ABSTRACT

Title of thesis: A HYBRID VARIATIONAL-LEVEL SET APPROACH TO HANDLE TOPOLOGICAL CHANGES

Shawn W. Walker, Master of Science, 2007

Thesis directed by: Professor Ricardo H. Nochetto
Program of Applied Mathematics and
Scientific Computation

We present a method for allowing explicit, Lagrangian meshes to undergo topological changes in an *automatic* way. We employ a method for detecting when topological changes are imminent. When a change occurs, we use a level set method to guide the change of topology of the domain mesh. This is then followed by an optimization step, using a variational formulation, that seeks to improve boundary mesh conformity to the zero level contour of the level set function. The advantage of this method is that it 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 level set update and optimization step are only needed during a topological change. Hence, our method does not significantly affect computational cost.

A HYBRID VARIATIONAL-LEVEL SET APPROACH TO HANDLE TOPOLOGICAL CHANGES

by

Shawn W. Walker

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2007

ACKNOWLEDGMENTS

I would like to thank my advisor, Ricardo Nochetto for his patience in reading and editing this document, as well as many fruitful discussions leading to its completion. I also thank the other committee members, Benjamin Shapiro and Tobias von Petersdorff for their involvement. I thank Andrea Bonito for his input and advice on the method described in this thesis. And I thank Marilyn Perry for allowing me to work in her closet, so I could concentrate on finishing the thesis.

The idea of this thesis had been around for awhile, but the bulk of the work in demonstrating the method and writing the thesis was done within the span of one month. Again, I thank the above mentioned for their willingness to take part in this endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This thesis is concerned with introducing a hybrid level set/variational method for enabling simulations (with Lagrangian grids) of fluid interfaces or free boundary problems involving topological changes (i.e. pinching or reconnection). In the following sections, we provide some background and motivation for doing this.

1.1 Free Boundary Problems with Large Deformations

Free boundary problems arise in many areas of mathematics and engineering. Understanding free surface dynamics is important for applications such as coating flows [5], simulating water wave dynamics for computer graphics [21], and surface tension/curvature driven flows in micro-fluidic devices such as Hele-Shaw flow [11], [25]. Other examples involve fluid-structure interactions, such as polymer filaments in an active flow field [43], interaction of a lipid biomembrane with a surrounding fluid [48], and animal locomotion in a fluid medium [1], [39]. Success in understanding and simulating free boundary problems would allow design of micro-scale devices, better understanding of cell-membrane dynamics, and more accurate simulations of industrial processes.

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 [4], droplet pinching in an electro-wetting device [10], and many other types of fluid flow [16].

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 'usual' 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 the behavior of the Navier-Stokes equations has been done for axi-symmetric fluid pinching [18], [19]. But one can certainly argue that a continuum model is not adequate and a model which includes atomistic behavior is more correct. Although recently in [29] and [30] 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 thesis, we develop a method for piecing together a simulation when a topological change happens. Specifically, we are concerned with simulations that use an explicit unstructured grid (e.g. a triangulation) for representing the deforming domain. This is the case when the continuum model is implemented through a finite element method. Our method is able to take a mesh that is deforming (in a Lagrangian frame), go through a topological change, and continue deforming automatically without user intervention. Furthermore, our method extends to three dimensional meshes as long as a high quality local re-meshing and mesh refinement tool is available.

Before describing the details of this method, we first give a literature survey of other methods for simulating deforming domain problems and how they go through topological changes.

1.2 Overview of Simulation Methods

At the numerical analysis and computational level, there are many issues concerned with creating robust and stable numerical methods that capture the physical model, while allowing for large domain deformations. The methods for handling free boundary problems are based, in part, on the standard numerical schemes for solving PDEs, which are Finite Difference (FDM), Finite Element (FEM), and Boundary Integral Methods (BIM). However, problems that depend on the domain geometry, such as curvature driven flows, require special enhancements to the standard techniques.

One popular method for capturing free surface motion is the level set method

[36], [37], which advects a scalar field function whose zero level set represents the interface. The numerical implementation can be done using either FDM or FEM. Level set methods have the advantage of being completely Eulerian and can automatically handle topological changes, though the physics underlying such changes is often left ill-understood. 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 [20] or refinement [32]. Another drawback 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 methods include the phase field method [46], [42], 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 (i.e. an interface mesh) and discretize the PDE using either FDM, FEM, or BIM. FEM and BIM are generally considered to be better conditioned and more robust than FDM, because they use an integral formulation. And there exist finite element and boundary integral methods that take advantage of the intrinsic representation of the interface [2], [17], [28]. However, one disadvantage to these explicit surface representations is the computational difficulty in handling large deformations of the mesh. In two dimensions, it is fairly straightforward to adjust the mesh through local re-meshing [41] or mesh smoothing [22]. But in three dimensions, it is not clear what the best methods are for adjusting a mesh as it deforms. Some methods [45] use elasticity models to modify the mesh, or take an optimization point of view [35], while others [8], [9] use a variational form to minimize the interpolation error to do local re-meshing. Also, some of this difficulty in 3-D is alleviated with BIM which requires no bulk interior mesh.

