to $\alpha = 2\max_{\Gamma_h}\sqrt{\lambda^n}$, evaluated at the beginning of each time step.

Similar to the case of the lamella, the discrete variational form (7.20) leads to a symmetric positive definite system for λ^{n+1}. In fact, the associated matrix can be made tridiagonal if the Lagrange basis functions are ordered by position along the Plateau border. However, despite the possibility of using an efficient tridiagonal solver, a simple Conjugate Gradient method has again been used to solve this system.

7.2.6 Implementation and parallelisation

The above finite element method is relatively straightforward to implement as it is almost identical to a one- or two-dimensional finite element problem in flat Euclidean space, except the vertices of the triangles/segments lie in $\mathbb{R}^3$. An unordered map (implementing a hash table) was used for the main data structure, and maps a globally unique vertex identifier to the λ and η values defined at that vertex. The VIIM framework provides identifiers for each type of surface, for example a Plateau border is the intersection of three different bubbles and can be identified by a 3-tuple of integers.

To parallelise the code, the inherent domain decomposition in the problem has been utilised: each lamella and Plateau border is assigned in its entirety to individual processors in an MPI implementation. To do this, the cost of solving the symmetric positive definite systems was estimated for each lamella and Plateau border (via a simple function of the number of mesh elements), and based on this, a simple load balancing algorithm was designed so that each processor had approximately the same amount of work to do, when possible. (For example, one processor might be responsible for a single, large lamella, while another processor may have a few small lamellae and several Plateau borders.) In this fashion, parallelisation has been greatly simplified since it is unnecessary to parallelise the matrix equation solvers. Furthermore, synchronisation among processors is only necessary for mesh vertices shared by multiple Plateau borders and/or lamellae. Such synchronisation is used to determine the flux boundary conditions and quadruple point boundary conditions, and can be performed with MPI's global gather operations. It follows that the scaling efficiency of the overall approach depends heavily on the number and size of the lamellae and Plateau borders, and how this is distributed to the processors. In the work presented here, this approach gave good parallel efficiency

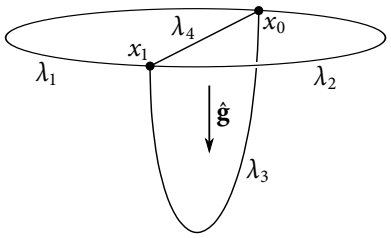

Figure 7.5. Geometry of the Plateau border convergence test, consisting of four Plateau borders meeting at two quadruple points.

in most cases. An exception is when the drainage phase is applied to relatively few bubbles, in which case many of the processors are idle. Usually, however, the computational cost of the macroscopic rearrangement phase dominates the cost of the drainage phase.

7.2.7 Convergence tests

Plateau border

Here, convergence is tested for the numerical scheme that is used to solve the Plateau border thin-film equations. A system of four Plateau borders meeting at two quadruple points is designed (as shown in Figure 7.5) with the aim to test all aspects of the numerical scheme, including coupling of solutions via the quadruple point boundary condition as well as surface tension and gravity. For each Plateau border we have the evolution equations

$$\partial_t \lambda_i + \frac{C_\Delta}{\mu} \frac{\partial}{\partial \ell} \Big(-\tfrac{1}{2} (\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}} \sigma \lambda_i^{\frac{1}{2}} \partial_\ell \lambda_i + \lambda_i^2 \rho g \hat{g}_{\tau_i} \Big) = S_i, \quad i = 1, 2, 3, 4, \tag{7.21}$$

which are coupled via the quadruple point boundary condition

$$\begin{cases} \sum_{i=1}^{4} \frac{1}{2} (\sqrt{3} - \frac{\pi}{2})^{\frac{1}{2}} \sigma \lambda_i^{\frac{1}{2}} \partial_\ell \lambda_i - \lambda_i^2 \rho g \hat{g}_{\tau_i} = 0, \\ \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4, \end{cases} \tag{7.22}$$

at the two quadruple points.

Two different tests are considered: one in which an exact solution is constructed, and another in which the solution is unknown and grid convergence is used. For the first case, an exact solution of this system is designed by substituting known expressions for λ_1, λ_2, and λ_3 into their respective evolution equations, thereby determining S_1, S_2, and S_3. The quadruple point boundary condition is then used to find a cubic polynomial in ℓ for λ_4 that has the correct value and derivative at each quadruple point, and this in turn generates S_4. Using the expressions for S_i, the numerical method is used to solve the system of PDEs, with the aim of recovering the exact solutions.

In more detail, consider the geometry shown in Figure 7.5. Here, Plateau borders number 1, 2, and 3 are circular arcs of arc length π while Plateau border number 4 is a straight line of arc length 2.

The parameterisation is such that $\ell = 0$ at quadruple point x_0, and $\ell = \pi$ or $\ell = 2$ at quadruple point x_1. Gravity points from the middle of Plateau border 4 to the middle of Plateau border 3. Let

$$\begin{aligned}\lambda_0(\ell, t) &= 2 + \cos(3\ell + t),\\ \lambda_1(\ell, t) &= 2 + \cos(3\ell + t) + 3\sin\ell,\\ \lambda_2(\ell, t) &= 2 + \cos(3\ell + t) + \tfrac{1}{2}\sin 2\ell.\end{aligned}$$

These were chosen to be nontrivial smooth solutions that are strictly positive for all time t. Using the quadruple point boundary condition (7.22), we can solve for both $\lambda_3(\ell, t)$ and $\lambda_3'(\ell, t)$ at $\ell = 0$ and $\ell = 2$. For each point in time, these four values uniquely determine a cubic polynomial in ℓ, which is chosen to be the remaining solution $\lambda_3(\ell, t)$. This yields

$$\begin{aligned}\lambda_3(\ell, t) = 2 + \cos t + x\frac{f_1(t)}{C_1\sqrt{2+\cos t}} - \tfrac{1}{2}x^2\Big(3\cos t - \frac{f_2(t)}{C_1\sqrt{2-\cos t}} + \frac{2f_1(t)}{C_1\sqrt{2+\cos t}}\Big)\\ + \tfrac{1}{4}x^3\Big(2\cos t - \frac{f_2(t)}{C_1\sqrt{2-\cos t}} + \frac{f_1(t)}{C_1\sqrt{2+\cos t}}\Big)\end{aligned}$$

where

$$\begin{aligned}C_1 &= \tfrac{1}{2}(\sqrt{3} - \tfrac{\pi}{2})^{\frac{1}{2}}\sigma,\\ f_1(t) &= \rho g(2+\cos t)^2 + C_1\sqrt{2+\cos t}(9\sin t - 4),\\ f_2(t) &= \rho g(2-\cos t)^2 + C_1\sqrt{2-\cos t}(9\sin t - 2).\end{aligned}$$

Substituting the expressions for $\lambda_1, \ldots, \lambda_4$ into (7.21) we can find closed-form expressions of $S_1, \ldots, S_4$. These are too cumbersome to repeat here, but pose no problem when evaluating numerically. Now we must choose specific values for the physical parameters, and we set $\mu = \sigma = \rho = g = 1$. These values were chosen to (i) make surface tension and gravity equally important effects for the purposes of convergence tests, and (ii) guarantees that λ_4 is strictly positive for all time t. Figure 7.6 (left pair) illustrates the exact solutions $\lambda_1, \ldots, \lambda_4$ at a two example points in time.

For the numerical method, each Plateau border is discretised with n equal sized line segments, so that $h \approx \pi/n$ for Plateau borders 1, 2, and 3, and $h = 2/n$ for Plateau border 4. The system of PDEs is solved using the method described in §7.2.5 and the method to solve the quadruple point boundary condition described in §7.2.2. The error of the numerical solution is defined as

$$e_p = \max_{i=1,2,3,4}\max_{t\in[0,1]} \|\lambda_i^h - \lambda_i\|_{L^p},$$

i.e., the maximum error over all (discrete) points in time of all Plateau borders, measured in the L^p norm for $p = 2$ and $p = \infty$, where λ_i^h denotes the numerical solution. To determine the order of accuracy, two time stepping schemes are considered: (i) one in which $\Delta t = 1/n \propto h$; and (ii) $\Delta t = 32/n^2 \propto h^2$. The convergence results are shown in Table 7.2, and confirms our expectation that the scheme is first order accurate in time and second order accurate in space.

For the second convergence test, the same Plateau border geometry and numerical discretisation is used, and the case when $S_i \equiv 0$ for all time is investigated. In particular, the Plateau borders are

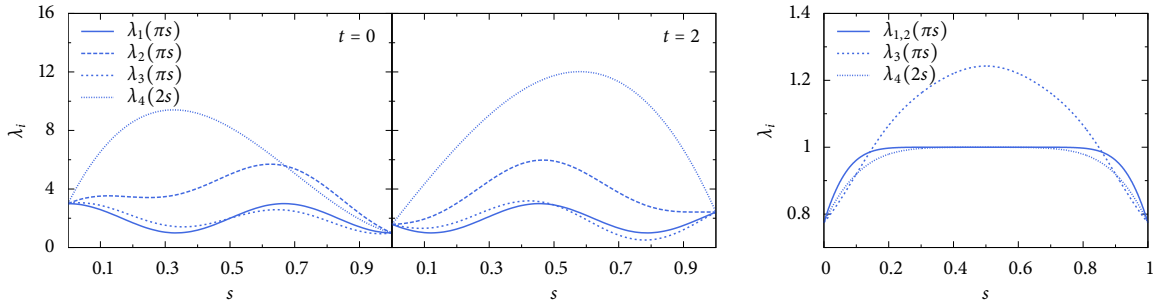

Figure 7.6. (Left pair) Exact solution for the Plateau border convergence test at times $t = 0$ and $t = 2$. (Right) Numerical solution at time $t = 10$ for the case when $S_i \equiv 0$ and $\lambda_i(t = 0) \equiv 1$ for all i.

	$\Delta t \propto h$				$\Delta t \propto h^2$			
n	e_∞	order	e_2	order	e_∞	order	e_2	order
32	0.06374	–	0.05624	–	0.06374	–	0.05624	–
64	0.03212	1.0	0.02846	1.0	0.01597	2.0	0.01415	2.0
128	0.01612	1.0	0.01433	1.0	0.00400	2.0	0.00354	2.0
256	0.00808	1.0	0.00720	1.0	0.00100	2.0	0.00089	2.0
512	0.00404	1.0	0.00361	1.0	0.00025	2.0	0.00022	2.0

Table 7.2. Results for the Plateau border convergence test using a known solution.

initialised with a uniform thickness so that $\lambda_i \equiv 1$ for all i at time $t = 0$. In this case, the system (7.21)–(7.22) conserves mass, i.e. $\frac{d}{dt}\sum_i \int_{\Gamma_i} \lambda_i = 0$. Since the exact solution is unknown, grid refinement is used to study convergence by defining

$$d_p(h) = \max_{i=1,2,3,4} \max_{t\in[0,10]} \|\lambda_i^{2h} - \lambda_i^h\|_{L^p},$$

and then estimating the convergence rate by using ratios of d_p with $\log_2\big(d(2h)/d(h)\big)$. As before, two time stepping schemes are considered: (i) $\Delta t = 5/n \propto h$ and (ii) $\Delta t = 320/n^2 \propto h^2$. The results are shown in Table 7.3, and a plot of the solution at time $t = 10$ is shown in Figure 7.6 (right). The results indicate first order accuracy in time and second order in space. It was also confirmed that mass of the discrete solution was conserved, independent of the grid size.

