Journal of Computational Physics 229 (2010) 6243–6269

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A hybrid variational front tracking-level set mesh generator for problems exhibiting large deformations and topological changes

Ricardo H. Nochetto [a], Shawn W. Walker [b,*]

[a] University of Maryland, Department of Mathematics, Mathematics Building, College Park, MD 20742-4015, USA
[b] New York University, Mathematics, Courant Inst., 251 Mercer Street, New York, NY 10012-1185, United States

ARTICLE INFO

Article history:
Received 26 October 2009
Received in revised form 19 April 2010
Accepted 21 April 2010
Available online 6 May 2010

Keywords:
Front tracking
Arbitrary-Lagrangian–Eulerian mesh method
Topological change
Shape optimization
Mesh generation
pinching
Level-set method

ABSTRACT

We present a method for generating 2-D unstructured triangular meshes that undergo large deformations and topological changes in an *automatic* way. We employ a method for detecting when topological changes are imminent via distance functions and shape skeletons. When a change occurs, we use a level set method to guide the change of topology of the domain mesh. This is followed by an optimization procedure, using a variational formulation of active contours, that seeks to improve boundary mesh conformity to the zero level contour of the level set function. Our method is advantageous for Arbitrary-Lagrangian–Eulerian (ALE) type methods and directly allows for using a variational formulation of the physics being modeled and simulated, including the ability to account for important geometric information in the model (such as for surface tension driven flow). Furthermore, the meshing procedure is not required at every time-step and the level set update is only needed during a topological change. Hence, our method does not significantly affect computational cost.

1. Introduction

1.1. Motivation

Free boundary problems arise in many areas of mathematics, physics, and engineering. Understanding free surface dynamics is important for applications such as coating flows [7], simulating water wave dynamics for computer graphics [30], and surface tension/curvature driven flows in micro-fluidic devices such as Hele-Shaw flow [16,34,92,89]. Other examples involve fluid–structure interactions, such as polymer filaments in an active flow field [82], interaction of a lipid bio-membrane with a surrounding fluid [99], and animal locomotion in a fluid medium [1].

However, in any application with a moving boundary, the deformation of the domain is the main obstacle in obtaining a tractable physical model. In addition, some of these applications exhibit topological changes (i.e. pinching or joining of disjoint parts of the interface) and prove even more difficult to model. Examples of this are budding of lipid bio-membranes [6], droplet pinching in an electro-wetting device [15,91], and many other types of fluid flow [25].

One of the difficulties in modeling a topological change is in handling the disparate length and time scales involved. For example, a pinching droplet may have two macroscopic pieces connected through a thin microscopic neck that is collapsing. And the time scale of the neck collapse may be quite small compared to the time scale of the bulk droplet motion. Furthermore, it is not clear how best to model the true physics when a topological change is occurring. Some asymptotic analysis of

* Corresponding author. Tel.: +1 301 742 3142.
E-mail address: walker@cims.nyu.edu (S.W. Walker).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.035

the behavior of the Navier–Stokes equations has been done for axisymmetric fluid pinching [27,28]. But one can argue that a continuum model is not adequate and a model which includes atomistic behavior is more correct. In [45,46] it was shown that adding a stochastic component to the Navier–Stokes equations was effective in modeling the behavior of nano-fluids in a non-vacuous environment when compared to a molecular dynamics simulation.

But some applications *do not require* a detailed understanding of the local behavior around a topological change. In the electro-wetting device it is enough to only acknowledge the fact that a droplet has pinched or joined. In this spirit, the remaining difficulty is in developing a simulation tool that can go through a topological change in a reasonable way, while properly "piecing" together the continuum model that governs the rest of the behavior.

In this paper, we develop a method for generating explicit unstructured triangular meshes in 2-D that conform to a smooth closed curve and can be used with Arbitrary-Lagrangian–Eulerian (ALE) methods. Furthermore, our method allows for topological changes of the domain and can continue deforming automatically without user intervention. Moreover, it has the potential for extension to 3-D tetrahedral meshes (see Section 6.5). Our method is targeted at variational problems where accurate knowledge of the boundary is critical to obtaining a robust solution. For example, problems involving higher order boundary information (i.e. surface tension flow, Willmore flow), optimization problems that require computation of surface quantities (i.e. shape optimization), and many fluid–structure interaction problems fall into this group. In particular, the finite element method is a common tool used to solve these types of problems and we feel our method would be useful in these areas.

1.2. Literature overview

One popular method for capturing free surface motion is the level set method [61,70], which advects a scalar field function whose zero level set represents the interface. Level set methods have the advantage of being completely Eulerian and can automatically handle topological changes, though the physics underlying such changes is often not well resolved. In particular, level set methods require a small amount of diffusion to allow for topological changes to occur. This can cause problems with mass conservation and requires special handling [29,75] or refinement [52]. An alternative approach is to use the coupled level set-volume of fluid (CLS-VOF) method to ensure mass conservation [80,81,86,87]. Another issue of the level set method, for curvature driven flows, is they typically use an explicit calculation of the interface curvature which can create numerical artifacts and noise. Other implicit surface methods include the phase field method [97,79], which uses a diffuse interface model (as opposed to a sharp or explicit interface). Phase field methods have similar advantages and drawbacks as the level set method.

Alternatively, one can use an explicit representation of the interface, such as an interface mesh or marker particles to "track" the interface. These are called front-tracking methods [35,85,18,2], some of which are designed to track shock fronts in hyperbolic equations [49,50,93]. Furthermore, there exist numerical PDE techniques that can take advantage of the intrinsic representation of the interface [4,26,43]. However, the main disadvantage of these explicit surface representations is the computational difficulty in handling large deformations of the mesh. In two dimensions, the mesh can be adjusted through local re-meshing [76] or mesh smoothing [31], but can still be awkward. In three dimensions, it is not clear what the best methods are for adjusting a mesh as it deforms.

There are a variety of mesh generation methods. Some take an optimization viewpoint [69,56,55] while others [13,14] use a variational form to minimize the interpolation error to do local re-meshing. Some methods use specific tilings of 3-D space [84] or marching cubes [51] or triangles [40]. Still others make use of implicit functions to create conforming meshes [58,59,57,11,63,64,94] as well as adaptive methods to create meshes adapted to the local feature size [39,48]. Some of these methods also include mesh smoothing operations (see [20,60,73,96,44] for more smoothing methods).

Currently, there exist some methods for taking explicit meshes through topological changes. Some use "surgery" [22,23,17,12] to cut the mesh or use a pre-defined bridge [66] in 3-D. This is a viable option when the topological change has a well-defined structure. But the general nature of topological changes is much more complicated, especially in 3-D. For example, a thinning neck of fluid could become very flattened and pinch in the middle leading to a torus like structure with one or many "handles". In this case, it is not clear how to reconstruct the mesh without a guide or indication of the new topological state of the domain.

Considering the trade-offs between implicit surface methods and explicit Lagrangian meshes, it is reasonable to suggest a hybrid approach. This would combine the accuracy of the explicit mesh methods with the ease of topological transformation of the level set method. One version of this is given by [5], which forms an explicit representation of the boundary at each time step that is coupled with their level set method and is advantageous for tracking of surface characteristics, such as texture coordinates, for use in rendering fluid interfaces for computer graphics. But their method does not generate an interior bulk mesh. Other relevant work includes [54], where they introduce the virtual node algorithm as a way of tracking topological changes of explicit triangular or tetrahedral meshes. However, their method is not concerned with the correct local geometry, since they were mainly concerned with solving elasticity equations, as opposed to surface tension driven flow.

The method we develop takes inspiration from some of the ideas in the above references and combines them in a novel way to generate meshes of arbitrary domains. In addition, we introduce a shape optimization approach for ensuring mesh conformity. We emphasize that our re-meshing method does not need to be executed at every time step of the simulation. The number of re-meshes only depends on the continuous deformation being approximated and the number of topological changes.

1.3. Algorithm overview

Our algorithm primarily consists of a special re-meshing routine that is embedded inside a time stepping loop. We make extensive use of distance functions and shape skeletons to resolve the shape and topology of the domain when generating a new mesh. We also use a shape optimization approach to ensure that the new mesh conforms to the boundary of the domain. Topological changes are implemented by locally diffusing the distance function in the neighborhood of the change.

The main point of our algorithm is to provide a way for generating meshes that can follow an arbitrarily complex deformation and can continue through topological changes *without having to specify the type of topological change, or specify geometric details, or perform surgery on the mesh.* Even if the physics of the topological change is well understood, it is not necessarily clear what the mesh should be after the change. This is especially important in three-dimensions. Therefore, this algorithm is an answer to the question of how to compute and mesh through a topological change, but *not* to the question of modeling the physics of the change itself.

We highlight some aspects of the algorithm in the following list. Details of each item are given in the sections that follow. Also, see the flowchart given in Fig. 1 for a high level summary.

- Time step adaptation. The size of the time-steps δt during a simulation are controlled by the desired accuracy and the amount of shear in the velocity field (see Section 2.2).
- Velocity extension. As the physical simulation of a moving domain progresses, the velocity on the domain boundary is extended to the entire mesh in a smooth way and used for updating the domain (see Section 2.3). This is done to prevent frequent re-meshing.
- Mesh smoothing. We use standard techniques to improve distorted elements (see Section 2.4).
- Re-meshing via the distance function. We develop an adaptive method for generation of unstructured triangular meshes that uses the distance function (with respect to the domain boundary) and shape skeleton. We are able to generate well-resolved and well shaped meshes by straightforward processing of the distance function and shape skeleton (see Section 3.2).
- Ensure mesh conformity. We use a shape optimization approach to ensure that the generated mesh conforms to the domain boundary (see Section 3.4).
- Detection of imminent topological changes. Any sufficiently "thin" region in the mesh is considered a topological change and can be found by simple processing of the distance function and shape skeleton. These regions are used to help guide mesh adaptation in those areas to ensure accurate resolution of the shape. See Section 4.1 for more discussion.
- Updating domain topology. This is achieved by locally diffusing the distance function in the thin regions only (see Section 4), followed by our shape optimization approach to ensure mesh conformity.

1.4. Notational convention

We now introduce some notation that will be used throughout (see Fig. 2 and Table 1). Let Γ be the interface or manifold between two distinct phases. We label the interior phase Ω_{int} and the exterior phase Ω_{ext}. The whole domain is defined by $\Omega := \overline{\Omega}_{\text{int}} \cup \overline{\Omega}_{\text{ext}}$ with outer boundary Γ_{ext}.

In general, we denote a fixed triangulation by $\mathcal{T}$ with possible superscripts. For example, $\mathcal{T}^{\text{int}}$ will denote a triangulation of Ω_{int}. We denote distance functions by a Greek letter and specify the triangulation that they are defined on, e.g. $\phi(\mathcal{T})$. The

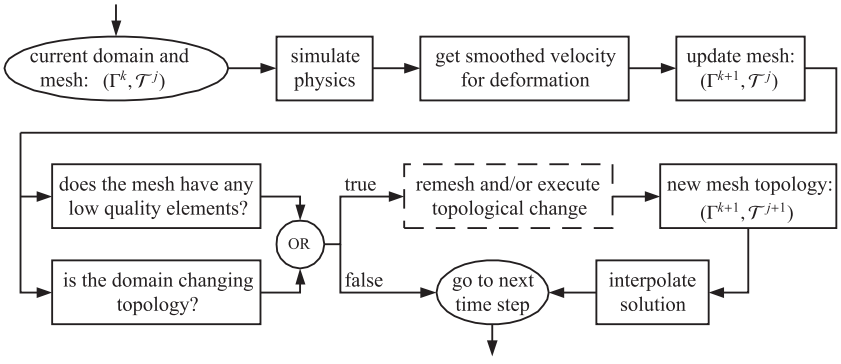

Fig. 1. High level block diagram of a single time step with possible re-meshing and topological changes. The current (interior) domain boundary $\Gamma^k \equiv \partial\Omega^k_{\text{int}}$ and mesh $\mathcal{T}^j$ have different indices because the mesh topology does not necessarily change at each time-step (i.e. only mesh vertices change). The physics (e.g. velocity field, pressure, etc...) is simulated with $(\Gamma^k, \mathcal{T}^j)$. Using a smooth update velocity, we obtain the domain shape at the next time index. Note that the mesh topology does not change, though the vertex positions do change. If the element qualities are not bad and a topological change is not imminent, then we proceed to the next time step. Otherwise, we execute the re-mesh routine. This generates a new mesh topology that geometrically conforms to the boundary Γ^{k+1}. Note that Γ^{k+1} also changes if a topological change occurs. Finally, we interpolate the solution variables from the old mesh topology $\mathcal{T}^j$ to the new topology $\mathcal{T}^{j+1}$ and proceed to the next time step. Section 3 further describes the re-mesh routine.

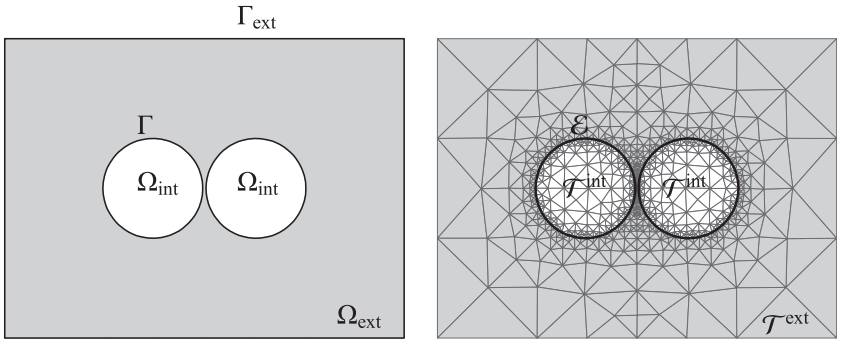

Fig. 2. Continuous domain and discrete mesh with symbolic notation. The interior domain is Ω_{int}, the shaded region is Ω_{ext}, with two-phase boundary $\Gamma := \overline{\Omega}_{int} \cap \overline{\Omega}_{ext}$. The outer (continuous) boundary is denoted Γ_{ext}. The entire domain is defined as $\Omega := \overline{\Omega}_{int} \cup \overline{\Omega}_{ext}$. The discretization of the continuous domain is represented as a set of interior triangles $\mathcal{T}^{int}$ and exterior triangles $\mathcal{T}^{ext}$, with a set of mutual edges $\mathcal{E}$ representing the discretization of the boundary Γ. The entire triangulation is given by $\mathcal{T} := \mathcal{T}^{int} \cup \mathcal{T}^{ext}$. In our algorithm, we use the signed distance function ϕ with respect to Γ and compute it over the entire triangulation $\mathcal{T}$ [8]; this dependence is denoted by $\phi(\mathcal{T})$. In this paper, we take ϕ to be positive (negative) over the interior (exterior). The zero level set of ϕ corresponds exactly to the discrete representation of Γ because the triangulation is conforming.

Table 1
Symbol definitions.

Symbol	Definition
Ω	Entire continuous domain (both phases)
Γ	Closed boundary $\Gamma := \overline{\Omega}_{int} \cap \overline{\Omega}_{ext}$ Contained in Ω
$\mathcal{T}$	A triangulation of Ω
$\mathcal{T}^{old}$	A previous triangulation of Ω
ϕ	Distance function to Γ
$\mathcal{V}_{Sk}$	Vertices (in $\mathcal{T}^{old}$) that define shape skeleton
ξ	Distance function to shape skeleton
$\mathcal{G}$	A new triangulation (default)
$\mathcal{G}(\mathcal{E} : \phi)$	Mesh conforms to $\{\phi = 0\}$
ϵ_{smooth}	Smoothing parameter (see Section 3.2.3)
t	Time
δt_{max}	Maximum allowed time step
$\mathbf{u}$	Vector velocity
$\mathbf{u}_{smooth}$	Smooth velocity extension
$\mathbf{x}_{displace}$	Displacement function to deform mesh
$D(\mathbf{u})$	Symmetrized gradient operator
ε_0	Artificial diffusion parameter
$\mathbf{X}$	Surface parameterization
$J(\Gamma)$	Cost functional; see Eq. (10)
$\delta J(\Gamma, \mathbf{V})$	Shape derivative in direction $\mathbf{V}$
$\mathcal{I}$	Contin. piecewise linear interpolation operator
$\tilde{\Gamma}$	Set of points given by $\mathcal{E}$
Ω_{int}, Ω_{ext}	Interior, exterior domain
Γ_{ext}	External boundary $\Gamma_{ext} := \partial\Omega$
$\mathcal{T}^{int}$, $\mathcal{T}^{ext}$	Triangulation of interior, exterior phases
$\mathcal{T}^{init}$	An initial triangulation
ψ_{top}	Level-set function after topological change
$\mathcal{V}_{thin}$	Subset of $\mathcal{V}_{Sk}$ that represent thin regions
ξ_{thin}	Distance function to thin regions
$\mathcal{E}$	Set of edges contained in $\mathcal{G}$
κ	Curvature
$\kappa_{\mathcal{P}_0}$, $\kappa_{\mathcal{P}_1}$	FEM approximation of curvature
δt	Time step
$\mathbf{x}$	Position coordinate
(u, v)	$\mathbf{u} = (u, v)$
$\boldsymbol{\nu}$	Outward normal vector
$\boldsymbol{\tau}$	Positively oriented tangent vector
d_{neck}	Minimum neck thickness
θ	Cut-off function (localizes diffusion)
∇_Γ	Surface gradient operator
$\mathbf{V}$	Vector perturbation of surface
$\boldsymbol{\varphi}$	Optimal descent direction
α	Shape optimization step size
Γ^k	Shape optimization iterate

symbol $\mathcal{E}$ will denote a set of edge segments (whose union is a closed manifold) that is shared by two separate triangulations, for instance at the interface between $\mathcal{T}^{\text{int}}$ and $\mathcal{T}^{\text{ext}}$. In other words, $\mathcal{E}$ will be the discrete representation of a two-phase interface. Lastly, we let $\mathcal{G}$ denote a generic triangulation that is not fixed, meaning that $\mathcal{G}$ is in the process of being modified (i.e. triangles are being added/removed, or adaptive refinement is currently running). Moreover, given a set of edge segments $\mathcal{E}$ contained in $\mathcal{G}$, we let $\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}))$ denote the dependence of $\mathcal{G}$ on the distance function ϕ, meaning that $\mathcal{G}$ has been deformed so that $\mathcal{E}$ conforms to the zero level set $\{\phi = 0\}$. This is important when we modify a generated mesh to conform to some zero level set. If we just write $\mathcal{G}$, this refers to the mesh in the default, unmodified state.