Currently, there are few methods for taking explicit meshes through topological changes. Some existing methods use 'surgery' [13], [14] to cut the mesh. This is a viable option for pinching in 2-D problems, but the nature of topological changes in 3-D is much more complicated. 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 level set and explicit mesh methods, 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 [3], which forms an explicit representation 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. Another method [7], not concerned with topological changes, seeks to create a 3-D tetrahedral mesh of a domain described by an implicit surface that is defined by the zero level set of a scalar function. The method we develop in this thesis is a hybrid method and combines some of the methods listed here. A detailed description follows in Section 2.

1.3 Outline

The outline of the thesis is the following. In Chapter 2, we describe in detail our hybrid method. Local mesh adjustments such as mesh smoothing and re-meshing are discussed in Section 2.2. Section 2.3 describes our method of mesh and velocity extension. Our time-stepping adaptation method is given in Section 2.4, followed by our method of detecting topological changes in Section 2.5. The actual topological change execution is discussed in section 2.6, with the level set method described in Section 2.6.1 and our optimization method for adjusting the boundary given in Section 2.6.2.

In Chapter 3, we show numerical experiments to demonstrate our method for taking triangular finite element meshes through topological changes. The first example uses a given rotating velocity field and is discussed in Section 3.1. Our second example involves a MEMS (Micro-Electro-Mechanical System) device that uses Electro-Wetting On Dielectric (EWOD) to move water droplets by augmenting surface tension effects. This example exhibits *multiple pinching* and is described in Section 3.2. The last example involves surface tension driven flow in a Hele-Shaw cell with water droplets joining and is given in Section 3.3.

We conclude in Chapter 4 by discussing some of the limitations of our method along with a comparison of the relevant literature.

Chapter 2

Algorithm Description

2.1 Overview

The algorithm mainly consists of using a level set update to indicate how the mesh topology changes when a topological change is imminent. This is followed by an optimization step that reconstructs the mesh around the region of topological change by again using the level set function. The main point of the algorithm is to provide a way for allowing meshes to go through topological changes *without having to make complicated decisions or 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 give an outline of the algorithm in the following list. Details of each item are then given in subsections that follow. The first four items are always performed (or checked) at every step of the simulation, while items 5 and 6 are only performed to execute a topological change. Also, see the flowchart given in Figure 2.1 for a high level summary.

- Mesh smoothing and re-meshing. The techniques we use for fixing distorted

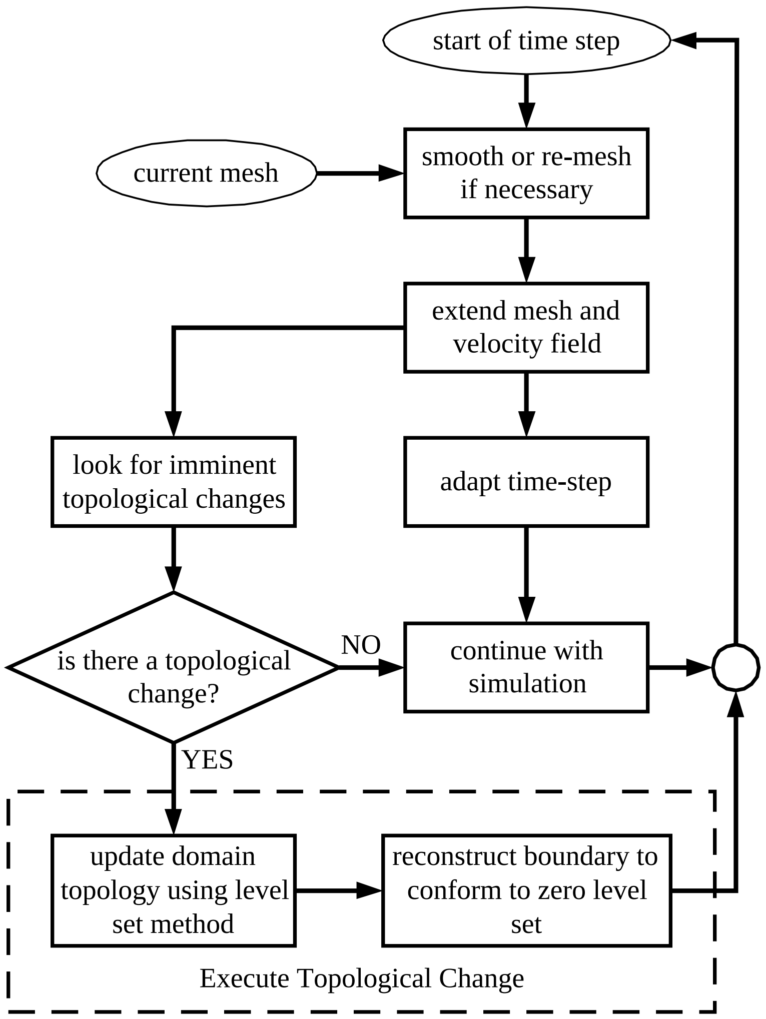

start of time step
current mesh
smooth or re-mesh if necessary
extend mesh and velocity field
look for imminent topological changes
adapt time-step
is there a topological change?
NO
continue with simulation
YES
update domain topology using level set method
reconstruct boundary to conform to zero level set
Execute Topological Change