	$\Delta t \propto h$				$\Delta t \propto h^2$			
n	d_∞	order	d_2	order	d_∞	order	d_2	order
64	0.01136	–	0.00308	–	0.01877	–	0.00482	–
128	0.00569	1.0	0.00119	1.4	0.00697	1.4	0.00141	1.8
256	0.00170	1.7	0.00038	1.6	0.00121	2.5	0.00031	2.2
512	0.00062	1.4	0.00016	1.2	0.00027	2.2	0.00008	2.0
1024	0.00028	1.1	0.00008	1.0	0.00007	2.0	0.00002	2.0

Table 7.3. Results for the Plateau border convergence test with unknown solution, using grid refinement.

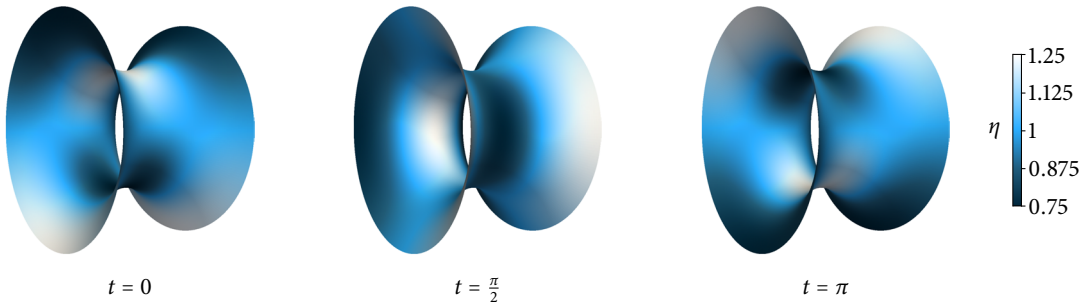

Figure 7.7. Time evolution of the exact solution for the lamella thin-film equation convergence test on a catenoid.

Lamella

Next, convergence is tested for the numerical scheme used to solve the lamella thin-film equations. Two tests are performed: one in which an exact solution is manufactured, and another in which the solution is unknown and grid convergence is used. In both cases, the lamella takes the shape of a catenoid. This is a surface of revolution with constant (in fact, zero) mean curvature, which has a nontrivial Gaussian curvature, so that $k_1^2 + k_2^2$ varies in space. The surface is parameterised by $u \in [0, 2\pi)$ and $v \in [-\frac{5}{4}, \frac{5}{4}]$ so that

$$x = \cosh v \cos u, \quad y = \cosh v \sin u, \quad z = v,$$

and is shown in Figure 7.7.

In the first test, an exact solution of the lamella thin-film equation is created by choosing $\eta = \eta(u, v) = 1 + \frac{1}{4}\sin(u + t)\cos(4\pi v/5)$ and calculating f such that

$$\eta_t + \frac{1}{3\mu}\nabla_s \cdot \left(\sigma\eta^3\nabla_s\left((k_1^2 + k_2^2)\eta + \Delta_s\eta\right) + \rho g\hat{\mathbf{g}}_s\,\eta^3\right) = f. \tag{7.23}$$

This is supplemented by the implied flux boundary condition $\mathbf{Q}\cdot\nu = Q$ that the exact solution satisfies. A closed-form expression for f and Q is most easily obtained by using computer algebra software, together with the curvilinear coordinate expressions for the surface Laplacian, etc. in the (u, v) coordinate system. The result is too involved to repeat here, but poses no problem in evaluating numerically. In the following, $\hat{\mathbf{g}} = \hat{z}$, and the parameters were chosen such that $\mu = 1$, $\sigma = 0.2$ and $\rho g = 0.5$. Altogether, this choice of exact solution η and parameters were designed to represent a solution that is relatively smooth, with competing effects of gravity and diffusion, in such a way that the numerics are nontrivial but still effectively examines convergence properties. (Different values of parameters, solutions, and geometries were also tested, and results similar to the following were obtained.)

A finite element mesh is generated for the catenoid based on a Cartesian grid dividing the interval $v \in [-\frac{5}{4}, \frac{5}{4}]$ into n equal sections and $u \in [0, 2\pi]$ into $2n$ equal sections. While the curvature term $k_1^2 + k_2^2$ can be shown to equal $2\,\mathrm{sech}^2 v$, we instead calculate this term numerically, using the procedure outlined in §7.2.4 based on a level set function whose zero level set coincides with the

		$\Delta t \propto h$				$\Delta t \propto h^2$			
n	h	e_∞	order	e_2	order	e_∞	order	e_2	order
8	0.843	0.15422	–	0.27302	–	0.15422	–	0.27302	–
16	0.445	0.06780	1.3	0.11975	1.3	0.04255	2.0	0.07190	2.1
32	0.229	0.03631	0.9	0.06287	1.0	0.01139	2.0	0.01895	2.0
64	0.117	0.01952	0.9	0.03263	1.0	0.00284	2.0	0.00482	2.0
128	0.059	0.01013	1.0	0.01663	1.0	0.00071	2.0	0.00121	2.0

Table 7.4. Results for the lamella convergence test using a known solution.

catenoid. In a similar fashion, the gravitational term $\hat{\mathbf{g}}_s$ is computed via the normal vector field, also using a level set function and finite differences. Thus, in the following, we are testing the combined effects of the finite element method and finite difference methods to calculate $k_1^2 + k_2^2$ and $\hat{\mathbf{g}}_s$. The error in the computed solution η_h is measured with

$$e_p = \max_{0 \le t \le \pi} \|\eta_h - \eta\|_{L^p}$$

for $p = 2$ and $p = \infty$. Two time stepping schemes are considered: (i) $\Delta t = \pi/4n = \mathcal{O}(h)$ and (ii) $\Delta t = 2\pi/n^2 = \mathcal{O}(h^2)$. The results are shown in Table 7.4 and agree with our expectation that the finite element scheme is first order in time and second order in space.

In the second test, the lamella thin-film equation is solved on the same catenoid, with $Q \equiv 0$, $f \equiv 0$, and the initial condition $\eta(t = 0) \equiv 1$. In this example, the direction of gravity is chosen to be $\hat{\mathbf{g}} = \hat{y}$, and all other parameters identical to the previous test. The resulting evolution of film thickness is shown in Figure 7.8 (computed using $n = 128$), and shows that the liquid drains in the direction of gravity, but also collects in regions of high curvature. Since the solution is unknown, grid refinement is used to measure convergence: defining the difference between solutions on two different grid sizes as

$$d_p = \max_{0 \le t \le 2} \|\eta_{2h} - \eta_h\|_{L^p},$$

the convergence rate can be estimated with ratios of d_p. For a time step $\Delta t = 4/n^2 = \mathcal{O}(h^2)$, the results are given in Table 7.5 and show second order convergence.

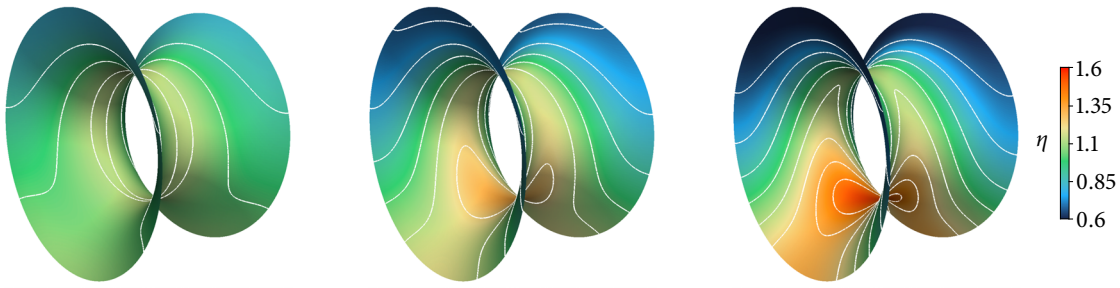

Figure 7.8. Evolution of film thickness on a catenoid with initially uniform thickness $\eta \equiv 1$ at time $t = 0$ (gravity points directly down). The white lines are contour lines of the thickness function η.

n	h	d_∞	order	d_2	order
16	0.445	0.06782	–	0.07931	–
32	0.229	0.01344	2.5	0.02018	2.1
64	0.117	0.00434	1.7	0.00665	1.7
128	0.059	0.00107	2.1	0.00161	2.1

Table 7.5. Corresponding to Figure 7.8, results for the lamella convergence test with unknown solution, using grid refinement.

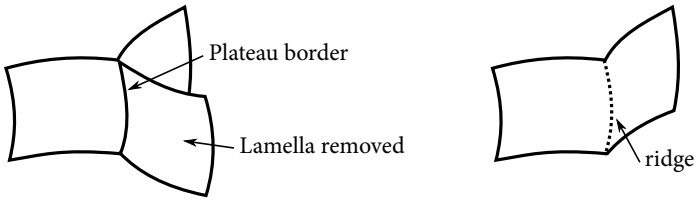

Figure 7.9. (Left) Three lamellae meet at a Plateau border. (Right) After removal of the indicated lamella, the associated Plateau border ceases to exist as well and what remains is a sharp ridge that will quickly smooth due to macroscopic effects of surface tension.

7.3 Coupling drainage and rearrangement via rupture

In the drainage phase, the coupled system of thin-film equations are evolved until the minimum lamella thickness falls below a critical threshold $\eta < \eta_{\text{threshold}}$. Rupture is then modelled as a three-step process that couples the results of the drainage phase with reinitialising the rearrangement phase. In the first step, the membrane which had minimal thickness is removed and the liquid mass it contained is uniformly distributed around the Plateau borders it was once connected to. However, once it is removed, the associated Plateau borders are no longer the intersection of three lamellae and instead become ridges in adjacent lamellae, as shown in Figure 7.9. Hence, in the second step, the liquid mass in these Plateau borders is locally "injected" into the adjacent lamellae, and this is accomplished in the same process as reinitialising the rearrangement phase. In more detail:

(i) In the first step, liquid mass in the ruptured lamella is temporarily reassigned to the Plateau borders. This is accomplished with a simple update of the appropriate Plateau border thickness functions: let Γ_L denote the lamella which has ruptured, and let $\{\Gamma_{\text{PB}_i}\}_{i=1}^N$ denote the set of Plateau borders (with thickness functions λ_i) that were on the boundary of Γ_L. An adjustment factor δ is computed with the formula

$$\delta = \frac{2\int_{\Gamma_L} \eta}{\sum_{i=1}^N \int_{\Gamma_{\text{PB}_i}} 1}$$

and is used to update $\lambda_i \leftarrow \lambda_i + \delta$. This update conserves liquid mass.

(ii) In the second step, the Lagrangian particle scheme described in §7.1.3 is reinitialised with the updated thickness functions η and λ obtained from the final step of the drainage phase. This uses an identical procedure to that used to re-seed the Lagrangian particle scheme: the mesh elements used in the finite element formulation are used as locations to seed $\{\eta_i\}$ and $\{\lambda_i\}$ particles with thickness values obtained from the finite element solver. Such particles are generated for all lamellae and Plateau borders which have not been removed by the rupture event. Fictitious $\{\lambda_i\}$ particles are also generated for the removed Plateau borders $\{\Gamma_{\text{PB}_i}\}$; however these are immediately injected onto the adjacent lamellae. This is done by imagining each of these particles to be a "ball of liquid" of finite radius, and collapsing this ball onto the adjacent lamellae in such a way that mass is conserved. In this work, these balls were chosen

to have radius h where h is the grid cell size, such that they have a radially symmetric density that decays to zero at the boundary of the ball.