2. Basic concerns

We start by stating some basic ideas that are useful for any methods using ALE techniques.

2.1. Main cause of mesh distortion

Mesh distortion for a triangular mesh that is moving with a given velocity field (which comes from the physics being simulated) is directly due to gradients in the field (i.e. the velocity field has some shear component). This clearly happens when a topological change is underway. In this section, we derive a basic estimate that relates the maximal time-step of a mesh update (while preventing mesh inversion) to the gradient of the velocity.

A diagram of a single triangle in some triangulation is given in Fig. 3. The 2-D velocity field is assumed to be *linear* over the triangle and is denoted by $\mathbf{u} = (u, v)$. For simplicity, we assume that $v = 0$ and that u at the points p_1 and p_2 is denoted by u_1 and u_2, respectively, and we assume that $u_1 > u_2$. The points p_1 and p_2 move with constant velocity u_1 and u_2 because we are updating the triangle vertex positions by taking a discrete time-step. We want to estimate how large the time step must be for the point p_1 to cross over p_2; this will invert the triangle. The relative distance between p_1 and p_2 (after moving one step) is given by $h_{max} - \delta t(u_1 - u_2)$, where δt is the time-step of the mesh update. Hence, if the relative distance becomes zero, then δt is given by

$$\frac{1}{\delta t} = \frac{u_1 - u_2}{h_{max}} = \frac{\partial u}{\partial x}.$$

A similar relation holds when looking for the time to cross-over of the points p_3 and p_4 with velocity $(0, v)$,

$$\frac{1}{\delta t} = \frac{\partial v}{\partial y}.$$

Naturally, a conservative estimate for the maximal time-step that will not cause the triangle to invert is

$$\delta t < \frac{C_\tau}{|\nabla \mathbf{u}|}, \tag{1}$$

for some positive constant $0 < C_\tau \leqslant 1$ (we use $C_\tau := 0.1$). Of course, the triangle may be very distorted after updating. Further consideration suggests that (1) should actually be

$$\delta t < \frac{C_\tau}{|D(\mathbf{u})|}, \quad D(\mathbf{u}) := \frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^\dagger}{2}. \tag{2}$$

Note that $D(\mathbf{u}) = 0$ for any rigid motion [83], which does not cause mesh distortion. The choice of $C_\tau = 0.1$ is conservative, meaning that making C_τ smaller would lead to unnecessarily small time steps (i.e. no parameter tuning is needed here).

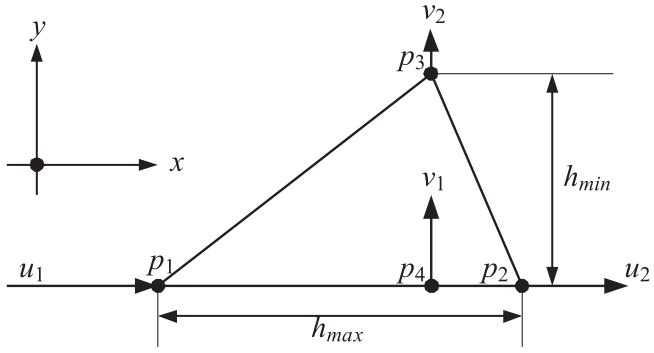

Fig. 3. A mesh triangle undergoing deformation. The velocity field over the triangle is labeled (u, v) and is linear over the triangle. The values of the x component of velocity are labeled u_1 and u_2 at the points p_1 and p_2, respectively (with $u_1 > u_2$). As p_1 and p_2 move in the x direction, their relative distance decreases. The rate of decrease depends on $\frac{\partial u}{\partial x}$. This gives an estimate of the largest time step δt that can be taken before p_1 and p_2 cross-over, which is $\delta t < 1/\frac{\partial u}{\partial x}$. Any larger time-step will cause the triangle to be inverted.

2.2. Time-stepping

We adopt a simple method for adapting the time-step. First, the maximum time step $\delta t_{\max}$ is set by the desired accuracy. Then, based on (2), we choose the current time step δt such that

$$\delta t := \min\left(\delta t_{\max}, \frac{C_\tau}{\max_T |D(\mathbf{u})|}\right), \tag{3}$$

where $\mathbf{u}$ is a piecewise linear approximation (over the triangulation) of the true velocity. If the true velocity is a Lipschitz function, the estimate (3) is essentially independent of the triangulation as long as the velocity field is well resolved. Otherwise, it depends on the triangle's diameter, shape regularity, and the nature of the singularity in the derivative of the velocity. For example, suppose the true velocity has a $\sqrt{|\mathbf{x}|}$ type singularity. Then one can show that $\delta t \lesssim \sqrt{\operatorname{diam}(T)}$, where T is a triangle in the neighborhood of the singularity.

2.3. Smooth velocity extension

In lieu of Section 2.2, it is desirable to update the domain mesh with a velocity field that has minimal shear (i.e. with $|\nabla \mathbf{u}|$ minimal). A simple way to do this is by harmonic extension. Let $\mathbf{u}_{\text{smooth}}$ be a piecewise linear function over the triangulation $\mathcal{T}$ which solves the standard weak formulation of the following vector Laplace equation

$$\begin{aligned} -\Delta \mathbf{u}_{\text{smooth}} &= 0, \ \Omega, \\ \mathbf{u}_{\text{smooth}} &= \mathbf{u}, \ \Gamma, \\ \frac{\partial \mathbf{u}_{\text{smooth}}}{\partial \boldsymbol{\nu}} &= 0, \ \Gamma_{\text{ext}}. \end{aligned} \tag{4}$$

where $\mathbf{u}$ is the true velocity field that comes from the physics of the problem. Solving the Laplacian guarantees that $|\nabla \mathbf{u}_{\text{smooth}}|$ will be minimized in the L^2 sense [32], thus subjecting the mesh triangles to minimal distortion. Moreover, updating the shape with $\mathbf{u}_{\text{smooth}}$ keeps the same shape evolution. It is not necessary to update the interior vertices of the mesh of Ω with the true velocity. Hence, we take advantage of this freedom by using a smooth extension of the true velocity.

Of course, solving (4) will incur extra computational cost in addition to simulating the physics. However, we make the following points: (1) if an iterative solver is used, it is not necessary to demand high accuracy in the solution because $\mathbf{u}_{\text{smooth}}$ plays no role in the physics; (2) if the mesh topology did not change from the previous time step, then the previous solution of (4) can be used to "warm-start" the iterative solver and (3) it may be possible for the user to take advantage of a canned solver/package for (4). Multilevel solvers are known to be quite efficient on unstructured grids [10,95]. Moreover, our mesh generation method in Section 3 can be trivially modified to generate a set of nested meshes for use in a multigrid algorithm [3,41,42,68].

2.4. Mesh smoothing

Local mesh smoothing is a useful tool for improving an existing mesh. Various techniques for improving a 2-D triangulation exist, such as Laplace smoothing which averages the positions of mesh vertices based on its neighbors. In addition, one can use an optimization method, such as in [31], which moves the vertices of the mesh in an attempt to optimize the local quality metric of the triangulation [47]. One advantage of this method is that it is guaranteed not to invert elements and produces well-shaped elements for the given mesh topology. Mesh smoothing is a supplementary tool that we use to prevent frequent re-meshing.

3. Mesh generation

This section describes our re-meshing algorithm. For simplicity, we will consider problems where only an interior phase is of interest, such as for fluid droplets in air. Thus the meshing of the exterior is not particularly critical. Our method can be easily generalized to the case of an arbitrary domain that contains two (or more) phases of interest (see Section 6.3).

3.1. Approximating the shape skeleton

Suppose we have a domain Ω that contains an interior closed manifold Γ. Assume we have the signed distance function ϕ (to Γ) defined on all of Ω (recall Fig. 2). The shape skeleton is the locus of points where $\nabla\phi$ is discontinuous [71,77]. Knowing the shape skeleton gives valuable information about the geometry and topology as well as potential locations of topological changes.

Finding the skeleton on a discrete grid is straightforward when $\nabla\ \phi$ points in nearly opposite directions on either side of the discontinuity. These points correspond directly to where a topological change may be imminent. Other points where

changes in $\nabla\phi$ are less abrupt are harder to detect, but less important for our purposes. Hence, we propose a method of estimating the location of part of the shape skeleton that only corresponds to abrupt changes in $\nabla\phi$.

Suppose we have a triangulation $\mathcal{T}^{\text{old}}$ (of Ω) that conforms to the boundary Γ. We denote the piecewise linear signed distance function (to Γ) on $\mathcal{T}^{\text{old}}$ by $\phi(\mathcal{T}^{\text{old}})$. We use the method in [8] to compute ϕ because it has no restriction on the triangulation. Note that the zero level set of $\phi(\mathcal{T}^{\text{old}})$ exactly represents Γ because the mesh is conforming. Let $\mathcal{V}_{\text{Sk}}$ be the set of vertices in $\mathcal{T}^{\text{old}}$ that locate discontinuities in $\nabla\ \phi$ according to some tolerance. $\mathcal{V}_{\text{Sk}}$ is found by executing Algorithm 1.

Algorithm 1. Sweep edges and vertices of mesh

1: Set tolerance Sk_{tol} such that $0.0 < \text{Sk}_{\text{tol}} < 1.0$. Default value: $\text{Sk}_{\text{tol}} = 0.5$.
2: Initialize $\mathcal{V}_{\text{Sk}} = \emptyset$.
3: # PART 1
4: **for all** edges E in $\mathcal{T}$ **do**
5: For each E, compute:

$$\text{EJ} := -\frac{\nabla\phi_+}{|\nabla\phi_+|}\cdot\frac{\nabla\phi_-}{|\nabla\phi_-|}, \quad -1 \leqslant \text{EJ} \leqslant 1, \tag{5}$$

where ϕ_+ and ϕ_- are evaluated on opposite sides of E.
6: **if** $\text{EJ} > \text{Sk}_{\text{tol}}$ **then**
7: include the end points of E in $\mathcal{V}_{\text{Sk}}$.
8: **end if**
9: **end for**
10: # PART 2
11: **for all** vertices v in $\mathcal{T}$ **do**
12: For each v, compute a local weighted average of $\nabla\phi/|\nabla\phi|$ at v by

$$\text{AVE} := 1 - \left|\frac{1}{|\omega|}\sum_{T\subset\omega}|T|\frac{\nabla\phi}{|\nabla\phi|}|_T\right|, \quad 0 \leqslant \text{AVE} \leqslant 1, \tag{6}$$

where ω is the local "star" of triangles that share v as a vertex.
13: **if** $\text{AVE} > \text{Sk}_{\text{tol}}$
14: include v in $\mathcal{V}_{\text{Sk}}$.
15: **end if**
16: **end for**
17: Remove all vertices from $\mathcal{V}_{\text{Sk}}$ that lie on the manifold Γ.
18: **return** $\mathcal{V}_{\text{Sk}}$.

Roughly speaking, Part 1 of Algorithm 1 looks for large jumps in $\nabla\phi$ across mesh edges and Part 2 identifies vertices where $\nabla\phi$ points towards or away from the vertex. In most cases, Part 1 is enough. However, consider the case when Γ is a circle, where the shape skeleton is just the center point. If a mesh vertex is perfectly aligned with the circle's center, then Part 1 would not detect it, but Part 2 would.

In subsequent sections, we use ϕ and $\mathcal{V}_{\text{Sk}}$ to create a new mesh that is adaptively refined so that it can resolve the topology and geometry of Γ. We emphasize that the complete shape skeleton can be complicated, i.e. lots of fingering due to small undulations of the boundary. But we only need the extreme parts of the skeleton to correctly resolve the topology.

The choice of $\text{Sk}_{\text{tol}} = 0.5$ was dictated by numerical experiments and appeared to be relatively robust; recall that we only need the abrupt changes in $\nabla\phi$. Note: choosing Sk_{tol} close to 1 is too restrictive and would only detect the *extreme* changes in $\nabla\phi$.

3.2. Generate mesh

Before handling any topological changes, we must first generate a new well-shaped mesh that conforms to the current manifold Γ.

3.2.1. Initial mesh

We start by creating an initial coarse mesh that contains Γ, followed by subsequent adaptive refinement to resolve the geometry and topology of Γ. Define a domain Ω to be a rectangular box with dimensions chosen such that it contains Γ and that $\text{dist}(\Gamma,\Gamma_{\text{ext}}) > C_{\text{buffer}}\ \text{diam}(\Gamma)$ (default value: $C_{\text{buffer}} := 0.3$). The initial triangulation of Ω (denoted $\mathcal{T}^{\text{init}}$) is taken to be a coarse cartesian like grid with a crisscross pattern (see Fig. 4, upper left corner). The choice of $C_{\text{buffer}} = 0.3$ is not particularly critical; it only needs to be sufficiently large to prevent potentially large deformations between Γ and Γ_{ext}.

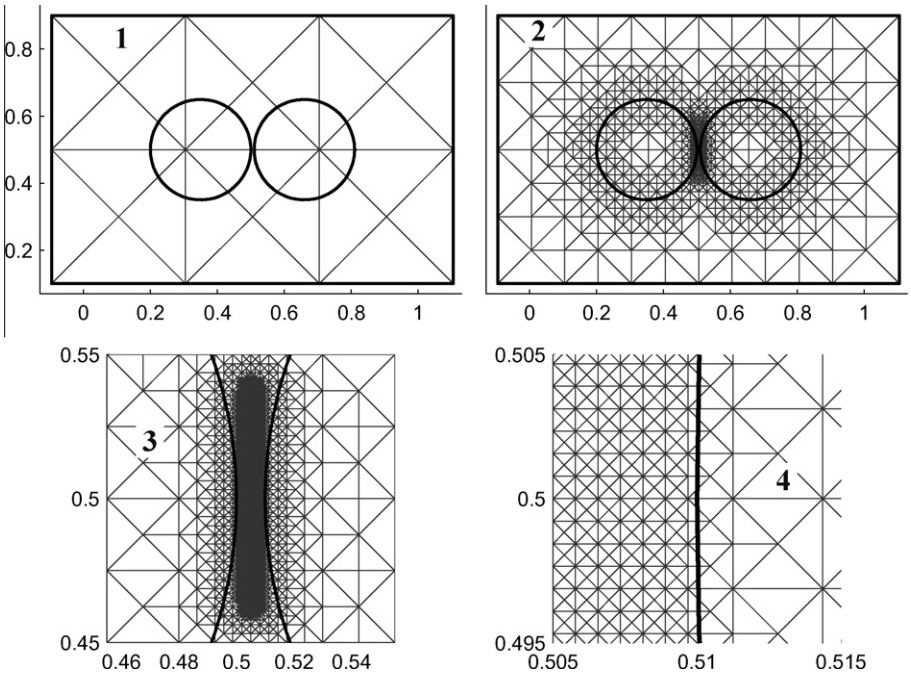

Fig. 4. Illustration of adaptive refinement. The bolded black curve corresponds to the zero level set of $\phi(\mathcal{T}^{\text{old}})$. (1) Initial coarse mesh that covers the interior manifold Γ with extra surrounding buffer region. (2) Resulting adaptively refined mesh that resolves the topology of Γ. (3) Zoom-in of region that is close to a topological change. The dense refinement in between the two circles is due to the presence of the shape skeleton (not shown) and the thinness there; see Sections 3.2.2 and 4.1.2. Extra refinement in narrow regions is useful for resolving topological changes. (4) Another zoom-in.