Figure 2.1: High level block diagram of algorithm. At the beginning of each time-step, the mesh quality is computed and smoothed or re-meshed if necessary. Next, the domain mesh is extended (using Triangle) to a box that contains the domain, and the velocity field is extended everywhere. Then, the time-step is adapted to avoid inverting triangles in the mesh. In addition, a check is made to determine if there are any topological changes imminent. If there are no topological changes, then the simulation continues as normal. If there is a topological change, then the level set method is used to obtain the new topology. This is followed by a boundary mesh reconstruction step to better approximate the new domain shape given by the level set method.

elements and remeshing are standard and are briefly discussed in Section 2.2.

- Mesh and velocity extension. As the physical simulation of a moving domain progresses, the domain mesh and velocity must be extended for later use (see Section 2.3).

- Time-Step adaptation. The size of the time-steps τ during a simulation are controlled by the desired accuracy, the amount of shear in the velocity field, and the time-scale of topological changes. The details are discussed in Section 2.4.

- Detection of imminent topological changes. This requires the user to define a tolerance, d_{neck}, for how close 'disjoint' parts of the boundary have to be before considering a topological change. In other words, d_{neck} refers to the minimum thickness of necking regions in the extended domain $E\Omega$ (defined in Section 2.3). This is made more precise in Section 2.5.

- Updating the level set function and mesh topology. Here we use a straightforward discretization of the level set equation (with local diffusion) and a simple criteria for updating the mesh topology (see Section 2.6.1).

- Reconstructing the region containing the topological change using a minimization approach. This step is meant to correct for errors in the position of the new surface obtained in the previous step, and is described in Section 2.6.2.

See Appendix A for a list of symbol definitions.

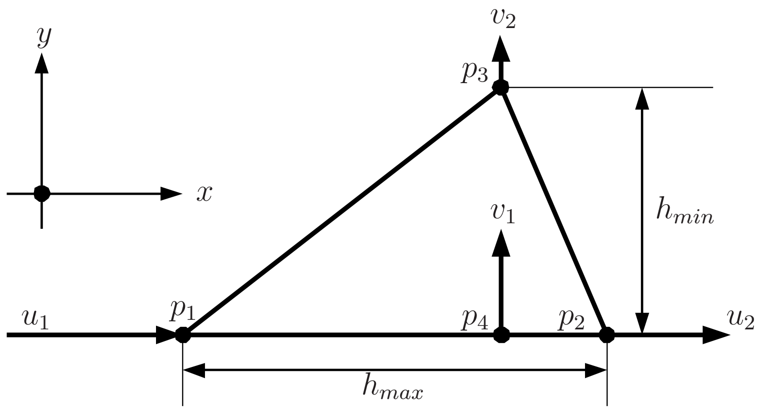


Figure 2.2: A mesh triangle (assumed to be shape regular) undergoing deformation. The velocity field over the triangle is labeled (u, v) and is linear over the triangle. The values of the x component of velocity are labeled u_1 and u_2 at the points p_1 and p_2, respectively (with $u_1 > u_2$). As p_1 and p_2 move in the x direction, their relative distance decreases. The rate of decrease depends on $u_1 - u_2$ or actually $\frac{\partial u}{\partial x}$. This gives an estimate of the largest time step τ that can be taken before p_1 and p_2 cross-over, which is $\tau < 1/\frac{\partial u}{\partial x}$. Any larger time-step will cause the triangle to be inverted.

2.2 Mesh Smoothing and Re-meshing

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 show a very basic estimate that relates the maximal time-step of a mesh update (while preventing mesh distortion) to the gradient of the velocity.

A diagram of a single triangle in some triangulation is given in Figure 2.2 and is assumed to be shape regular (i.e. no skinny triangles). The 2-D velocity field over the triangle is denoted by $\vec{u} = (u, v)$. Velocity components in the x direction, at the points p_1 and p_2, are denoted by u_1 and u_2, respectively. The points are moving with those velocities. In addition, we assume that $u_1 > u_2$ and the velocity field is assumed to be linear over the triangle. 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 $h_{max} - \tau(u_1 - u_2)$, where τ is the time-step of the mesh update. Hence, if the relative distance becomes zero, then τ is given by

$$\frac{1}{\tau} = \frac{u_1 - u_2}{h_{max}} = \frac{\partial u}{\partial x},$$

where the second equality is because u is assumed linear. A similar relation holds when looking for the time to cross-over of the points p_3 and p_4:

$$\frac{1}{\tau} = \frac{\partial v}{\partial y}.$$

It should then be clear that a conservative estimate (that does not depend on the size of the triangle) for the maximal time-step which will not cause the triangle to invert is

$$\tau < \frac{1}{|\nabla \vec{u}|}.$$

Of course, the triangle may be very distorted after updating and will not be shape regular. Hence, this argument cannot be used for estimating the maximal time-step unless periodic mesh smoothing or re-meshing is used.