(iii) Finally, to complete the coupling between the drainage phase and rearrangement phase, the VIIM is notified of the removal of a lamella by merging the two bubbles on either side of the lamella. This is done with a simple update of the indicator function χ used in the VIIM, which re-identifies the two bubbles on either side of the lamella as now being the same, now larger, bubble. The rearrangement phase can now be re-executed with new system of interconnected interfaces and film thicknesses.

This procedure of reassigning mass from the removed lamella into adjacent Plateau borders, which themselves are removed as their liquid mass is injected locally into remaining lamellae, provides a first approximation of the physical rupture mechanism. In actuality, the rupture process occurs over a very short time scale and, depending on spatial scales and film thicknesses, can lead to different behaviours, as studied in [90]. For the spatial scales and foam solutions considered in this work, it is expected that the redistribution process implemented above provides at least a qualitatively accurate account of the rupture process. However, it would be ideal to consider more carefully the effect of non-uniform rupture and how this affects redistribution of liquid mass. Such a study could use, for example, a fully three-dimensional direct numerical simulation of the incompressible Navier-Stokes equations for the liquid-gas system, and attempt to resolve the extreme scales of isolated rupturing films, at least for the short amount of time it takes for rupture to occur. This could be achieved with adaptive mesh refinement techniques, and is the subject of possible future work.

7.4 Results

In the previous set of sections, several numerical schemes have been developed for use in the rearrangement, drainage, and rupture phases of the multiscale model of foam dynamics. For each of these phases, the corresponding methods have been designed to accurately solve the underlying evolution equations, with the ability to couple to the other phases. Here, several results of the methods are presented. In the first set of results, individual components of the foam model are tested and verified, demonstrating various physical mechanisms of the model. The entire system is then coupled to study two problems exhibiting nontrivial foam dynamics that involve foam collapse via bubble rupture cascades.

In all of these results, the physical parameters have been chosen to represent a typical soap bubble foam: the density and viscosity of the liquid is similar to that of water, gravity is normal Earth gravity, and the gas phase represents typical ambient air at room temperature. The precise values of the physical parameters used in the following results are given in Table 7.6.

7.4.1 Rearrangement phase

To demonstrate rearrangement and surface area minimisation, Figure 7.10 shows the effect of removing a specific lamella from a cluster which is otherwise in macroscopic equilibrium. After removal,

Parameter and units	Notation	Fig. 7.10	Fig. 7.11	Fig. 7.13	Fig. 7.14	Fig. 7.15[a]	Fig. 7.16
Surface tension ($\mathrm{kg\,s^{-2}}$)	σ	0.034	0.034	0.034	0.034	0.034	0.034
Gas density ($\mathrm{kg\,m^{-3}}$)	ρ_g	1.15	1.15	–	1.15	1.15	1.15
Gas viscosity ($\mathrm{kg\,m^{-1}\,s^{-1}}$)	μ_g	1.8×10^{-5}	3.42×10^{-5}	–	1.8×10^{-5}	5×10^{-5}	5×10^{-5}
Liquid density ($\mathrm{kg\,m^{-3}}$)	ρ	–	–	1000	1000	1000	1000
Liquid viscosity ($\mathrm{kg\,m^{-1}\,s^{-1}}$)	μ	–	–	0.001	0.001	0.001	0.001
Gravity ($\mathrm{m\,s^{-2}}$)	g	–	–	9.8	9.8	9.8	9.8
Initial lamellae thickness (μm)	$\eta(t=0)$	–	–	5	5	10	1
Initial Plateau border cross-sectional area ($\mathrm{mm^2}$)	$\lambda(t=0)$	–	–	0.05	0.05	0.002	0.05
Typical bubble diameter (mm)		10	40	2	3	0.1 – 0.4	3
Rupture threshold (nm)		–	–	–	10	40	10

Table 7.6. Physical parameters used in the numerical simulations. [a]For the purposes of limiting bubble rearrangement to the lamellae cluster, in the simulation of Fig. 7.15, rupture of the background membrane was inhibited by including an additional diffusion term of the form $\Delta_s^2\eta$ into its thin-film equation.

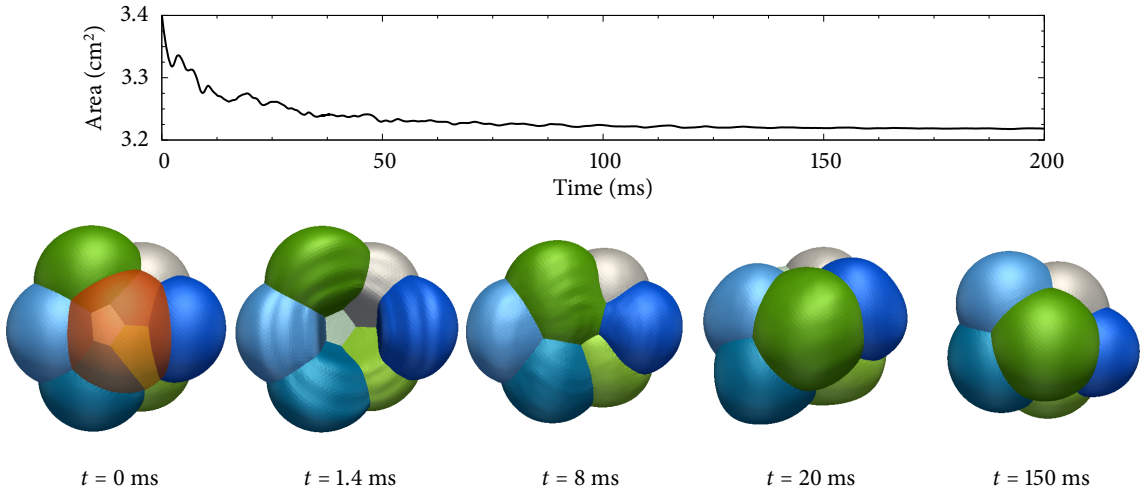

Figure 7.10. A small cluster, initially in equilibrium, undergoes rearrangement due to the removal of a lamella (orange) at time $t = 0$. (Top) Total surface area as a function of time. (Bottom) Evolution of cluster during rearrangement.

surface tension drives the cluster into a new configuration, undergoing various topological changes in the process. Figure 7.10 (top) plots the total surface area as a function of time and shows that it reaches a local minimum; the bottom figures illustrate how the "hole" made by removing the lamella is filled-in, generating capillary waves as it does so, with 120° angle conditions satisfied throughout the process, ultimately leading to an equilibrium where each lamella has constant mean curvature.

To test the accuracy of the Navier-Stokes solver, in Figure 7.11, numerical results are compared to that of a bubble oscillation experiment by Kornek *et al.* [95]. In this work, several high-speed movies were captured of bubbles colliding and merging together; once merged, the resulting larger bubbles oscillate due to effects of surface tension. One of these movies was used to determine the radii of two slightly overlapping bubbles, as shown by the $t = 0$ inset in Figure 7.11. Experimental parameters quoted in [95] were then used as parameters for the Navier-Stokes solver. In particular, the density of the gas was quoted as $1.2\,\mathrm{kg\,m^{-3}}$, however the authors were uncertain as to precisely what percentage of butane was contained in the gas mixture. It was found that better agreement between numerical results and the experiment were obtained by slightly altering the density to $1.15\,\mathrm{kg\,m^{-3}}$. Overall, the numerical results illustrated in Figure 7.11 show good qualitative agreement with the experiment. It is possible to more carefully analyse the results by measuring modes and frequencies of oscillation, as well as dampening rates, as was done in [95], however this has not been considered here.

In the next example, the effect of rearrangement on changes in film thickness is demonstrated by considering an oscillating soap bubble, as shown in Figure 7.12. A bubble with an initial shape of an ellipsoid (semi-principal axes of lengths of $\frac{20}{3}$ mm, 5 mm and 4 mm) is initialised with a uniform lamella thickness of $\eta \equiv 0.3\,\mu$m. Figure 7.12 (top) shows the evolution of the subsequent variations in film thickness: areas where the bubble has "compressed" tend to increase in thickness, while the thickness decreases near points of expansion. In Figure 7.12 (bottom), the same results are shown

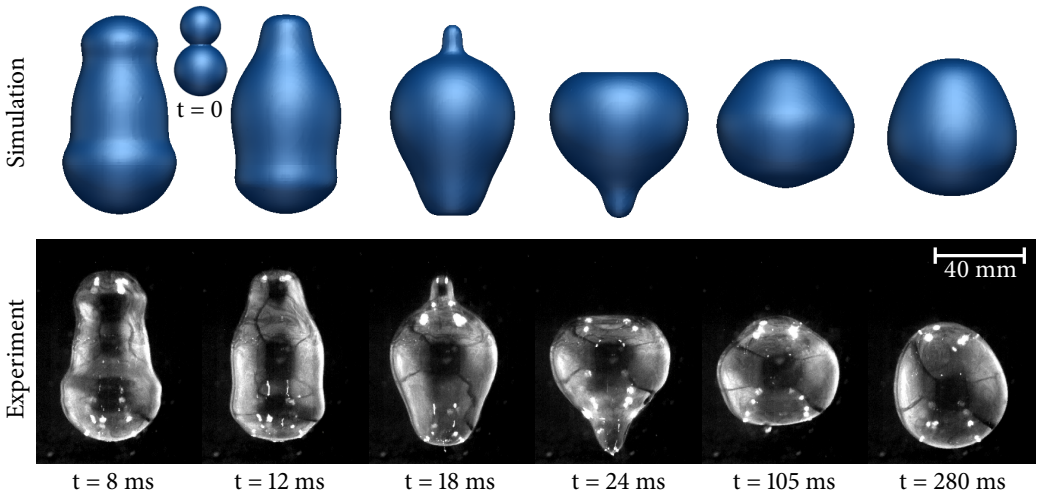

Figure 7.11. Comparison of numerical results with experiment. Two spherical soap bubbles merge at $t = 0$, subsequently causing surface tension driven oscillations that eventually lead to a larger spherical bubble. Experimental results reproduced from [95] (by permission of IOP Publishing); numerical simulation uses identical physical parameters and time scale, and was computed on a $64 \times 64 \times 96$ grid.

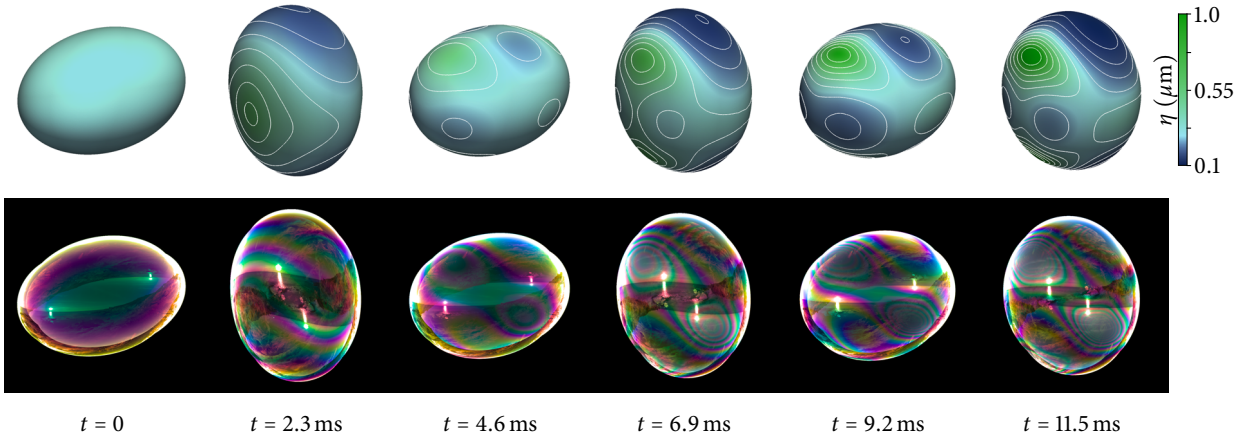

Figure 7.12. Evolution of lamella thickness for an oscillating bubble. (Top row) Colours indicate thickness η of the lamella and the white curves are contour lines of η. (Bottom row) The same bubble oscillation visualised with thin-film interference. Simulation computed on a $256 \times 256 \times 256$ grid in a cubic domain of side length 20 mm with periodic boundary conditions.