3.2.2. Adaptive refinement

Let $\phi\left(\mathcal{T}^{\text{old}}\right)$ be the signed distance function (to Γ) on the old triangulation $\mathcal{T}^{\text{old}}$. Also, let $\xi\left(\mathcal{T}^{\text{old}}\right)$ be the distance function to the skeleton represented by the set of vertices $\mathcal{V}_{\text{Sk}}$; we can compute this by the same method we used for ϕ [8]. Our refinement method is given in Algorithm 2. The idea is to adaptively refine the mesh until none of the triangles intersects *both* the shape skeleton and the interface (see Fig. 4, lower left corner). It is guaranteed to terminate as long as dist $(\{\phi = 0\}, \{\xi = 0\}) > 0$. This is the case as long as Γ approximates a smooth curve. The mesh $\mathcal{G}$ produced by this algorithm well resolves the topology of Γ. Note that we need to interpolate ϕ and ξ onto the mesh $\mathcal{G}$ in Algorithm 2.

Algorithm 2. Adaptive refinement

1: Initialize triangulation $\mathcal{G} := \mathcal{T}^{\text{init}}$. Set C_{adapt} to be a constant such that $0.0 < C_{\text{adapt}} \leqslant 2.0$.
   Default value: $C_{\text{adapt}} = 2.0$.
2: **loop**
3: Initialize $\mathcal{M} = \emptyset$.
4: **for all** triangles T in $\mathcal{G}$ **do**
5: Estimate triangle diameter: let L_T be the length of the longest edge of T.
6: **if** $L_T \geqslant C_{\text{adapt}} \min_T \phi$ and $L_T \geqslant C_{\text{adapt}} \min_T \xi$ **then**
7: include T in $\mathcal{M}$.
8: **end if**
9: **end for**
10: **if** $\mathcal{M} \neq \emptyset$ **then**
11: Execute the longest edge bisection routine [67] on $\mathcal{G}$ with the marking $\mathcal{M}$.
12: **else**
13: Exit loop.
14: **end if**
15: **end loop**
16: **return** $\mathcal{G}$.

The choice of $C_{\text{adapt}} = 2.0$ is not critical and was found to work for a wide range of test cases. Choosing C_{adapt} smaller only leads to excessive refinement near the interface Γ. Ergo, for most applications, $C_{\text{adapt}} = 2.0$ should be sufficient.

3.2.3. Refine by curvature

In order to ensure that $\mathcal{G}$ resolves the geometry of Γ, we further refine it using the curvature of Γ as a guide. To facilitate this, we must estimate the curvature of Γ on the old mesh $\mathcal{T}^{\text{old}}$. One method involves computing second derivatives of the distance function ϕ. However, we computed ϕ as a piecewise linear function over $\mathcal{T}^{\text{old}}$, so second derivatives would not make sense.

But we do have an explicit mesh for the manifold Γ, which we can use directly. Let $\boldsymbol{\nu}$ be the piecewise constant normal vector of the polygonal boundary Γ and $\mathbf{x}_i$ be the position of a vertex of Γ. Then we define a continuous piecewise linear approximation $\hat{\boldsymbol{\nu}}$ of the normal vector by solving the following variational problem for all continuous piecewise linear vector functions $\mathbf{v}$ on Γ:

$$\epsilon_{\text{smooth}} \int_\Gamma \nabla_\Gamma \boldsymbol{\eta} \cdot \nabla_\Gamma \mathbf{v} + \int_\Gamma \boldsymbol{\eta} \cdot \mathbf{v} = \int_\Gamma \boldsymbol{\nu} \cdot \mathbf{v}, \quad \Rightarrow \quad \hat{\boldsymbol{\nu}}(\mathbf{x}_i) := \frac{\boldsymbol{\eta}(\mathbf{x}_i)}{|\boldsymbol{\eta}(\mathbf{x}_i)|} \tag{7}$$

i.e. (7) is a smoothed $L^2(\Gamma)$ projection with a re-normalization to ensure unit length. With this, we can compute a piecewise constant approximation of the curvature via $\kappa_{\mathcal{P}_0} := \nabla_\Gamma \cdot \hat{\boldsymbol{\nu}} = (\partial_{\boldsymbol{\tau}} \hat{\boldsymbol{\nu}}) \cdot \boldsymbol{\tau}$, where $\partial_{\boldsymbol{\tau}}$ is the tangential derivative. The smoothing parameter ϵ_{smooth} is used to prevent over-estimating the curvature in the case where the mesh is under resolved. In our computations, we take $\epsilon_{\text{smooth}} = 10^{-6}$.

In Algorithm 3, we need to interpolate the manifold curvature onto nearby triangles. Therefore, for convenience, we define a continuous piecewise linear approximation $\kappa_{\mathcal{P}_1}$ of $\kappa_{\mathcal{P}_0}$ by a standard L^2 projection, i.e. solve

$$\int_\Gamma \kappa_{\mathcal{P}_1} \mu = \int_\Gamma (\nabla_\Gamma \cdot \hat{\boldsymbol{\nu}}) \mu \tag{8}$$

for all continuous piecewise linear functions μ on Γ. This gives an estimate of the curvature at the vertices of Γ. We then define an extension κ_{extend} of $\kappa_{\mathcal{P}_1}$ to the entire triangulation $\mathcal{T}^{\text{old}}$ by solving a scalar Laplace problem just like (4), except the boundary data is given by $\kappa_{\mathcal{P}_1}$. For efficiency, one only needs to extend the curvature to a narrow band of triangles in the neighborhood of Γ.

The purpose of Algorithm 3 is to ensure that all triangles close to Γ are small enough to resolve the curvature of Γ. This allows for better approximation of Γ in Section 3.4. The choice of $C_R = 0.2$ is conservative and works well for our test cases. Setting C_R to a smaller value just gives more refinement near the interface Γ.

Algorithm 3. Refine for curvature

1: Initialize triangulation $\mathcal{G}$ with result from Algorithm 2. Set C_R to be a constant such that $0.0 < C_R \leqslant 0.3$.
   Default value: $C_R = 0.2$.
2: **loop**
3: Initialize $\mathcal{M} = \emptyset$.
4: **for all** triangles T in $\mathcal{G}$ **do**
5: Estimate triangle diameter: let L_T be the length of the longest edge of T.
6: Estimate minimum radius of curvature near T: $R_{\min}(T) := 1/\max_T \kappa_{\text{extend}}$.
7: **if** $L_T \geqslant \min_T \phi$ and $L_T \geqslant C_R R_{\min}(T)$ **then**
8: include T in $\mathcal{M}$.
9: **end if**
10: **end for**
11: **if** $\mathcal{M} \neq \emptyset$ **then**
12: Execute the longest edge bisection routine [67] on $\mathcal{G}$ with the marking $\mathcal{M}$.
13: **else**
14: Exit loop.
15: **end if**
16: **end loop**
17: **return** $\mathcal{G}$.

3.3. Select candidate manifold

Now that we have a new well-shaped mesh $\mathcal{G}$, we must deform it so that it conforms to the zero level set of $\phi(\mathcal{T}^{\text{old}})$. In other words, we want $\mathcal{G}$ to conform to Γ. To do this, we must first select a candidate manifold that is embedded in $\mathcal{G}$ as a closed set of edge segments. We do this by choosing a subset of triangles in $\mathcal{G}$ to be the interior phase; the embedded manifold is just the outer boundary of the interior triangles. Finding a "good" selection of triangles to be in the interior phase is

non-trivial because the discrete nature of the mesh will introduce an aliasing effect. This section describes how we handle this and takes inspiration from [11]. We emphasize that this procedure is the only part of our algorithm that does not extend to 3-D (see Section 6.5 for some discussion).

The background reference mesh $\mathcal{G}$ comes from adaptively refining, via longest edge bisection, an initial uniform crisscross mesh (recall previous sections). As a result, all triangles in $\mathcal{G}$ are self-similar isosceles right triangles; note that no mesh smoothing has been used at this stage of the algorithm. Thus, we will take advantage of this property. Let $\mathcal{T}^{\text{int}}$ be the set of triangles such that

$$\mathcal{T}^{\text{int}} := \{T \in \mathcal{G} : \phi(\mathcal{T}^{\text{old}})|_{\mathbf{x}_c(T)} \geqslant 0\}, \tag{9}$$

i.e. we evaluate the signed distance at the barycenter $\mathbf{x}_c$ of T and if it is non-negative, we say it belongs to the interior phase. However, cutting the mesh like this can lead to an irregular (initial) manifold shape because of aliasing effects. Thus, we define $\mathcal{E}$ to be the boundary of $\mathcal{T}^{\text{int}}$ and proceed to modify the set $\mathcal{T}^{\text{int}}$ by performing local operations based on the shape of $\mathcal{E}$.

First we check if any pairs of adjacent edges of $\mathcal{E}$ shares the same triangle. If so, then that triangle will become crushed when we enforce mesh conformity (see Section 3.4). We avoid this by adding (or removing) the offending triangle to (or from) the set $\mathcal{T}^{\text{int}}$ and updating $\mathcal{E}$ accordingly. However, this may not be enough.

The next part of our selection process takes advantage of the crisscross nature of the mesh. From this, it can be seen that the angle between two adjacent edge segments in $\mathcal{E}$ is either 45°, 90°, 135°, or 180°. Hence, we loop through each vertex of $\mathcal{E}$, check the angle there, and adjust $\mathcal{T}^{\text{int}}$ and $\mathcal{E}$ accordingly. We summarize these checks in the following list (see Algorithm 4 for a full description).

1. If the angle is 45°, then the two adjacent edge segments must share the same exterior (or interior) triangle. In this case, we simply add (or remove) the shared triangle to (from) the set $\mathcal{T}^{\text{int}}$ and adjust $\mathcal{E}$.
2. If the angle is 90°, then three different cases can arise. We adjust $\mathcal{T}^{\text{int}}$ by following the method described in Fig. 5.
3. If the angle is 135° or 180°, then nothing needs to be done. The discrete manifold is already well-shaped.

We consider the above method a single pass through the mesh to adjust $\mathcal{T}^{\text{int}}$. We then loop this entire procedure until the set $\mathcal{T}^{\text{int}}$ no longer changes. Typically, only one pass is needed with an additional pass to check for consistency. Note that one can check the topology of $\mathcal{E}$ directly and compare it to the topology of the boundary Γ in the old mesh $\mathcal{T}^{\text{old}}$ to ensure they are the same.

Upon completion of Algorithm 4, the topology of $\mathcal{G}$ and $\mathcal{E}$ becomes fixed for the remainder of this section. Also, recall our notation from Section 1.4 and note that $\mathcal{G} = \mathcal{G}(\mathcal{E} : \emptyset)$ denotes the default geometry of the mesh $\mathcal{G}$, i.e. we have not deformed the mesh to conform to anything, yet.

Algorithm 4. Select candidate manifold

1: Interpolate the value of $\phi\left(\mathcal{T}^{\text{old}}\right)$ at the barycenter of all triangles in $\mathcal{G}$. Initialize $\mathcal{T}^{\text{int}}$ to be the set of triangles with $\phi \geqslant 0$ at their barycenter. This induces a closed manifold consisting of the set of edge segments $\mathcal{E}$ that is the outer boundary of the triangles $\mathcal{T}^{\text{int}}$ contained in $\mathcal{G}$.
2: **while** the set $\mathcal{T}^{\text{int}}$ was updated **do**
3: If any two adjacent edge segments in $\mathcal{E}$ share the same *exterior* triangle T_{ext}, then we *add* T_{ext} to the set $\mathcal{T}^{\text{int}}$. We check all pairs of adjacent edge segments and update $\mathcal{T}^{\text{int}}$ and $\mathcal{E}$.
4: If any two adjacent edge segments in $\mathcal{E}$ share the same *interior* triangle T_{int}, then we *remove* T_{int} from the set $\mathcal{T}^{\text{int}}$. We check all pairs of adjacent edge segments and update $\mathcal{T}^{\text{int}}$ and $\mathcal{E}$.
5: **for all** vertices v in $\mathcal{E}$ **do**
6: If two edge segments that share v have an angle of 90° or less, and the corner points toward the interior phase, then execute the procedure described in Fig. 5 (i.e. add triangles to $\mathcal{T}^{\text{int}}$). Update $\mathcal{T}^{\text{int}}$ and $\mathcal{E}$.
7: **end for**
8: **for all** vertices v in $\mathcal{E}$ **do**
9: If two edge segments that share v have an angle of 90° or less, and the corner points toward the exterior phase, then execute the procedure described in Fig. 5 (i.e. remove triangles from $\mathcal{T}^{\text{int}}$). Update $\mathcal{T}^{\text{int}}$ and $\mathcal{E}$.
10: **end for**
11: **end while**
12: **return** $\mathcal{T}^{\text{int}}, \mathcal{E}$. $\mathcal{E}$ now has a fixed topology.

3.4. Active contours for mesh conformity

For notational convenience, let $\widetilde{\Gamma}$ be the continuous manifold defined by the discrete mesh $\mathcal{E}$. Clearly, $\widetilde{\Gamma}$ approximates the original manifold Γ but is "jagged" and will not necessarily conform to Γ. Since we have the boundary Γ captured exactly as

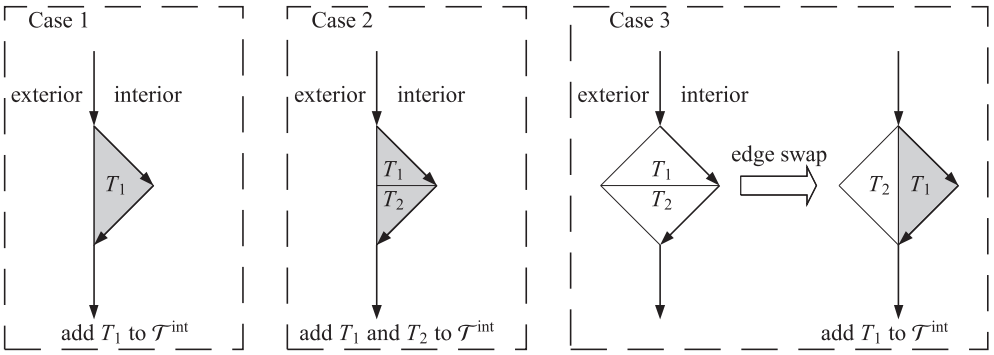

Fig. 5. Description of adding and removing triangles for candidate manifold selection (only addition is shown here). Bold directed arrows show the oriented manifold edge segments for the current set of interior triangles $\mathcal{T}^{\text{int}}$. The diagram shows three cases where the candidate manifold has a corner pointing *inward* (90° angle) due to our initial selection (Section 3.3) of interior triangles $\mathcal{T}^{\text{int}}$; this is an example of aliasing. Cases 1 and 2 are a problem because of the mesh conformity phase (see Section 3.4), which will crush the shaded triangles when the manifold is made to conform to the zero level set. Thus, we add the shaded triangles to the set $\mathcal{T}^{\text{int}}$ which makes the manifold more regular and avoids crushing triangles. If we ignore case 3, triangles T_1 and T_2 will not be crushed during the mesh conformity phase, however there will be a moderate amount of mesh distortion. So to improve the mesh quality, we do an edge swap and include the new T_1 triangle which makes the manifold boundary shape more regular. When the corner is pointing *outward*, the process is the same except the triangles are removed from the set $\mathcal{T}^{\text{int}}$ (instead of included). If the corner angle is 45°, then only case 1 can occur.

the zero level set of $\phi(\mathcal{T}^{\text{old}})$, we can adjust $\widetilde{\Gamma}$ by solving a minimization problem. This can be done for higher dimension surfaces, which makes this approach attractive. Thus, in the following discussion, we consider a general surface (i.e. a 1-D curve or 2-D surface).

3.4.1. Shape optimization problem

For the moment, we abuse notation and let Γ be some arbitrary surface (not necessarily the manifold in question). We define an energy functional (dependent on Γ) to be minimized:

$$J(\Gamma) = \frac{1}{2}\int_{\Gamma} \phi^2. \tag{10}$$

With this, we want to find a new surface Γ^* that minimizes J:

$$\Gamma^* = \arg\min_{\Gamma} J(\Gamma). \tag{11}$$

Clearly, the minimum solution is a surface that lies along the zero level set of ϕ.

3.4.2. Gradient flow

We find the surface that minimizes the functional (10) by defining an L^2 gradient flow [24,23]. This is a gradient descent method that seeks to move the surface in a direction that is guaranteed to minimize the cost J. We first give a short proof for the shape derivative of boundary functionals [19,78,65].

