Journal of Computational Physics 231 (2012) 6051–6085

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Analysis and applications of the Voronoi Implicit Interface Method

R.I. Saye, J.A. Sethian [*]

Department of Mathematics and Lawrence Berkeley National Laboratory, University of California, Berkeley, Berkeley, CA 94720, United States

ARTICLE INFO

Article history:
Received 3 November 2011
Received in revised form 23 March 2012
Accepted 5 April 2012
Available online 25 April 2012

Keywords:
Multiple interface dynamics
Multiphase physics
Level set methods
Foams
Fluid flow
Curvature flow

ABSTRACT

We analyze a new mathematical and numerical framework, the "Voronoi Implicit Interface Method" ("VIIM"), first introduced in Saye and Sethian (2011) [R.I. Saye, J.A. Sethian, The Voronoi Implicit Interface Method for computing multiphase physics, PNAS 108 (49) (2011) 19498–19503] for tracking multiple interacting and evolving regions ("phases") whose motion is determined by complex physics (fluids, mechanics, elasticity, etc.). From a mathematical point of view, the method provides a theoretical framework for moving interface problems that involve multiple junctions, defining the motion as the formal limit of a sequence of related problems. Discretizing this theoretical framework provides a numerical methodolology which automatically handles multiple junctions, triple points and quadruple points in two dimensions, as well as triple lines, etc. in higher dimensions. Topological changes in the system occur naturally, with no surgery required. In this paper, we present the method in detail, and demonstrate several new extensions of the method to different physical phenomena, including curvature flow with surface energy densities defined on a per-phase basis, as well as multiphase fluid flow in which density, viscosity and surface tension can be defined on a per-phase basis.

We test this method in a variety of ways. We perform rigorous analysis and demonstrate convergence in both two and three dimensions for a variety of evolving interface problems, including verification of von Neumann–Mullins' law in two dimensions (and its analog in three dimensions), as well as normal driven flow and curvature flow with and without constraints, demonstrating topological change and the effects of different boundary conditions. We couple the method to a second order projection method solver for incompressible fluid flow, and study the effects of membrane permeability and impermeability, large shearing torsional forces, and the effects of varying density, viscosity and surface tension on a per-phase basis. Finally, we demonstrate convergence in both space and time of a topological change in a multiphase foam.

1. Introduction

Many scientific and engineering problems are characterized by a large number of different regions that touch in many different configurations, and whose interaction depends on local geometries, complex physics, intricate jump conditions, and other associated boundary conditions and constraints. Examples include the motion of foams, crystal grain growth, and multicellular structures in man-made and biological materials, as well as mathematical problems, such as geometric motion, domain decomposition and surface area minimization problems.

There are a host of mathematical and computational challenges associated with computing the solutions to these problems. Often, the physics, chemistry, and mechanics that drive the interface motion are complex, requiring the solution of

[*] Corresponding author.
E-mail address: sethian@math.berkeley.edu (J.A. Sethian).

fluid mechanical equations with specified jumps, elasticity solvers with different properties in each membrane, as well as diffusion and transport effects of species both within and across the region boundaries, etc. It is challenging to construct good and well-posed mathematical models that adequately describe the fundamental forces. At the same time, formulating a consistent mathematical model and numerical methodology for the interface motion itself can be daunting, especially in the presence of multi-junction points (triple points, quadruple points, etc.) in two dimensions and analogous structures in three dimensions, including triple lines where multiple surfaces meet, etc.

1.1. Mathematical formulations

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 2D) and surfaces (in 3D) that connect them. As examples of how such motion may be specified for a purely geometric motion given by curvature flow in two dimensions:

- One option might be to employ 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 minimize 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 crystal grain growth and soap bubble foams. In this situation, generally speaking, quadruple points are unstable: they will quickly destabilize 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 this work, we concentrate mainly on the case relevant to many physical problems in which triple points are allowed to move, often satisfying certain triple point angle conditions that naturally arise from the type of physics being considered. This will carry over to three-dimensional analogs with no change to the fundamental formulation.

1.2. Numerical formulations

From a numerical perspective, we want a stable method that correctly includes the prescribed motion at triple points, correctly represents moving-interface boundary conditions and associated physics, performs efficiently and accurately, and handles rapid changes in topology and structure.

A variety of numerical methods have been proposed to track the evolution of these multiphase problems. Somewhat broadly, major approaches include:

- *Front tracking methods*: Here, the interface is explicitly represented with a Lagrangian geometric representation, typically taken as connected line segments in two dimensions and triangles in three dimensions (see, for example, [7,6,13]). Junctions (e.g. triple points) have shared nodes, and the positions of the elements making up the front are updated in time. In these methods, an explicit surgery mechanism for topological change is required, which may be challenging in three dimensions due to a range of possible types of topology change. For multiphase systems, Young's law, which determines the angles in a triple point, is explicitly assumed and enforced as a boundary condition.
- *Volume of fluid methods*: Here, each phase/region is described by a characteristic function, which has a value of one inside the phase and zero outside [18]. In discrete form, these become volume fractions on a mesh, in which multiple phases can occupy the same cell, with respective volume fractions adding up to one. The values of these fractions inside the cells are updated according to local reconstruction algorithms based on neighboring cell values. Junctions correspond to mixed regions with more than two phases in a cell [4,8].
- *Level set methods*: Level set methods, introduced by Osher and Sethian [19], embed the interface as the zero level set of an implicitly defined signed distance function, and rely in part on the theory of curve and surface evolution given in [24] and on the link between front propagation and hyperbolic conservation laws discussed in [25]. Level set methods recast interface motion as a time-dependent Eulerian initial value partial differential equation, and use viscosity solutions of the appropriate differential equations to update the position of the front, using an interface velocity that is derived from the relevant physics both on and off the interface. Topological change is handled as a straightforward part of the implicit embedding, and geometric quantities such as normal directions and curvature are naturally calculated. The appropriate viscosity solutions are obtained by exploiting schemes from the numerical solution of hyperbolic conservation laws to build appropriate finite difference and finite element approximations, and the underlying level set methods are made computationally efficient through the use of adaptive narrow banding, see [1]. For a general review, see [27].

  The extension of these methods to multiphase systems typically involves using multiple level set functions, one for each phase/region (or some other encoding scheme): the first such method was introduced in [17]. These different level set functions must be coupled together correctly in order to avoid creating vacuums or overlaps. This is typically done through a "repair" procedure which is required at the end of every time step to "seal" the level set functions back together. Later approaches include [30], in which a projection method is employed to project the values of the multiple level set functions to a specific manifold that does not allow vacuums or overlaps.

When the desired motion is purely geometric, and other physics (fluid motion, elasticity, etc.) are ignored, alternative techniques have been developed. These include:

- *Variational methods*: Variational methods derive their evolution from gradient descent on an energy functional, and time measures the progress towards the desired minimization 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 [34]. Multiple level set functions are again employed, and, rather than repairing the gaps or overlaps that occur from the decoupled motion of individual phases, a penalty function is employed with appropriate constraints that keep the separate level set functions from developing gaps or overlaps beyond some tolerance. In these methods, which so far have been limited to geometric motion, Young's law is not an explicit constraint, but results automatically from the minimization.
- *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 [12,17]. The interfacial layer must be resolved sufficiently well for numerical accuracy.
- *Diffusion-generated methods*: These methods alternate between two steps: first, a diffusion step corresponding to convolution with a kernel, and then a reconstructive step. Here, the phases are typically represented using either multiple level set functions or multiple characteristic functions, and Young's law is automatically satisfied. These methods can be used to generate mean curvature flow and can be made unconditionally stable. See for example [11], which shows the modeling of curvature flow on a system with tens of thousands of phases.

1.3. Overview of the Voronoi Implicit Interface Method

1.3.1. New contributions

The Voronoi Implicit Interface Method was first introduced in [23]. In this paper, we analyze the method and provide numerous new extensions. In particular:

- We present the mathematical formulation of moving multiphase multiple junction interface problems in more detail.
- We consider different approaches in numerically approximating the mathematical formulation, and analyze their applicability across a range of problems.
- We provide extensive convergence tests verifying the accuracy of the method, including two- and three-dimensional analyses of motion in the presence of complex topological changes.
- We present extensions to different types of evolution, including energy minimization with different surface energy densities.
- We couple the method to incompressible fluid mechanics in which density, viscosity and surface tension can be defined on a per-phase basis, showing both two- and three-dimensional simulations.

1.3.2. Summary of the VIIM

The Voronoi Implicit Interface Method has a variety of important features, including:

- Mathematical features:
  - *Single representation*: The general method, in its mathematical form, uses only one function for an entire multiphase system, plus an additional indicator function.
  - *Accurate representation of geometric quantities*: Key geometric quantities required for computing interface speeds, including normal direction and curvature, are calculated in a natural manner without explicitly reconstructing the interface between phases.
  - *Coupling with physics*: The equation of motion for the interface evolution couples naturally with underlying physics in such a way that time has physical meaning. The physics influences the evolution of the multiphase system, and the evolution of the multiphase system influences the physics.
  - *Dimension independent*: The formulation works in any number of spatial dimensions with no change to the fundamental algorithm.
  - *Multiple junctions and topological change*: The formulation automatically deals with the evolution of triple points, triple lines, and topological change in the multiphase system. Regions can disappear and new phases may be created.
  - *Consistency of multiphase motion*: By construction, regions of overlap or vacuum separation cannot occur.
  - *Consistency with level set methods*: The formulation captures the same motion as the standard level set method in the case of only two phases.

- Numerical/algorithmic features:
  - *Discretization*: The method works on a fixed Eulerian spatial mesh. We used a straightforward finite difference approximations on a rectangular grid; a finite element formulation on an unstructured mesh is also possible.
  - *Data structures*: At any time step, each computational mesh point carries only a pair of function values, independent of the number of phases.
  - *Accuracy*: The method is first order accurate at triple points/lines (and higher order junctions), and can be made of arbitrarily high order away from these degeneracies on smooth interfaces between two neighboring phases.
  - *Efficiency*: Using narrow banding, the computational complexity of the method does not depend directly on the number of phases, but instead depends only on the number of grid cells containing the interface.

The outline of the paper is as follows. First, after a brief review of level set methods, we introduce the basic Voronoi Implicit Interface Method in Section 2. Aspects of the corresponding numerical algorithm and implementation are then discussed in Section 3, followed by convergence tests for various types of interface evolution in two and three dimensions, including topological changes and different boundary conditions in Section 4. We then demonstrate in Sections 5 and 6 various applications of the VIIM, including different types of geometric evolution, and motion coupled to incompressible fluid dynamics.

2. Mathematical formulation

2.1. Level set methods

As background, we briefly summarize a level set formulation for an interface separating two phases. (For an introduction, including numerical methods and applications of the level set method, see [27].) We start with an $n-1$ dimensional hypersurface Γ in $\mathbb{R}^n$, which is the location of the interface boundary between the two phases at time $t = 0$, as well as a speed function F defined on the interface. We then embed Γ as the zero level set of a signed distance function $\phi(x, t = 0)$, with the sign chosen to be positive in one region and negative inside the other. The associated level set evolution equation is an initial value PDE which evolves ϕ in time in such a way that the zero level set always corresponds to the evolving front, namely

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

This formulation embeds the interface in a higher-dimensional function ϕ defined throughout the entire domain and, hence, adds unnecessary computational labor. More sophisticated versions employ the Narrow Band Level Set Method introduced in [1], which limits the signed distance function to a small neighborhood around the evolving front, and hence reduces the computational complexity to roughly the number of elements on the front. We additionally note that the above equation assumes that the speed function F exists off of the interface itself: efficient ways of building such “extension velocities” were introduced in [2]. Finally, we note that periodic reinitialization may be required: an accurate reinitialization method based on bicubic/tricubic interpolation was introduced in [9].

2.2. The Voronoi reconstruction

Recall that the Voronoi diagram of a set of nodes in $\mathbb{R}^n$ is a tessellation of $\mathbb{R}^n$ into Voronoi cells, defined so that all the points in each cell are closer to one particular node than to any other node. 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 subdivision of $\mathbb{R}^n$ such that all points in a particular cell are either inside one of these regions or closer to it than any other region. The *Voronoi interface* is defined to be the boundary of these cells.

The key idea in our approach is to rely on one of the main features of level set methods: the interface is embedded as the zero level set of an implicit function (most commonly, the signed distance function to the interface), and the ensuing level set function is updated according to an initial value PDE, defined everywhere in the domain Ω. This means that the motion of a level set (including the zero level set) is bracketed by the motion of the surrounding neighboring level sets. This property is an immediate consequence of the comparison theorem under suitable restrictions on the speed function that moves the level sets, and these restrictions are often satisfied through the construction of extension velocities developed in [2].

In the Voronoi Implicit Interface Method, we utilize this property by letting the evolution of a multiphase system be determined by hypersurfaces that are nearby the interface. 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 surfaces each existing solely in one phase. While an interface in a multiphase system may have intricate complexity where multiple phases touch (such as at triple points, triple lines, etc.), neighboring hypersurfaces will be remarkably well behaved as the system evolves. The motion of the interface between multiple phases is constructed from the position of these nearby hypersurfaces, using the Voronoi interface of these hypersurfaces.

Less abstractly, we first embed the interface of a multiphase system as the zero level set of an *unsigned* distance function ϕ. We also embed the speed function F that moves this interface in a higher dimensional function as well to create an “extension velocity" which smoothly varies away from the interface, satisfies certain properties, and defaults to the given velocity on the interface itself: this idea was introduced in [16], and efficient ways to construct this extension velocity were presented in [2]. The level set evolution PDE is then applied to ϕ for a small time Δt. Under this motion, the zero level set of ϕ will generally not remain a co-dimension one surface. However, and this is the key point, for a sufficiently short time, nearby level sets (with value $\epsilon > 0$, say) will remain a co-dimension one surface. One can then reconstruct the interface after time Δt as the Voronoi interface of the nearby level sets of ϕ with value ϵ.

2.3. One- and two-dimensional examples

We illustrate with a simple one dimensional example (see Fig. 1). Imagine an interface, which is a single point located at $X(t)$ at time t, moving with speed $F(x,t)$, thus $\frac{dX(t)}{dt} = F(X(t),t)$. 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 update the distance function everywhere away from the zero level set in a straightforward manner. 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 front 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.