There are various techniques for improving or creating a 2-D triangulation. In this paper, we make extensive use of the freely available program "Triangle" by Shewchuk [41]. Triangle is able to produce well shaped 2-D meshes of entire domains quite easily and can also be used for local re-meshing. Whenever the quality metric [31] of the mesh triangulation deteriorates, we use Triangle to re-mesh the domain. In addition, we use an optimization method for "smoothing meshes" (i.e. for reducing the distortion of triangles in the mesh) that does not require re-meshing the entire domain. This method [22] moves the vertices of the mesh in an attempt to optimize the local quality metric of the triangulation [31]. One advantage of this optimization method is that it is guaranteed not to invert elements. By smoothing, we are able to delay having to re-mesh the domain.

In addition, to prevent excessive re-meshing or mesh manipulation, we update the mesh with a velocity field $\vec{u}_{smooth}$ that has minimal shear (i.e. with $|\nabla\vec{u}_{smooth}|$ minimal). This is done by solving a vector Laplace equation with Dirichlet boundary condition given by the vector velocity $\vec{u}$ at the boundary

$$\begin{aligned} -\triangle\vec{u}_{smooth} &= 0,\ \Omega, \\ \vec{u}_{smooth} &= \vec{u},\ \Gamma, \end{aligned} \tag{2.1}$$

where Ω is the domain and $\Gamma := \partial\Omega$. Solving the Laplacian guarantees that $|\nabla\vec{u}_{smooth}|$ will be minimized in the L^2 sense [23]. This ensures the boundary will move with the correct velocity (coming from the physics) and the interior triangles will be subjected to minimal distortion. 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. Combining this with local re-meshing and mesh smoothing gives us a robust way to maintain mesh quality.

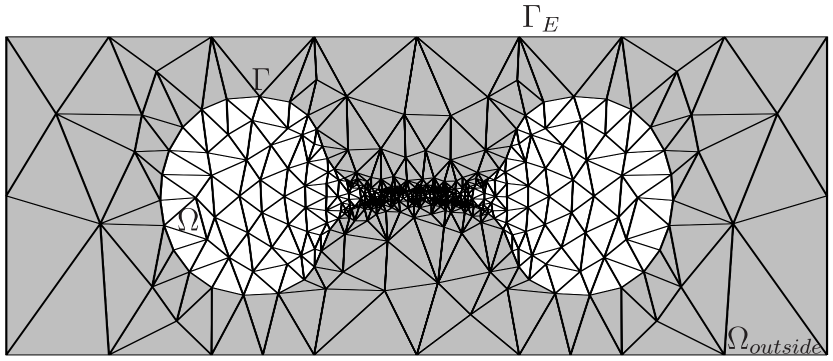


Figure 2.3: A domain mesh with its extension to an enclosing box. The interior mesh is Ω, the shaded region is $\Omega_{outside}$, and $E\Omega = \Omega \bigcup \Omega_{outside}$. The domain boundary is denoted by Γ and the boundary of $E\Omega$ is Γ_E. In Section 2.6.1, the signed distance function to Γ is computed on $E\Omega$, and is positive over Ω and negative over $\Omega_{outside}$ (i.e. the shaded region). The zero level set of the distance function corresponds exactly to the set Γ.

2.3 Mesh and Velocity Extension

At each step of the simulation, the domain mesh must be extended beyond the domain boundary for the following reasons: checking for imminent topological changes and for solving the level set equation if there is a change (both are discussed in subsequent sections). In this thesis, the mesh is extended to a rectangular box that contains the domain mesh, though in principle, the mesh only needs to be extended a distance of d_{neck} to allow for detection of close boundaries. For clarity (see Figure 2.3), let Ω refer to the domain which contains a set of triangles $\top_\Omega$ that define a triangulation (i.e. the domain mesh) and let $\Gamma := \partial\Omega$ (i.e. the boundary of Ω). Denote the extended domain (i.e. the whole box) by $E\Omega$, with boundary

$\Gamma_E := \partial E\Omega$, which contains a larger set of triangles $\top_{E\Omega}$ with edges that conform to Γ. And denote the exterior or 'outside' domain by $\Omega_{outside} := E\Omega \setminus \Omega$ whose boundary, $\Gamma_{outside} := \Gamma \bigcup \Gamma_E$. The triangle mesh $\top_{E\Omega}$ for $E\Omega$ is produced using the program Triangle.

Lastly, given a velocity field (coming from the physics being simulated) defined on Γ, we need an extension of the velocity, denoted $\vec{u}_{smooth} = (u_{smooth}, v_{smooth})$, to Ω and to $\Omega_{outside}$ (i.e. to all of $E\Omega$). This is done by first solving a vector Laplace equation (see equation (2.1)). Then we solve another vector Laplace equation over $\Omega_{outside}$ with the same Dirichlet data on Γ and zero Neumann data on Γ_E for each velocity component

$$-\triangle \vec{u}_{smooth} = 0, \ \Omega_{outside}, \tag{2.2}$$

$$\vec{u}_{smooth} = \vec{u}, \ \Gamma, \tag{2.3}$$

$$\frac{\partial \vec{u}_{smooth}}{\partial \nu} = 0, \ \Gamma_E.$$

This velocity field will be used in detecting a topological change and for solving the level set equation.