Lemma 1 (Shape Derivative). *Suppose f is a fixed smooth function on $\mathbb{R}^n$ and let Γ be a smooth closed manifold of dimension $n-1$. Let $\mathbf{V}$ be a smooth vector field defined on $\mathbb{R}^n$ that induces a flow such that points $\mathbf{x}$ in Γ move with the field $\mathbf{V}(\mathbf{x})$. Then, for the functional $Q := \int_\Gamma f$, we have that the shape derivative of Q in the direction $\mathbf{V}$ is*

$$\delta Q(\Gamma;\mathbf{V}) := \int_{\Gamma} \nabla f\cdot\mathbf{V} + f[\nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma \mathbf{V}], \tag{12}$$

where ∇_Γ is the surface gradient operator on Γ and $\mathbf{X}$ is the identity map on Γ.

Proof. We begin with a standard result from the shape derivative literature [19,78,65]

$$\delta Q(\Gamma;\mathbf{V}) := \int_{\Gamma} (\mathbf{V}\cdot\boldsymbol{\nu})(\boldsymbol{\nu}\cdot\nabla f) + f(\kappa\boldsymbol{\nu}\cdot\mathbf{V}), \tag{13}$$

which we will further manipulate. Using the fact that $\kappa\nu = -\Delta_\Gamma \mathbf{X}$ (where Δ_Γ is the surface Laplacian), we have

$$\begin{aligned} &\delta Q(\Gamma;\mathbf{V}) = \int_{\Gamma} (\mathbf{V}\cdot\boldsymbol{\nu})(\boldsymbol{\nu}\cdot\nabla f) - \Delta_\Gamma \mathbf{X}\cdot(f\mathbf{V}),\\ &\text{integrate by parts } \rightarrow = \int_{\Gamma} \mathbf{V}\cdot[\boldsymbol{\nu}\otimes\boldsymbol{\nu}]\nabla f + \nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma(f\mathbf{V}). \end{aligned} \tag{14}$$

Since $\nabla_\Gamma \mathbf{X} = \mathbf{I} - \nu \otimes \nu$ (i.e. the projection onto the tangent space of the manifold Γ), we re-write (14) as

$$\begin{aligned}\delta Q(\Gamma;\mathbf{V}) &= \int_\Gamma \mathbf{V}\cdot[\boldsymbol{\nu}\otimes\boldsymbol{\nu}]\nabla f + f[\nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma \mathbf{V}] + \mathbf{V}\cdot[\nabla_\Gamma \mathbf{X}]\nabla_\Gamma f \\ &= \int_\Gamma \mathbf{V}\cdot[\boldsymbol{\nu}\otimes\boldsymbol{\nu}]\nabla f + f[\nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma \mathbf{V}] + \mathbf{V}\cdot[\mathbf{I}-\boldsymbol{\nu}\otimes\boldsymbol{\nu}]\nabla f = \int_\Gamma \nabla f\cdot\mathbf{V} + f[\nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma \mathbf{V}],\end{aligned} \tag{15}$$

which gives the assertion. □

Therefore, by (12), the first variation with respect to shape perturbations of the functional J (10) is

$$\delta J(\Gamma;\mathbf{V}) = \int_\Gamma \phi\nabla\phi\cdot\mathbf{V} + \frac{1}{2}\int_\Gamma \phi^2\nabla_\Gamma \mathbf{X}\cdot\nabla_\Gamma \mathbf{V}, \tag{16}$$

where $\mathbf{V}$ is a vector perturbation of Γ. A more detailed derivation of (16) can be found in [88].

A simple gradient flow follows by first defining a vector velocity $\boldsymbol{\varphi}$ on the surface Γ by

$$\int_\Gamma \boldsymbol{\varphi}\cdot\mathbf{V} = -\delta J(\Gamma;\mathbf{V}) \tag{17}$$

for all perturbations $\mathbf{V}\in C^\infty(\Gamma)$. We then define a flow by

$$\frac{d}{dt}\mathbf{X}(\cdot,t) = \boldsymbol{\varphi}(\cdot), \quad \mathbf{X}(\cdot,t) = \Gamma(t), \tag{18}$$

which means the surface Γ will move with the velocity $\boldsymbol{\varphi}$.

3.4.3. Semi-implicit time discretization

We solve the gradient flow problem by using a semi-implicit time-discretization. This is done by setting $\boldsymbol{\varphi}$ to $\boldsymbol{\varphi}^{k+1}$ in (17) and using a backward Euler method for (18). Combining with Eq. (16) gives

$$\begin{aligned}&\int_{\Gamma^k}\boldsymbol{\varphi}^{k+1}\cdot\mathbf{V} = -\int_{\Gamma^k}\phi\nabla\phi\cdot\mathbf{V} - \frac{1}{2}\int_{\Gamma^k}\phi^2\nabla_{\Gamma^k}\mathbf{X}^{k+1}\cdot\nabla_{\Gamma^k}\mathbf{V}, \\ &\mathbf{X}^{k+1} = \mathbf{X}^k + \alpha\boldsymbol{\varphi}^{k+1},\end{aligned} \tag{19}$$

where the superscript is the iteration index and α is the step size to use in updating Γ^k. Rearranging gives the following variational formulation: given $\mathbf{X}^k$ and ϕ, find $\boldsymbol{\varphi}^{k+1}\in H^1(\Gamma)$ such that

$$\int_{\Gamma^k}\boldsymbol{\varphi}^{k+1}\cdot\mathbf{V} + \alpha\int_{\Gamma^k}\frac{1}{2}\phi^2\nabla_{\Gamma^k}\boldsymbol{\varphi}^{k+1}\cdot\nabla_{\Gamma^k}\mathbf{V} = -\int_{\Gamma^k}\phi\nabla\phi\cdot\mathbf{V} - \frac{1}{2}\int_{\Gamma^k}\phi^2\nabla_{\Gamma^k}\cdot\mathbf{V} \tag{20}$$

for all $\mathbf{V}\in H^1(\Gamma)$. Note that we used the identity $\nabla_{\Gamma^k}\mathbf{X}^k\cdot\nabla_{\Gamma^k}\mathbf{V} = \nabla_{\Gamma^k}\cdot\mathbf{V}$. Given the solution $\boldsymbol{\varphi}^{k+1}$, the new position Γ^{k+1} is obtained by the discrete update in Eq. (19). This process is then iterated until the sequence of surfaces $\{\Gamma^k\}$ reaches a minimum of (10). This minimization process is quite general and can be applied to the discrete manifold $\mathcal{E}$.

3.4.4. Finite element discretization

For computational purposes, we discretize $\boldsymbol{\varphi}^{k+1}$ and $\mathbf{V}$ in Eq. (20) with piecewise linear "hat" functions over the polygonal boundary Γ^k at each iteration index k. We then use a finite element implementation of Eq. (20) to solve for $\boldsymbol{\varphi}^{k+1}$, where the integrals are computed over Γ^k. The initial condition $\Gamma^0 := \widetilde{\Gamma}$ is given by the closed manifold of edge segments $\mathcal{E}$ resulting from Algorithm 4.

We also require interpolation of ϕ and $\nabla\phi$ with normalization. Let $\mathcal{I}^k$ be the continuous piecewise linear interpolant defined on Γ^k. Then we define the interpolation of ϕ, $\nabla\phi$ (with normalization) by

$$\begin{aligned}\mathcal{I}^k\phi(\mathbf{x}) &= \frac{\phi(\mathcal{T}^{\text{old}})(\mathbf{x})}{|\nabla\phi(\mathcal{T}^{\text{old}})(\mathbf{x})|}, \\ \mathcal{I}^k\nabla\phi(\mathbf{x}) &= \frac{\nabla\phi(\mathcal{T}^{\text{old}})(\mathbf{x})}{|\nabla\phi(\mathcal{T}^{\text{old}})(\mathbf{x})|}\end{aligned} \tag{21}$$

for all vertices $\mathbf{x}$ of the polygon Γ^k. Note that we interpolate the value of ϕ and $\nabla\phi$ from the old mesh $\mathcal{T}^{\text{old}}$. Therefore, for convenience in computing the integrals in (20), we use the interpolant (21).

The purpose of the normalization is to act as a pre-conditioner when ϕ is far from being a distance function. When ϕ is a distance function, then $|\nabla\phi| = 1$ and (21) reduces to standard linear interpolation. Otherwise, ϕ may have a small slope in some regions, which can affect the speed of convergence of our shape optimization method. This normalization procedure avoids that. See Section 4.3, for more discussion.

We used a step size $\alpha = 1.0$ and checked convergence of our optimization method by evaluating $\|\mathcal{I}^k\phi\|_{L^\infty(\Gamma^k)}$. Typically, about 10 to 30 iterations are needed to achieve $\|\mathcal{I}^k\phi\|_{L^\infty(\Gamma^k)} < 10^{-15}$; the actual number depends on how well the initial guess $\Gamma^0 := \widetilde{\Gamma}$ approximates Γ. Then we define Γ_{new} to be the converged shape.

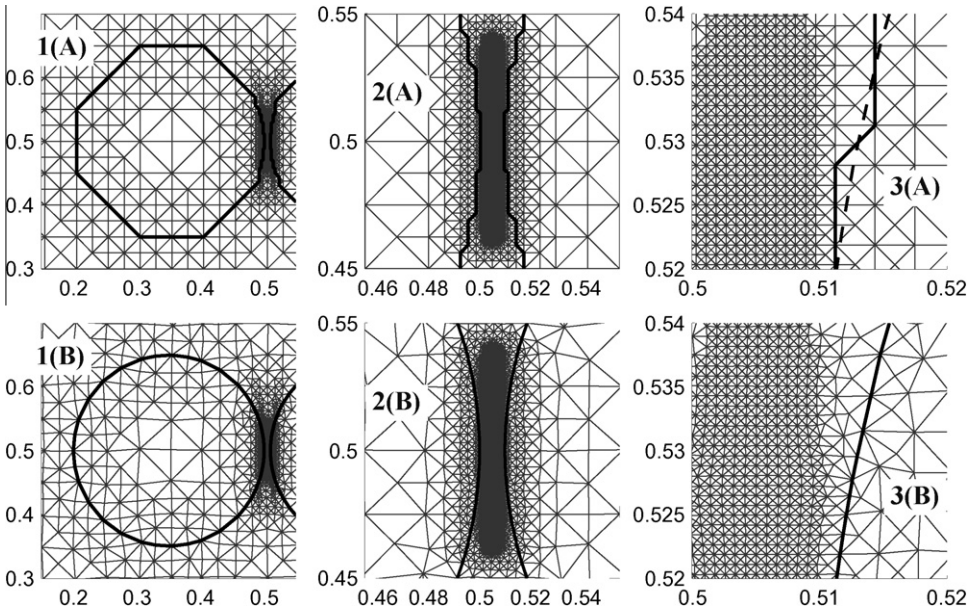

Fig. 6. Illustration of ensuring mesh conformity. The bolded black curve corresponds to the discrete manifold $\mathcal{E}$. Plots 1, 2, and 3 show different zoom-in levels of the mesh $\mathcal{G}$; (A) is before conformity and (B) is after. The dashed curve in three shows the zero level set of $\phi(\mathcal{T}^{\text{old}})$. In 3(B), the discrete manifold $\mathcal{E}$ clearly overlaps the dashed curve. In addition to the gradient flow method in Section 3.4.2, we also perform a few sweeps of standard mesh optimization (see Section 2.4). We denote the resulting conformed mesh by $\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}^{\text{old}}))$.

3.4.5. Final mesh conformity phase

In computing the final shape Γ_{new}, we ignored the interior and exterior mesh vertices that are not part of the manifold $\mathcal{E}$. We use a similar technique as in Section 2.3 to smoothly deform the remaining vertex positions into place. Let $\mathbf{D}$ be the net displacement in moving $\widetilde{\Gamma}$ to Γ_{new}. We then solve a vector Laplace problem on the mesh $\mathcal{G}(\mathcal{E} : \emptyset)$:

$$\begin{aligned} -\Delta \mathbf{x}_{\text{displace}} &= 0, \ \Omega, \\ \mathbf{x}_{\text{displace}} &= \mathbf{D}, \ \widetilde{\Gamma}, \\ \frac{\partial \mathbf{x}_{\text{displace}}}{\partial \nu} &= 0, \ \Gamma_{\text{ext}}, \end{aligned} \tag{22}$$

where $\mathbf{x}_{\text{displace}}$ is a displacement function for all points in the triangulation $\mathcal{G}$. Now we define the new deformed mesh $\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}^{\text{old}}))$ by adding $\mathbf{x}_{\text{displace}}$ to all the vertex positions in $\mathcal{G}(\mathcal{E} : \emptyset)$. In our implementation, we divide the total displacement into five incremental steps and use multiple solves of (22) to displace the vertices. This is necessary for smoothly deforming the mesh (see Fig. 6). In addition to the smoothed deformation, we also run a few sweeps of standard mesh optimization (see Section 2.4) on the mesh $\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}^{\text{old}}))$ while keeping the manifold vertices fixed.

There is no guarantee that the above method will not create inverted triangles (mesh entanglement). Therefore, we include a check for mesh inversion in our code. This never happened in the test cases we ran. In fact, one of the main reasons for the manifold selection algorithm in Section 3.3 is to help prevent mesh entanglement.

If no topological changes are imminent, the mesh $\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}^{\text{old}}))$ is the final output of the overall algorithm and is returned to the main simulation for the next time step.

4. Topological changes

Only a relatively small amount of work remains to account for topological changes of the interior manifold Γ. This is achieved by diffusing the distance function in the local region of the topological change. The details now follow.

4.1. Topological change detection

The detection of topological changes can be complicated. Essentially, one must look for regions where "disjoint" parts of the boundary are *close and collapsing together*. Detecting "closeness" is a common problem in computer graphics and collision detection, where the closest point transform or fast marching methods are used [9,53]. The determination of whether boundaries are collapsing depends on the nature of the velocity field coming from the physics. This can be especially difficult if the velocity field is becoming asymptotically slow near the point of pinch-off, as in a fluid droplet. Therefore, to avoid this difficulty, we assume the following hypothesis:

- If any region of the domain is sufficiently "thin" (i.e. has a thickness less than d_{neck}, a user specified tolerance), then a topological change is assumed to be occurring. In other words, d_{neck} acts as a resolution scale.

4.1.1. Locating thin regions

Deciding if there are thin regions in the shape of Γ is straightforward. We simply evaluate $|\phi|$ at all vertices in $\mathcal{V}_{\text{Sk}}$ (shape skeleton). If any vertex in the skeleton has a distance that is smaller than $d_{\text{neck}}/2$, then it is flagged as a "thin" vertex. We denote the set of thin vertices by $\mathcal{V}_{\text{thin}} \subset \mathcal{V}_{\text{Sk}}$. If $\mathcal{V}_{\text{thin}} = \emptyset$, then there are no imminent topological changes. Otherwise, we continue with the rest of this section.

4.1.2. Extra refinement near thin regions

We execute the change by solving a heat equation (discussed in the next section), which means we need an accurate solution near the thin regions. This requires the triangles in the thin regions to be sufficiently dense. We achieve this by adding an additional refinement step to the method described in Section 3. Algorithm 5 basically performs extra local refinement near the thin regions and is slightly graded (see Fig. 4, lower left corner). This requires us to compute the distance to the vertices in $\mathcal{V}_{\text{thin}}$. Let $\xi_{\text{thin}}(\mathcal{T}^{\text{old}})$ denote this distance function, which is interpolated onto the new mesh $\mathcal{G}$ when we execute Algorithm 5.

Algorithm 5. Extra refinement of thin regions

1: Set A_{outer}, A_{inner}, and B_{inner} to be constants such that $0 < A_{\text{inner}} \leqslant A_{\text{outer}} \leqslant 1$ and $0 < B_{\text{inner}} \leqslant 0.5$.
   Default values: $A_{\text{outer}} = 0.4$, $A_{\text{inner}} = 0.1$, and $B_{\text{inner}} = 0.2$.
2: Initialize triangulation $\mathcal{G}$ with result from Algorithm 3.
3: **loop**
4: Initialize $\mathcal{M} = \emptyset$.
5: **for all** triangles T in $\mathcal{G}$ **do**
6: Estimate triangle diameter: let L_T be the length of the longest edge of T.
7: # create an inner and outer region, such that the inner region is more refined.
8: **if** ($\min_T \xi_{\text{thin}} \leqslant d_{\text{neck}}$ and $L_T \geqslant A_{\text{outer}} d_{\text{neck}}$) OR ($\min_T \xi_{\text{thin}} \leqslant B_{\text{inner}} d_{\text{neck}}$ and $L_T \geqslant A_{\text{inner}} d_{\text{neck}}$)
9: include T in $\mathcal{M}$.
10: **end if**
11: **end for**
12: **if** $\mathcal{M} \neq \emptyset$ **do**
13: Execute the longest edge bisection routine [67] on $\mathcal{G}$ with the marking $\mathcal{M}$.
14: **else**
15: Exit loop.
16: **end if**
17: **end loop**
18: **return** $\mathcal{G}$.