In higher dimensions, and in the presence of triple points, this same construction works. In Fig. 2(left), we see a triple point of an unsigned distance function, which is non-negative everywhere. In Fig. 2(middle), we show the ϵ-level sets, and in Fig. 2(right), the Voronoi reconstruction from these ϵ-level sets.

2.4. 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 a *ϵ-smoothed solution* ϕ_ϵ, given by

$$\phi_\epsilon = \lim_{\Delta t \to 0, n\to\infty} (V_\epsilon \circ E_{\Delta t})^n(\phi_0), \tag{1}$$

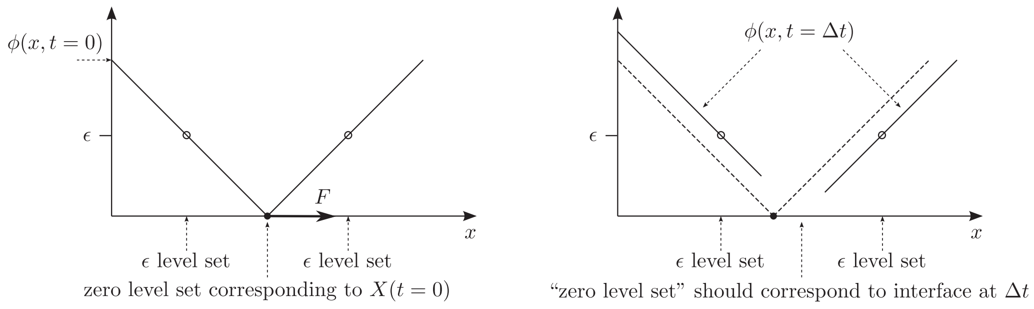

Fig. 1. Evolution of an unsigned distance function ϕ for an interface in one dimension.

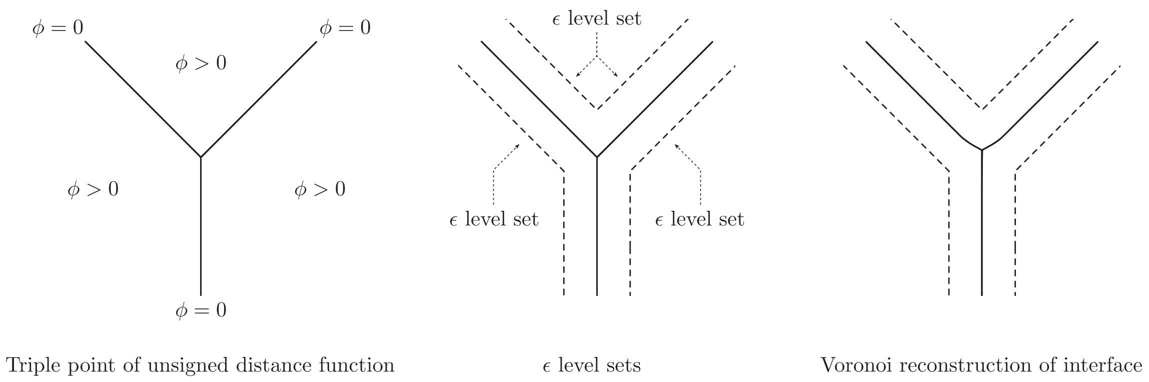

Fig. 2. Reconstruction of a triple point from an unsigned distance function and the Voronoi interface.

where ϕ_0 is the initial condition. We point out that this construction defines ϕ_ϵ at intermediate times $0 \leqslant t \leqslant T$ as well as the final time T.

We can now take the limit of these ϵ-smoothed solutions as $\epsilon \to 0$ from above, to obtain a solution given by

$$\phi_{\epsilon=0^+} = \lim_{\epsilon\to 0^+} \phi_\epsilon. \tag{2}$$

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 (2).

2.5. 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.

We suggest here that the above mathematical formulation of multiphase interface evolution is the correct way to formally define the solution to multiphase problems. As we show in our results below, it does indeed provide a solution that is physically relevant. This formulation may serve as a possible way to analyze mathematically these flows, however the mathematical analysis is not straightforward. We shall report on this elsewhere [21].

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

Our goal now is to build numerical approximations to the formal definition given by the $\epsilon = 0^+$ limit in Eqs. (1) and (2). In this section, we discuss different approaches of discretizing time and space to find the limit and what key algorithmic components are needed to achieve this. This paper focuses on an implementation that uses standard finite difference approximations to spatial and temporal derivatives on a rectangular grid. We emphasize however that the VIIM approach is suited to many other settings (finite elements, unstructured meshes, etc.), and we make some additional comments on this later. In this section, we also comment on the efficiency of the discrete algorithm, as well as a possible parallel implementation with MPI.

3.1. Spatial and temporal discretizations of the formal $\epsilon = 0^+$ limit

We begin with the usual discretizations having grid cell size h and time step Δt. We write our formal definition in more detail in two steps as

$$\phi_\epsilon = \lim_{n\to\infty, h\to 0} \left(V_\epsilon^h \circ E_{\Delta t}^h\right)^n (\phi_0), \tag{3}$$

followed by the limit

$$\phi_{\epsilon=0^+} = \lim_{\epsilon\to 0^+} \phi_\epsilon. \tag{4}$$

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. We assume that the time step Δt is suitably linked to h, as is usually required for stability. The VIIM contains an additional parameter ϵ, corresponding to the choice of which neighboring level sets to use to rebuild the unsigned distance function.

The formal definition, given by $\phi_{\epsilon=0^+}$, is the limit as $\epsilon \to 0^+$ of the limit as the space and time discretizations 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* $\phi_{\epsilon=0^+}$:
A discretized version of the formal limit given in (4) is to fix $\epsilon > 0$, compute a converged (in space and time) solution corresponding to the ϵ-smoothed solution given in (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 any 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 ϵ goes to zero to obtain $\phi_{\epsilon=0^+}$.

We note several important aspects of this approach:
- Imagine some CFL-type requirement relating Δt to h. Then, as h and Δt go to zero, we obtain a converged solution to the ϵ-smoothed solution.
- The physical region between the ϵ-level sets and the multiphase interface is resolved as the grid cell size goes to zero.
- If narrow bands are used to reduce the computational complexity, the size of these narrow bands increases as the grid is resolved, since they must cover all of the domain between the ϵ-level sets and the interface.

In the tests below, we use this approach to validate that this approximation produces a converged limit to our formal definition.

(ii) *The coupled formal limit $\phi_{\epsilon=0^+}$: couple ϵ and h, such that $\epsilon > 2h$:*
Here, we couple ϵ 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. This means that we always use the same grid resolution between the ϵ-level sets and the multiphase interface: as the grid size is decreased, the value of ϵ is correspondingly reduced. 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 emphasize that the choice of h dictates the value of ϵ.
- We then study convergence as h and Δt go to zero to obtain $\phi_{\epsilon=0^+}$.

We note several important aspects of this version:
- Linking ϵ and h, and choosing $\epsilon > 2h$, is an easy method to implement, and converges to the correct solution (see Section 4). Here, we choose $\epsilon > 2h$ 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.
- Setting ϵ to be a multiple of h means that the number of grid cells in the narrow band does not need to increase as h decreases.

In the tests below, we select several cases to verify that this limiting process, which couples to ϵ to h, obtains 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 $(\Delta t, h) \to 0$:*
Alternatively, the limits in (3) and (4) may be exchanged: first compute the inner limit as ϵ goes to zero, to produce a grid/time discretized solution $\phi^{\Delta t,h}_{\epsilon=0+}$, and then evaluate this solution as Δt, h vanish. More precisely:
- Compute $\phi^{\Delta t,h}_{\epsilon=0+}$ by finding $\lim_{\epsilon\to 0^+}\phi^{\Delta t,h}_{\epsilon}$.
- Now, take the limit as Δt, h goes to zero to produce $\phi_{\epsilon=0+} = \lim_{(\Delta t,h)\to 0}\phi^{\Delta t,h}_{\epsilon=0+}$.

We note several important aspects of this version:
- This approach uses the motion of the ϵ-level sets, in the limit ϵ goes to zero from above. This limit can be evaluated directly rather than through a limiting process.
- One approach is to use 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, we 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. This method is used below in the convergence tests. However, this requires considerable programming complexity, and incurs additional computational expense in building the extensions.

In the tests below, and similar to the approaches (i) and (ii) above, we use this approach to verify that it obtains the same solution as the formal definition.

3.2. Key algorithmic steps

Here we discuss some of the key steps in finding the discrete solutions using the different approaches presented above. We shall not give details on the finite difference schemes, and instead refer the reader to standard references on level set methods, see for example [27]. In each of the approaches above, 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.

3.2.1. The Voronoi interface

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

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

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

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

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

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 set. In other words, we define

$$\Gamma_V = \{x \in \Omega : \text{ there exists } i \neq j \text{ such that } d(x,\Gamma_{\epsilon,i}) = d(x,\Gamma_{\epsilon,j}) \leqslant d(x,\Gamma_{\epsilon,k}) \text{ for all } k \neq i,j\},$$

where $d(x,\Gamma)$ is the distance between a point x and hypersurface Γ. (This formulation is useful when pursuing a front tracking approximation to VIIM, as mentioned later.)

- *Construction via solution of Eikonal equations*: A related formulation is to pose a boundary value Eikonal equation, 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

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

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 $\Gamma_{\epsilon,i}$,

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

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

$$\Gamma_V = \left\{ x \in \Omega : \text{ there exists } i \neq j \text{ such that } \phi_i(x) = \phi_j(x) \geqslant \max_{k \neq i,j} \phi_k(x) \right\}. \tag{5}$$

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.

3.2.2. 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

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

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 utilizing 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 (5), then the new indicator function is given by $\chi(x) = \arg\max_i \ \phi_i(x)$.

3.2.3. Solving the Eikonal equation

In our work, we have extensively used the Eikonal equation method of rebuilding the unsigned distance function. This is advantageous since the ϵ-level sets and the Voronoi interface need never be explicitly constructed. To do this, we need to solve the Eikonal equation on a fixed mesh with a robust, fast and accurate method. We make extensive use of the accurate and fast Eikonal reinitialization algorithm developed by Chopp (see [9]), which utilizes bicubic (tricubic in 3D) interpolation to accurately initialize the Fast Marching Method [26]. In more detail, if a grid cell is identified to contain the zero level set of a function ψ (which need not be a distance function), then ψ is interpolated in that cell using a bicubic/tricubic patch. A Newton solver is then used to find the closest point on the interpolated interface, from which the exact distance to the interpolated interface is computed at the nodes of all such grid cells. This procedure creates a small initial band that can be input to the efficient Fast Marching Method [26], which is a Dijkstra-like ordered upwind finite difference scheme for solving the full Eikonal equation outside this initial band. A different Dijkstra-like control theoretic discretization of the Eikonal equation stemming from optimal control was developed in [32], and we refer the reader to [29] for a detailed discussion and extensions of Fast Marching Methods to general front propagation problems.

3.2.4. Adding physics and evaluation of the speed function F

A speed function F which captures the appropriate physics must also be specified with our multiphase problem. At each time step, we imagine that we are given a speed function F, defined in all of space and which can depend on position, normal, and curvature. This speed function is produced 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 additional 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, our formulation leads to a natural meshing algorithm, which will be presented below.

We note again that the formulation of the VIIM, by construction, cannot create overlaps or vacuums: each point in the domain is part of a unique phase at all times. Our reconstruction differs from existing algorithms in that there is neither a "no overlap/vacuum" condition, nor is there a penalty term that penalizes region of vacuums or overlap. The interface is always well-defined at every time step. This means that relevant physics in each phase correctly interact, and there is no need to invent physics in regions of "vacuum".

3.3. 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 interface 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 material on both sides of the interface is given the same value for the indicator function.

3.4. Efficiency

To obtain computational efficiency, we use the ideas of the Narrow Band Level Set Method [1]. In particular, the unsigned distance function ϕ is only defined 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, 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.

3.5. Interface extraction and visualization

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 visualization to display the evolving structures. In the two-phase case, there are several standard visualization algorithms extracting level sets of implicit functions in two and three dimensions, including marching cubes [14], marching tetrahedrons, and their variants.

In our case, we have a more complex, multiphase structure. Here, we introduce a new extraction algorithm applicable to multiphase flow, and which capitalizes on the Voronoi interpretation, coupled to aspects of marching tetrahedrons. Using the approach of solving Eikonal equations to find the Voronoi interface, as given in Section 3.2.1, 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 (5):

$$\Gamma_{ij} = \{x \in \Omega : \phi_i(x) = \phi_j(x) \geqslant \phi_k(x) \quad \text{for all } k \neq i,j\}. \tag{6}$$

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 $\{E_\ell : 1 \leqslant \ell \leqslant n\}$, where each E_ℓ is a straight line segment in 2D or a triangle in 3D. Since the last condition in (6) 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 $\phi_i(x) = \phi_j(x) \geqslant \phi_k(x)$ 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 $\phi_i \geqslant \phi_k$. Pieces on which $\phi_i < \phi_k$ are thrown away. The result is a collection of surface elements whose union is Γ_{ij}.

We note several features about the above procedure.

- Although we used piecewise linear interpolation, in principle, any interpolation in which the resulting surface elements can be chopped accurately could be used, for example, quadratic surface elements chopped by other quadratic surface elements.
- 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 visualization purposes, since line segments and triangles are naturally displayed, and chopping such elements with other elements is straightforward. We additionally note that piecewise linear interpolation is second order accurate wherever the function being interpolated is smooth, which is accurate enough for many purposes. We use a marching tetrahedrons based interpolation, in which each grid cell is divided into two triangles (in 2D) or six tetrahedrons (in 3D). This provides a well defined, predictable piecewise linear interpolant, unlike in March-

ing Cubes where the structure of the interpolant can change between cells and indeed between different interfaces in the same cell.
- 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, yields efficient extraction of mesh elements and efficient chopping operations. The total operation count 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.
- Unlike methods based on contour plotting or triangulations, the use of the Voronoi reconstruction produces a result with no artifacts at junctions where multiple interfaces meet.