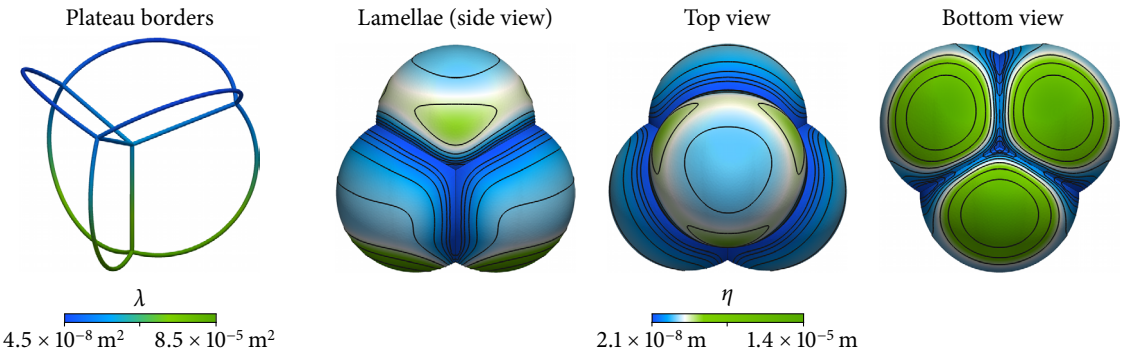

Figure 7.13. Solution of the coupled lamella and Plateau border thin-film equations on a pyramid of four spherical bubbles. Colours indicate thickness η of the lamellae and cross-sectional area λ of the Plateau borders, the black curves are contour lines of η, and (except for the top and bottom views) gravity points down.

using thin-film interference: constructive and destructive interference of reflected light, together with variations in film thickness, lead to interference patterns ("rainbows") seen in everyday soap films. Here, the physically-based ray tracing rendering engine *LuxRender* [129] has been used to solve the Fresnel equations to obtain reflection, refraction, and thin-film interference effects. In particular, the bubble has been globally illuminated by a beach scene: in the reflected image, one can see two suns (from the front and back surfaces of the bubble), with the sky in the upper half. Comparing the patterns with the film thickness shown in the top row, we can see there is a correlation between film thickness variation and the interference patterns.

7.4.2 Drainage phase

In this example, liquid drainage in a coupled lamellae and Plateau border system is demonstrated. Figure 7.13 shows a pyramid of four spheres with diameters 2 mm, forming a network of six lamellae and ten Plateau borders. The lamellae are initialised at time $t = 0$ with a uniform thickness of $\eta = 5\,\mu$m and the Plateau borders with uniform cross-sectional area $\lambda = 0.05\,\text{mm}^2$. Figure 7.13 shows the thickness after draining for 16.1 s. The effect of gravity is seen with the accumulation of liquid at the bottom of the lamellae and Plateau borders, while the effect of the flux boundary condition can be observed with the reduced thickness of the lamellae at the junctions. In the case of the Plateau borders, the thickness profile has essentially attained an equilibrium: as the liquid drains to the bottom due to gravity, the Plateau borders become thin at the top, thereby reducing the liquid pressure, which in turn leads to a pressure gradient opposing the force of gravity.

7.4.3 Rupture and redistribution of mass

To demonstrate rupture and redistribution of liquid mass, in Figure 7.14, a cluster of bubbles with non-uniform thickness has been draining, and the internal lamella separating the two front facing bubbles ruptures immediately after time $t = 0$. The liquid originally contained in the lamella, together

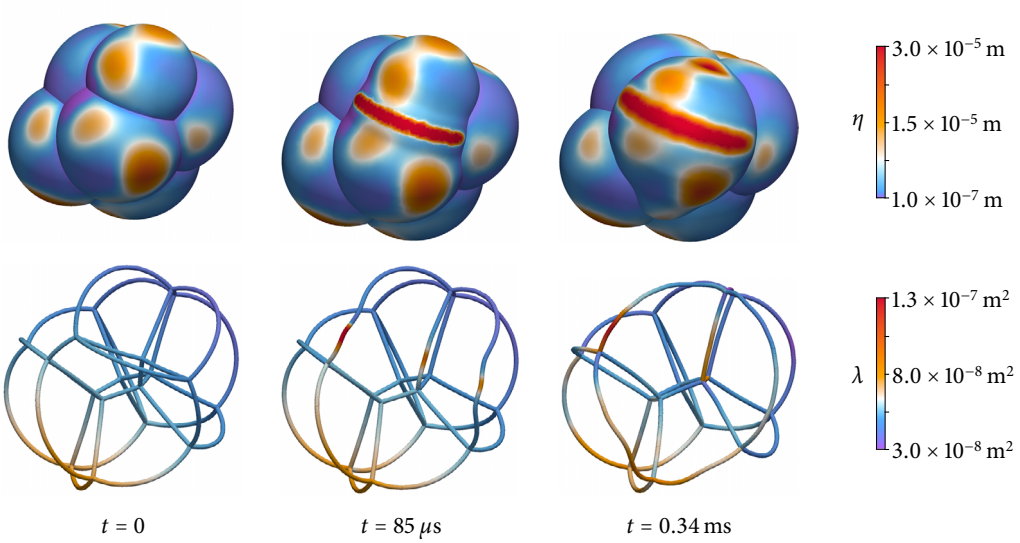

Figure 7.14. Evolution during rupture. An internal lamella joining the two front facing bubbles ruptures and is removed, leading to rearrangement of bubbles and varying film thicknesses.

with the Plateau borders it was once connected to, is locally distributed to the remaining lamellae, as shown by the sudden increase in thickness. The system, driven by macroscopic rearrangement, quickly moves into a new configuration.

7.4.4 Coupled rearrangement, drainage, and rupture

By assembling the complete physical system, the multiscale model can be used to predict the evolution of foam cluster dynamics, under the combined effects of rearrangement, drainage, and rupture. This is demonstrated here with two foam collapse problems.

The first example serves to highlight how foam dynamics can crucially depend on the interaction between the three phases – in other words, while the drainage phase determines which lamellae are ruptured, both the rupture phase and rearrangement phase contribute significantly to this process as well, through nontrivial transport of membrane liquid. To motivate the design of this example, suppose that the lamellae start with a uniform thickness of η_0. As seen in the drainage example in Figure 7.13, it is often the case that as lamellae drain over time, a boundary layer in their thickness develops near the Plateau borders, due in part to the flux boundary condition. Scaling arguments (see §6.5.4) applied to the lamella thin-film equation together with the flux boundary condition, suggest that the width of the boundary layer after a fixed amount of time is $\mathcal{O}(\eta_0^{1/2}\lambda_0^{1/4})$, where λ_0 is a typical thickness of the Plateau border. For typical film thicknesses and drainage times, the length predicted by the scaling is on the order of 0.1 mm, and this was confirmed by numerical tests, as is the result that Plateau borders tend to have the same order of magnitude thickness across the entire network. It follows that for a cluster of bubbles which initially have the same lamellae thickness, all lamellae drain at approximately the same rate, and thus those bubbles smaller than the boundary layer will thin more rapidly and rupture first.

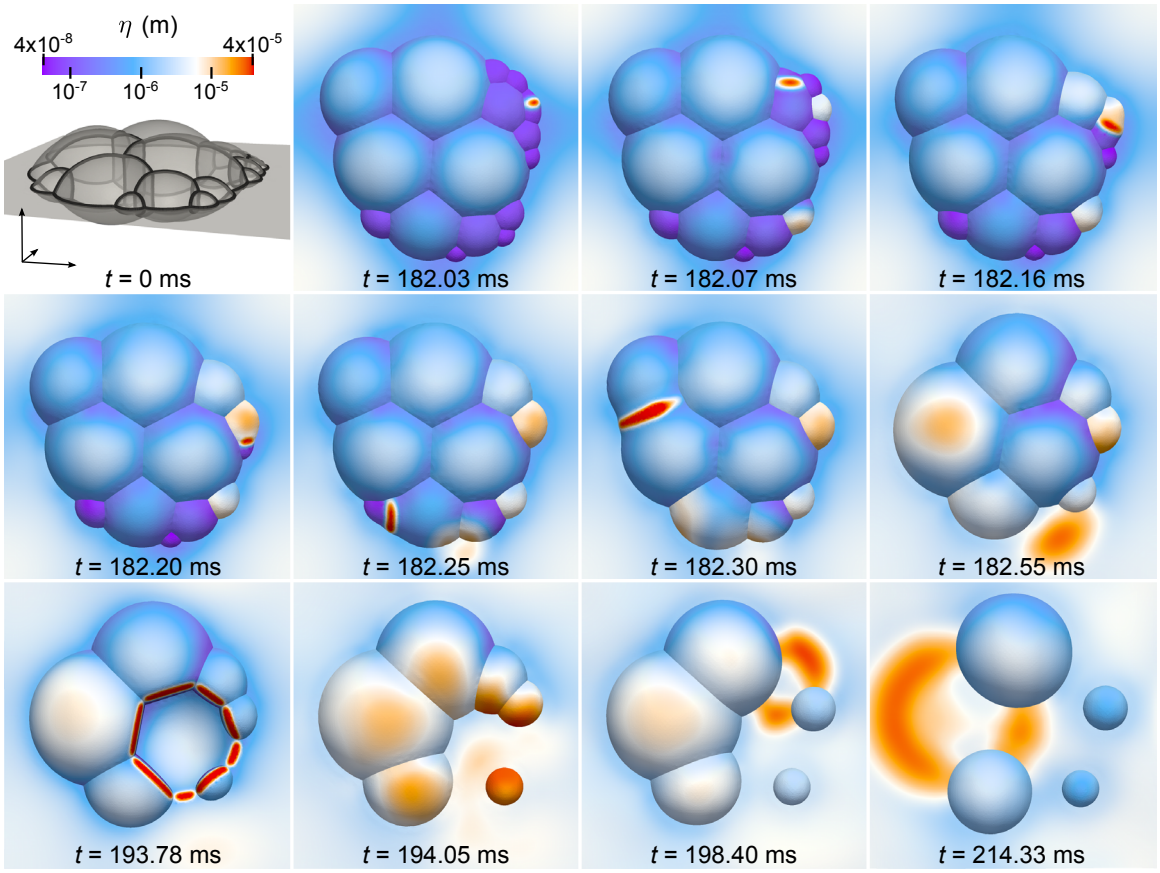

Figure 7.15. Results of the coupled multiscale model for a cluster of bubbles attached to a membrane. In the top-left frame, a side-view of the initial configuration is shown, using semi-opaque lamellae and emphasising the Plateau borders, to highlight the 3D structure of the results. In the rest of the frames, a top-down view is given, showing the lamellae film thickness η, corresponding to the indicated colour scale. The background membrane absorbs some of the drainage, but is chosen not to rupture. As the system evolves, rupture events can be identified by the localised increases in lamellae thickness.