If we ignore Algorithm 5, the rest of our method still works. The extra refinement just gives better resolution of the shape after the topological change. Our choice of A_{outer}, A_{inner}, and B_{inner} is not very critical, but seemed to work well for the test cases we tried.

Remark 1. The output of Algorithm 5 is the base mesh $\mathcal{G}$. We then proceed just as before with the manifold selection method in Section 3.3 to find $\mathcal{E}$. And just as before, we create a deformed mesh $\mathcal{G}\left(\mathcal{E} : \phi\left(\mathcal{T}^{\text{old}}\right)\right)$ that conforms to the zero level set of ϕ *before* the topological change (recall Section 3.4). We use $\mathcal{G}\left(\mathcal{E} : \phi\left(\mathcal{T}^{\text{old}}\right)\right)$ in Section 4.2.3 for re-computing the distance function and for diffusing the distance function to simulate a topological change. This was done to ensure an adequate mesh for *resolving* the topological change.

4.2. Obtain new domain topology

4.2.1. Level-set method

An alternative to front-tracking of explicit boundaries is to use a level set method [61,70]. In this case, the boundaries are represented implicitly as the zero level set of some scalar function ψ. The evolution of the boundaries is captured by solving the following hyperbolic equation

$$\begin{aligned} &\partial_t \psi(\mathbf{x}, t) + \mathbf{u}(\mathbf{x}) \cdot \nabla \psi(\mathbf{x}, t) = 0 \quad \text{for all} \quad \mathbf{x} \quad \text{and all} \quad t > 0, \\ &\psi(\mathbf{x}, 0) = \psi_0(\mathbf{x}), \end{aligned} \tag{23}$$

where $\mathbf{u}$ is the velocity field that moves the boundaries and ψ_0 is usually taken to be the distance function to the two-phase boundary. The main advantage of level set methods is that they handle topological changes automatically. There is no

explicit decision needed to determine when a topological change happens nor *how* it happens or how the local geometry should look. It is this aspect that we wish to take advantage of in our method.

4.2.2. Viscosity solution

Eq. (23) is linear and well-posed as long as the velocity $\mathbf{u}$ is smooth [98]. In order to have two boundaries (i.e. curves defined by the zero level set of ψ) touch in finite time, it is necessary to have a velocity field that is not Lipschitz [5] (this follows from standard uniqueness theorems for ODE's). But in this case, the solvability of (23) is questionable, especially in the case of a topological change. To address this, we add a small diffusion term:

$$\partial_t \psi + \mathbf{u} \cdot \nabla \psi = \varepsilon_0 \Delta \psi. \tag{24}$$

This guarantees that the equation is well-posed. In effect, we obtain the "viscosity" solution [32] of (23) which allows for splitting and reconnection of implicitly defined boundaries.

4.2.3. Computing new topology by diffusion

First we compute the signed distance function $\phi(\mathcal{G}(\mathcal{E} : \phi(\mathcal{T}^{\text{old}})))$ on the new mesh that was just created (recall Remark 1 in Section 4.1.2). Then we get the new topology of Γ by locally diffusing ϕ, which we do in two steps. First, we solve one time step of the following time-discrete heat equation with small parameter ε_0 and some step size δt, i.e. compute ψ such that

$$\psi - \delta t \varepsilon_0 \Delta \psi = \phi, \tag{25}$$

where ϕ is the initial condition. We omit the convective part that appears in (24) for the following reason. The term $\mathbf{u} \cdot \nabla \psi$ is only needed to capture the motion of the interface. But at this stage of our algorithm, the domain motion has already been accounted for earlier in the time-stepping method (see Fig. 1). If we did include the convective part, we would need to solve the full physics again, which would complicate our method. In fact, later we show that the product $\delta t \varepsilon_0$ is independent of δt (see (28)). The zero level set of ψ captures the topological change of the zero level set of ϕ. This is an artificial step, but is in the spirit of viscosity solutions which allows colliding fronts (level sets) to merge and reconnect.

Solving (25) globally diffuses the level set, which is undesirable. We make the diffusion local by performing a simple procedure. Let θ:$[0,\infty) \to [0,1]$ be a cut-off function defined by:

$$\theta(s) = \begin{cases} 1, & 0 \leqslant s \leqslant d_{\text{neck}}, \\ \left[\cos\left(\frac{\pi}{2}\frac{s - d_{\text{neck}}}{d_{\text{neck}}}\right) + 1\right] \Big/ 2, & d_{\text{neck}} < s < 3d_{\text{neck}}, \\ 0, & 3d_{\text{neck}} \leqslant s. \end{cases} \tag{26}$$

Next we define a new level set function that is only locally diffused

$$\psi_{\text{top}}(\mathbf{x}) = \theta(\xi_{\text{thin}}(\mathbf{x}))\psi + [1 - \theta(\xi_{\text{thin}}(\mathbf{x}))]\phi \quad \text{for all} \quad \mathbf{x} \quad \text{in} \Omega. \tag{27}$$

Eq. (27) smoothly localizes the diffusion of the level set function. Thus, only regions of topological change are affected.

The parameter ε_0 must be chosen to guarantee that thin regions will connect or pinch-off. A classical result on the diffusion length [62] indicates that ε_0 should satisfy $\sqrt{\delta t \varepsilon_0} \approx O(d_{\text{neck}})$. Hence, we set the diffusion parameter as

$$\frac{1}{2}\frac{d_{\text{neck}}^2}{\delta t} \leqslant \varepsilon_0 \leqslant \frac{d_{\text{neck}}^2}{\delta t}, \tag{28}$$

The addition of the diffusion term is directly analogous to artificial viscosity methods used for solving hyperbolic equations, which adds a small amount of diffusion on the order of the mesh size. In our computations, δt plays no role because of cancelation in (25), so we set $\delta t = 1$ (in this section) and $\varepsilon_0 = 0.7 d_{\text{neck}}^2$. The exact choice of ε_0 is not very critical, as long as (28) is satisfied.

4.2.4. Finite element discretization

We use a standard finite element approximation of (25) when solving for the new topology. Specifically, ψ and ϕ are represented as piecewise linear functions over the triangulation $\mathcal{G}\left(\mathcal{E} : \phi\left(\mathcal{T}^{\text{old}}\right)\right)$ which gives a standard well-posed system that can be solved by many techniques.

A simple way to reduce computational cost is to only solve (25) locally by limiting the computational domain to a neighborhood of $\mathcal{V}_{\text{thin}}$. The solution procedure does not change, except a Neumann condition is applied on the outer boundary of the small computational domain. This is easily implemented if an iterative method is used to solve (25).

4.3. Rerun mesh conformity phase

We must conform the original, adaptively refined mesh $\mathcal{G}$ to the new zero level set of ψ_{top} (see Eq. (27)), which is defined on the previously deformed mesh $\mathcal{G}\left(\mathcal{E} : \phi\left(\mathcal{T}^{\text{old}}\right)\right)$ (recall Remark 1). We do this by first selecting a new candidate manifold $\mathcal{E}_{\text{top}}$ by running Algorithm 4 except with ψ_{top} rather than $\phi\left(\mathcal{T}^{\text{old}}\right)$. This is then followed by the method described in Section 3.4 for ensuring mesh conformity, i.e. this generates the mesh $\mathcal{G}(\mathcal{E}_{\text{top}} : \psi_{\text{top}})$ which is the final output of our algorithm.

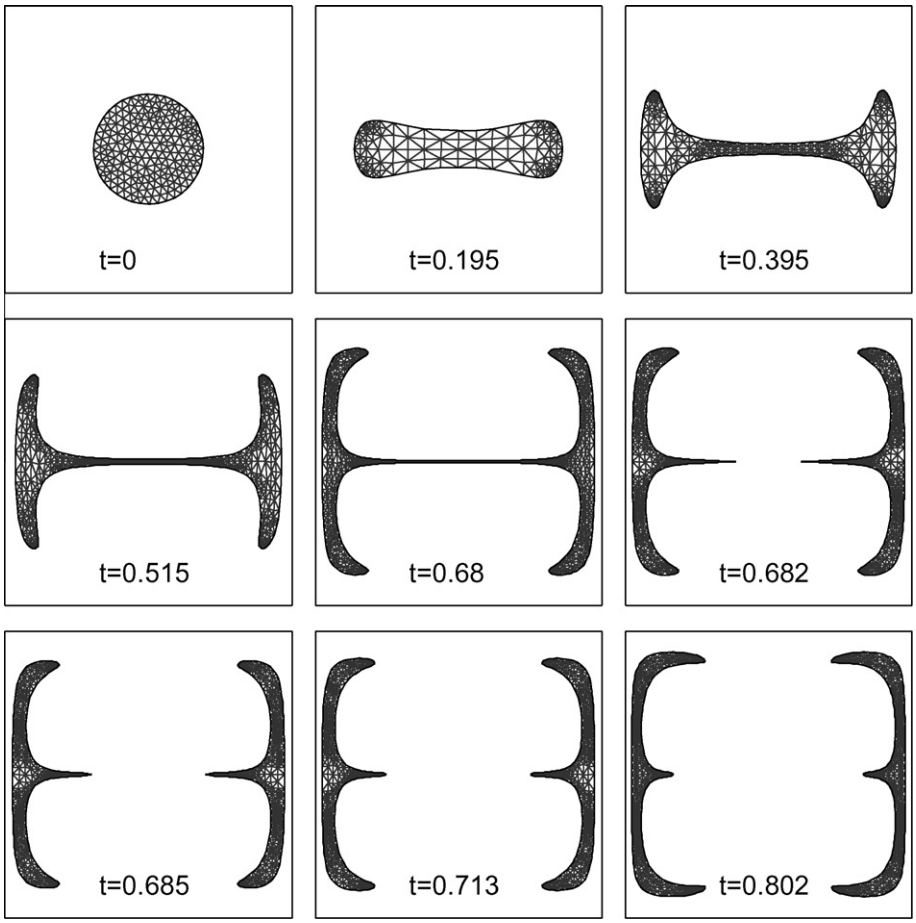

Fig. 7. Initial circular domain (first frame) and subsequent deformation with topological change. Only the interior mesh is shown. Outer box is a unit square. Eventually, the domain becomes very thin in the center and a topological change is executed.

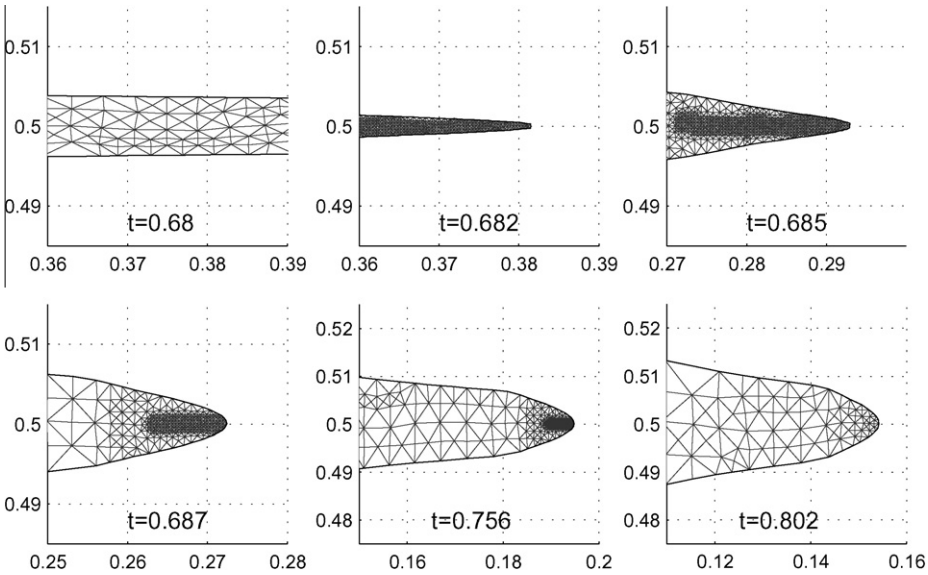

Fig. 8. Zoom-in of the pinching region in Fig. 7 ($d_{\text{neck}} = 5 \times 10^{-3}$). All of the triangles that were in the pinching region become part of the exterior triangulation (not shown). Multiple topological changes happen because of the thinness of the filament. Note that the axis limits are not the same in all frames.

We now discuss the reason for the pre-conditioning step in (21). Because the level set update (25) is a time-discrete heat equation, it acts to smooth the initial data and drive it towards a constant value (when Neumann boundary conditions are used). This causes ψ_{top} to be relatively flat in the region of topological change compared to the initial condition which was a distance function. Ergo, the ϕ and $\nabla\phi$ terms that appear in (20) would give weak forcing for moving the manifold. Therefore, without the normalization in (21), the active contour algorithm would take many more iterations to converge.

5. Numerical experiments

We present three simulations to demonstrate the method described in Section 1.3. The first simulation contains no physics and consists of a mesh that is moving with a prescribed velocity field. The second simulation comes from an application known as electro-wetting [15,16,91], which consists of a Hele-Shaw cell [74,37] with the ability to modify surface tension effects through electric fields. These devices are capable of splitting and merging droplets and have potential applications

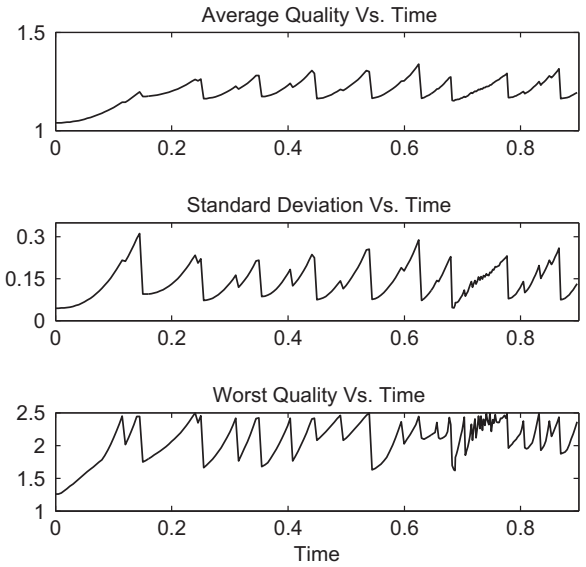

Fig. 9. Plots of triangle qualities versus simulation time (Section 5.1). Triangle qualities (of the entire mesh) were computed using the formulas given in [47]. Here, we plot the mean, standard deviation, and worst case triangle quality at each time-step of the simulation. The quality measure ranges from 1.0 to ∞, where 1.0 is the best and corresponds to an equilateral triangle. Note that the isosceles right triangles that make up the generated meshes before ensuring mesh conformity have a quality measure of approximately 1.1547.

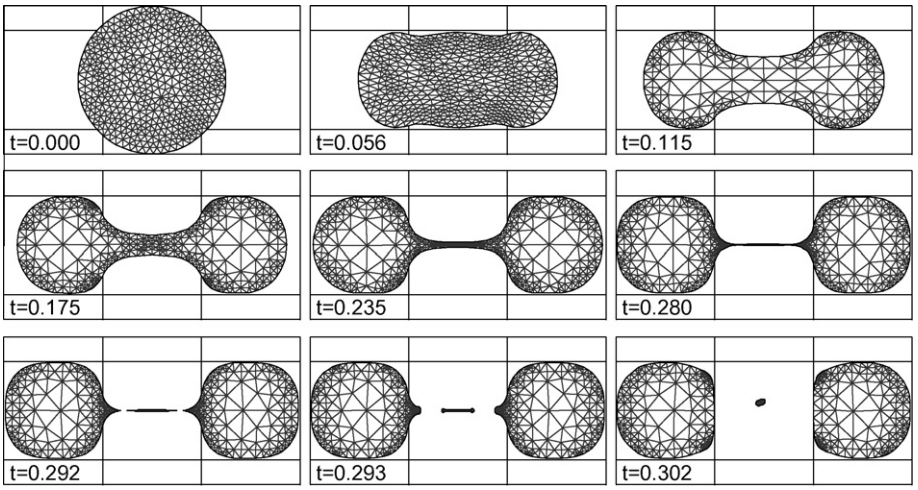

Fig. 10. EWOD driven droplet motion; only the interior mesh is shown. Plot box has units of 1×0.5. The droplet starts in a circular shape and is pulled apart because the left and right electrodes are turned on. Eventually, the droplet pinches in two places (symmetrically) resulting in an elongated satellite droplet. The time-scale of relaxation of the satellite droplet is very fast (see Fig. 11).

for "lab-on-a-chip" devices [33,38]. The third simulation demonstrates reconnection of droplets in a Hele-Shaw cell due solely to surface tension (no electro-forcing).