3.6. Parallel implementation using MPI

We have developed a parallel framework, 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. In this work, the domain is a simple rectangular Cartesian grid, and this is sub-divided into smaller grids that are assigned to individual processors in the MPI implementation. Synchronization of data on the subgrids is performed using ghost layers of sufficient size. Many algorithms can be parallelized easily with this approach, while other algorithms require more care, such as the solution of Eikonal equations and elliptic equation solvers.

To parallelize the Eikonal equation solver, we 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 subgrid 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 initialized with bicubic/tricubic interpolation, the normal Fast Marching Method can be used. In this fashion, each processor can solve the Eikonal equation locally and independentally of the other processors. The parallel efficiency of this method depends on the geometry of the interface and the relative size of the subgrid vs. narrow band: if the interface is sufficiently dense, as it often is in multiphase simulations, the method scales very well.

For fluid flow equations, it is usually the Navier–Stokes solver that needs the most computing time, due to requiring the solution of an elliptic equation (for pressure) at each time step. Here, we have parallelized a multigrid solver using standard approaches that use, for instance, processor idling. We have tested the parallel efficiency of the solver, and found that multiphase fluid simulations scale very well to thousands of processors.

3.7. Alternative approaches

Finally, before turning to convergence studies based on our implementation of the VIIM on a regular Cartesian grid, we mention that other approaches are possible. Viewed from a very high level, our formulation is to

1. Advance the ϵ-level sets (defined as sets a distance ϵ 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.
3. Loop to 1.

In the above, we have concentrated on a level set finite difference methodology. We emphasize that these two steps may be approximated using a host of numerical technologies, such as finite element methods, semi-Lagrangian techniques, and front tracking approaches. For example, a front tracking method version may be constructed as follows:

- First, given a collection of phases, build a discrete approximation of the set of points located a distance ϵ from the multiphase interface. In two dimensions, this discrete approximation might consist most naturally of connected line segements; in three dimensions, a triangulated representation might be most natural. Regardless, the important salient fact is that each of these discrete approximations will lie in a single phase.
- Second, update their positions by advancing the discretization in a Lagrangian manner, which typically involves updating the nodes of the discretization.
- Finally, find the Voronoi interface to build a new multiphase interface and repeat.

We will discuss these alternative numerical technologies for approximating the solution to the VIIM elsewhere [22].

4. Convergence tests and verification

In this section, we perform convergence tests of our methods in two and three dimensions, verifying the accuracy, robustness, and convergence properties under refinement of numerical parameters. We consider three different ways to test convergence in our method, as outlined in Section 3.1:

(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 our mathematical definition of multiphase evolution, and produces a converged solution. In the following, we denote this approach 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. Our results will show that coupling ϵ to h in this manner also produces the same, converged solution. We denote this approach 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 labeled 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 we shall use grid convergence. 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 co-dimension one), we let

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

Roughly speaking, the Hausdorff distance measures the maximum width of the region between Γ_1 and Γ_2. In our convergence tests we measure convergence in time as well as in space. We use an averaged L^1 norm in time[1] and 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

$$d(\Gamma_1, \Gamma_2) = \frac{1}{T}\int_0^T d_H(\Gamma_1(t), \Gamma_2(t))\; dt. \tag{7}$$

To numerically evaluate (7) 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 our purposes.

In all of our tests, the domain is a unit square in 2D (or a unit cube in 3D) and we use a uniform Cartesian grid with cell size h. 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. We measure convergence as $h \to 0$ 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 in the usual fashion we measure the difference of two solutions on two different grids, in our case using $d(\Gamma_h^\epsilon, \Gamma_{2h}^\epsilon)$, and calculate ratios between these differences 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, we note that 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).

4.1. Two-dimensions: basic tests

We first verify that our method produces correct results in the case of straightforward two-phase problems. While these could easily be modeled with traditional level set methods, we include these for completeness.

4.1.1. Circle expanding with unit speed

We first test the case of a circle expanding with unit speed $F = 1$, as illustrated in Fig. 3(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 $\Delta t < h$ and we have fixed $\Delta t = \frac{1}{2}h$. 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 our results confirm this. Fig. 3 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. Fig. 4 plots the error compared with the exact solution, $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$, and shows that we obtain first order accuracy in space and time, independent of the choice of ϵ.

4.1.2. Initially square interface expanding with unit speed

We now consider an initially square interface expanding with unit speed $F = 1$, as illustrated in Fig. 5 (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 $\Delta t < h$

[1] We also ran tests using the maximum norm in time, e.g. $\max_{0\leqslant t\leqslant T} d_H(\cdot,\cdot)$. Convergence is still obtained under this norm as well, with the same overarching 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, we use the L^1 norm in time, which for the tests we perform here, does not weaken any conclusions we make about convergence.

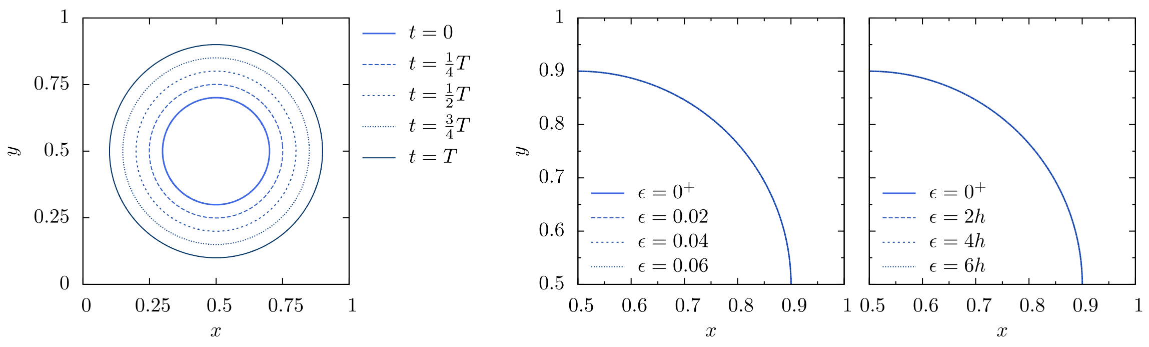

Fig. 3. Evolution of a circle expanding with unit speed $(T = \frac{1}{5})$ (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

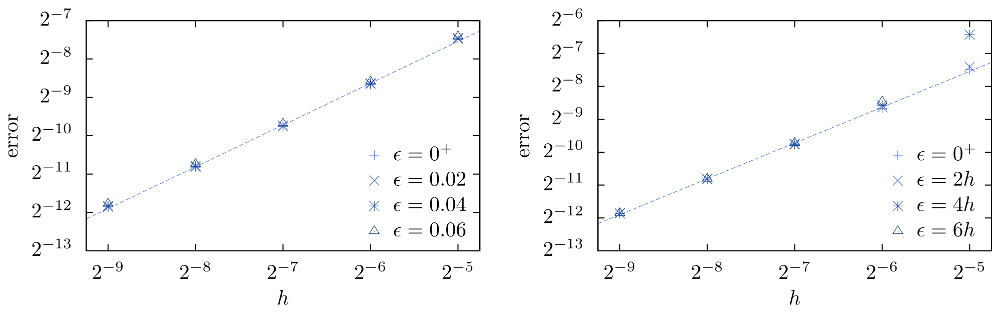

Fig. 4. Corresponding to the test case in Fig. 3, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 1.0.

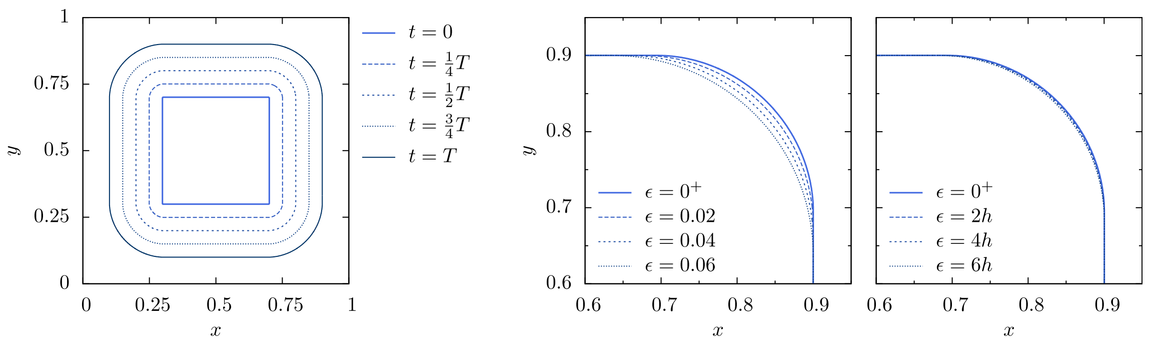

Fig. 5. Evolution of an initially square interface expanding with unit speed $(T = \frac{1}{5})$ (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

and we have fixed $\Delta t = \frac{1}{2}h$. In this test case, due to the corners present in the initial interface, the Voronoi reconstructed interface changes with the choice of ϵ. Fig. 5 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 Fig. 5(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 1 and Fig. 6. In Fig. 6(left) we see that if $\epsilon > 0$ is fixed, then convergence to the correct solution is not obtained. However, in Table 1, and in the same regime, we observe that despite not obtaining convergence to the correct solution, the numerical results are nevertheless converging (to the ϵ-smoothed solution). In the regime where ϵ is proportional to h, we obtain approximate first order convergence to the solution as $h \to 0$.

4.1.3. Square shrinking with unit speed

Now we consider the case of a square shrinking with unit speed, as illustrated in Fig. 7(left). The square has an initial width of $\frac{4}{5}$ and is evolved over a time $T = \frac{1}{5}$ to finish with a width of $\frac{2}{5}$. Similar to before, the Voronoi reconstructed interface has an error depending on the choice of ϵ. Fig. 7 shows the numerical results at final time $t = T$ obtained on the same grid size

Table 1
Corresponding to the test case in Fig. 5, convergence results for the fixed ϵ regime.

h	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

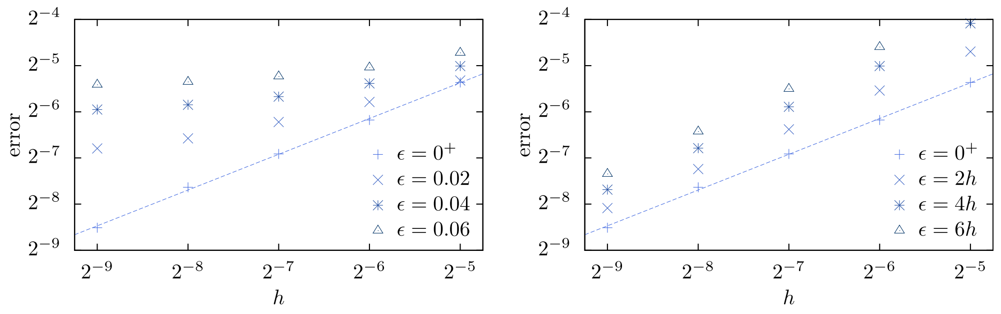

Fig. 6. Corresponding to the test case in Fig. 5, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 0.8.

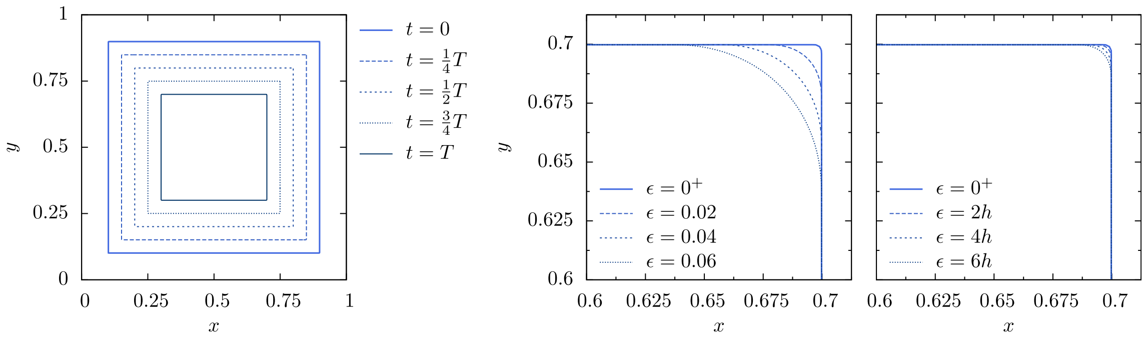

Fig. 7. Evolution of a square shrinking with unit speed ($T = \frac{1}{5}$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

($h = 1/512$) for different ϵ values. In particular, in Fig. 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 is presented in Table 2 and Fig. 8, which show similar convergence properties to the correct solution as in the previous test.

4.1.4. Curvature flow on a circle

We next consider curvature flow applied to a circle with speed $F = \kappa$, as illustrated in Fig. 9(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 we use standard finite difference stencils that 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 $F_{\text{actual}} = F_{\text{exact}} + \mathcal{O}(\epsilon^2)$.) Fig. 9 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 quantitative in Fig. 10 and Table 3. In the regime where ϵ is proportional to h, we obtain second order convergence to the solution as $h \to 0$, independent of the constant of proportionality. In this case, second order convergence is obtained because we are using second order finite difference methods and forward Euler time stepping with a second order time step $\Delta t = \mathcal{O}(h^2)$.

Table 2
Corresponding to the test case in Fig. 7, convergence results for the fixed ϵ regime.

h	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

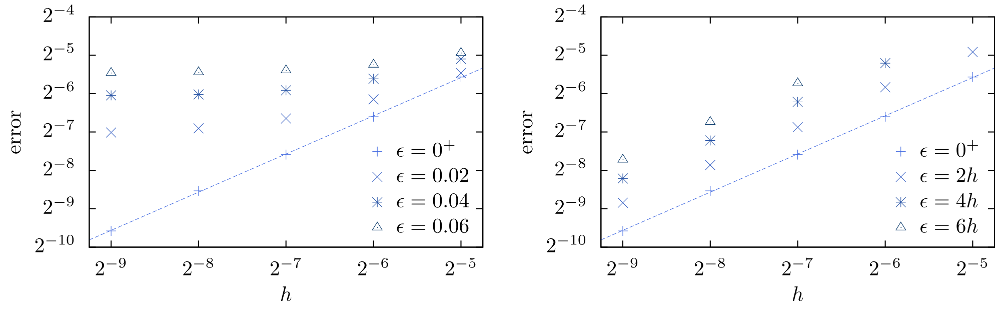

Fig. 8. Corresponding to the test case in Fig. 7, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 1.0.