2.4 Time-Stepping

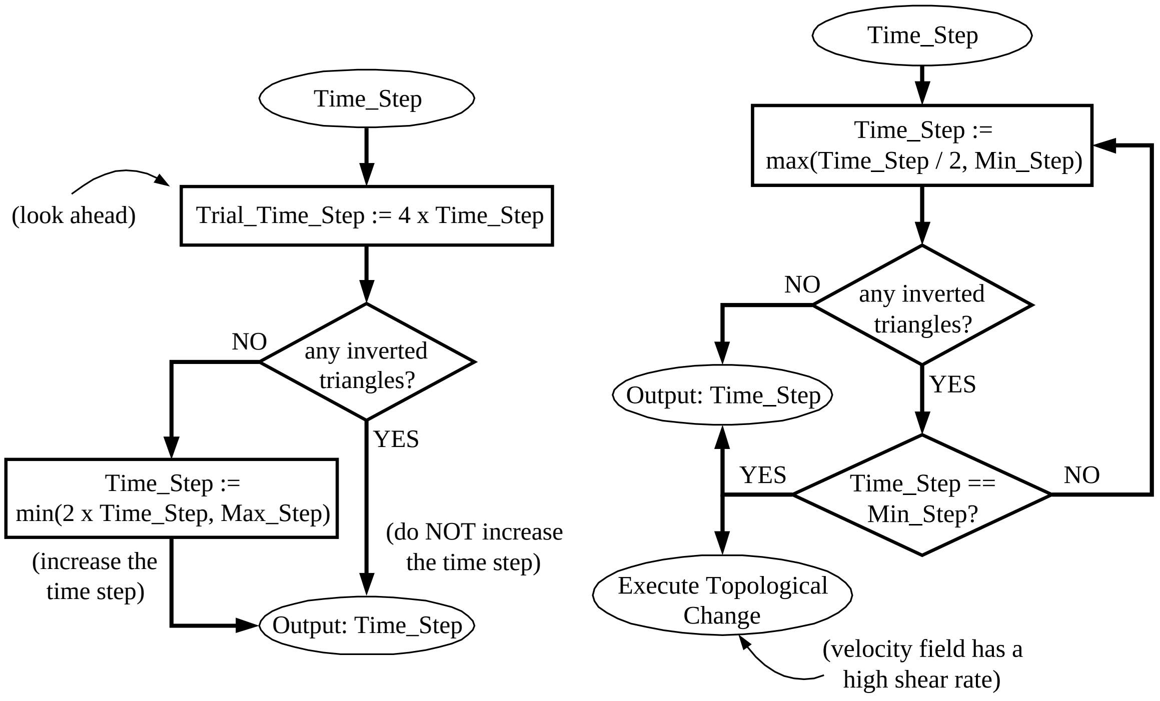
We adopt a fairly simple method for adapting the time-step. First, the maximum time-step τ_{max} is set by the desired accuracy. The minimum time-step τ_{min} is connected with the time-scale of the fastest dynamics of the physical situation being simulated (i.e. τ_{min} must be chosen small enough to allow the simulation to resolve large gradients in the velocity field without causing mesh inversion). The algorithm proceeds by finding the largest time-step $\tau \in [\tau_{min}, \tau_{max}]$ such that the mesh of the extended domain ($E\Omega$) can be updated (using $\vec{u}_{smooth}$ from equation (2.2)) without inverting any triangles. The details are summarized by the flowcharts in Figure 2.4.

2.5 Topological Change Detection

The detection of when to execute a topological change is slightly complicated. Essentially, one must look for regions where 'different' parts of the boundary are *close and collapsing together.* Detecting 'closeness' is a common problem in collision detection, where the closest point transform or fast marching methods are used [6], [33]. However, the geometry of the domain is already captured by the triangulation and can be used to look for these 'thin' regions (see Figure 2.5). But 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 (see Chapter 4 for some discussion). Therefore, to avoid this difficulty, *if regions of the domain are 'thin', then they are assumed to be undergoing a topological change.* The rest of this section describes how we find these 'thin' regions.

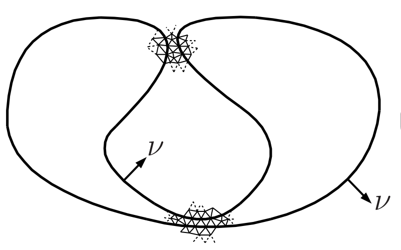
2.5.1 Check For Thinness

We check for 'thinness' in the following way. First, one defines how thin a piece of the domain must be, d_{neck}, in order to be considered a topological change.