Remark 2. In all experiments, the mesh quality [47] was maintained within the following criteria. The worst quality value for any triangle must be less than 2.5 and no more than 5% of all triangles can have a quality above 2.0. Note that the quality metric in [47] is scale invariant and is defined so that an equilateral triangle has quality 1.0, and any other triangle shape has a higher value (worse quality).

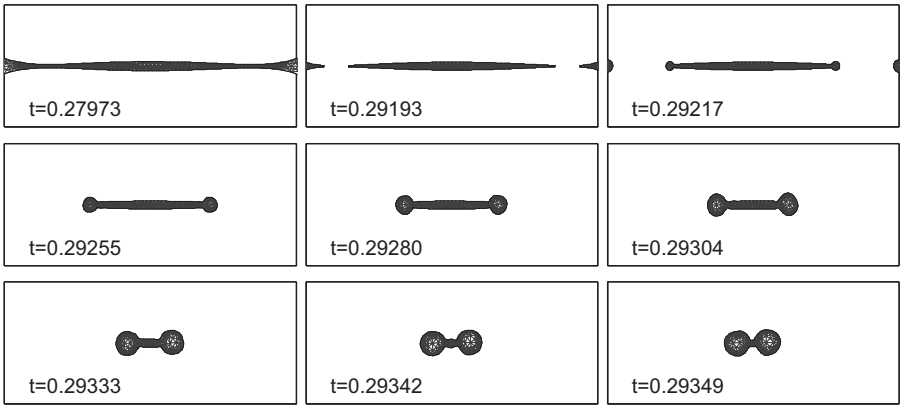

Fig. 11. Zoom-in of satellite droplet relaxation. Plot box has units of 0.24×0.1. The droplet begins elongated and rapidly collapses together. The "dumbbell" shape arises because the velocity at the ends is extremely large, while the velocity in the central region is almost zero. Thus the end pieces of fluid overtake the more quiescent region, which causes the shape to "bunch up" at the ends. Evolution continues in Fig. 12.

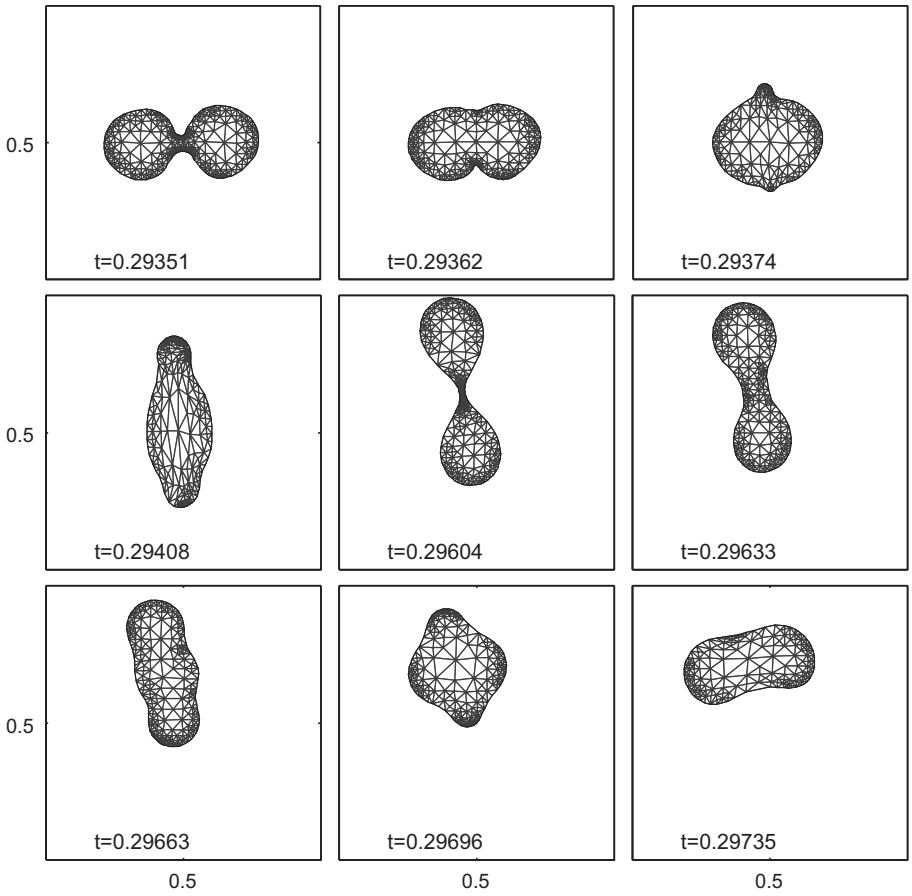

Fig. 12. Zoom-in of satellite droplet relaxation; continued from Fig. 11. Plot box has units of 0.08×0.08. Evolution continues in Fig. 13.

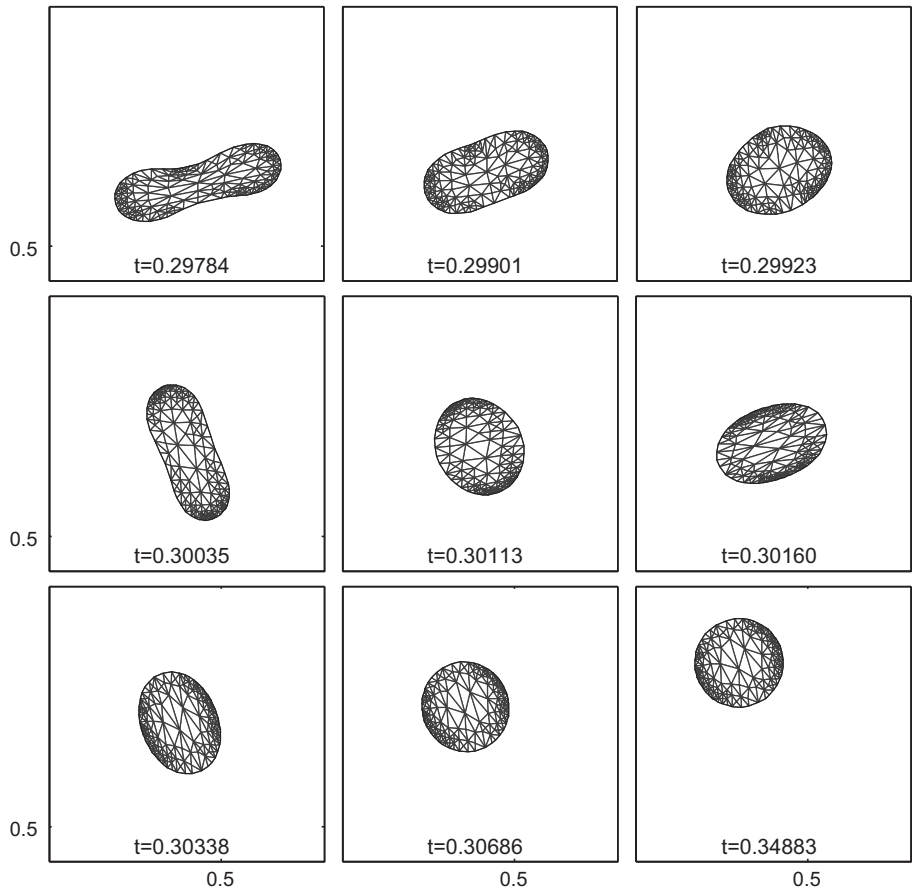

Fig. 13. Zoom-in of satellite droplet relaxation; continued from Fig. 12. Plot box has units of 0.08×0.08. The droplet comes to rest at a position that is not symmetric (i.e. not at (0.5,0.5)). This is because the PDE solution inside the satellite droplet requires more resolution. The velocity field is very large (initially) at the ends of the satellite drop, is near zero in the center, and rapidly changes in a small region. Thus, a *highly* refined mesh, in the interior of the droplet, is needed to resolve the dynamics. However, our main point here is to demonstrate the ability of our method to handle the extreme deformations exhibited by topological changes.

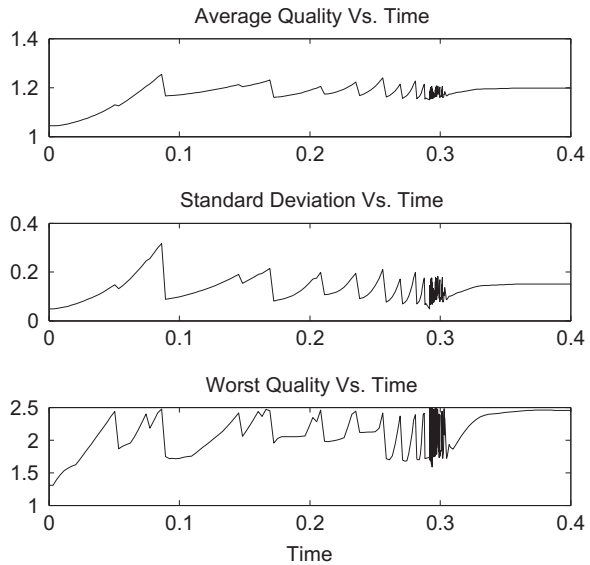

Fig. 14. Plots of triangle qualities versus simulation time (Section 5.2.3). Same format as in Fig. 9.

If this criteria is violated, then we perform 4 sweeps of mesh optimization [31] in an attempt to satisfy the criteria. If the criteria is still violated, then we re-mesh via our mesh generation algorithm (Section 3). Recall the block diagram in Fig. 1.

5.1. Rotating vortices

In this simulation, we prescribe a velocity field $\mathbf{u} = (u, v)$ of the form

$$u(x,y) = \sin(2\pi x)\cos(2\pi y),$$
$$v(x,y) = -\cos(2\pi x)\sin(2\pi y),$$

which is a two-by-two array of counter-rotating vortices, and the divergence of $\mathbf{u}$ is zero. The initial domain shape is a circle inside a unit square, shown in Fig. 7; the initial mesh was generated by the commercial package "MeshGen". The vertices of the boundary move with the given velocity field and the rest of the vertices move by extending the vector velocity on the boundary using a Laplace solve (see Eq. (4)). The mesh undergoes severe deformation due to the counter-rotating vortices, though the vector Laplace solve does limit the amount of mesh distortion. As the domain becomes thin in the middle, and reaches a minimum thickness of $d_{\text{neck}} = 5 \times 10^{-3}$, the topological change routine is executed (in addition to our general re-meshing algorithm).

In Fig. 8, we show a closeup of the pinching region depicted in Fig. 7. Of course, the dynamics of the flow after the pinch do not change since the velocity field is prescribed.

The extreme deformation shown by this example demonstrates the ability of our method to compensate for mesh distortion and detect thin regions. The optimization of the mesh boundary is also satisfactory. In Fig. 9, we show triangle quality

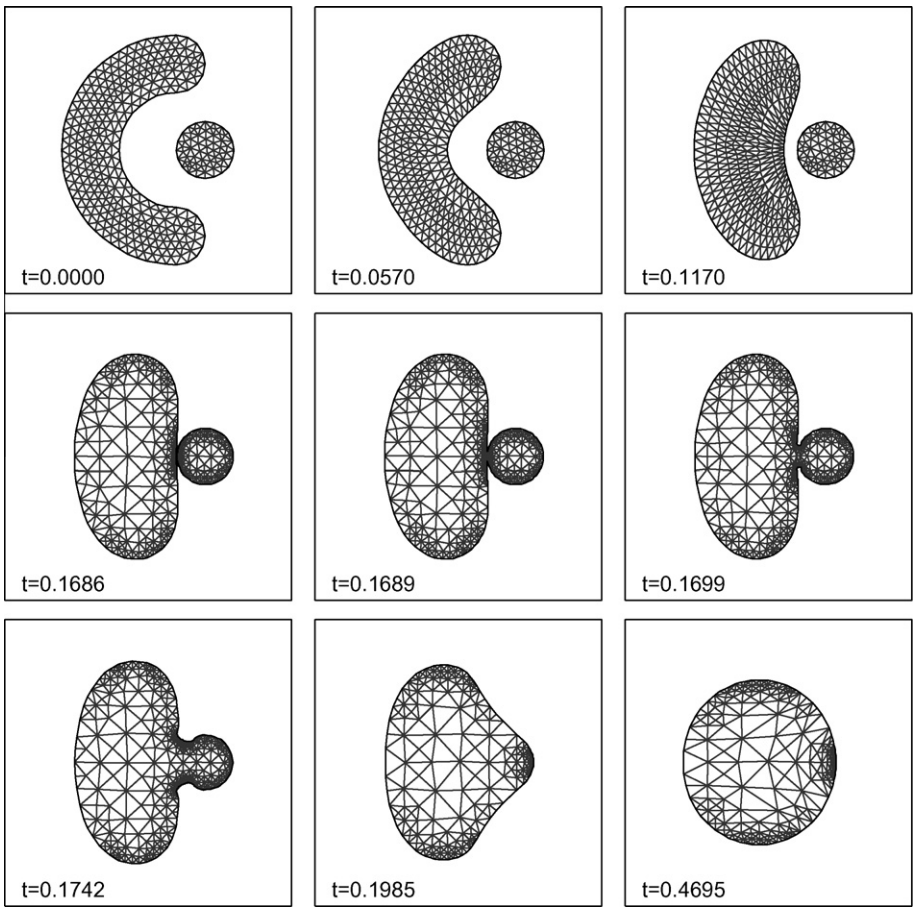

Fig. 15. Droplet motion by surface tension; only the interior mesh is shown. Plot box has units of 1×1. As the left droplet unbends, it comes closer to the stationary drop on the right. Eventually, the droplets connect when the spacing between them drops below $d_{\text{neck}} = 2 \times 10^{-3}$. This creates two high curvature regions (symmetric with respect to the midline of the plot box) on the top and bottom of the reconnection region. The time-scale of relaxation is very fast (see Figs. 16 and 17).

statistics as the domain deforms in time. The large jumps in the curves correspond to re-meshing the domain. The smaller jumps happen when the mesh vertex positions are optimized (see Section 2.4) to improve quality (i.e. no changes in mesh topology).

5.2. EWOD driven droplets

5.2.1. Summary of model

Next, we use a simulation of an Electro-Wetting On Dielectric (EWOD) device to drive the motion of a water droplet to a topological change (droplet pinching). The device consists of two parallel plates very close together with a water droplet squashed in between with air surrounding it, hence the problem is effectively 2-D Hele-Shaw flow with surface tension. A 3×3 array of square electrodes is embedded in the bottom plate, which are used for applying voltages that can change the effective surface tension locally [72]. This allows for the ability to force a circular droplet to pinch-off. The governing equations are given by

$$\begin{aligned} &\beta_1 \frac{\partial \mathbf{u}}{\partial t} + \beta_2 \mathbf{u} + \nabla p = 0, \quad \text{in } \Omega, \\ &\nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega, \\ &p - (\kappa + E) = 0, \quad \text{on } \Gamma, \end{aligned} \tag{29}$$

where β_1 and β_2 are non-dimensional parameters, κ is the curvature of Γ, and E is a given forcing function that comes from the electric field. Here, Ω is the interior domain and Γ is the manifold between the interior and exterior phases. Note the presence of the inertial term $\partial \mathbf{u}/\partial t$. The interface equation of motion is given by

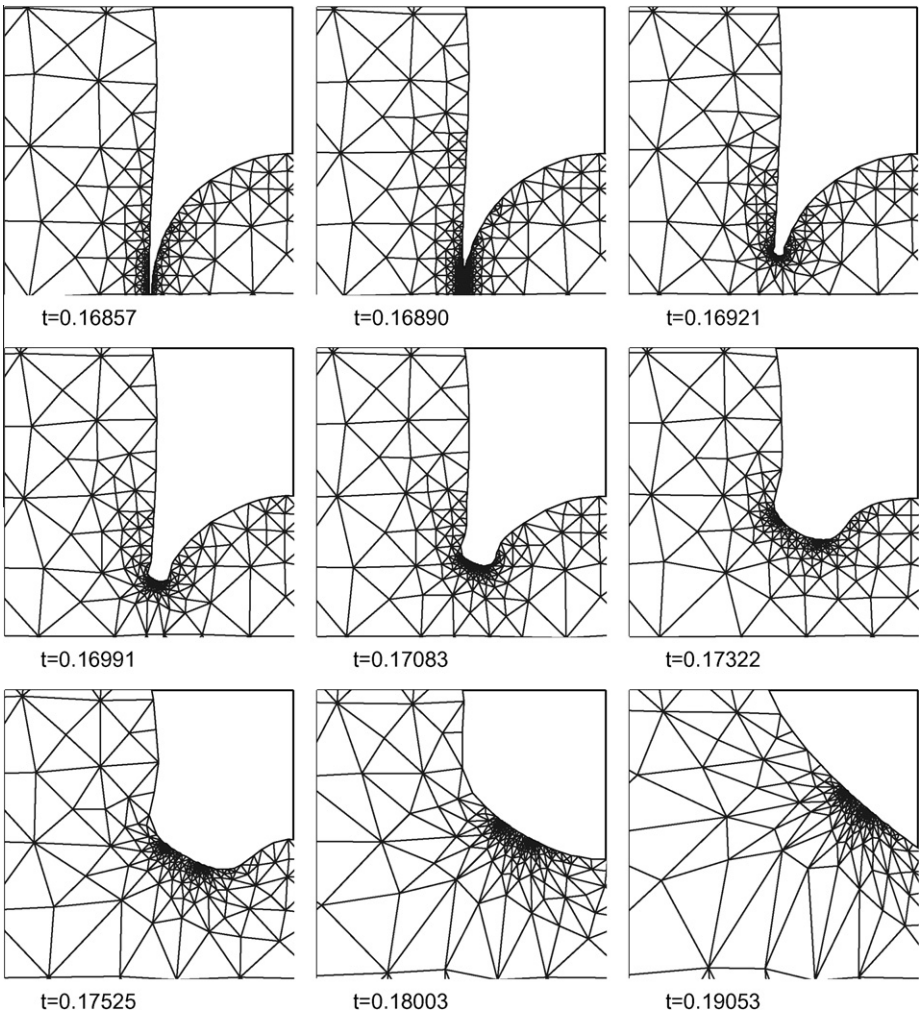

Fig. 16. Zoom-in of sharp reconnection region relaxing (**top**). Plot box has units of 0.2×0.2.