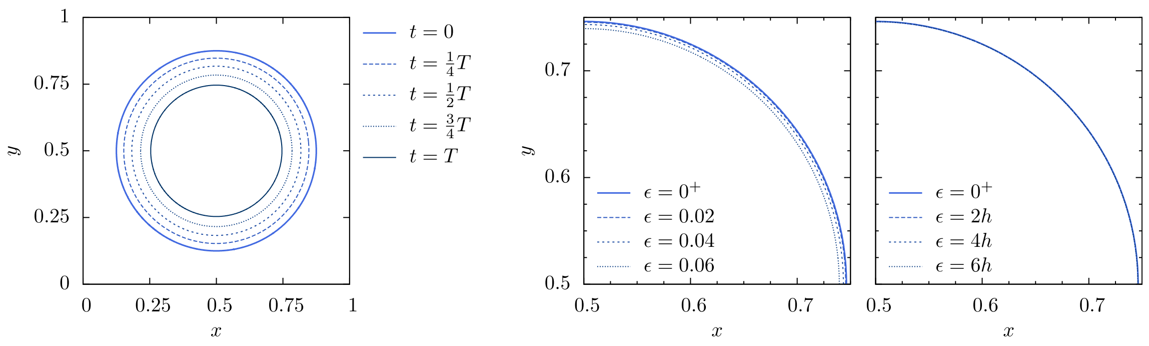

Fig. 9. Evolution of a circle collapsing under curvature flow with speed $F = \kappa$ ($T = 0.04$) (left), and the solution at time $t = T$ obtained on the same grid ($h = 1/512$), with ϵ fixed (middle) and ϵ coupled to h (right).

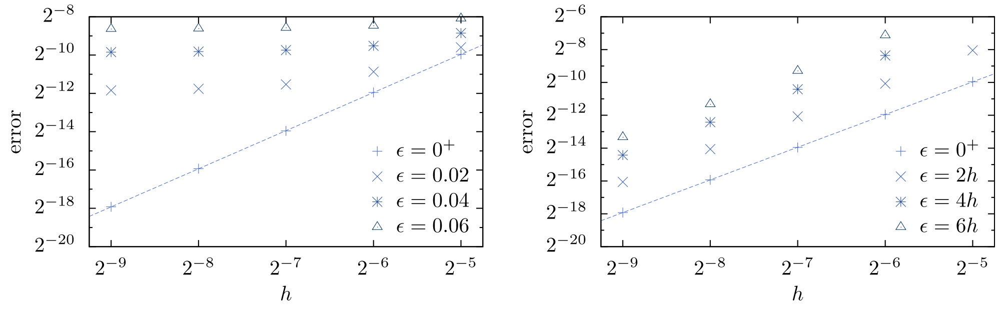

Fig. 10. Corresponding to the test case in Fig. 9, the error $d(\Gamma_h^\epsilon, \Gamma_{\text{exact}})$ as a function of grid size h for the two regimes: ϵ fixed (left), and ϵ coupled to h (right). The slope of both lines is 2.0.

4.2. 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, we set $\Delta t = \frac{1}{4}h^2$ and evolve over a time

Table 3
Corresponding to the test case in Fig. 9, convergence results for the fixed ϵ regime.

h	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

of $T = \frac{125}{512} \approx 0.244$ (corresponding to 1000 time steps on the coarsest grid). Fig. 11 illustrates the evolution for both types of boundary conditions. We see 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.

Fig. 12 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 convergence test, the exact solution is not known and so we rely on grid refinement to measure convergence. Table 4 (fixed ϵ regime) and Table 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, we also verified that in the fixed ϵ regime, that in the limit $\epsilon \to 0$, all solutions ultimately converged to the same solution.

4.3. von Neumann–Mullins' law in 2D

4.3.1. 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 view-

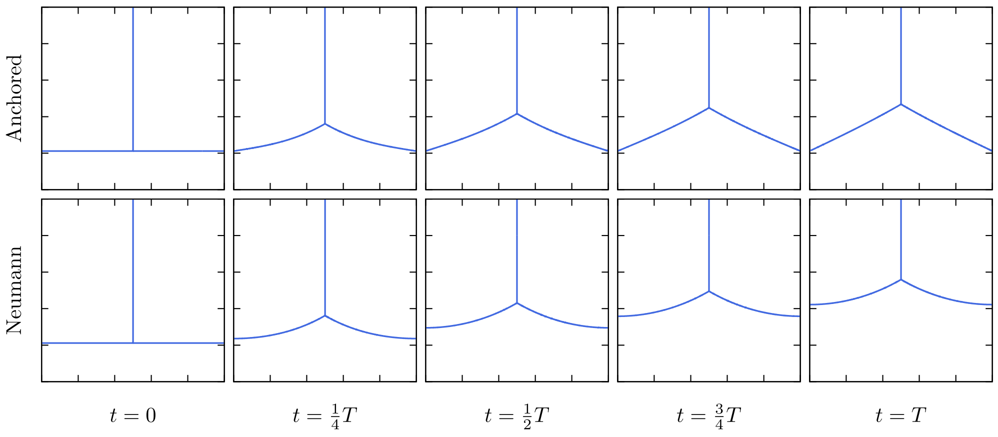

Fig. 11. Evolution of curvature flow applied to an interface that is initially a T junction ($T = \frac{125}{512}$), with anchored boundary conditions (top) and Neumann boundary conditions (bottom).

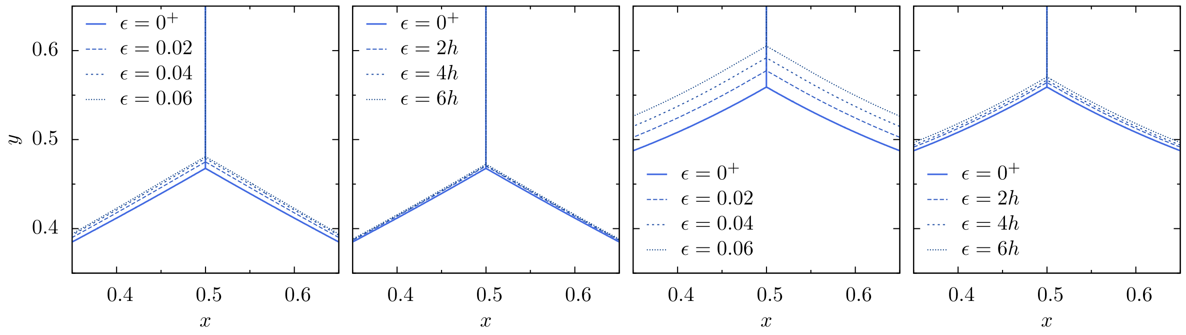

Fig. 12. Corresponding to the test in Fig. 11, the solution at time $t = T$ obtained on the same grid ($h = 1/512$) with different choices of ϵ for the two different ϵ regimes; anchored boundary conditions (left pair) and Neumann boundary conditions (right pair).

Table 4
Corresponding to the test in Fig. 11, convergence results for the fixed ϵ regime.

	h	$\epsilon = 0^+$		$\epsilon = 0.02$		$\epsilon = 0.04$		$\epsilon = 0.06$	
		d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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 5
Corresponding to the test in Fig. 11, convergence results for regime in which ϵ is coupled to h.

	h	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
		d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

ing curvature flow as minimizing 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 and Mullins 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

$$\frac{dA}{dt} = 2\pi\gamma\left(\frac{n}{6} - 1\right). \tag{8}$$

Our algorithm assigns a true meaning to time evolution and this relation was used in [23] to test convergence of the VIIM, and in Figs. 13 and 14, we perform a test of this law. We start with a set of 25 randomly positioned phases and apply curvature flow (with $\gamma = 1$), using periodic boundary conditions (grid size 256×256). As the system evolves, a phase changes area according to the above law: the area vs. time plot should be a straight line until the number of sides changes due to topological changes in the network. Once this happens, this creates a new constant rate $\frac{dA}{dt}$ for a given phase, depending on the number of sides, so that the area of each phase as a function of time is piecewise linear. Fig. 13 illustrates the evolution. We pick nine of the initial phases, and, for each of the nine initial phases, we plot the evolution of area in time in Fig. 14. The results show correct match with von Neumann–Mullins' law throughout the evolution.

4.3.2. 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.

We first make some remarks about the reconstruction frequency, i.e. how often the unsigned distance function is reconstructed from the Voronoi interface. In the basic VIIM algorithm presented above, the interface is rebuilt after every time step. However, we have found that it can be advantageous to reconstruct less frequently, say every 10 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 a multiphase system. The optimal balance depends on the exact application and the time step being used.

To test the numerical results of the VIIM against von Neumann–Mullins' law, we consider n sided shapes, for $n = 4$, 6 and 8, suitably connected to the boundary (see Fig. 15). (In addition to the tests shown here, we have also tested cases with $n = 3$, 5, 7.) The initial shape is chosen to be circular with radius 0.3, 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 our convergence tests, we examined the role of reinitialization frequency/reconstruction interval in convergence. 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, we found that consistent convergence rates were obtained for approximately $k \geqslant 8$,

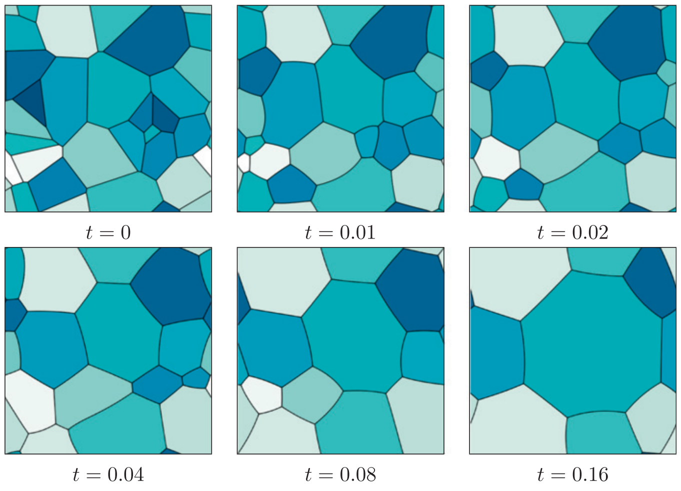

Fig. 13. Curvature flow applied to an initial set of 25 randomly positioned phases. According to von Neumann–Mullins' law, phases with more than six sides grow, those with less than six shrink, and those with six sides conserve their area. This process leads to "coarsening" of the system.

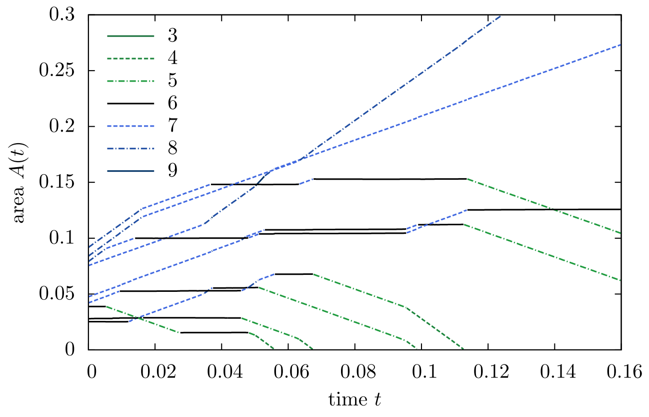

Fig. 14. Area as a function of time for a selected set of nine phases corresponding to the evolution shown in Fig. 13. According to von Neumann–Mullins' law, the area of a particular phase should be a piecewise linear function, with a derivative that is an affine function of the number of sides of the phase. We have therefore colored 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.

and we have set $k = 16$ in the following. 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 by von Neumann–Mullins' law. To simplify matters, we skip over convergence for the regime where ϵ is fixed, and instead only consider the case when $\epsilon = \alpha h$ is coupled to the grid size h, as $h \to 0$. For each n, we have given:

- a time sequence illustrating the evolution of the n-sided shape over time (Fig. 15);
- a plot of the error in $\frac{dA}{dt}$ as a function of h (Fig. 16); and
- a table with convergence rates obtained with grid refinement, measuring convergence of the interface evolution in space and time (Table 6).

According to von Neumann–Mullins' law (8), 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. Including the test cases not shown here (with $n = 3, 5, 7$), in most cases we obtain well behaved convergence, 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

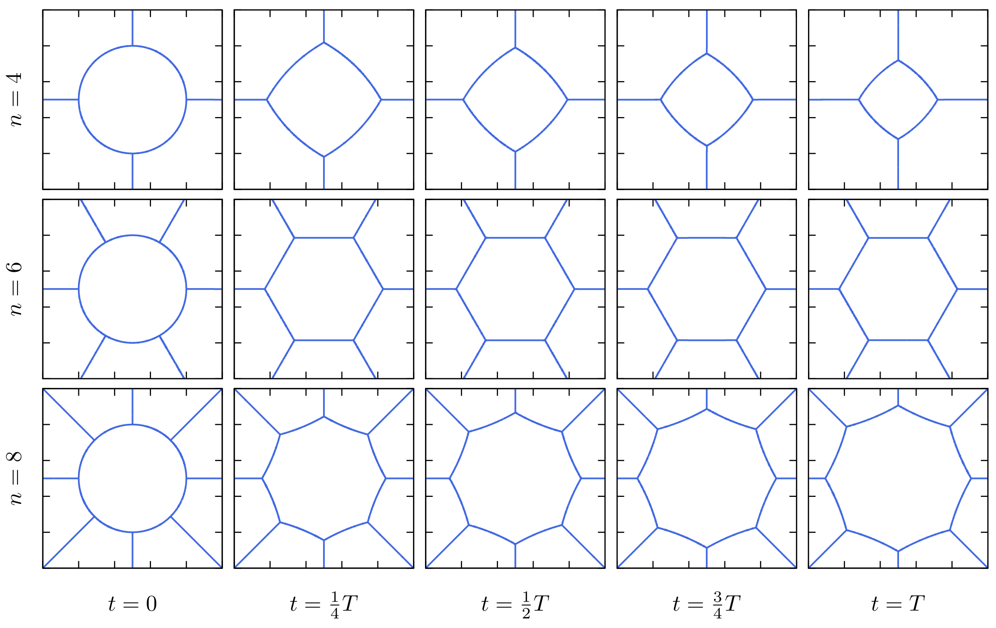

Fig. 15. Evolution of initially circular n-sided shapes under curvature flow with $\gamma = 1$ ($T = 0.08$).