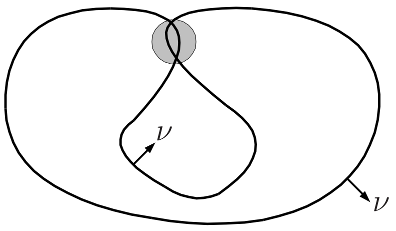

Time_Step
(look ahead)
Trial_Time_Step := 4 x Time_Step
NO
any inverted triangles?
YES
Time_Step := min(2 x Time_Step, Max_Step)
(increase the time step)
(do NOT increase the time step)
Output: Time_Step
Time_Step
Time_Step := max(Time_Step / 2, Min_Step)
NO
any inverted triangles?
YES
Output: Time_Step
YES
Time_Step == Min_Step?
NO
Execute Topological Change
(velocity field has a high shear rate)

(a) Increasing the time-step. (b) Decreasing the time-step.

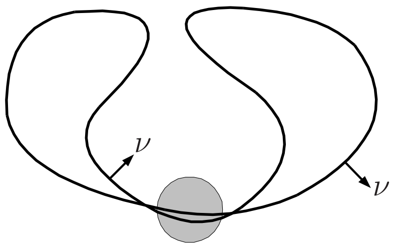
Figure 2.4: Flowchart for determining the time-step. During the simulation a check is made to determine if updating the mesh vertices with the previous time-step, and the velocity $\vec{u}_{smooth}$, will cause any triangles to be inverted. If not, the algorithm executes the 'Increase Time-Step' routine, which only allows the time-step to increase incrementally from one simulation step to the next. Otherwise, the 'Decrease Time-Step' routine is called, which decreases the time-step until no triangles are inverted. If the mesh still gets inverted with the minimum time-step, then there are extremely high gradients in the velocity field. This can be due to an imminent topological change or a rather extreme velocity field. In both cases, our algorithm proceeds with executing a topological change (see Section 2.6). In this case, the set of inverted triangles $\top_{pinch}$ (defined in Section 2.5.1) is determined using the velocity field $\vec{u}_{smooth}$ and the minimum time-step τ_{min}. Note that if the physics produces velocity fields with high gradients and are <u>not</u> associated with a topological change, then it is necessary to choose τ_{min} small enough to allow the simulation to resolve these situations.



(a) Domain with extended mesh shown locally at the two thin regions.



(b) 'Inflated' mesh.



(c) 'Deflated' mesh.

Figure 2.5: Domain with boundaries that are close. The outward pointing normal vector is ν and is shown in all three figures. In Figure a, the domain boundary is shown by a thick black line with its extended mesh shown in two local regions only. In Figure b, the mesh has been 'inflated' by moving all boundary vertices in the normal direction (i.e. along ν). All interior vertices are moved by using an extension of the normal vector field on the boundary through a vector Laplace solve. This causes the triangles in the top 'thin' region to become inverted. In Figure c, the mesh has been shrunk or 'deflated' by moving along the negative normal direction ($-\nu$). This causes the triangles in the lower 'thin' region to become inverted. Therefore, a convenient way to check for boundaries that are close is to inflate and deflate the mesh and look for triangles that are inverted.

Next, we define a vector field $\vec{u}_\nu$ on the boundary of the mesh that is given by the outward pointing normal vector, ν (see Figure 2.5). For a polygonal boundary, this is done by computing an average normal at each vertex. Next, this vector field $\vec{u}_\nu$ is extended to $E\Omega$ by solving the vector Laplace equation (2.2). Then, the mesh is inflated and deflated (see Figure 2.5) by using the following updates:

$$\vec{x}_{inflate} := \vec{x}_E + \frac{d_{neck}}{2}\vec{u}_\nu \tag{2.4}$$

$$\vec{x}_{deflate} := \vec{x}_E - \frac{d_{neck}}{2}\vec{u}_\nu \tag{2.5}$$

where $\vec{x}_E$ is the position vector of the mesh vertices in $E\Omega$. Finally, we check for inverted triangles in both of the meshes defined by $\vec{x}_{inflate}$ and $\vec{x}_{deflate}$. Let $\top_{pinch}$ be the set of triangles in $E\Omega$ that become inverted due to inflation and deflation ($\top_{pinch}$ indicates the thin regions of the domain). See the flowchart in Figure 2.6 for a summary of these steps.

If there is a thin region, then $|\nabla\vec{u}_\nu|$ (in the thin region) can be estimated by $2/d$, where d is the thickness of the neck (see Figure 2.7). By the discussion in the previous section, the step size needed to invert a triangle in the thin region is given by $\tau = d/2$. Therefore, if $d < d_{neck}$ then some triangles in the necking regions will become inverted after inflating and deflating the mesh. The location of the inverted triangles then indicate regions where a topological change is possible. If there are no thin regions nor places of high curvature, then $|\nabla\vec{u}_\nu|$ will be much smaller than $2/d$, hence the inflation and deflation steps will not invert any triangles in $E\Omega$. Figure 2.8 depicts why high curvature is bad. If the physical problem being simulated exhibits regions of high curvature, then it is necessary to choose d_{neck} sufficiently small to