To demonstrate this behaviour, and how it effects rearrangement of bubbles, an example is shown in Figure 7.15. A cluster of 17 bubbles is suspended by a membrane, so that bubbles protrude below and above the membrane. This configuration was designed in order to make the rearrangement simpler to visualise with a top-down perspective. The cluster has a range of bubble sizes, from 0.1 to 0.5 mm in diameter, and at time $t = 0$ is initially in equilibrium, such that each lamella has a uniform thickness of 10 μm, and each Plateau border a uniform cross-sectional area of 0.002 mm^2. After draining for a time of 182 ms, some of the smallest lamellae rupture in quick succession. As this occurs, adjacent bubbles grow in size and increase in thickness. Initially, much of the rupture events are associated with the smaller lamellae, but because rupture effects the macroscopic dynamics of the bubbles, in some cases, larger lamellae rupture due to membrane stretching. On this small spatial scale, the rearrangement phase typically takes 0.1 ms to equilibrate, while drainage steps takes tens of milliseconds. The results show how a nontrivial sequence of rupture events is obtained, and

how bubble rearrangement affects rupture events, both locally and globally, due to changes in film thickness and macroscopic hydrodynamics.

Finally, in the last example, Figure 7.16 shows the results for a larger cluster of 27 bubbles with a typical bubble diameter of 3 mm. In this example, a cluster starts in macroscopic equilibrium with a uniform lamellae thickness of 1 μm, and each Plateau border a uniform cross-sectional area of 0.05 mm^2. Compared to the case in Figure 7.15, in this example the typical bubble size is much larger. As a result, the rupture of a single lamella can have a greater impact on the global dynamics. We see this in Figure 7.16 – after draining for 6.3 s, a single lamella ruptures, and this causes a rapid collapse of the entire structure.

7.5 Concluding remarks

In this chapter and the previous, a multiscale model of the dynamics of a foam has been developed, permitting the study of the effects of fluid properties, topology, bubble shape, and distribution, on drainage, rupture, and rearrangement. Several numerical methods were developed to accompany this model, ranging from Lagrangian-based schemes for transporting film thickness during rearrangement, to biharmonic-modified finite element methods and techniques for treating the coupled boundary conditions in the system of thin-film equations in the drainage phase. Using two foam collapse problems, it was demonstrated how rupture, liquid drainage, and gas hydrodynamics each affect one another on both local and global scales.

Both the scale-separated model and the underlying numerical algorithms are general enough to allow extension of the physics at individual scales to include other phenomena. For example, some types of surfactant solution give rise to films with long lifetimes. In such films, disjoining pressures and van der Waals force can be important in liquid drainage and rupture initiation; these additional physics can be modelled by adding extra terms to the thin-film equations. Diffusive coarsening, which can also be important for long lifetime foams, could be added by generalising the thin-film equations to allow slow movement of the membranes, using equations similar to those derived in [103]. In other types of surfactant solution, mobile/stress-free boundary conditions at the liquid-gas interface are more appropriate than the no-slip boundary conditions used here [106, 108, 130]. In principle, it is possible to derive different thin-film equations on the curved lamellae, taking into account these boundary conditions. In so doing, it is expected that this would lead to coupled PDEs for film thickness evolution: one for the film thickness, which is coupled to a second equation for the tangential velocity field of the liquid inside the membrane. Similar approaches may also allow surface viscosities, evaporation dynamics, and heating of a foam to be modelled.

Additional future work could include more detailed considerations of some of the assumptions made in this model. For example, in the macroscopic rearrangement phase, the films were idealised as massless and infinitely thin, so that the liquid was essentially passively advected by the gas dynamics. Thus, inertial and viscous effects of the liquid inside the films were neglected. Some experimental studies indicate that such effects can dampen the motion of the films. In the work of [95], several experiments were performed which indicated that the dampening rate for soap bubble oscillations is 40%–60% faster in practice, compared to small-scale perturbation theory and numerical experi-

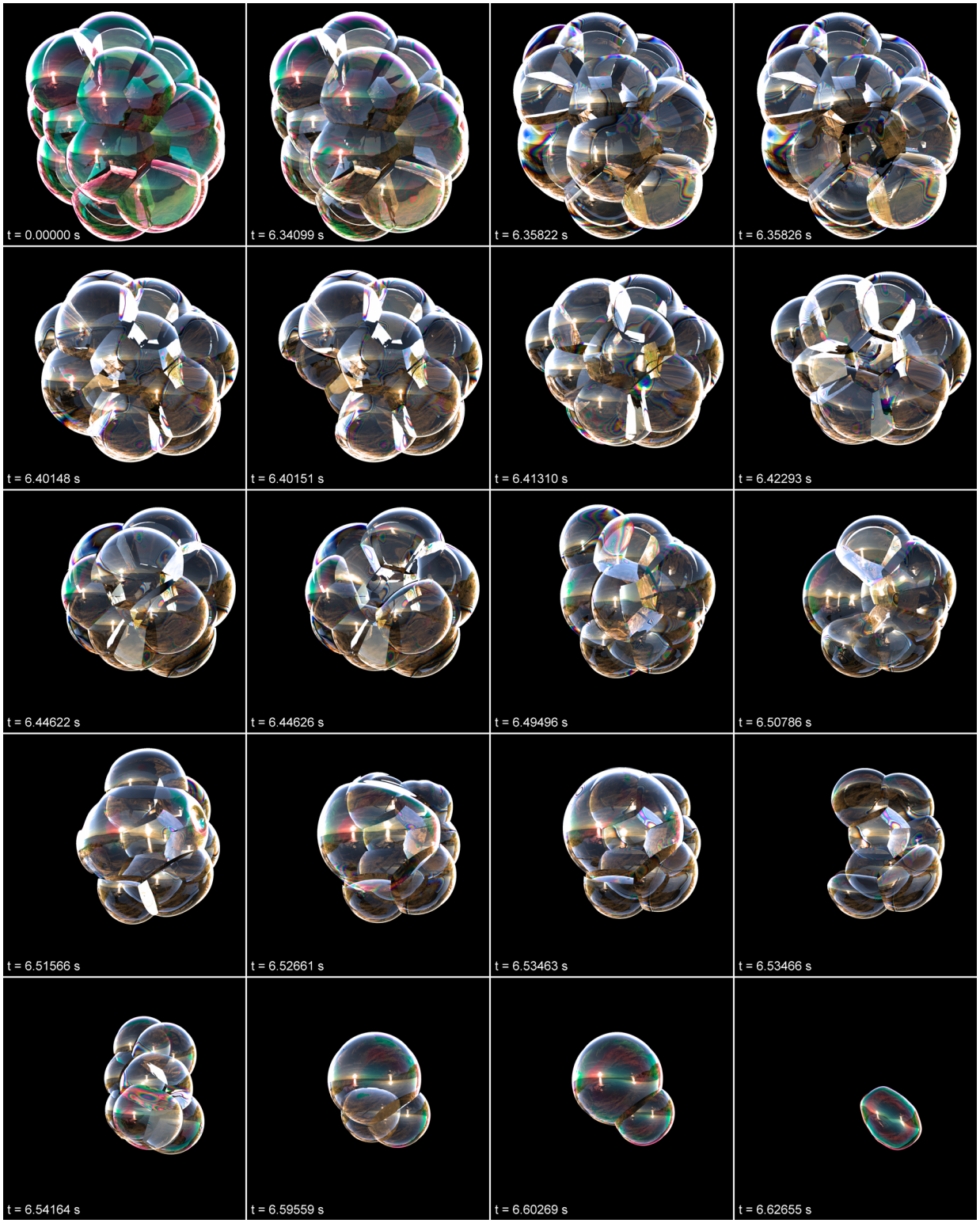

Figure 7.16. Collapse of a foam cluster, visualised with thin-film interference.

ments. Nevertheless, the frequency of oscillation of the experimental data matches theoretical and numerical predictions. To address the difference in dampening, it may be possible to suitably modify the surface tension force in Navier-Stokes, and/or the local viscosity of the gas, to take into account these inertial effects.

Chapter 8

Summary and Future Directions

A central theme in this work has been in the study of multiphase problems, including multiphase curvature flow and multiphase incompressible fluid dynamics, generating high-quality meshes for interconnected surfaces, and multiscale modelling of rupture dynamics in foams. While several new models and numerical methods have been developed here, a main contribution is that of the Voronoi Implicit Interface Method. The VIIM provides a general purpose and flexible method for tracking the evolution of multiple interacting interfaces. It can be coupled to geometry and physics, automatically handles topological changes in the interface, works in any number of spatial dimensions, and is accurate, robust, and efficient.

Recall the core idea of the VIIM: the motion of surfaces nearby to the interface determines the motion of the interface itself. These nearby surfaces are defined as the ϵ-level sets of the distance function to the interface, and the interface motion is reconstructed from the motion of the ϵ-level sets using the Voronoi interface. Thus, the speed of the interface is in some sense the "average" speeds of the nearby ϵ-level sets. This property leads to several advantages. For example, since the neighbouring surfaces can move under quite general speed laws (as we saw in the theory of level set methods in Chapter 2), the VIIM provides a natural method for determining the motion of triple junctions under these speed laws. In the case of curvature flow, this naturally leads to triple point angle conditions that are observed in a range of physical systems, such as foams and grain growth. As was shown in many convergence tests and applications, defining the motion of the interface in this manner leads to a interface tracking method that accurately evolves junctions in space and time. Furthermore, the VIIM idea also leads to a type of regularisation: in the case that speed laws are discontinuous across the interface (as they were, for example, in curvature flow with constraints, or specifying different curvature coefficients, or even in some sense during topological changes in surface tension driven fluid flow), the VIIM regularises the motion by averaging the speed laws across the interface.

Several possibilities exist for future work involving the VIIM, such as developing a mathematical theory for multiphase interface evolution and in designing advanced numerical methods to further increase its accuracy. In addition to further work on multiphase fluid flow problems, there are also several multiphase problems that have not been considered here, which the VIIM could be applied to. We conclude by discussing some of these possibilities.

Mathematical theory of the VIIM

The basic mathematical formulation of the VIIM is to find an ϵ-smoothed solution given by

$$\phi_\epsilon = \lim_{\Delta t \to 0, n \to \infty} (V_\epsilon \circ E_{\Delta t})^n (\phi_0),$$

and then take the limit as $\epsilon \to 0^+$ to find

$$\phi_{\epsilon=0^+} = \lim_{\epsilon \to 0^+} \phi_\epsilon,$$

where V_ϵ is the operator which reconstructs the unsigned distance function from the ϵ-level sets using the Voronoi interface, and $E_{\Delta t}$ is the evolution operator which evolves the ϵ-level sets for a single time step Δt. The numerical tests in Chapter 3 suggest that ϕ_ϵ is well-defined, and so is the limit as $\epsilon \to 0^+$. Clearly, it would be worthwhile to develop a mathematical theory to accompany these numerical results. From a mathematical perspective, key questions to ask include:

- Under what conditions does the VIIM coincide with the theory of surface evolution via level set methods in the case of only two phases? For example, in the "figure 8" problem considered by Evans and Spruck [131], for curvature flow in the level set method, the zero level set "fattens" and develops a non-empty interior. In the VIIM, with the same initial condition and speed law, the interface will not fatten. Although this particular scenario is widely considered as pathological in the practical application of interface tracking methods, it is nevertheless of interest in a theoretical setting.