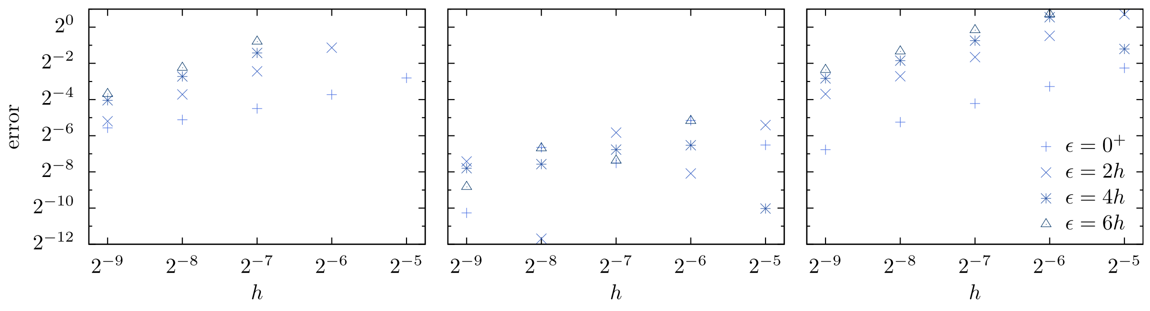

Fig. 16. Plot of the error in the computed value of $\frac{dA}{dt}$ as a function of grid size h for the n-sided shapes of Fig. 15, with $n = 4$ (left), $n = 6$ (middle) and $n = 8$ (right).

Table 6
Convergence results for the n-sided shape undergoing curvature flow in Fig. 15.

n	h	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
		d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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
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
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

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 based errors dominate.

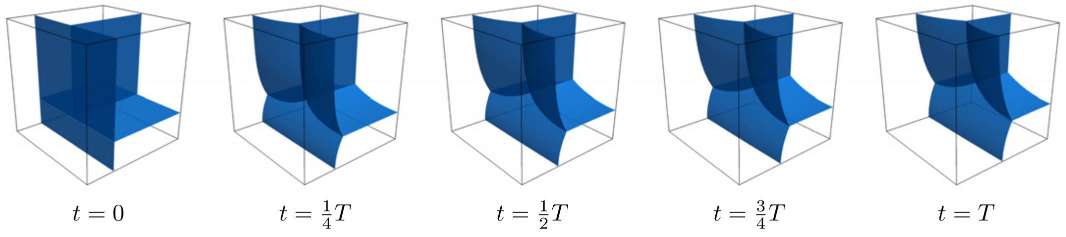

Fig. 17. Evolution of a three-dimensional analogue of a T junction subject to curvature flow ($T = \frac{125}{1024}$, computed on a 128^3 grid).

Table 7
Convergence results for the curvature flow illustrated in Fig. 17.

h	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

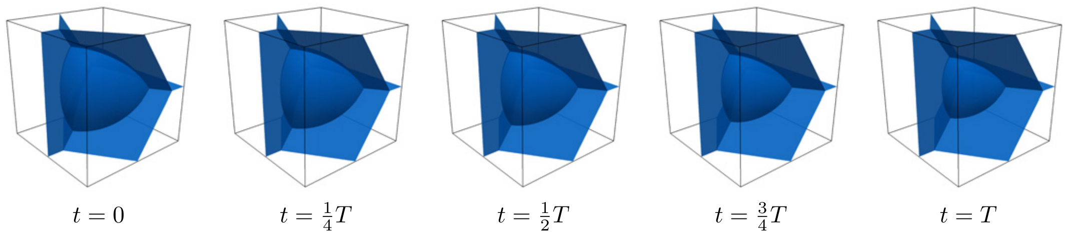

Fig. 18. Evolution of a Reuleaux tetrahedron shrinking under multiphase curvature flow ($T = 0.025$, computed on a 128^3 grid).

4.4. Three dimensions: evolution of triple lines

The VIIM easily extends to three dimensions and correctly handles the three-dimensional analogues of triple points. In Fig. 17, we repeat results from [23] showing 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 7 shows that the VIIM has first order convergence in both space and time.

4.5. von Neumann–Mullins’ law in three dimensions

The two-dimensional von Neumann–Mullins’ law was recently generalized to three or more dimensions and provides a formula for $\frac{dV}{dt}$ involving a “mean-width” [15], 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 the phase/region has. In [15], 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 our algorithm 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 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 Fig. 18. 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

[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 $\frac{d}{dt}V^{\frac{2}{3}} = \frac{2}{3}V^{-\frac{1}{3}}\frac{d}{dt}V = \frac{4}{3}C_V^{-1/3}C_S =: C \approx 5.28854633$.

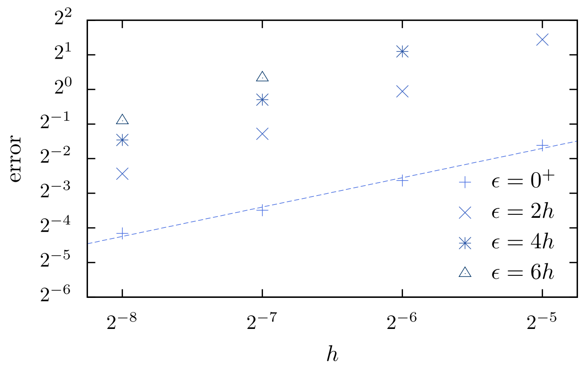

Fig. 19. Plot of the error in the computed value of $\frac{d}{dt}V^{2/3}$ for the shrinking Reuleaux tetrahedron in Fig. 18. The slope of the line is 0.85.

Table 8
Convergence results for the curvature flow illustrated in Fig. 18.

h	$\epsilon = 0^+$		$\epsilon = 2h$		$\epsilon = 4h$		$\epsilon = 6h$	
	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate	d_h^ϵ	Rate
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

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 Fig. 19 and Table 8 and shows that the VIIM once again obtains convergence as $h \to 0$ for every value of ϵ.

4.6. Summary of convergence results

The preceding shows that the Voronoi Implicit Interface Method converges under several different tests. Our convergence tests are in both time and space, verifying that the method correctly computes complex flows in a converged manner: we obtained first order convergence in almost all cases. The method yields the correct evolution in the case of two phase flow, convergence for triple junctions and multiply-connected regions, 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. We remark that 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.

5. Geometric flows

In this section, we illustrate the capabilities of the VIIM with various geometric examples in two and three space dimensions.

5.1. Constant 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 minimize 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 regularization properties.

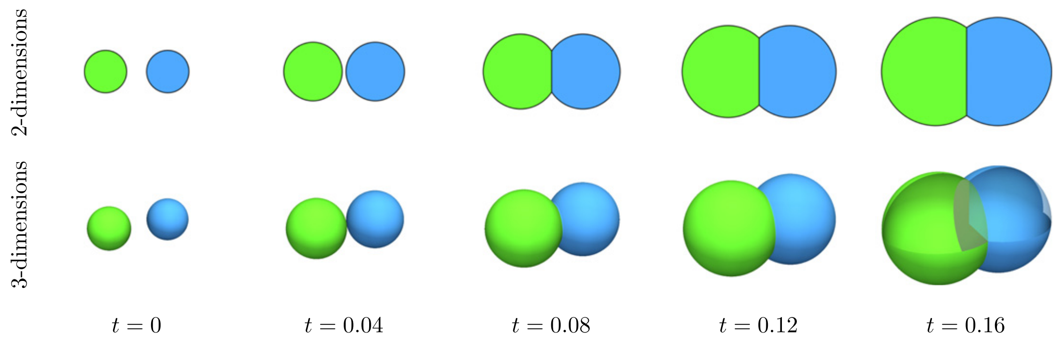

Fig. 20. Results of a constant normal driven flow such that Green and Blue grow with unit speed into White and stop at each other: $F_{GW} = F_{BW} = 1$, $F_{GB} = 0$. The initial radii of the circles (spheres) is 0.1 with a separation of 0.3 between their centers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

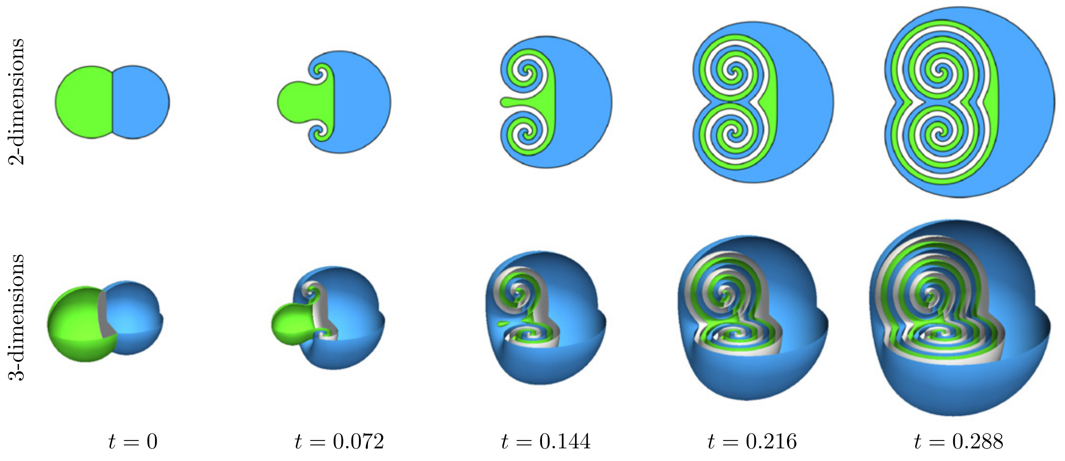

Fig. 21. Results of a constant normal driven flow where White expands into Green, which in turn expands into Blue, which in turn expands into White, all with unit speed ($F_{WG} = 1$, $F_{BW} = 1$, $F_{GB} = 1$). This flow generates two spiraling triple points in 2D and a spiraling triple line in 3D 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 centers. In the 3D case, we have cut away a quadrant to render the inside structure visible. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Fig. 20, we show 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 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, $F_{BG} = F_{GB} = 0$.[3] 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 Fig. 20, we have cut away a sector 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 $\epsilon = 0^+$.

In Fig. 21, we show 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 [20], 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 Fig. 21 of the VIIM applied to this problem correspond to adding an artificial surface tension with strength proportional to the grid size h.

[3] For interpretation of color in Figs. 20, 21, 31 and 32, the reader is referred to the web version of this article.

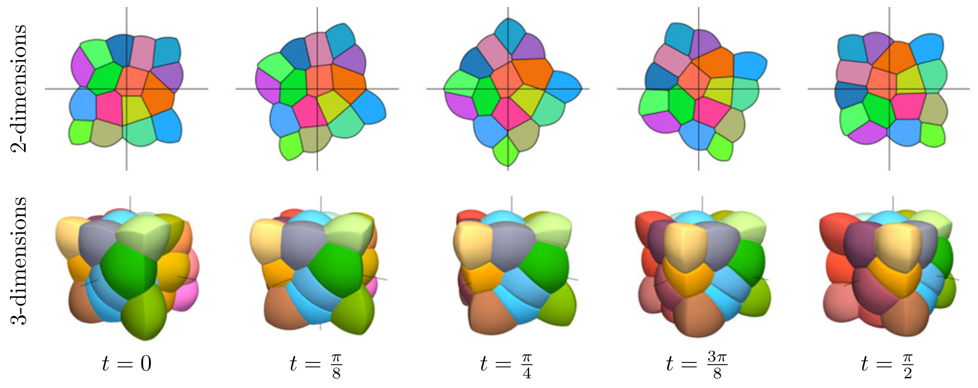

Fig. 22. Advection by an external velocity field that implements counterclockwise rotation such that $t = 2\pi$ corresponds to one complete revolution about the indicated origin (axis of rotation pointing up in the case of 3D).

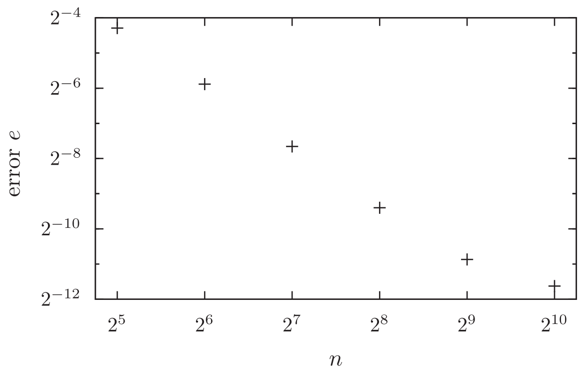

Fig. 23. Convergence analysis for area conservation corresponding to the two-dimensional simulation of Fig. 22. Here, the error is measured using (9) with $N = 17$ phases and a grid size of $n \times n$.

5.2. 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

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

In Fig. 22, we show both two- and three-dimensional simulations under a constant velocity field implementing rotation, using a simple second order ENO upwinding scheme for the advection term and forward Euler in time. Here, the simulation was computed on a 256×256 grid in 2D and a 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 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 $0 \leqslant t \leqslant T$ and summing this over each phase, i.e.

$$e = \sum_{i=1}^{N} \max_{t\in[0,T]} |A_i(t) - A_i(0)|, \tag{9}$$

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 Fig. 22, the error e as a function of grid size is shown in Fig. 23. 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 Fig. 23 is predominantly superlinear.

5.3. Mean curvature flow with constraints

5.3.1. 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, see [23]. 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

$$s^n(x) = \frac{V_i^0 - V_i^n}{A_i^n \Delta t} \quad \text{where } i \text{ is such that } x \in \Omega_i. \tag{10}$$

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 $s^n|\nabla\phi^n|$ 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 3.5.

Despite each phase potentially growing or shrinking at different rates, the VIIM robustly and smoothly handles the discontinuity of the source term (10) and conserves volume almost exactly. In Fig. 24, repeated from [23], we demonstrate the method in two and three dimensions on a set of 100 initially random phases (using zero Neumann boundary conditions). Mean curvature flow minimizes perimeter (surface area in 3D), which, subject to the constraint of area (volume) conservation, eventually attains an equilibrium. We see this in Fig. 24, and in particular, one can observe various topological changes occur and at all times triple junctions have 120° angles.