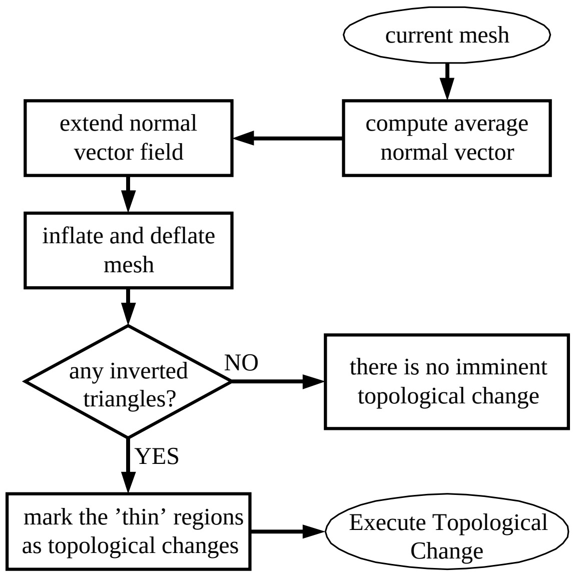

current mesh
compute average normal vector
extend normal vector field
inflate and deflate mesh
any inverted triangles?
NO
there is no imminent topological change
YES
mark the 'thin' regions as topological changes
Execute Topological Change

Figure 2.6: Flowchart for procedure to detect topological change. If a wait period is in effect because of a recent topological change within the last W_τ time-steps (defined in Section 2.5.3), then this routine does not execute (i.e. all 'thin' regions are ignored). If not, then given the current mesh, the average normal vector at each boundary vertex is computed and used to define a vector field $\vec{u}_\nu$ on the boundary. The vector field is extended to the extended mesh $E\Omega$ by a vector Laplace solve and used to inflate and deflate the extended mesh (see Figure 2.5). Any triangles that become inverted from the inflation or deflation step are marked as regions of topological change (i.e. let $\top_{pinch}$ be the set of triangles that become inverted). If $\top_{pinch}$ is empty, then no topological change is imminent and this routine terminates). If $\top_{pinch}$ is non-empty, a covering of the 'pinched' triangles is produced, and this routine signals to execute a topological change.

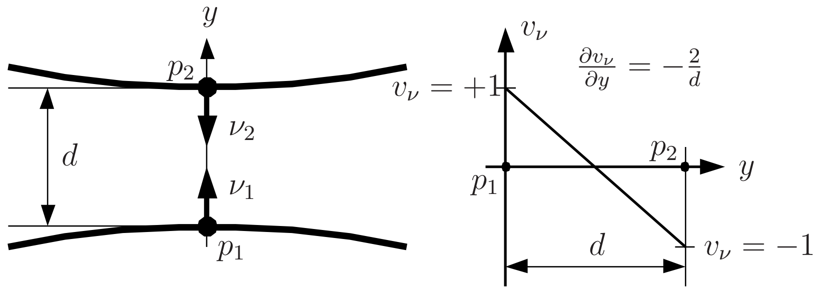


Figure 2.7: A zoom in of a thin region. The normal vectors ν_1 and ν_2 depicted here point along the y axis. Hence, the normal vector field $\vec{u}_\nu = (u_\nu, v_\nu) = (0, +1)$ at p_1 and $\vec{u}_\nu = (0, -1)$ at p_2. This implies $\nabla u_\nu \approx 0$ and $\frac{\partial v_\nu}{\partial x} \approx 0$ in the thin region. And $\frac{\partial v_\nu}{\partial y} \approx -\frac{2}{d}$ in the thin region, so $|\nabla \vec{u}_\nu| \approx \frac{2}{d}$ also.

avoid mistaking these regions for topological changes.

2.5.2 Define Covering Of Pinching Regions

If the set $\top_{pinch}$ is empty, then the algorithm stops for the current step of the simulation (meaning there is no topological change). If it is non-empty, then we define a covering of the region of topological change in the following way. For each $T \in \top_{pinch}$, we define a disk D_T of radius d_{neck} and center coordinate given by the barycenter of T. Let C_{pinch} be the collection of disks and let R_{pinch} be the region covered by the union of the disks (see Figure 2.9). This will be used later in updating the level set function and mesh topology.

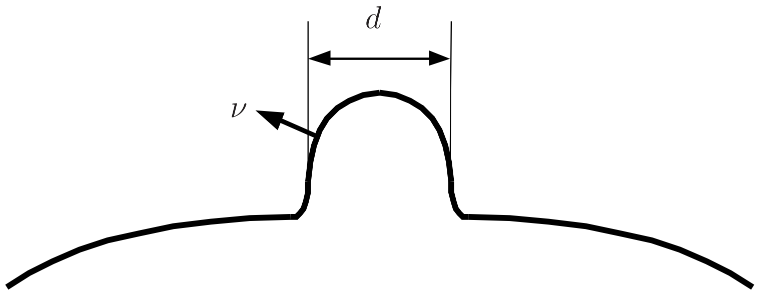


Figure 2.8: A hypothetical domain with a bump with large curvature. If d is less than d_{neck}, then the bump will be detected as a 'thin' region because $|\nabla \vec{u}_\nu|$ will be large there. To avoid mistaking the bump for a topological change, d_{neck} must be chosen sufficiently small in order to allow the simulation to resolve the bump region.