- Are there statements similar to the comparison principle in level set methods for multiphase flow? The comparison principle is a key result in proving various well-posedness properties of surface evolution problems using the level set method. Finding an equivalent in multiphase problems is subtle, essentially due to the inherent coupling at junctions.

- What requirements are there on the speed laws that ensures the $\epsilon \to 0^+$ definition leads to a well-posed definition of interface evolution?

This is the subject of ongoing research.

Numerical methods for the VIIM

The convergence tests in Chapters 3 and 4 showed that, generally speaking, the VIIM converges with first order accuracy in space and time. In this Eulerian framework, first order accuracy at junctions is basically the best one can hope for. Thus, to increase the accuracy of the VIIM, methods using adaptive mesh refinement may be useful. In such an approach, the computational grid would be refined around junctions, while smooth interfaces away from junctions could continue to use a relatively coarse grid. Since junctions are codimension-two surfaces, the amount of refinement is minimal. However, by introducing smaller space scales, smaller time scales may also be needed for stability reasons. Since using a small time step everywhere in the domain could be prohibitive, techniques based on "sub-cycling" time could be used [132].

From a high level point of view, the VIIM moves a multiphase interface with the following formulation:

1. Advance the ϵ-level sets (defined as sets a distance ϵ away from the multiphase interface), for one time step.
2. Redefine the interface at the next time step as the Voronoi interface of these ϵ-level sets, and loop to 1.

In this work, techniques based on level set methods and finite differences were used to implement this scheme. Note however that these steps may be approximated using a host of numerical technologies, such as finite element methods, semi-Lagrangian techniques, and front tracking methods. Different approaches may lead to different features, and this is worthy of investigation.

Future applications

Several potential applications of the VIIM include:

- *Statistics for Navier-Stokes driven diffusive coarsening.* Gas diffusion across permeable membranes in a dry foam leads to diffusive coarsening (sometimes referred to as Ostwald ripening), in which the gaseous regions change volume over time, similar to multiphase curvature flow (Figure 3.15). To study the long-time statistics of this effect, multiphase curvature flow in 2D and 3D is often used, for example in the computational studies in [37–39], as well as statistical relations derived in [96]. Multiphase curvature flow is a good approximation of diffusive coarsening when the gas dynamics are negligible. It would therefore be interesting to investigate how full surface tension-driven Navier-Stokes dynamics affect coarsening statistics (similar to Figure 4.11), especially when inertial and viscous effects balance or overpower permeability. Two-dimensional dry foam coarsening results with Navier-Stokes were partially studied in [17]. Here, the VIIM provides an opportunity to study this effect in generality in both two and three dimensions.

- *Multi-region image segmentation.* In a similar fashion, the application of the VIIM to multi-region image segmentation is also worthy of investigation. A common method to perform image segmentation is to evolve the boundary of different regions, again by minimising certain energy functionals, in such a way to identify objects and their boundaries in a digital image. Dubrovina and Kimmel [133] have made preliminary investigations into using the VIIM for this purpose and tested simple two-dimensional image segmentation problems. Using the same techniques, three-dimensional image segmentation could also be performed on a range of image segmentation problems.

- *Grain growth in materials design.* Many metal and ceramic materials exhibit "grains", i.e. regions of different crystal orientation or alignment. The geometry and evolution of the grains is often crucial in determining overall material properties, such as strength, resistance to corrosion, ability to fracture, magnetism, etc. Depending on external factors such as temperature,

the grain boundaries move in order to reduce their interfacial energy. For example, in a simple model of grain growth, grains move to minimise their total surface area. This leads to curvature flow with triple junctions satisfying 120° angle conditions – an example was shown in Figure 3.15. In this simple case, a variety of numerical approaches are available to model multiphase curvature flow, such as diffusion generated motion and phase-field methods. However, in complex grain growth problems, more complicated laws of motion are needed, such as taking into account the "roughness" of grain-boundaries that can slow grain movement [134]. When more complex laws of motion are specified, existing numerical methods become more difficult to apply. Here, the flexibility of the VIIM provides a unique opportunity to solve this problem, with the potential to model complex grain growth.

Finally, in regards to the multiscale model for foam dynamics, several possibilities exist for extending the model to include additional physics. Some of these were discussed in the conclusions of Chapter 7. Additional possibilities include:

- *Three-dimensional simulations of topological changes and rupturing films.* The dynamics of topological changes, such as a T1 event in a two-dimensional foam shown in Figure 4.7 and Figure 7.2, have received relatively little attention [97]. It would be worthwhile to study such topological changes using a fully three-dimensional numerical simulation that attempts to resolve the extreme scales, including film thicknesses. This would entail studying the dynamics in a small spatial window and using adaptive mesh refinement techniques. The results would give a better indication of how liquid mass inside the film is redistributed as a result of the topological change, and this could be used to develop quantitative models to be coupled to the multiscale model. In a similar fashion, the rupturing of a film as it retracts back on itself could also be modelled by resolving the scales (and again using adaptive mesh refinement). The results could be compared to models using asymptotic arguments and also be incorporated into the multiscale model.

- *Foams in contact with walls.* In the multiscale model, the foam was assumed to be isolated and not in contact with the wall of a chamber. When Plateau borders are in contact with a fixed wall, their cross-sectional shape changes. In principle, this could be incorporated into the model by using a different value for the coefficient C_Δ in the thin-film equations for these Plateau borders. However, modelling the slip of lamellae and Plateau borders in contact with walls may require carefully constructed contact-angle models and slip boundary conditions. This could also be studied in the context of the multiscale model of foam dynamics.

References

1. R. I. Saye & J. A. Sethian. The Voronoi Implicit Interface Method for Computing Multiphase Physics. *Proceedings of the National Academy of Sciences*, **108**(49), 19498–19503 (2011). doi:10.1073/pnas.1111557108
2. R. I. Saye & J. A. Sethian. Analysis and applications of the Voronoi Implicit Interface Method. *Journal of Computational Physics*, **231**(18), 6051–6085 (2012). doi:10.1016/j.jcp.2012.04.004
3. R. I. Saye & J. A. Sethian. The Voronoi Implicit Interface Method and Computational Challenges in Multiphase Physics. *Milan Journal of Mathematics*, **80**(2), 369–379 (2012). doi:10.1007/s00032-012-0187-6
4. R. I. Saye. "An algorithm to mesh interconnected surfaces via the Voronoi interface". (under review, 2013)
5. R. I. Saye & J. A. Sethian. Multiscale Modeling of Membrane Rearrangement, Drainage, and Rupture in Evolving Foams. *Science*, **340**(6133), 720–724 (2013). doi:10.1126/science.1230623
6. D. Curran. *Closed cell metal foam.* http://commons.wikimedia.org/wiki/File:Closed_cell_metal_foam_with_large_cell_size.JPG. Online – accessed May 2, 2013.
7. E. Pleshakov. *Microstructure of VT22 after quenching.* http://en.wikipedia.org/wiki/File:CrystalGrain.jpg. Online – accessed May 2, 2013.
8. M. L. Manning, R. A. Foty, M. S. Steinberg & E.-M. Schoetz. Coaction of intercellular adhesion and cortical tension specifies tissue surface tension. *Proceedings of the National Academy of Sciences*, **107**(28), 12517–12522 (2010). doi:10.1073/pnas.1003743107
9. K. Brakke. The Surface Evolver. *Experimental Mathematics*, **1**(2), 141–165 (1992). doi:10.1080/10586458.1992.10504253
10. R. Phelan, D. Weaire & K. Brakke. Computation of Equilibrium Foam Structures Using the Surface Evolver. *Experimental Mathematics*, **4**(3), 181–192 (1995). doi:10.1080/10586458.1995.10504320
11. K. Li, M.-Z. Xie, H. Wang & H. Liu. Computational study on cell structure evolution of random liquid and metal-melt foams. *Applied Physics A*, **97**(3), 595–605 (2009). doi:10.1007/s00339-009-5259-2

12. F. Wakai, N. Enomoto & H. Ogawa. Three-dimensional microstructural evolution in ideal grain growth – general statistics. *Acta Materialia*, **48**(6), 1297–1311 (2000). doi:10.1016/S1359-6454(99)00405-X

13. M. Fujita & S. Onami. Cell-to-Cell Heterogeneity in Cortical Tension Specifies Curvature of Contact Surfaces in Caenorhabditis elegans Embryos. *PLoS ONE*, **7**(1) (2012). doi:10.1371/journal.pone.0030224

14. S. Hilgenfeldt, S. Erisken & R. W. Carthew. Physical modeling of cell geometric order in an epithelial tissue. *Proceedings of the National Academy of Sciences*, **105**(3), 907–911 (2008). doi:10.1073/pnas.0711077105

15. C. S. Peskin. The Immersed Boundary Method. *Acta Numerica*, **11,** 479–517 (2002). doi:10.1017/S0962492902000077

16. Y. Kim, M.-C. Lai & C. S. Peskin. Numerical simulations of two-dimensional foam by the Immersed Boundary Method. *Journal of Computational Physics*, **229**(13), 5194–5207 (2010). doi:10.1016/j.jcp.2010.03.035

17. Y. Kim, Y. Seol, M.-C. Lai & C. S. Peskin. The Immersed Boundary Method for Two-Dimensional Foam with Topological Changes. *Communications in Computational Physics*, **12**(2), 479–493 (2012). doi:10.4208/cicp.181210.080811s

18. L. Bronsard & B. T. R. Wetton. A Numerical Method for Tracking Curve Networks Moving with Curvature Motion. *Journal of Computational Physics*, **120**(1), 66–87 (1995). doi:10.1006/jcph.1995.1149

19. M.-C. Lai, C.-W. Hsu & H. Huang. A Front-Tracking Method for Motion by Mean Curvature with Surfactant. *Advances in Applied Mathematics and Mechanics*, **1**(2), 288–300 (2009)

20. W. F. Noh & P. Woodward. SLIC (Simple Line Interface Calculation), in *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics* (eds A. I. van de Vooren & P. J. Zandbergen) **59** (Springer Berlin Heidelberg, 1976), 330–340. doi:10.1007/3-540-08004-X_336

21. D. J. Benson. Volume of fluid interface reconstruction methods for multi-material problems. *Applied Mechanics Reviews*, **55**(2), 151–165 (2002). doi:10.1115/1.1448524

22. B. Y. Choi & M. Bussmann. A piecewise linear approach to volume tracking a triple point. *International Journal for Numerical Methods in Fluids*, **53**(6), 1005–1018 (2007). doi:10.1002/fld.1317

23. S. Osher & J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, **79**(1), 12–49 (1988). doi:10.1016/0021-9991(88)90002-2

24. B. Merriman, J. K. Bence & S. J. Osher. Motion of Multiple Junctions: A Level Set Approach. *Journal of Computational Physics*, **112**(2), 334–363 (1994). doi:10.1006/jcph.1994.1105

25. K. A. Smith, F. J. Solis & D. L. Chopp. A projection method for motion of triple junctions by level sets. *Interfaces and Free Boundaries*, **4**(3), 263–276 (2002). doi:10.4171/IFB/61

26. F. Losasso, T. Shinar, A. Selle & R. Fedkiw. Multiple interacting liquids, in *ACM SIGGRAPH 2006 Papers* (2006), 812–819. doi:10.1145/1179352.1141960