5.3.2. Volume targeting

One can easily replace the constant V_i^0 in the source term in (10) 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. Fig. 25 shows the resulting evolution for the same set of phases used in Fig. 24. 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.

5.3.3. 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 generalize this to the case where each interface Γ_{ij} has different surface energy densities, i.e. different coefficients of curvature. In this situation, there is a generalized Young's law, which states that the angles θ_i of a triple point satisfy

$$\frac{\sin\theta_i}{\gamma_{jk}} = \frac{\sin\theta_j}{\gamma_{ik}} = \frac{\sin\theta_k}{\gamma_{ij}}, \tag{11}$$

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

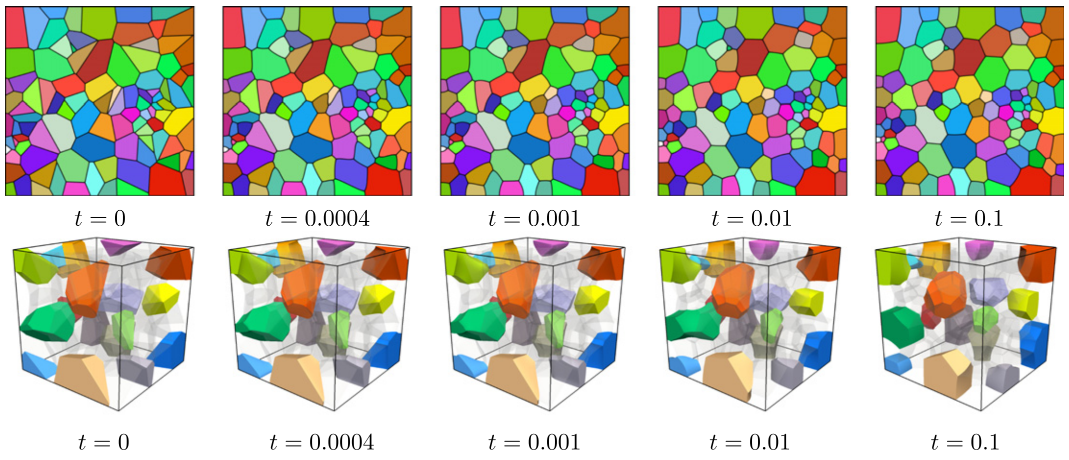

Fig. 24. Mean curvature flow ($\gamma = 1$) with area/volume conservation on a set of 100 randomly created phases (zero Neumann boundary conditions). By the time $t = 0.1$, the solution has approximately attained equilibrium. (Top) Two-dimensional results, computed on a 256 × 256 grid on the domain $[0,1]^2$ (approximately 13,000 time steps, taking about 20 min on a quad core laptop). (Bottom) Three-dimensional results, in which a selection of phases have been colored solid, computed on a 128^3 grid in a unit cube (approximately 3300 time steps, taking about 150 min on an eight-core desktop).

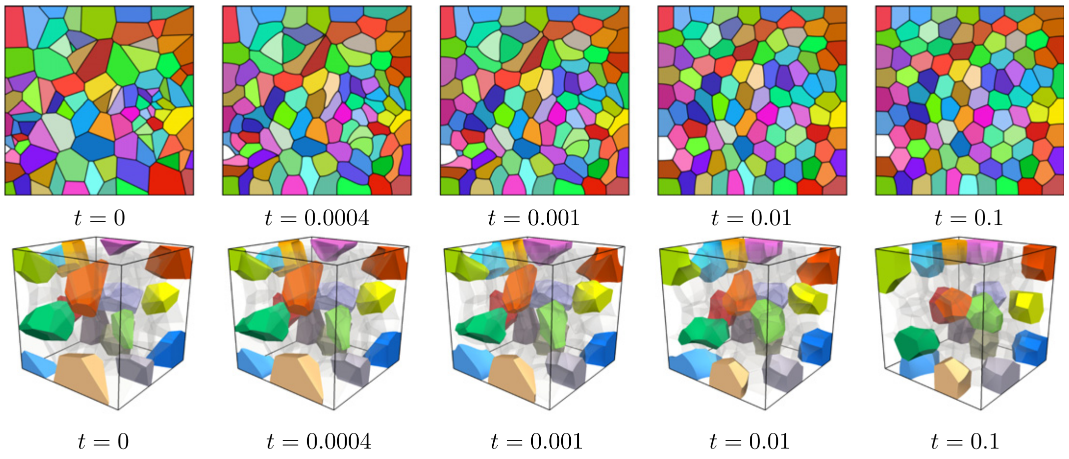

Fig. 25. Mean curvature flow ($\gamma = 1$) with area/volume targeting, so that every phase ultimately has the same area/volume (zero Neumann boundary conditions). By the time $t = 0.1$, the solution has approximately attained equilibrium. (Top) Two-dimensional results, computed on a 256×256 grid on the domain $[0, 1]^2$. (Bottom) Three-dimensional results in which a selection of phases have been colored solid.

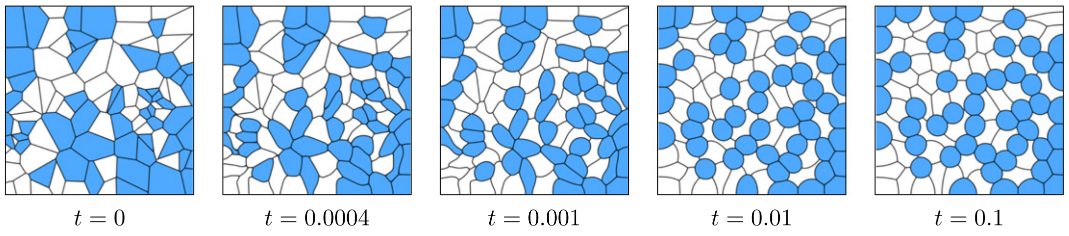

Fig. 26. Mean curvature flow with area targeting, so that all phases ultimately have the same area, with variable coefficients of curvature: blue phases have $\gamma = 1$, while white phases have $\gamma = 0.1$. By the time $t = 0.1$, the solution has approximately attained equilibrium. Simulation computed on a 256×256 grid in the unit square, with zero Neumann boundary conditions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\phi_t = \gamma\kappa|\nabla\phi|, \quad \text{where } \gamma(x) = \gamma_i \quad \text{if } x \in \Omega_i,$$

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 Fig. 26 we demonstrate this method 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 $\gamma_i = 1$ (blue phases), moving under curvature flow with area targeting. According to Young's law (11), 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 Fig. 26, in which the blue phases surrounded by white phases are approximately circular. By using a suitably modified von Neumann–Mullins' law, we have confirmed that the VIIM correctly converges to a solution satisfying the generalized Young's law (11).

6. Fluid flow with permeability

Aside from geometric examples, the VIIM can also be applied to multiphase evolution determined by complex physics, and we demonstrate this here by coupling the method to incompressible multiphase fluid flow. In [23], we presented an application in simulating dry foams. 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% [33]. 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 neighbor, 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 neighborship 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, we consider the case where the membranes can be idealized as massless and infinitely thin. This system was studied in detail in [13], using an immersed boundary method. There, the authors coupled the fluid mechanics of the gas to the evolution of the interface so that foams far from equilibrium can be modeled, which, for example, allowed them to apply large shearing forces to a foam. A large number of physical configurations were studied, showing a range of physical effects. Two-dimensional motion was considered, and topological changes were not allowed.

We now extend the framework developed in [23] to the case where different phases can have different densities and viscosities, and different membranes can have different surface tensions.

The set up of our multiphase fluid system is as follows. Suppose there are N phases, labeled $i = 1,\ldots,N$. 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 $\boldsymbol{u}$ of the fluid, assumed to be continuous across the interface. When there is permeability, the rate of diffusion of fluid is proportional to the pressure difference [33]. We adopt the approach used in [13], and model permeability as a slip of the interface in the normal direction relative to $\boldsymbol{u}$. The velocity of the interface in this case is thus $\boldsymbol{u} - M\sigma_{ij}\kappa\boldsymbol{n}$, where $M \geqslant 0$ is a physical parameter denoting the amount of permeability, and $\boldsymbol{n}$ is the unit normal of the interface (with the same orientation as that used to calculate κ).

In practice, the discontinuity in density and viscosity of the fluid across the interface must be regularized. A standard approach used in two-phase fluid flow problems (see the review [28]), is to smooth the density and viscosity across the interface, through the use of a smoothed Heaviside function. In the multiphase case, we define

$$\rho(x) = \frac{\sum_{i=1}^{N}\rho_i H_\varsigma(\phi_i)}{\sum_{i=1}^{N} H_\varsigma(\phi_i)}, \quad \mu(x) = \frac{\sum_{i=1}^{N}\mu_i H_\varsigma(\phi_i)}{\sum_{i=1}^{N} H_\varsigma(\phi_i)}, \tag{12}$$

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

$$H_\varsigma(x) = \begin{cases} 0 & \text{if } x \leqslant -\varsigma, \\ \frac{1}{2}\left(1 + \frac{x}{\varsigma} + \frac{1}{\pi}\sin\frac{\pi x}{\varsigma}\right) & \text{if } |x| \leqslant \varsigma, \\ 1 & \text{if } x \geqslant \varsigma. \end{cases} \tag{13}$$

Here, ς measures the width of the smoothing transition (and is usually denoted by ϵ, but we do not wish to cause confusion with the ϵ used in the VIIM). We have set $\varsigma = 2h$ for our simulations. Note that we have normalized by the sum of Heaviside functions in (12). This only matters at junctions, e.g. triple points, and is required to ensure $\min_i \rho_i \leqslant \rho(x) \leqslant \max_i \rho_i$.

Surface tension in the multiphase system is modeled as a body force through the use of a Dirac delta function [5,31,28]. In a two-phase fluid flow problem, this force takes the form $\boldsymbol{st} = -\sigma\kappa\delta(\phi)\nabla\phi$ where ϕ is a signed level set function for the interface. In our 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 phase separately, sum all of these, and normalize 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 5.3.3. Using this formulation for surface tension for a multiphase system, we therefore define

$$\boldsymbol{st} = -\frac{1}{2}\sum_{i=1}^{N}\sigma_i\kappa(\phi_i)\delta(\phi_i)\nabla\phi_i, \tag{14}$$

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 (14) directly but with smoothed Dirac delta functions, i.e.

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

where δ_ς is a smoothed one-dimensional delta function, e.g. the derivative of the smoothed Heaviside function in (13),

$$\delta_\varsigma(x) = \begin{cases} \frac{1}{2\varsigma}\left(1 + \cos\frac{\pi x}{\varsigma}\right) & \text{if } |x| < \varsigma, \\ 0 & \text{otherwise}. \end{cases} \tag{15}$$

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. We have obtained better results by instead *mollifying* the surface tension to smooth it, i.e. defining $\boldsymbol{st}_\varsigma := \boldsymbol{st} * \delta_\varsigma^n$ 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

$$\begin{aligned}\boldsymbol{st}_\varsigma(x) = (\boldsymbol{st} * \delta_\varsigma^n)(x) &= \int_\Omega \boldsymbol{st}(y)\delta_\varsigma^n(x-y)dy = -\frac{1}{2}\sum_{i=1}^N \int_\Omega \delta(\phi_i)(y)(\sigma_i\kappa(\phi_i)\nabla\phi_i)(y)\delta_\varsigma^n(x-y)dy \\ &= -\frac{1}{2}\sum_{i=1}^N \int_{\Gamma_i} \sigma_i(\kappa\boldsymbol{n})(y)\delta_\varsigma^n(x-y)dS(y). \qquad (16)\end{aligned}$$

We see that the n-dimensional convolution becomes a surface integral of $\kappa\boldsymbol{n}$ weighted by the mollifer. 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 3.5, since then $\kappa\boldsymbol{n}$ 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 our implementation, we have used the n-dimensional mollifier $\delta_\varsigma^n(x) = \prod_{i=1}^n \delta_\varsigma(x_i)$ where δ_ς is given in (15) and $x = (x_1, \ldots, x_n)$, which 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

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

where ρ, μ are given by (12), $\boldsymbol{st}_\varsigma$ is given by (16), and $\boldsymbol{F}$ is any additional body forces (such as gravity). To solve the Navier–Stokes equations numerically, we have used a second order variable density approximate projection method [3], which is based on Chorin's projection method [10]. We have used a second order upwinding ENO scheme for the advection term (for both $\boldsymbol{u}$ and ϕ), Crank–Nicholson for the diffusion term and forward Euler in time. The projection method was implemented with a variable coefficient multigrid solver, and the entire code has been parallelized with MPI.

6.1. Testing convergence during topological change

In a multiphase fluid flow calculation, accurately simulating topological changes represents a significant challenge. Here, we perform a convergence analysis to examine the ability of the VIIM and our Navier–Stokes solver to accurately calculate the evolution of a "T1" topological change in a two-dimensional foam. A T1 change (see [33]) 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.

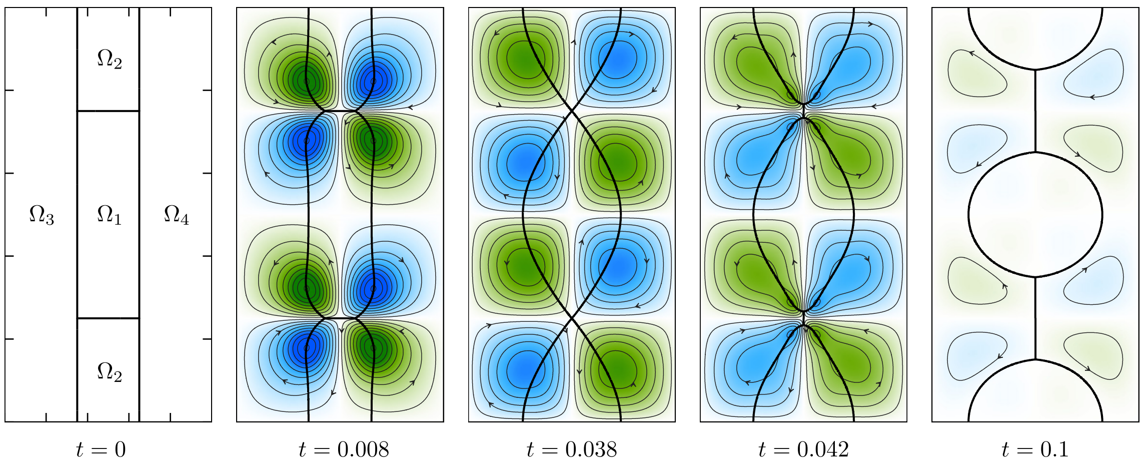

Fig. 27. Evolution of a four-phase fluid system which has two T1 topological changes. Streamlines are shown, and the colors indicate the magnitude of the stream function ψ. At $t = 0$, we have drawn a schematic of the initial condition having periodic boundary conditions on the bottom and top (and the 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.