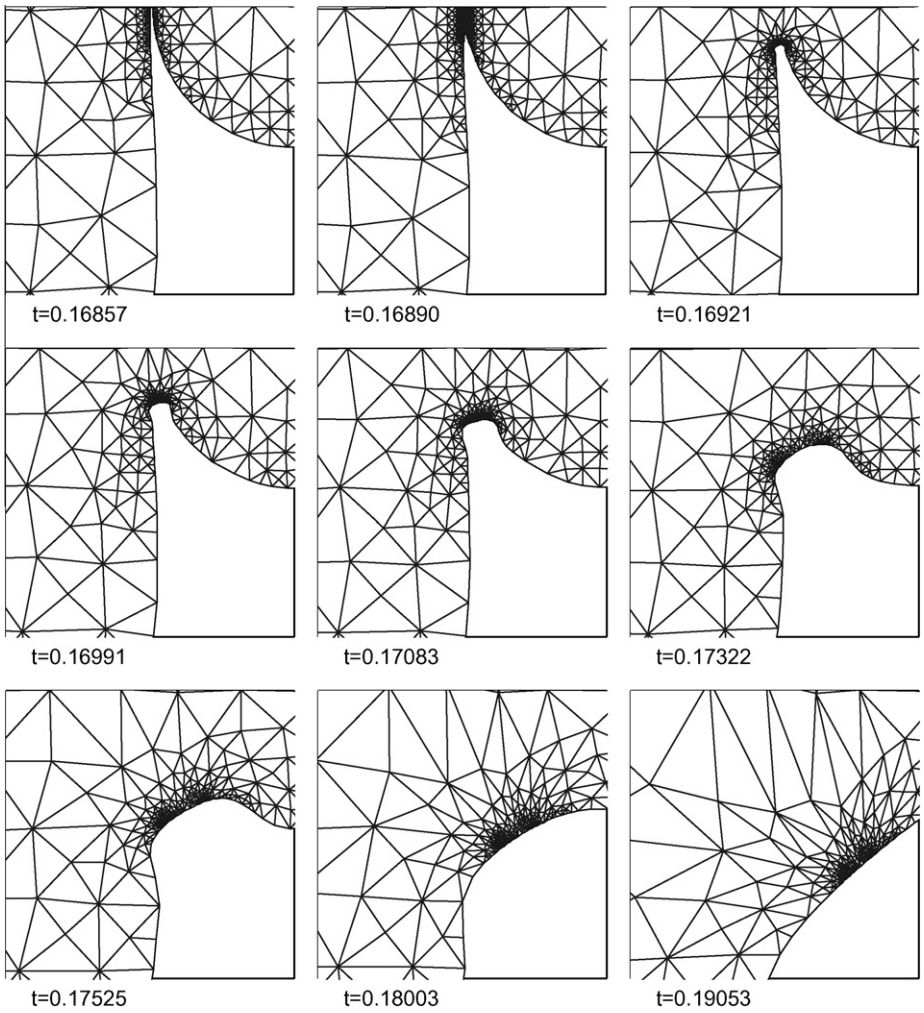

Fig. 17. Zoom-in of sharp reconnection region relaxing (**bottom**). Plot box has units of 0.2×0.2.

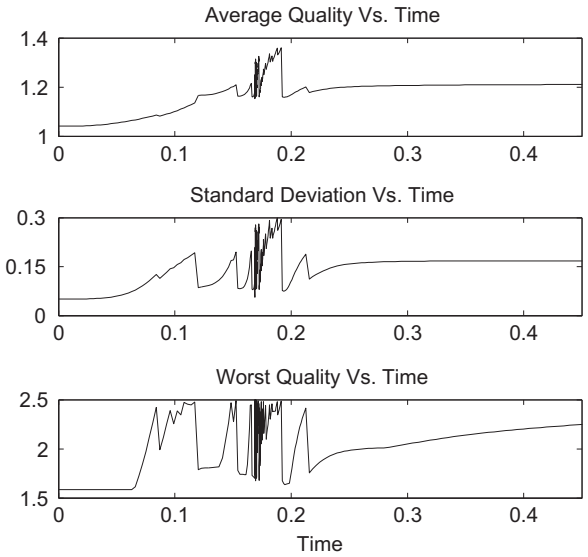

Fig. 18. Plots of triangle qualities versus simulation time (Section 5.2.4). Same format as in Fig. 9.

$$\partial_t \mathbf{x} = \mathbf{u}(\mathbf{x}, t) \quad \text{for all } \mathbf{x} \quad \text{in } \Gamma. \tag{30}$$

For more details about the model and numerical method, see [89,91,90,92].

5.2.2. Wait period for topological changes

For the droplet flow experiments, we define a "wait" period for topological changes to happen. In fluid pinching, it is likely that a thin "spike" will be present after the pinch-off has occurred. Which means that our method of detecting topological changes could trigger another change immediately after because of the thin region near the sharp corner. In fact, this may cause a sequence of topological changes that can "eat" the spike away. This is undesirable in some cases, because the natural dynamics may resolve the spike naturally without any extra topological changes occurring.

Therefore, in these experiments, we define a wait period of $T_{\text{wait}} := 10^{-3}$. If a topological change is executed, then for the next T_{wait} seconds (non-dimensional), topological changes are not allowed to occur. General re-meshing is still allowed. We do this so that the natural dynamics of the flow can have a chance to smooth out the domain.

5.2.3. A pinching droplet

Fig. 10 shows a droplet overlaying a 3×3 grid of square electrodes. The voltage is actuated on the left and right electrodes which causes the droplet to be pulled apart. Eventually, the droplet develops a thin neck which pinches off at two separate regions when the thickness drops below $d_{\text{neck}} = 5 \times 10^{-4}$. This type of pinching singularity in Hele-Shaw flow was shown in [74,37,36].

Immediately after the pinch-off, the time-scale becomes extremely small. This is due to the large curvature that is present at the end points of the elongated satellite droplet. As the flow progresses, the satellite droplet takes on a "dumbbell" shape. This is reasonable given that the velocity field is mostly concentrated at the end points and is negligible everywhere else. Essentially, the end regions overtake the stationary part of the droplet. See Fig. 11.

A closeup of the satellite droplet evolution is shown in Figs. 12 and 13. The bulging ends of the droplet slam into each other which causes the droplet to elongate in the vertical direction and stretch into another dumbbell shape. This happens because of the inertial term in (29). The droplet continues to oscillate with decreasing amplitude until it relaxes to a circular shape (Fig. 13). Despite the extreme deformation, our algorithm is able to capture this evolution. Fig. 14 shows triangle quality statistics as the domain deforms in time.

5.2.4. Joining of droplets by surface tension

In this last experiment, we use the EWOD simulation without any electrical forcing (i.e. $E = 0$). Hence, the flow is purely due to surface tension. This example shows how our method handles connecting or joining droplets. Fig. 15 shows two droplets, one circular and the other elongated and bent, that eventually coalesce. The bent droplet relaxes and develops an "air" gap with the smaller drop. The reconnection takes place when the gap drops below $d_{\text{neck}} = 2 \times 10^{-3}$. As the droplet relaxes, the large curvature regions get smoothed out.

After the reconnection, the time-scale becomes very small because of the large curvature near the cusp like regions. Similarly to the previous case, the instantaneous flow is highly localized near the region of reconnection and negligible everywhere else. A zoom-in of this flow is given in Figs. 16 and 17. In both figures, we get a "mushroom" like shape as the high curvature region smooths out. This is essentially the same effect observed in the dumbbell shape of Fig. 11. Fig. 18 shows triangle quality statistics as the domain deforms in time.

6. Conclusions

We have presented a method for mesh generation of 2-D domains undergoing large deformations and topological changes. The method uses a level set formulation to indicate how the topology changes, and is only used during the time-step of the topological change. In addition, an active contour method using a shape optimization technique is used to improve boundary mesh conformity to the zero level contour of the level set function. Topological changes happen when narrow regions of the domain become thinner than the user specified tolerance d_{neck}. All simulations were implemented within the MATLAB/C++ toolbox FELICITY (Finite ELement Implementation and Computational Interface Tool for You) developed by the second author.

6.1. Mass preservation during topological changes

Our method for allowing topological changes is essentially a level set method with a small amount of diffusion added. Hence, mass may not be conserved during a pinching or reconnection event. The amount of mass loss or gain is directly controlled by the user specified tolerance d_{neck}, so can be tuned if desired. Moreover, the loss or gain only happens during a topological event. The rest of the time, our method is just front-tracking. Preserving mass at all times is a desirable property and the subject of future work.

6.2. How to start the algorithm

One issue with our method is that it requires a global inside and outside mesh that conforms to the initial boundary. This can be inconvenient if only a polygonal representation of the boundary is available. We basically need a reference mesh in order to compute the distance function to the boundary which is then used in our mesh generation algorithm.

One solution would be the following. Start with a coarse reference mesh and adaptively refine all triangles that intersect the manifold, and continue until some minimum feature size is reached. Next, the distance function could be computed in a narrow band around the boundary then extended by the method in [8]. This would give a distance function whose zero level set approximates the boundary to within the desired feature size. One could then use this distance function in our active contour routine for generating a conforming mesh. Unfortunately, this would most likely be more expensive than the method we describe in this paper if the minimum feature size is very small in order to account for some regions of high curvature. But this would only be done once and could be considered as a form of pre-processing. In the case where the initial manifold is known as the level set of some function, then one can take advantage of this directly.

6.3. Handling exterior and interior phases

We focused on the case where the exterior phase was not important, hence the mesh of the exterior (particularly of Γ_{ext}) was not important. Examples where it is important are: flow of gas bubbles through a liquid phase inside a closed box, multiple materials that deform in a problem of elasticity, etc.. . .Handling these cases is a straightforward extension of what we have described. If the enclosing "box" (denoted Γ_{box}) is rectangular, then one can easily modify the initial triangulation $\mathcal{T}^{\text{init}}$ to be the box (i.e. $\Gamma_{\text{ext}} = \Gamma_{\text{box}}$, recall Section 3.2.1). Otherwise, one must first cover the enclosing container with a rectangular shaped initial mesh (with a sufficiently large surrounding buffer region).

Then we generate a mesh $\mathcal{G}$ that resolves the two-phase manifold Γ *and* the enclosing shape Γ_{box}. Next, candidate manifolds must be found that approximate Γ and Γ_{box}. Finally, during the mesh conformity phase, the mesh is deformed so as to conform to both Γ and Γ_{box} simultaneously.

In fact, this can be generalized further to include any number of internal boundaries that may or may not be interacting. If they do interact, then care must be taken in defining what a topological change is and how they occur.

6.4. Remove wait period

The need for a wait period T_{wait} in Section 5.2.2 is due to the way that we characterize a topological change, i.e. we only view thinness as an indicator. This can be overcome if we include some other information, such as the nature of the flow field in a thin region. Likewise, processing the shape skeleton better may help identify the correct regions of topological change [71,77]. This will be a point of future work.

6.5. Generalize to 3-D

Our method mostly generalizes to 3-D. Our shape skeleton computation can be directly extended to tetrahedral meshes in 3-D or one could possibly use the variational approach in [71,77] to approximate a smoothed skeleton. In general, our philosophy is that PDE based/variational methods can be quite effective for discrete mesh generation. But a mesh, no matter how refined it is, is an inherently discrete structure and any algorithm for mesh generation must deal with that, which our candidate manifold selection does (see Section 3.3). However, this method does not generalize to 3-D. On the other hand, we think the method in [11] could be adapted to our needs for 3-D tetrahedral meshes.

Estimating the curvature in 3-D will require a slightly different method than computing $\nabla_{\Gamma} \cdot \boldsymbol{v}$, because that only gives the total curvature. Instead, we will need to estimate $\nabla_{\Gamma} \boldsymbol{v}$ (i.e. the second fundamental form [21]). The largest eigenvalue of the 3×3 matrix $\nabla_{\Gamma} \boldsymbol{v}$ corresponds to the largest principle curvature of the surface and can be used for refining by curvature in Algorithm 3. Estimating $\nabla_{\Gamma}\, \boldsymbol{v}$ can be done by a similar method as given in (8) of Section 3.2.3.

More improvements could be made, including having a method to adapt the mesh boundary (in some sense) when doing the shape optimization/smoothing step. One criteria could be to maximize the shape regularity of the boundary mesh (while smoothing), which is especially important for using our method in 3-D.

6.6. Meshing domains with corners

Lastly, we mention the possibility of extending our method to handle manifolds with corners (in 2-D). If the corners are specified, then one can add another stage for ensuring conformity that occurs before the main phase given in Section 3.4. The initial stage would deform the mesh so that appropriately chosen points in the manifold $\mathcal{E}$ are made to conform to the corner points. This can be done by computing distance functions to each individual corner. Next, note that the corner points would automatically partition the 1-D manifold into disjoint connected segments. Thus, the active contour phase would consist of optimizing each individual segment so as to conform to the whole shape. The final mesh conformity phase described in Section 3.4.5 would remain the same.

However, corners can be complicated in 3-D [69] and would require multiple phases to handle the corner tip, followed by the corner edges, and finally the remaining patches of smooth surface. Of course, the shape regularity of the mesh will be limited by the angle of the corners.

Acknowledgments

We wish to acknowledge funding support. Walker was partially supported by the NSF-VIGRE grant of the Department of Mathematics, University of Maryland, College Park, in addition to support from NSF-RTG grant DMS-0602235 and DOE grant DE-FG02–88ER25053. Nochetto was partially supported by NSF grants DMS-0505454 and DMS-0807811, with current support by NSF grant CBET 0754983 (program managers Maria Burka and William Schultz).