27. L. A. Vese & T. F. Chan. A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model. *International Journal of Computer Vision*, **50**(3), 271–293 (2002). doi:10.1023/A:1020874308076

28. J. Lie, M. Lysaker & X.-C. Tai. A variant of the level set method and applications to image segmentation. *Mathematics of Computation*, **75**(255), 1155–1174 (2006). doi:10.1090/S0025-5718-06-01835-7

29. H.-K. Zhao, T. Chan, B. Merriman & S. Osher. A Variational Level Set Approach to Multiphase Motion. *Journal of Computational Physics*, **127**(1), 179–195 (1996). doi:10.1006/jcph.1996.0167

30. H.-K. Zhao, B. Merriman, S. Osher & L. Wang. Capturing the Behavior of Bubbles and Drops Using the Variational Level Set Approach. *Journal of Computational Physics*, **143**(2), 495–518 (1998). doi:10.1006/jcph.1997.5810

31. S. Esedoglu & P. Smereka. A variational formulation for a level set representation of multiphase flow and area preserving curvature flow. *Communications in Mathematical Sciences*, **6**(1), 125–148 (2008)

32. H. Garcke, B. Nestler & B. Stoth. A Multiphase Field Concept: Numerical Simulations of Moving Phase Boundaries and Multiple Junctions. *SIAM Journal on Applied Mathematics*, **60**(1), 295–315 (1999). doi:10.1137/S0036139998334895

33. S. J. Ruuth. A Diffusion-Generated Approach to Multiphase Motion. *Journal of Computational Physics*, **145**(1), 166–192 (1998). doi:10.1006/jcph.1998.6028

34. S. Esedoglu, S. Ruuth & R. Tsai. Diffusion generated motion using signed distance functions. *Journal of Computational Physics*, **229**(4), 1017–1042 (2010). doi:10.1016/j.jcp.2009.10.002

35. L. C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana University Mathematics Journal*, **42,** 533–557 (1993)

36. G. Barles & C. Georgelin. A Simple Proof of Convergence for an Approximation Scheme for Computing Motions by Mean Curvature. *SIAM Journal on Numerical Analysis*, **32**(2), 484–500 (1995). doi:10.1137/0732020

37. M. Elsey, S. Esedoglu & P. Smereka. Diffusion generated motion for grain growth in two and three dimensions. *Journal of Computational Physics*, **228**(21), 8015–8033 (2009). doi:10.1016/j.jcp.2009.07.020

38. M. Elsey, S. Esedoglu & P. Smereka. Large-scale simulation of normal grain growth via diffusion-generated motion. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **467**(2126), 381–401 (2010). doi:10.1098/rspa.2010.0194

39. S. Esedoglu & F. Otto. "Threshold dynamics for networks with arbitrary surface tensions". (Preprint 2013)

40. J. E. Taylor. The Motion of Multiple-Phase Junctions under Prescribed Phase-Boundary Velocities. *Journal of Differential Equations*, **119**(1), 109–136 (1995). doi:10.1006/jdeq.1995.1085

41. F. Reitich & H. Soner. Three-phase boundary motions under constant velocities. I: The vanishing surface tension limit. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, **126**(4), 837–865 (1996). doi:10.1017/S0308210500023106

42. X. Chen & J.-S. Guo. Self-similar solutions of a 2-D multiple-phase curvature flow. *Physica D: Nonlinear Phenomena*, **229**(1), 22–34 (2007). doi:10.1016/j.physd.2007.03.009

43. P. R. Rios & M. E. Glicksman. Polyhedral model for self-similar grain growth. *Acta Materialia*, **56**(5), 1165–1171 (2008). doi:10.1016/j.actamat.2007.11.010

44. L. Bronsard & F. Reitich. On three-phase boundary motion and the singular limit of a vector-valued Ginzburg-Landau equation. *Archive for Rational Mechanics and Analysis*, **124**(4), 355–379 (1993). doi:10.1007/BF00375607

45. P.-O. Persson & G. Strang. A simple mesh generator in Matlab. *SIAM Review*, **46**(2), 329–345 (2004). doi:10.1137/S0036144503429121

46. J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences* (Cambridge University Press, 1999)

47. S. Osher & R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces* (Springer, 2003)

48. Y. Giga. *Surface Evolution Equations: A Level Set Approach* (Springer, 2006)

49. J. A. Sethian. Numerical Methods for Propagating Fronts, in *Variational Methods for Free Surface Interfaces, Proceedings of the Sept, 1985 Vallambrosa Conference* (eds P. Concus & R. Finn) (Springer-Verlag, 1987). doi:10.1007/978-1-4612-4656-5_18

50. D. Adalsteinsson & J. A. Sethian. A Fast Level Set Method for Propagating Interfaces. *Journal of Computational Physics*, **118**(2), 269–277 (1995). doi:10.1006/jcph.1995.1098

51. D. L. Chopp. Some Improvements of the Fast Marching Method. *SIAM Journal on Scientific Computing*, **23**(1), 230–244 (2001). doi:10.1137/S106482750037617X

52. J. A. Sethian. A Fast Marching Level Set Method for Monotonically Advancing Fronts. *Proceedings of the National Academy of Sciences*, **93**(4), 1591–1595 (1996). doi:10.1073/pnas.93.4.1591

53. R. Malladi, J. A. Sethian & B. C. Vemuri. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(2), 158–175 (1995). doi:10.1109/34.368173

54. D. Adalsteinsson & J. A. Sethian. The Fast Construction of Extension Velocities in Level Set Methods. *Journal of Computational Physics*, **148**(1), 2–22 (1999). doi:10.1006/jcph.1998.6090

55. D. L. Chopp. Another Look at Velocity Extensions in the Level Set Method. *SIAM Journal on Scientific Computing*, **31**(5), 3255–3273 (2009). doi:10.1137/070686329

56. W. E. Lorensen & H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, **21**(4), 163–169 (1987). doi:10.1145/37402.37422

57. A. Guéziec & R. Hummel. Exploiting Triangulated Surface Extraction Using Tetrahedral Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, **1**(4), 328–342 (1995). doi:10.1109/2945.485620

58. B. A. Payne & A. W. Toga. Surface Mapping Brain Function on 3D Models. *IEEE Computer Graphics and Applications*, **10**(5), 33–41 (1990). doi:10.1109/38.59034

59. S. L. Chan & E. O. Purisima. A new tetrahedral tesselation scheme for isosurface generation. *Computers and Graphics*, **22**(1), 83–90 (1998). doi:10.1016/S0097-8493(97)00085-X

60. J. von Neumann. in *Metal Interfaces* (ed C. Herring) (American Society for Metals, Cleveland, 1952), 108–110

61. W. W. Mullins. Two-Dimensional Motion of Idealized Grain Boundaries. *Journal of Applied Physics*, **27**(8), 900–904 (1956). doi:10.1063/1.1722511

62. R. D. MacPherson & D. J. Srolovitz. The von Neumann relation generalized to coarsening of three-dimensional microstructures. *Nature*, **446,** 1053–1055 (2007). doi:10.1038/nature05745

63. D. L. Weaire & S. Hutzler. *The Physics of Foams* (Oxford University Press, 2001)

64. J. A. Sethian & P. Smereka. Level Set Methods for Fluid Interfaces. *Annual Review of Fluid Mechanics*, **35,** 341–372 (2003). doi:10.1146/annurev.fluid.35.101101.161105

65. J. U. Brackbill, D. B. Kothe & C. Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, **100**(2), 335–354 (1992). doi:10.1016/0021-9991(92)90240-Y

66. M. Sussman, P. Smereka & S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, **114**(1), 146–159 (1994). doi:10.1006/jcph.1994.1155

67. A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell & M. L. Welcome. A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, **142**(1), 1–46 (1998). doi:10.1006/jcph.1998.5890

68. A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, **22**(104), 745–762 (1968). doi:10.1090/S0025-5718-1968-0242392-2

69. J. Bloomenthal & K. Ferguson. Polygonization of non-manifold implicit surfaces, in *Proceedings of the 22nd annual conference on computer graphics and interactive techniques* (ACM, 1995), 309–316. doi:10.1145/218380.218462

70. H. Suzuki, T. Fujimori, T. Michikawa, Y. Miwata & N. Sadaoka. Skeleton Surface Generation from Volumetric Models of Thin Plate Structures for Industrial Applications, in *Mathematics of Surfaces XII* (eds R. Martin, M. Sabin & J. Winkler) 442–464 (Springer Berlin Heidelberg, 2007). doi:10.1007/978-3-540-73843-5_27

71. M. H. Shammaa, H. Suzuki & Y. Ohtake. Extraction of isosurfaces from multi-material CT volumetric data of mechanical parts, in *Proceedings of the 2008 ACM symposium on solid and physical modeling* (ACM, 2008), 213–220. doi:10.1145/1364901.1364931

72. H.-C. Hege, M. Seebass, D. Stalling & M. Zöckler. *A generalized marching cubes algorithm based on non-binary classifications.* Tech. rep. (Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997)

73. S. Yamazaki, K. Kase & K. Ikeuchi. Non-manifold implicit surfaces based on discontinuous implicitization and polygonization, in *Proceedings of the Geometric Modeling and Processing* (2002), 138–146. doi:10.1109/GMAP.2002.1027505

74. J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec & R. Keriven. High-Quality Consistent Meshing of Multi-label Datasets, in *Information Processing in Medical Imaging* (eds N. Karssemeijer & B. Lelieveldt) (Springer Berlin Heidelberg, 2007), 198–210. doi:10.1007/978-3-540-73273-0_17

75. T. K. Dey, F. Janoos & J. A. Levine. Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers*, **28**(1), 71–82 (2012). doi:10.1007/s00366-011-0217-y

76. D. Boltcheva, M. Yvinec & J.-D. Boissonnat. Mesh Generation from 3D Multi-material Images, in *Medical Image Computing and Computer-Assisted Intervention* (Springer-Verlag, 2009), 283–290. doi:10.1007/978-3-642-04271-3_35

77. K. Shimada & D. C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing, in *Proceedings of the third ACM symposium on solid modeling and applications* (ACM, 1995), 409–419. doi:10.1145/218013.218095

78. M. Meyer, R. Whitaker, R. M. Kirby, C. Ledergerber & H. Pfister. Particle-based Sampling and Meshing of Surfaces in Multimaterial Volumes. *IEEE Transactions on Visualization and Computer Graphics*, **14**(6), 1539–1546 (2008). doi:10.1109/TVCG.2008.154

79. B. Reitinger, E. Bornik & R. Beichel. Constructing smooth non-manifold meshes of multi-labeled volumetric datasets, in *Proceedings of WSCG 2005* (UNION Agency Science Press, 2005), 227–234

80. Y. Zhang & J. Qian. Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering*, **247,** 166–178 (2012). doi:10.1016/j.cma.2012.07.022

81. Y. Zhang, T. J. R. Hughes & C. L. Bajaj. Automatic 3D Mesh Generation for a Domain with Multiple Materials, in *Proceedings of the 16th International Meshing Roundtable* (eds M. L. Brewer & D. Marcum) 367–386 (Springer Berlin Heidelberg, 2008). doi:10.1007/978-3-540-75103-8_21