Table 9
Convergence results for the evolution in Fig. 27. Here, $T_1(h)$ is the time of the T1 event on grid with cell size h, and supposing that to leading order, $T_1(h) = t_0 + C\,h^p$, the "rate" 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 $0 \leqslant t \leqslant 0.1$.

h	Time of T1 event			Evolution	
	$T_1(h)$	$T_1(2h) - T_1(h)$	Rate	d_h	Rate
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

As a specific example, we consider the situation shown in Fig. 27, 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 Fig. 27, we have indicated the flow of the liquid 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 (labeled 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 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 favorable 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, we have set all four phases to have $\rho = 1$ and $\mu = 0.1$.

With a grid size of $\frac{n}{2} \times n$ (so that $h = 1/n$), we perform a convergence analysis on the computed time of the T1 event, as well as measure grid convergence on the evolution of the interface. 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 suppose that $T_1(h) = t_0 + \mathcal{O}(h^p)$, and measure the convergence rate p. 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 (7). This is the same method we used in Section 4 to measure convergence of the interface location in both space and time. Table 9 contains the results of these convergence measurements for our 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.

6.2. Application to dry foams

The application of the VIIM to dry foams was first demonstrated, for constant density, viscosity and surface tension, in [23], in which the above framework was used to simulate a foam 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. Here, we expand on those results for some of the test cases.

6.2.1. Two-dimensional results

For convenience, we restate the simulation parameters. 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, we use an external force $\boldsymbol{F}$ in the Navier–Stokes momentum equations given by

$$\boldsymbol{F}(x, y, t) = 10(\sin \pi x \sin 2\pi y, -\sin 2\pi x \sin \pi y) \sin \pi t, \quad 0 \leqslant x, y \leqslant 1,$$

which corresponds to a spinning force, in the counterclockwise direction about the center point $x = y = \frac{1}{2}$ for $0 \leqslant t \leqslant 1$, and clockwise about the center point for $1 \leqslant t \leqslant 2$. The factor of 10 in $\boldsymbol{F}$ has been chosen to give a relatively strong shearing/spinning force that dominates the stabilization effects of surface tension.

A case with no permeability, in which $M = 0$, was shown in detail in [23], and we do not repeat it here. We instead consider a case where there is permeability, and set $M = 0.04$, which leads to significant coarsening over the time interval $0 \leqslant t \leqslant 2$. We start with 25 random phases and evolve the Navier–Stokes equations with agitator forcing. The simulation was performed on a 1024×1024 grid with periodic boundary conditions and $\epsilon = 2h$. Fig. 28 illustrates the results with plots of phase evolution, streamlines and stream function, and pressure fields, over five different points in time, $t = 0.01, 0.5, 1, 1.5$ and 2 corresponding to extremums in the agitator forcing. We see that there is significant shearing and several topological changes occur as the foam coarsens. 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

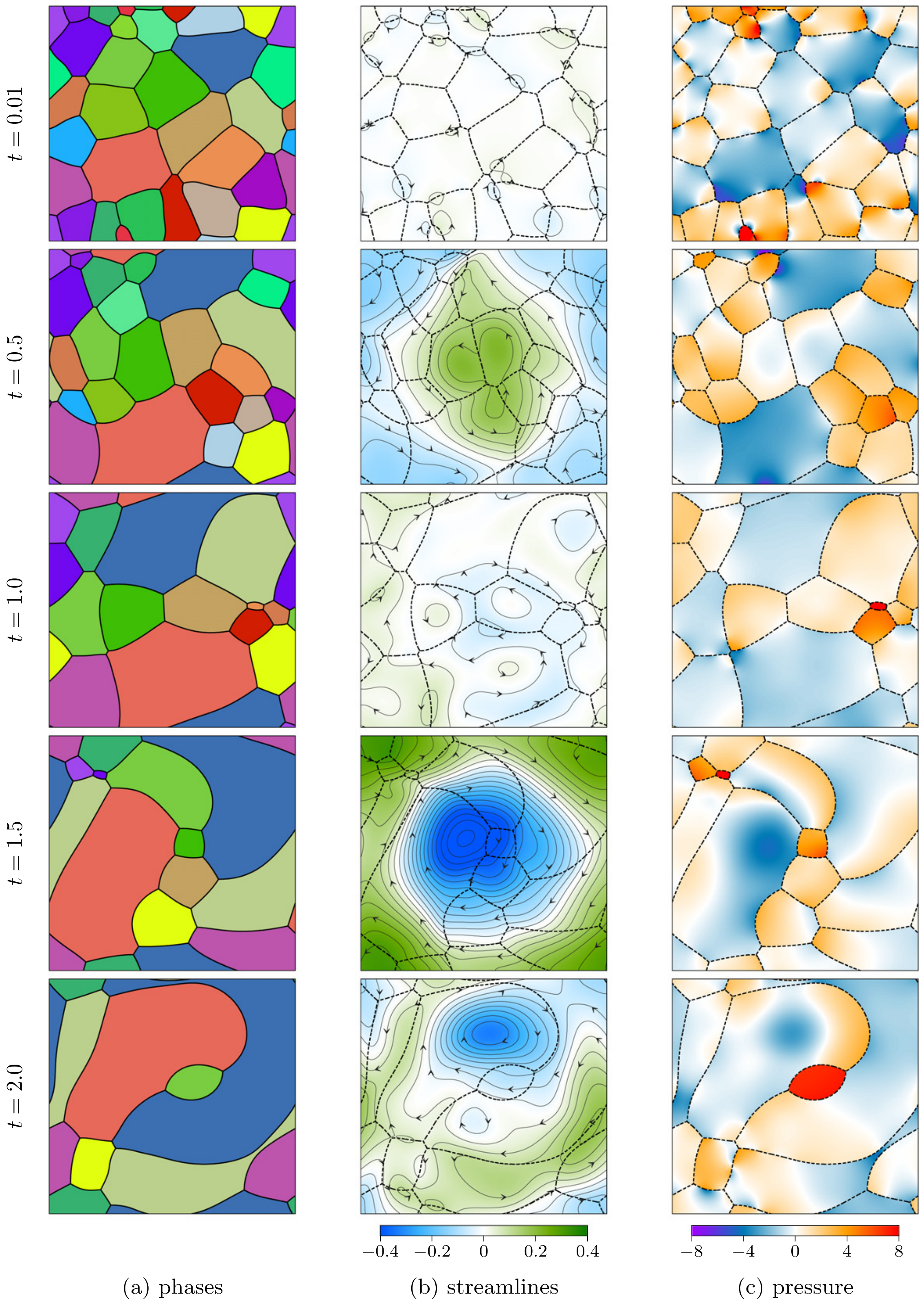

Fig. 28. Results of a fluid flow simulation with an external agitator force and permeability.

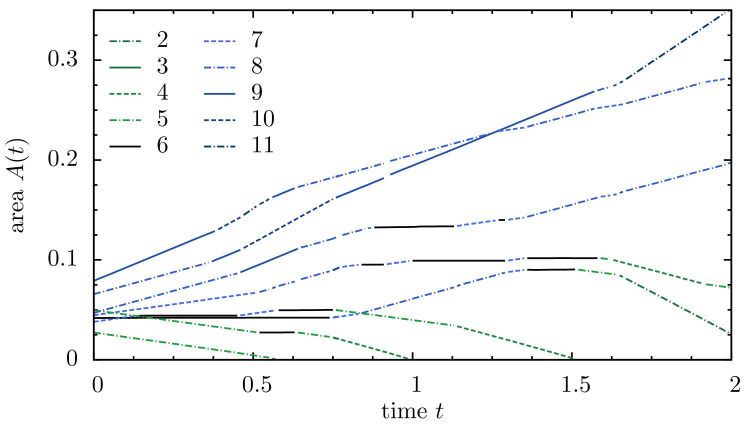

Fig. 29. Area as a function of time for some of the phases in the simulation of Fig. 28. Each line corresponds to a different phase, and the color and style is determined by the number of sides that phase had at that particular instant.

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

where n is the number of sides of a particular phase (or one of its connected components if it has more than one component). We have plotted the area as a function of time for some of the phases in Fig. 29. Similar to the plot obtained in our convergence tests (i.e. Fig. 14), 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.

6.2.2. 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.

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

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

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; Fig. 30 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 $256 \times 256 \times 256$ grid with periodic boundary conditions and $\epsilon = 2h$. We have colored solid 13 representative phases in order to visualize 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 colored according to the azimuthal component of the velocity field, relative to the center line of the agitation, computed by $\boldsymbol{u}\cdot\hat{\boldsymbol{\theta}}$ where

$$\hat{\boldsymbol{\theta}} = \frac{\left(-y+\frac{1}{2}, x-\frac{1}{2}, 0\right)}{\left[\left(-y+\frac{1}{2}\right)^2 + \left(x-\frac{1}{2}\right)^2\right]^{\frac{1}{2}}} \tag{17}$$

is the azimuthal tangent vector corresponding to a cylindrical coordinate system with center line $x = y = \frac{1}{2}$. Thus, the color of the streamlines give an indication of the direction the flow is spinning about the center 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 localized in nature. This is especially noticeable in the case of no permeability, in which the stabilization 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.

6.3. Variable fluid properties

Our final application couples the VIIM 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 connected 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

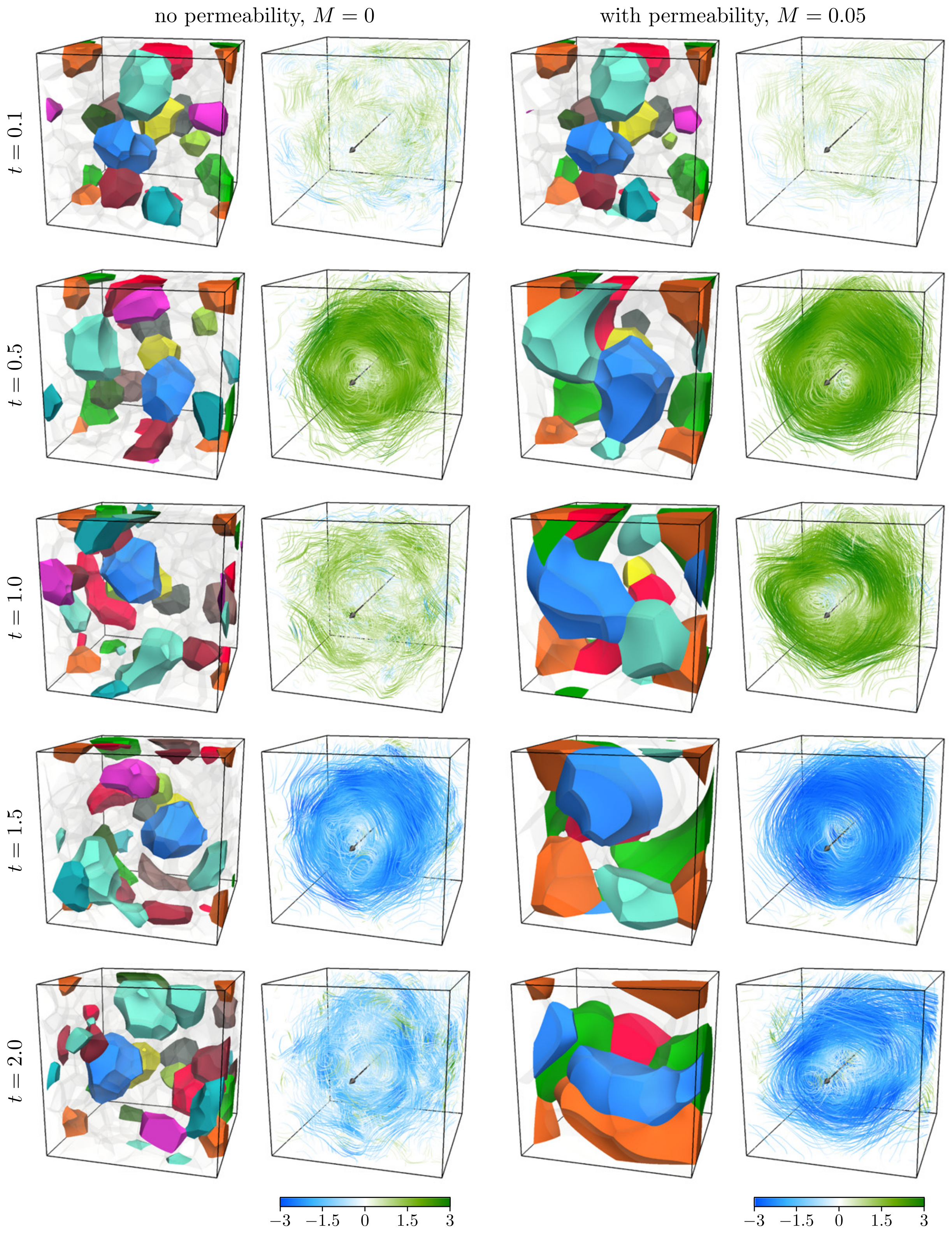

Fig. 30. Results of a fluid flow simulation with an external agitator force in three dimensions, with and without and permeability. Streamlines are colored by the azimuthal component of the velocity field, computed by $\boldsymbol{u} \cdot \hat{\boldsymbol{\theta}}$, where $\hat{\boldsymbol{\theta}}$ is defined in (17).

gravity: regions of denser liquid need to be sufficiently large in size for the force of gravity to dominate the stabilization 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. We also demonstrate the effect of different interfaces having different surface tensions, giving rise to triple junctions with different angle configurations.