References

[1] Silas Alben, Michael J. Shelley, Coherent locomotion as an attracting state for a free flapping body, in: Proceedings of the National Academy of Sciences USA, vol. 102, 2005, pp. 11163–11166.
[2] Eugenio Aulisa, Sandro Manservisi, Ruben Scardovelli, A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking, Journal of Computational Physics 197 (2) (2004) 555–584.
[3] Randolph E. Bank, PLTMG: A Software Package for Solving Elliptic Partial Differential Equations: Users' Guide 8.0, Society for Industrial Mathematics, 1987.
[4] Eberhard Bänsch, Pedro Morin, Ricardo H. Nochetto, A finite element method for surface diffusion: the parametric case, Journal of Computational Physics 203 (2005) 321–343.
[5] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, John A. Strain, A semi-Lagrangian contouring method for fluid simulation, ACM Transactions on Graphics 25 (1) (2006).
[6] Tobias Baumgart, Samuel T. Hess, Watt W. Webb, Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension, Nature 425 (2003) 821–824.
[7] T.D. Blake, A. Clarke, E.H. Stattersfield, An investigation of electrostatic assist in dynamic wetting, Langmuir 16 (6) (2000) 2928–2935.
[8] Folkmar Bornemann, Christian Rasch, Finite-element discretization of static Hamilton–Jacobi equations based on a local variational principle, Computing and Visualization in Science 9 (2) (2006) 57–69.
[9] David E. Breen, Sean Mauch, Ross T. Whitaker, 3d Scan conversion of csg models into distance volumes, in: VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization, ACM Press, New York, NY, USA, 1998, pp. 7–14.
[10] Susanne C. Brenner, L. Ridgway Scott, The Mathematical Theory of Finite Element Methods, second ed., Springer, New York, NY, 2002.
[11] R. Bridson, J. Teran, N. Molino, R. Fedkiw, Adaptive physics based tetrahedral mesh generation using level sets, Engineering with Computers 21 (2005) 2–18.
[12] Tyson Brochu, Robert Bridson, Robust topological operations for dynamic explicit surfaces, SIAM Journal of Scientific Computing 31 (4) (2009) 2472–2493.
[13] Long Chen, Mesh smoothing schemes based on optimal delaunay triangulations, in: 13th International Meshing Roundtable, Sandia National Laboratories, 2004, pp. 109–120.
[14] Long Chen, Jinchao Xu, Optimal delaunay triangulations, Journal of Computational Mathematics 22 (2) (2004) 299–308.
[15] Sung Kwon Cho, Hyejin Moon, Jesse Fowler, S.-K. Fan, Chang-Jin Kim, Splitting a liquid droplet for electrowetting-based microfluidics, in: International Mechanical Engineering Congress and Exposition, ASME Press, New York, NY, Nov. 2001, ISBN: 0791819434.
[16] Sung Kwon Cho, Hyejin Moon, Chang-Jin Kim, Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits, Journal of Microelectromechanical Systems 12 (1) (2003) 70–80.
[17] V. Cristini, J. Blawzdziewicz, M. Loewenberg, An adaptive mesh algorithm for evolving surfaces: simulation of drop breakup and coalescence, Journal of Computational Physics 168 (2001) 445.
[18] F.S. de Sousa, N. Mangiavacchi, L.G. Nonato, A. Castelo, M.F. Tomé, V.G. Ferreira, J.A. Cuminato, S. McKee, A front-tracking/front-capturing method for the simulation of 3d multi-fluid flows with free surfaces, Journal of Computational Physics 198 (2) (2004) 469–499.
[19] Michel C. Delfour, Jean-Paul Zolésio, Shapes and Geometries: Analysis, Differential Calculus, and Optimization, Advances in Design and Control, vol. 4, SIAM, 2001.
[20] Mathieu Desbrun, Mark Meyer, Peter Schröder, Alan H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, in: SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999, pp. 317–324.
[21] M.P. do Carmo, Differential Geometry of Curves and Surfaces, Prentice Hall, Upper Saddle River, New Jersey, 1976.
[22] G. Doğan, P. Morin, R.H. Nochetto, A variational shape optimization approach for image segmentation with a Mumford-Shah functional, SIAM Journal of Scientific Computing 30 (6) (2008) 3028–3049.
[23] Gunay Doğan, A Variational Shape Optimization Framework for Image Segmentation, Ph.D. Thesis, University of Maryland at College Park, 2006.
[24] Gunay Doğan, Pedro Morin, Ricardo H. Nochetto, Marco Verani, Discrete gradient flows for shape optimization and applications, Computer Methods in Applied Mechanics and Engineering 196 (2007) 3898–3914.
[25] Milton Van Dyke, An Album of Fluid Motion, Parabolic Press, 1982.
[26] Gerhard Dziuk, An algorithm for evolutionary surfaces, Numerische Mathematik 58 (1) (1990) 603–611.
[27] Jens Eggers, Universal pinching of 3d axisymmetric free-surface flow, Physical Review Letters 71 (21) (1993) 3458–3460.
[28] Jens Eggers, Singularities in droplet pinching with vanishing viscosity, SIAM Journal of Applied Mathematics 60 (2000) 1997.
[29] D. Enright, R.P. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, Journal of Computational Physics 183 (2002) 83–116.
[30] Douglas Enright, S. Marschner, Ronald Fedkiw, Animation and rendering of complex water surfaces, in: ACM Trans. Graph. (SIGGRAPH Proc.), 2002, pp. 736–744.
[31] J.M. Escobar, G. Montero, R. Montenegro, E. Rodríguez, An algebraic method for smoothing surface triangulations on a local parametric space, International Journal for Numerical Methods in Engineering 66 (4) (2006) 740–760.
[32] L.C. Evans, Partial Differential Equations, American Mathematical Society, Providence, Rhode Island, 1998.
[33] R.B. Fair, V. Srinivasan, H. Ren, P. Paik, V.K. Pamula, M.G. Pollack, Electrowetting-based on-chip sample processing for integrated microfluidics, in: IEEE International Electron Devices Meeting (IEDM), 2003.
[34] Petri Fast, L. Kondic, Michael J. Shelley, Peter Palffy-Muhoray, Pattern formation in non-newtonian Hele-Shaw flow, Physics of Fluids 13 (5) (2001) 1191–1212.
[35] James Glimm, John Grove, Brent Lindquist, Oliver A. McBryan, Gretar Tryggvason, The bifurcation of tracked scalar waves, SIAM Journal of Science and Statistical Computation 9 (1) (1988) 61–79.
[36] Raymond E. Goldstein, Adriana I. Pesci, Michael J. Shelley, Topology transitions and singularities in viscous flows, Physical Review Letters 70 (20) (1993) 3043–3046.

[37] Raymond E. Goldstein, Adriana I. Pesci, Michael J. Shelley, Instabilities and singularities in Hele-Shaw flow, Physics of Fluids 10 (11) (1998) 2701–2723.
[38] J. Gong, S.K. Fan, C.J. Kim, Portable digital microfluidics platform with active but disposable lab-on-chip, in: 17th IEEE International Conference on Micro Electro Mechanical Systems (MEMS), IEEE Press, Maastricht, The Netherlands, Jan. 2004, pp. 355–358, ISBN: 0-7803-8265-x.
[39] Roberto Grosso, Christoph Lnrig, Thomas Ertl, The multilevel finite element method for adaptive mesh optimization and visualization of volume data, in: In Proceedings Visualization, IEEE Computer Society Press, 1997, pp. 387–394.
[40] A. Hilton, A.J. Stoddart, J. Illingworth, T. Windeatt, Marching triangles: range image fusion for complex object modeling, Image Processing 1 (1996) 381–384.
[41] Michael Holst, Mclite: An adaptive multilevel finite element matlab package for scalar nonlinear elliptic equations in the plane, Technical report, University of California, San Diego, 2000.
[42] Michael Holst, Computational multiscale modeling: adaptive methods and fetk with applications in biophysics, 2008. Lecture notes (from the CTBP and NBCR Summer Workshop at UCSD).
[43] Thomas Y. Hou, John S. Lowengrub, Michael J. Shelley, Boundary integral methods for multicomponent fluids and multiphase materials, Journal of Computational Physics 169 (2001) 302–362.
[44] H. Huang, U. Ascher, Surface mesh smoothing, regularization, and feature detection, SIAM Journal of Scientific Computing 31 (1) (2008) 74–93.
[45] Wei Kang, Uzi Landman, Breakup of liquid nanobridges: molecular dynamics simulations and stochastic hydrodynamics, Physical Review Letters (Accepted) (2006).
[46] Wei Kang, Uzi Landman, Universality crossover of the pinch-off shape profiles of collapsing liquid nanobridges in vacuum and gaseous environments, Physical Review Letters 98 (6) (2007) 064504.
[47] P.M. Knupp, Algebraic mesh quality metrics, SIAM Journal of Scientific Computing 23 (2001) 193–218.
[48] Andrew P. Kuprat, Daniel R. Einstein, An anisotropic scale-invariant unstructured mesh generator suitable for volumetric imaging data, Journal of Computational Physics 228 (3) (2009) 619–640.
[49] Randall J. LeVeque, Keh-Ming Shyue, Two-dimensional front tracking based on high resolution wave propagation methods, Journal of Computational Physics 123 (2) (1996) 354–368.
[50] Xinfeng Liu, Yuanhua Li, J. Glimm, X.L. Li, A front tracking algorithm for limited mass diffusion, Journal of Computational Physics 222 (2) (2007) 644–653.
[51] William E. Lorensen, Harvey E. Cline, Marching cubes: a high resolution 3d surface construction algorithm, Computer Graphics 21 (4) (1987).
[52] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, in: ACM Trans. Graph. (SIGGRAPH Proc.), Los Angeles, 2004, pp. 457–462.
[53] Sean Mauch, A fast algorithm for computing the closest point and distance function, 2000.
[54] N. Molino, Z. Bao, R. Fedkiw, A virtual node algorithm for changing mesh topology during simulation, SIGGRAPH, ACM TOG 23 (2004) 385–392.
[55] R. Montenegro, J.M. Cascón, J.M. Escobar, E. Rodrfguez, G. Montero, An automatic strategy for adaptive tetrahedral mesh generation, Applied Numerical Mathematics 59 (9) (2009) 2203–2217. Second Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2007).
[56] R. Montenegro, G. Montero, J.M. Escobar, E. Rodríguez, Efficient strategies for adaptive 3-d mesh generation over complex orography, Neural, Parallel and Scientific Computation 10 (1) (2002) 57–76.
[57] Paul Ning, Jules Bloomenthal, An evaluation of implicit surface tilers, IEEE Computer Graphics and Applications 13 (6) (1993) 33–41.
[58] Yutaka Ohtake, Alexander Belyaev, Alexander Pasko, Dynamic meshes for accurate polygonization of implicit surfaces with sharp features, in: SMI '01: Proceedings of the International Conference on Shape Modeling and Applications, IEEE Computer Society, Washington, DC, USA, 2001, p. 74.
[59] Yutaka Ohtake, Alexander G. Belyaev, Dual/primal mesh optimization for polygonized implicit surfaces, in: SMA '02: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, ACM, New York, NY, USA, 2002, pp. 171–178.
[60] Yutaka Ohtake, Alexander G. Belyaev, Ilia A. Bogaevski, Polyhedral surface smoothing with simultaneous mesh regularization, Geometric Modeling and Processing 0 (2000) 229.
[61] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, NY, 2003.
[62] R.L. Panton, Incompressible Flow, second ed., John Wiley and Sons, Inc., New York, NY, 1996.
[63] Per-Olof Persson, Mesh size functions for implicit geometries and pde-based gradient limiting, Engineering with Computations 22 (2) (2006) 95–109.
[64] Per-Olof Persson, Gilbert Strang, A simple mesh generator in matlab, SIAM Review 46 (2) (2004) 329–345.
[65] Olivier Pironneau, Optimal Shape Design for Elliptic Systems, Springer Series in Computational Physics, Springer, New York, NY, 1984.
[66] Shaoping Quan, Jing Lou, David P. Schmidt, Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations, Journal of Computational Physics 228 (7) (2009) 2660–2675.
[67] Maria-Cecilia Rivara, Lepp-bisection algorithms, applications and mathematical properties, Applied Numerical Mathematics 59 (9) (2009) 2218–2235. Second Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2007).
[68] Alfred Schmidt, Kunibert G. Siebert, Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA, first ed., Springer, Heidelberg, Germany, 2005.
[69] Joachim Schöberl, Netgen an advancing front 2d/3d-mesh generator based on abstract rules, Computing and Visualization in Science 1 (1) (1997) 41–52.
[70] S.A. Sethian, Level Set Methods and Fast Marching Methods, second ed., Cambridge University Press, New York, NY, 1999.
[71] Jayant Shah, Scale Space and PDE Methods in Computer Vision, Lecture Notes in Computer Science, vol. 3459, Springer, 2005, pp. 339–350 (Chapter: Skeletons of 3D Shapes).
[72] Benjamin Shapiro, Hyejin Moon, Robin Garrell, Chang-Jin Kim, Equilibrium behavior of sessile drops under surface tension, applied external fields, and material variations, Journal of Applied Physics 93 (9) (2003) 5794–5811.
[73] Vadim Shapiro, Igor Tsukanov, Implicit functions with guaranteed differential properties, in: SMA '99: Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications, ACM, New York, NY, USA, 1999, pp. 258–269.
[74] Michael J. Shelley, Raymond E. Goldstein, Adriana I. Pesci, Topological transitions in Hele-Shaw flow. Singularities in fluids, plasmas and optics (Heraklion, 1992), NATO Advanced Science and Instrumentation Series C Mathematical and Physical Science 404 (1993) 167–188.
[75] S.V. Shepel, S. Paolucci, Finite element level set formulations for modelling multiphase flows, in: ICCMSE '03: Proceedings of the International Conference on Computational Methods in Sciences and Engineering, River Edge, World Scientific Publishing Co. Inc., NJ, USA, 2003, pp. 593–597.
[76] Jonathan Richard Shewchuk, Triangle: engineering a 2D quality mesh generator and delaunay triangulator, in: Ming C. Lin, Dinesh Manocha (Eds.), Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science, vol. 1148, Springer-Verlag, 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.
[77] Kaleem Siddiqi, Stephen M. Pizer, Medial Representations, Computational Imaging, vol. 37, Springer, 2008.
[78] J. Sokolowski, J.-P. Zolésio, Introduction to Shape Optimization. Springer Series in Computational Mathematics, Springer-Verlag, 1992.
[79] Y. Sun, C. Beckermann, Diffuse interface modeling of two-phase flows based on averaging: mass and momentum equations, Physica D 198 (2004) 281.
[80] M. Sussman, E.G. Puckett, A coupled level set and volume of fluid method for computing 3-d and axisymmetric incompressible two-phase flows, Journal of Computational Physics 162 (2000) 301.
[81] Mark Sussman, A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles, Journal of Computational Physics 187 (1) (2003) 110–136.
[82] Anna-Karin Tornberg, Michael J. Shelley, Simulating the dynamics and interactions of flexible fibers in stokes flows, Journal of Computational Physics 196 (2004) 8–40.
[83] Clifford A. Truesdell, A First Course in Rational Continuum Mechanics. Pure and Applied Mathematics, A Series of Monographs and Textbooks, Academic Press, 1976.

[84] Alper Üngör, Tiling 3d euclidean space with acute tetrahedra, in: Canadian Conference on Computational Geometry, Aug. 2001.
[85] Salih Ozen Unverdi, Grétar Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics 100 (1) (1992) 25–37.
[86] S.P. van der Pijl, A. Segal, C. Vuik, P. Wesseling, A mass-conserving level-set method for modelling of multi-phase flows, International Journal for Numerical Methods in Fluids 47 (2005) 339361.
[87] S.P. van der Pijl, A. Segal, C. Vuik, P. Wesseling, Computing three-dimensional two-phase flows with a mass-conserving level set method, Computing and Visualization in Science 11 (2008) 221–235.
[88] Shawn W. Walker, A Hybrid Variational-level Set Approach to Handle Topological Changes, Master's Thesis, University of Maryland at College Park, May 2007.
[89] Shawn W. Walker, Andrea Bonito, Ricardo H. Nochetto, Mixed finite element method for electrowetting on dielectric with contact line pinning, Interfaces and Free Boundaries 12 (1) (2010) 85–119.
[90] Shawn W. Walker, Benjamin Shapiro, A control method for steering individual particles inside liquid droplets actuated by electrowetting, Lab on a Chip 5 (2005) 1404–1407.
[91] Shawn W. Walker, Benjamin Shapiro, Modeling the fluid dynamics of electrowetting on dielectric (ewod), Journal of Microelectromechanical Systems 15 (4) (2006) 986–1000.
[92] Shawn W. Walker, Benjamin Shapiro, Ricardo H. Nochetto, Electrowetting with contact line pinning: computational modeling and comparisons with experiments, Physics of Fluids 21 (10) (2009) 102103.
[93] J.A.S. Witteveen, B. Koren, P.G. Bakker, An improved front tracking method for the euler equations, Journal of Computational Physics 224 (2) (2007) 712–728.
[94] Zoë J. Wood, Peter Schröder, David Breen, Mathieu Desbrun, Semi-regular mesh extraction from volumes, in: VIS '00: Proceedings of the Conference on Visualization '00, IEEE Computer Society Press, Los Alamitos, CA, USA, 2000, pp. 275–282.
[95] Jinchao Xu, Wavelets, Multilevel Methods and Elliptic PDEs. An Introduction to Multilevel Methods, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 1997. pp. 213–302 (Chapter 5).
[96] Yingying Yao, Chang Seop Koh, Dexin Xie, Robust mesh regeneration based on structural deformation analysis for 3d shape optimization of electromagnetic devices, in: ICEMS 2003, Sixth International Conference on Electrical Machines and Systems, vol. 2, 2003, pp. 732–735.
[97] P. Yue, J.J. Feng, C. Liu, J. Shen, A diffuse-interface method for simulating two-phase flows of complex fluids, Journal of Fluid Mechanics 515 (2004) 293.
[98] E.C. Zachmanoglou, Dale W. Thoe, Introduction to Partial Differential Equations with Applications (Paperback), Dover Publications, 1987.
[99] Ou-Yang Zhong-can, Elastic theory of biomembranes, Thin Solid Films 393 (2001) 19–23.