82. M. O. Bloomfield, D. F. Richards & T. S. Cale. The Use of Conformal Voxels for Consistent Extractions from Multiple Level-Set Fields, in *Computational Science - ICCS 2005* (eds V. S. Sunderam, G. D. van Albada, P. A. Sloot & J. Dongarra) 49–56 (Springer Berlin Heidelberg, 2005). doi:10.1007/11428862_7

83. V. d'Otreppe, R. Boman & J.-P. Ponthot. Generating smooth surface meshes from multi-region medical images. *International Journal for Numerical Methods in Biomedical Engineering*, **28**(6-7), 642–660 (2012). doi:10.1002/cnm.1471

84. J. C. Anderson, C. Garth, M. A. Duchaineau & K. I. Joy. Smooth, Volume-Accurate Material Interface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, **16**(5), 802–814 (2010). doi:10.1109/TVCG.2010.17

85. D. A. Field. Laplacian smoothing and Delaunay triangulations. *Communications in Applied Numerical Methods*, **4**(6), 709–712 (1988). doi:10.1002/cnm.1630040603

86. P.-O. Persson. "Mesh generation for implicit geometries". PhD thesis (Massachusetts Institute of Technology, 2005)

87. M. Bern & P. Plassmann. Mesh generation, in *Handbook of Computational Geometry* (eds J.-R. Sack & J. Urutia) (Elsevier Science, 1999)

88. D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, **47**(4), 887–906 (2000). doi:10.1002/(SICI)1097-0207(20000210)47:4<887::AID-NME804>3.0.CO;2-H

89. S. Hilgenfeldt, S. Arif & J.-C. Tsai. Foam: a multiphase system with many facets. *Philosophical Transactions of the Royal Society A*, **366**(1873), 2145–2159 (2008). doi:10.1098/rsta.2008.0004

90. J. C. Bird., R. de Ruiter, L. Courbin & H. A. Stone. Daughter bubble cascades produced by folding of ruptured thin films. *Nature*, **465,** 759–762 (2010). doi:10.1038/nature09069

91. J. Plateau. *Statique expérimentale et théorique des liquides soumis aux seules forces moléculaires* (Gauthier-Villars, Trubner et cie., Paris, France, 1873)

92. D. L. Chopp. Computing Minimal Surfaces via Level Set Curvature Flow. *Journal of Computational Physics*, **106**(1), 77–91 (1993). doi:10.1006/jcph.1993.1092

93. K. Polthier. Computational Aspects of Discrete Minimal Surfaces, in *Global Theory of Minimal Surfaces, Proceedings of the Clay Mathematics Institute 2001 Summer School* (ed D. Hoffman) (American Mathematical Society, 2002)

94. C. A. Miller & L. E. Scriven. The oscillations of a fluid droplet immersed in another fluid. *Journal of Fluid Mechanics*, **32**(3), 417–435 (1968). doi:10.1017/S0022112068000832

95. U. Kornek, F. Müller, K. Harth, A. Hahn, S. Ganesan, L. Tobiska & R. Stannarius. Oscillations of soap bubbles. *New Journal of Physics*, **12**(7), 073031 (2010). doi:10.1088/1367-2630/12/7/073031

96. S. Hilgenfeldt, A. M. Kraynik, S. A. Koehler & H. A. Stone. An Accurate von Neumann's Law for Three-Dimensional Foams. *Physical Review Letters*, **86**(12), 2685–2688 (2001). doi:10.1103/PhysRevLett.86.2685

97. M. Durand & H. A. Stone. Relaxation Time of the Topological T1 Process in a Two-Dimensional Foam. *Physical Review Letters*, **97**(22), 226101 (2006). doi:10.1103/PhysRevLett.97.226101

98. N. Vandewalle & J. F. Lentz. Cascades of popping bubbles along air/foam interfaces. *Physical Review E*, **64**(2), 021507 (2001). doi:10.1103/PhysRevE.64.021507

99. N. Vandewalle, J. F. Lentz, S. Dorbolo & F. Brisbois. Avalanches of Popping Bubbles in Collapsing Foams. *Physical Review Letters*, **86**(1) (2001). doi:10.1103/PhysRevLett.86.179

100. H. Ritacco, F. Kiefer & D. Langevin. Lifetime of Bubble Rafts: Cooperativity and Avalanches. *Physical Review Letters*, **98**(24), 244501 (2007). doi:10.1103/PhysRevLett.98.244501

101. A. Oron, S. H. Davis & S. G. Bankoff. Long-scale evolution of thin liquid films. *Reviews of Modern Physics*, **69**(3), 931–980 (1997). doi:10.1103/RevModPhys.69.931

102. T. G. Myers. Thin Films with High Surface Tension. *SIAM Review*, **40**(3), 441–462 (1998). doi:10.1137/S003614459529284X

103. P. D. Howell. Surface-tension-driven flow on a moving curved surface. *Journal of Engineering Mathematics*, **45**(3-4), 283–308 (2003). doi:10.1023/A:1022685018867

104. Z. Wang & G. Narsimhan. Model for Plateau border drainage of power-law fluid with mobile interface and its application to foam drainage. *Journal of Colloid and Interface Science*, **300**(1), 327–337 (2006). doi:10.1016/j.jcis.2006.03.023

105. A. V. Nguyen. Liquid Drainage in Single Plateau Borders of Foam. *Journal of Colloid and Interface Science*, **249**(1), 194–199 (2002). doi:10.1006/jcis.2001.8176

106. S. A. Koehler, S. Hilgenfeldt & H. A. Stone. Foam drainage on the microscale: I. Modeling flow through single Plateau borders. *Journal of Colloid and Interface Science*, **276**(2), 420–438 (2004). doi:10.1016/j.jcis.2003.12.061

107. C. J. W. Breward & P. D. Howell. The drainage of a foam lamella. *Journal of Fluid Mechanics*, **458,** 379–406 (2002). doi:10.1017/S0022112002007930

108. P. D. Howell & H. A. Stone. On the absence of marginal pinching in thin free films. *European Journal of Applied Mathematics*, **16**(05), 569–582 (2005). doi:10.1017/S095679250500625X

109. L. W. Schwartz & H. M. Princen. A Theory of Extensional Viscosity for Flowing Foams and Concentrated Emulsions. *Journal of Colloid and Interface Science*, **118**(1), 201–211 (1987). doi:10.1016/0021-9797(87)90449-8

110. R. Valéry Roy, A. J. Roberts & M. E. Simpson. A lubrication model of coating flows over a curved substrate in space. *Journal of Fluid Mechanics*, **454,** 235–261 (2002). doi:10.1017/S0022112001007133

111. T. G. Myers, J. P. F. Charpin & S. J. Chapman. The flow and solidification of a thin fluid film on an arbitrary three-dimensional surface. *Physics of Fluids*, **14**(8), 2788–2803 (2002). doi:10.1063/1.1488599

112. T. Erneux & S. H. Davis. Nonlinear rupture of free films. *Physics of Fluids A*, **5**(5), 1117–1122 (1993). doi:10.1063/1.858597

113. M. Prévost & D. Gallez. Nonlinear rupture of thin free liquid films. *Journal of Chemical Physics*, **84**(7), 4043–4048 (1986). doi:10.1063/1.450065

114. G. K. Batchelor. *An Introduction to Fluid Dynamics* (Cambridge University Press, 2000)

115. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell & M. L. Welcome. An Adaptive Level Set Approach for Incompressible Two-Phase Flows. *Journal of Computational Physics*, **148**(1), 81–124 (1999). doi:10.1006/jcph.1998.6106

116. D. Adalsteinsson & J. Sethian. Transport and diffusion of material quantities on propagating interfaces via level set methods. *Journal of Computational Physics*, **185**(1), 271–288 (2003). doi:10.1016/S0021-9991(02)00057-8

117. J. Sethian & Y. Shan. Solving partial differential equations on irregular domains with moving interfaces, with applications to superconformal electrodeposition in semiconductor manufacturing. *Journal of Computational Physics*, **227**(13), 6411–6447 (2008). doi:10.1016/j.jcp.2008.03.001

118. H. Stone. A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface. *Physics of Fluids A*, **2**(1), 111–112 (1990). doi:10.1063/1.857686

119. G. Grün & M. Rumpf. Nonnegativity Preserving Convergent Schemes for the Thin Film Equation. *Numerische Mathematik*, **87**(1), 113–152 (2000). doi:10.1007/s002110000197

120. J. Becker, G. Grün, M. Lenz & M. Rumpf. Numerical Methods for Fourth Order Nonlinear Degenerate Diffusion Problems. *Applications of Mathematics*, **47**(6), 517–543 (2002). doi:10.1023/B:APOM.0000034537.55985.44

121. L. Zhornitskaya & A. L. Bertozzi. Positivity-preserving numerical schemes for lubrication-type equations. *SIAM Journal on Numerical Analysis*, **37**(2), 523–555 (2000). doi:10.1137/S0036142998335698

122. G. Grün. On the convergence of entropy consistent schemes for lubrication type equations in multiple space dimensions. *Mathematics of Computation*, **72**(243), 1251–1279 (2003). doi:10.1090/S0025-5718-03-01492-3

123. A. L. Bertozzi, N. Ju & H.-W. Lu. A biharmonic-modified forward time stepping method for fourth order nonlinear diffusion equations. *Discrete and Continuous Dynamical Systems*, **29**(4), 1367–1391 (2011). doi:10.3934/dcds.2011.29.1367

124. J. Douglas, Jr. & T. Dupont. Alternating-direction Galerkin methods on rectangles, in *Numerical Solution of Partial Differential Equations, II (SYNSPADE 1970) (Proc. Sympos. Univ. of Maryland, College Park, Md., 1970)* (Academic Press, New York, 1971), 133–214

125. P. Beltrame & U. Thiele. Time Integration and Steady-State Continuation for 2d Lubrication Equations. *SIAM Journal on Applied Dynamical Systems*, **9**(2), 484–518 (2010). doi:10.1137/080718619

126. G. Dziuk & C. M. Elliott. Surface finite elements for parabolic equations. *Journal of Computational Mathematics*, **25**(4), 385–407 (2007)

127. M. Bertalmio, L.-T. Cheng, S. Osher & G. Sapiro. Variational Problems and Partial Differential Equations on Implicit Surfaces. *Journal of Computational Physics*, **174**(2), 759–780 (2001). doi:10.1006/jcph.2001.6937

128. J. B. Greer, A. L. Bertozzi & G. Sapiro. Fourth order partial differential equations on general geometries. *Journal of Computational Physics*, **216**(1), 216–246 (2006). doi:10.1016/j.jcp.2005.11.031

129. *LuxRender.* http://www.luxrender.net. 2012

130. S. A. Koehler, S. Hilgenfeldt & H. A. Stone. Liquid Flow through Aqueous Foams: The Node-Dominated Foam Drainage Equation. *Physical Review Letters*, **82**(21), 4232–4235 (1999). doi:10.1103/PhysRevLett.82.4232

131. L. C. Evans & J. Spruck. Motion of level sets by mean curvature. I. *Journal of Differential Geometry*, **33,** 635–681 (1991)

132. C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield & J. B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, **3**(3), 147–157 (2000). doi:10.1007/PL00013544

133. A. Dubrovina & R. Kimmel. *Active contours for multi-region image segmentation with a single level set function.* Tech. rep. CIS-2012-06 (Technion - Computer Science Department, 2012)

134. E. A. Holm & S. M. Foiles. How Grain Growth Stops: A Mechanism for Grain-Growth Stagnation in Pure Materials. *Science*, **328**(5982), 1138–1141 (2010). doi:10.1126/science.1187833