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

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

where $\boldsymbol{n}$ is the normal to the boundary and $\boldsymbol{\tau}$ is any tangent vector to the boundary. We identify the heavy phase with the label $\mathcal{H}$, and for each phase i, we set

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

The system therefore has density ratios of 10 and viscosity ratios of 100, in such a way that there is a Reynolds number ratio of 10 (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 generalized Young's law (11), the angles $(\theta_i, \theta_j, \theta_k)$ made by phases i, j, k at a triple point are

$$(\theta_i, \theta_j, \theta_k) = \begin{cases} (120^\circ, 120^\circ, 120^\circ) & \text{if none of } i, j, k \text{ equal } \mathcal{H}, \\ (170^\circ, 95^\circ, 95^\circ) & \text{if } i = \mathcal{H} \text{ and } j, k \neq \mathcal{H}. \end{cases} \tag{18}$$

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

We have implemented a 90° contact angle model, 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 $\boldsymbol{u}$, and $\boldsymbol{u}$ 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, we make 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 we make use of connected components 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, we think of the less dense phases 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 multi-scale phenomena of foam production, whereby microscale dynamics of thin-film membranes, surfactant flow and surface tension, determine the macroscopic creation and evolution of foams.

6.3.1. Two-dimensional results

We begin with a two-dimensional simulation of the above variable density fluid flow problem. Fig. 31 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 colored orange. The other phases, of which there are initially approximately 35, are colored shades of blue and green. Fig. 31 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 our 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

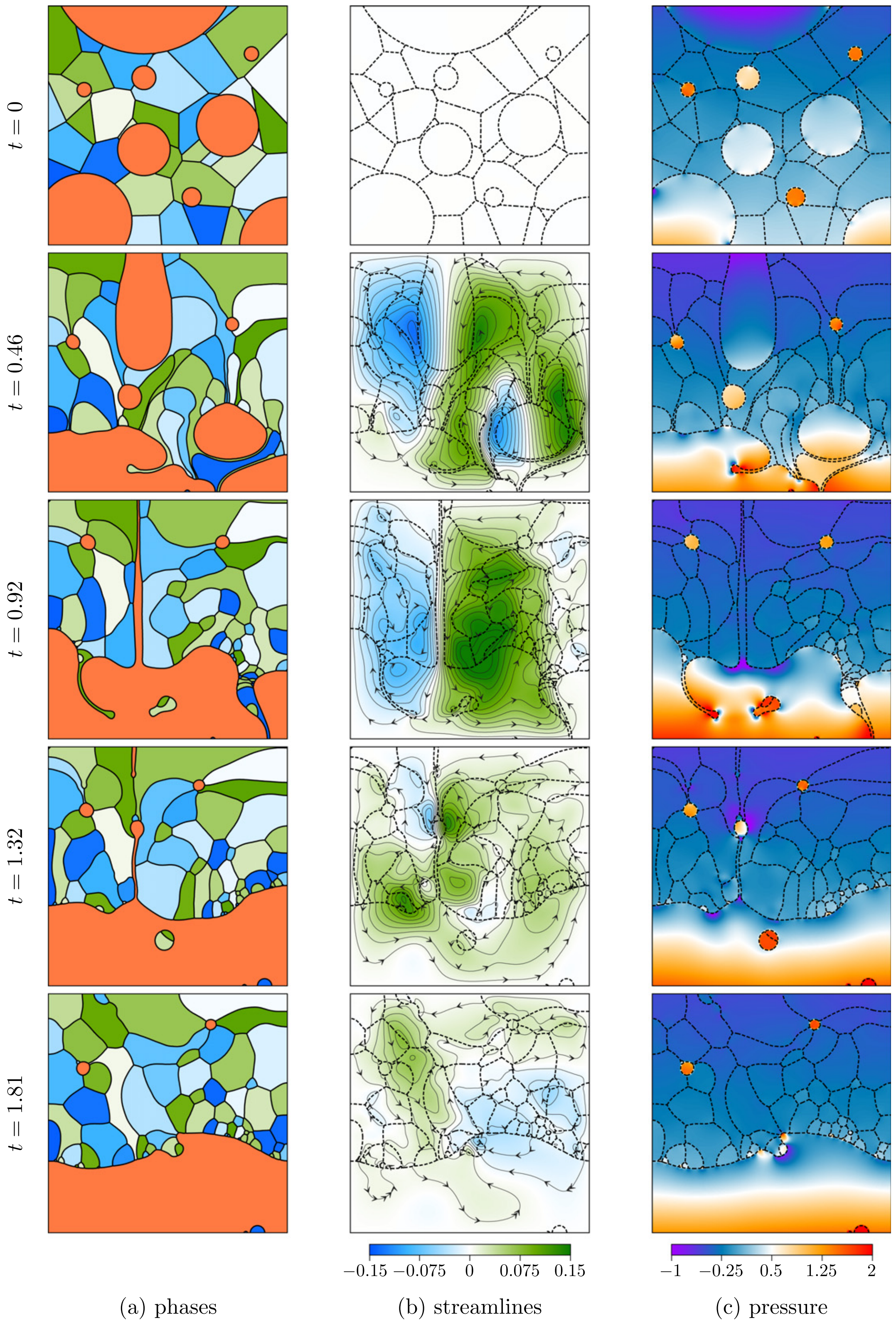

Fig. 31. Results of a fluid flow simulation with gravity, in which the orange colored phase is more viscous and more dense than the other phases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 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 (18): 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.

6.3.2. Three-dimensional results

Fig. 32 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 Fig. 32, the heavier phase $\mathcal{H}$ is colored 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 $t = 1.8$, we have rendered the bulk foam 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 traveling 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 generalized Young's law.

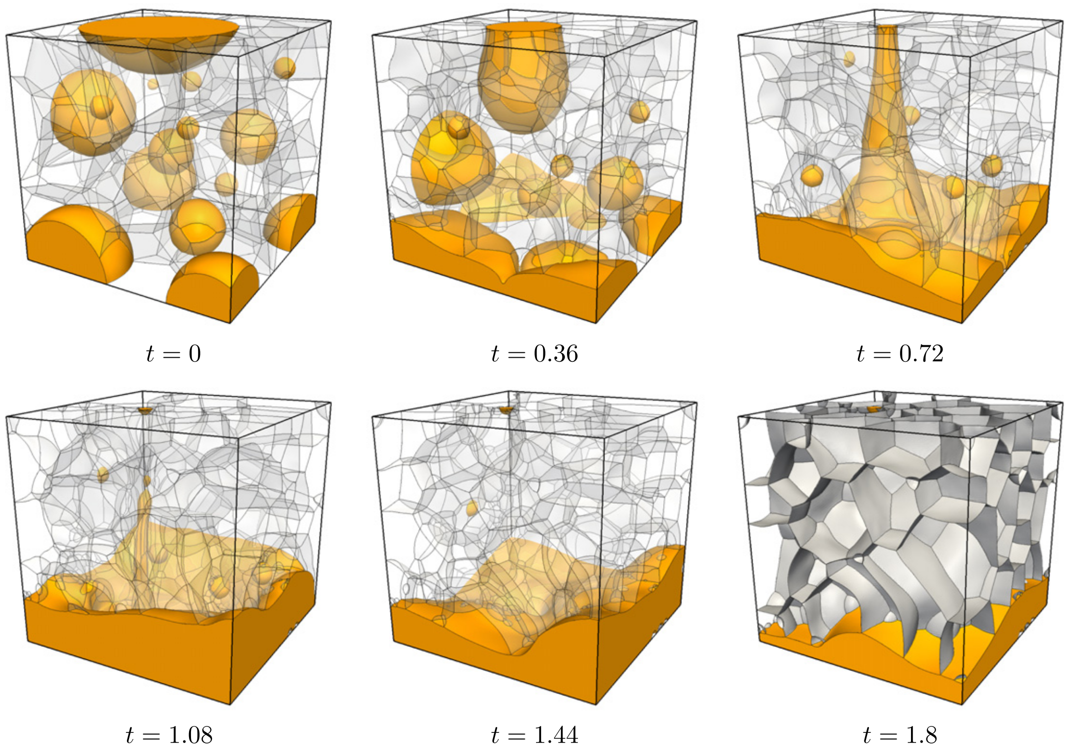

Fig. 32. Results of a fluid flow simulation in three dimensions with gravity, in which the orange colored phase is more viscous and more dense than the other phases. The bulk foam is rendered mostly transparent except for the last frame, where it is rendered opaque to make the structure more prominent. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusion

The Voronoi Implicit Interface Method tracks a large number of evolving, interconnected interfaces moving under complex interactions of geometry, physics, constraints and boundary conditions. It combines properties of implicit interface evolution with a generalization of the Voronoi tessellation that yields robustness through a wide range of topological changes, and it works in any number of spatial dimensions with no changes to the underlying algorithm. In this paper, the mathematical framework of the method has been presented, and a numerical discretization has been discussed. The method has been tested and convergence results have been provided, including verification of von Neumann–Mullins' law and convergence through topological changes in multiphase fluid flow. We have also demonstrated its application to various geometric flows, including mean curvature flow with different surface energy densities, and to multiphase fluid flow where density, viscosity and surface tension can be defined on a per-phase basis.

Acknowledgements

This work was supported in part by the Applied Mathematical Science subprogram of the Office of Energy Research, US Department of Energy, under Contract Number DE-AC02-05CH11231, by the Division of Mathematical Sciences of the National Science Foundation, and by NCI U54CA143833. Some computations used the resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02–05CH11231. J.S. was also supported by the Miller Foundation at UC Berkeley, and as an Einstein Visiting Fellow of the Einstein Foundation, Berlin. R.S. was also supported by an American Australian Association Sir Keith Murdoch Fellowship.

References

[1] David Adalsteinsson, James A. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118 (2) (1995) 269–277.
[2] David Adalsteinsson, James A. Sethian, The fast construction of extension velocities in level set methods, Journal of Computational Physics 148 (1) (1999) 2–22.
[3] Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, Michael L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, Journal of Computational Physics 142 (1) (1998) 1–46.
[4] David J. Benson, Volume of fluid interface reconstruction methods for multi-material problems, Applied Mechanics Reviews 55 (2) (2002) 151–165.
[5] J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (2) (1992) 335–354.
[6] K. Brakke, The surface evolver, Experimental Mathematics 1 (2) (1992) 141–165.
[7] Lia Bronsard, Brian T.R. Wetton, A numerical method for tracking curve networks moving with curvature motion, Journal of Computational Physics 120 (1) (1995) 66–87.
[8] Benjamin Y. Choi, Markus Bussmann, A piecewise linear approach to volume tracking a triple point, International Journal for Numerical Methods in Fluids 53 (2007) 1005–1018.
[9] David L. Chopp, Some improvements of the Fast Marching Method, SIAM Journal Scientific Computing 23 (1) (2001) 230–244.
[10] Alexandre Joel Chorin, Numerical solution of the Navier–Stokes equations, Mathematics of Computation 22 (104) (1968) 745–762.
[11] Matt Elsey, Selim Esedoglu, Peter Smereka, Diffusion generated motion for grain growth in two and three dimensions, Journal of Computational Physics 228 (21) (2009) 8015–8033.
[12] Harald Garcke, Britta Nestler, Barbara Stoth, A multiphase field concept: numerical simulations of moving phase boundaries and multiple junctions, SIAM Journal on Applied Mathematics 60 (1) (1999) 295–315.
[13] Yongsam Kim, Ming-Chih Lai, Charles S. Peskin, Numerical simulations of two-dimensional foam by the Immersed Boundary Method, Journal of Computational Physics 229 (13) (2010) 5194–5207.
[14] William E. Lorensen, Harvey E. Cline, Marching cubes: a high resolution 3d surface construction algorithm, Computer Graphics 21 (4) (1987) 163–169.
[15] Robert D. MacPherson, David J. Srolovitz, The von Neumann relation generalized to coarsening of three-dimensional microstructures, Nature 446 (2007) 1053–1055.
[16] 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) (1995).
[17] Barry Merriman, James K. Bence, Stanley J. Osher, Motion of multiple junctions: a level set approach, Journal of Computational Physics 112 (2) (1994) 334–363.
[18] W.F. Noh, P. Woodward, SLIC (Simple Line Interface Calculation), in: A.I. van de Vooren, P.J. Zandbergen (Eds.), Lecture Notes in Physics, Proceedings of 5th International Conference of Fluid Dynamics, vol. 59, Springer-Verlag, 1976, pp. 330–340.
[19] Stanley Osher, James A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49.
[20] F. Reitich, H. Soner, Three-phase boundary motions under constant velocities. I: The vanishing surface tension limit, Proceedings of the Royal Society of Edinburgh 126A (1996) 837–865.
[21] Robert I. Saye, James A. Sethian, Analysis of multiphase interface motion, in preparation.
[22] Robert I. Saye, James A. Sethian, Finite element and front tracking discretizations of the Voronoi Implicit Interface Method, in preparation.
[23] Robert I. Saye, James A. Sethian, The Voronoi Implicit Interface Method for computing multiphase physics, Proceedings of the National Academy of Sciences 108 (49) (2011) 19498–19503.
[24] J.A. Sethian, Curvature and the evolution of fronts, Communications in Mathematical Physics 101 (1985) 487–499.
[25] J.A. Sethian, Numerical methods for propagating fronts, in: P. Concus, R. Finn (Eds.), Variational Methods for Free Surface, Interfaces Proceedings of the Sept, 1985 Vallambrosa Conference, Springer-Verlag, 1987.
[26] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences 93 (1996) 1591–1595.
[27] 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.
[28] J.A. Sethian, Peter Smereka, Level set methods for fluid interfaces, Annual Review of Fluid Mechanics 35 (2003) 341–372.
[29] J.A. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilon-Jacobi equations: theory and algorithms, SIAM Journal Numerical Analysis 41 (2003) 325–363.
[30] 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) (2002) 263–276.

[31] Mark Sussman, Peter Smereka, Stanley Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1) (1994) 146–159.
[32] J.N. Tsitskiklis, Efficient algorithms for globally optimal trajectories, IEEE Transactions of Automatic Control 40 (1995) 1528–1538.
[33] D. Weaire, S. Hutzler, The Physics of Foams, Oxford University Press, 1999.
[34] Hong-Kai Zhao, T. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, Journal of Computational Physics 127 (1) (1996) 179–195.