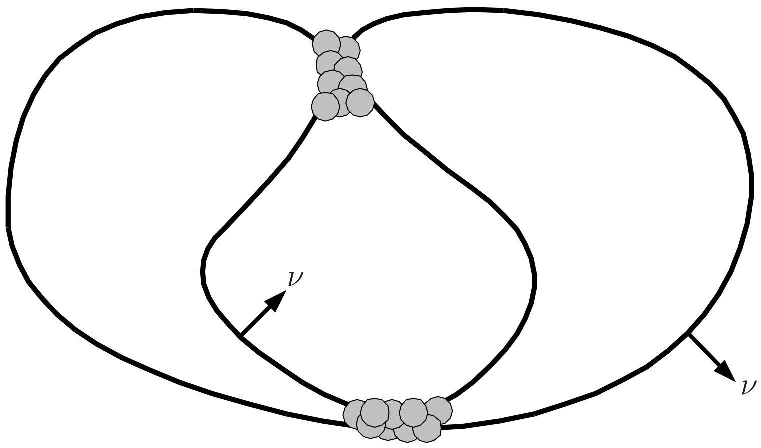


Figure 2.9: Domain with thin regions and local covering. The collection of disks C_{pinch} are shown as shaded circles here. The set of points that the disks cover is denoted R_{pinch}. This covering region is used in updating the topology of the mesh in Section 2.6.1.

2.5.3 Wait Period For Topological Changes

Lastly, we define a certain '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 would trigger another change immediately after because of the high curvature region. In fact, this may cause a sequence of topological changes until the 'spike' is completely eaten away! This is undesirable in some cases, because the natural dynamics may resolve the 'spike' naturally without any extra topological changes occurring. Therefore, we define a wait period to prevent a spurious sequence of pinches by W_τ, which is a whole number of time-steps.

This works in the following way. If a topological change is executed, then for the next W_τ time-steps, the 'thinness' criteria is not evaluated in Section 2.5.1. If there are other regions in the domain that are close to topological change, then those will not be executed until the wait period is done. An exception to this is allowed in the time-adaptation scheme in Section 2.4. If the velocity field is very abrupt and wants to pinch, then the time-stepping scheme will allow for this (see Figure 2.4). This can happen if the velocity field has a compressive shock in a thin region.

2.6 Execute Topological Change

This part of the algorithm takes the given triangular mesh through a topological change. A level set method is used to indicate the topology and domain shape after the change. The details follow in subsequent sections.

2.6.1 Update Level Set and Topology

Now that the location of the topological change is known, it is possible to use the level set method to indicate how the domain topology changes. The main steps here are

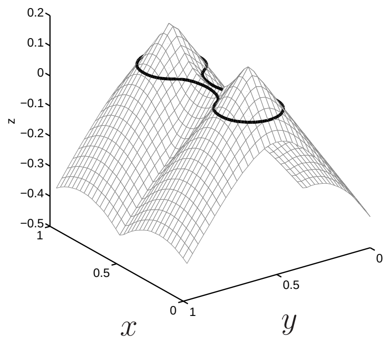
- Compute the signed distance function.
- Convect the distance function with the level set equation.
- Reconstruct the new domain from the updated level set function.

In the next section, we derive the level set equation using characteristics and then pose it in weak form. Following this, we state the algorithm for getting the new domain topology after a pinch.

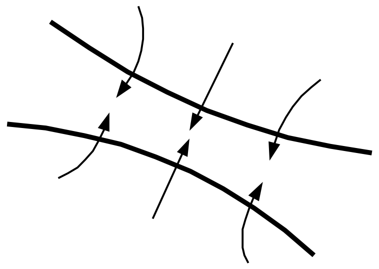
Derivation of Level Set Equation

We first derive the level set equation by tracking characteristics. Consider two curves that are colliding together from a given velocity field $\vec{u}$ (see Figure 2.10). The curves are part of a global boundary and are represented by the zero level set of some scalar function $\psi(\vec{x})$, where $\vec{x}$ is the coordinate position in the plane. The motion of the two curves can be captured by using the *method of characteristics* to obtain a new scalar function whose zero level set corresponds to the new position of the curves. More precisely, let $\vec{y}(t)$ be the position of a point in the plane at time t, and suppose the point moves with the velocity field $\vec{u}$, i.e.

$$\frac{d}{dt}\vec{y}(t) = \vec{u}(\vec{y}(t), t).$$



(a) Scalar level set function with zero level contour in thick black.



(b) Zoom-in of pinching region with characteristics.

Figure 2.10: Level set function with characteristic curves indicating motion. In Figure (a), the level set function is a scalar function defined over a 2-D domain. The zero level set is shown as a thick black line and depicts a domain approaching a topological change (i.e. a pinch). In Figure (b), a zoom-in is shown of the pinching region with curvy arrows indicating the flow field.

Next, suppose there exists a scalar function $\phi(\vec{x}, t)$ that is defined for all $\vec{x} \in E\Omega$ and $t \in \mathbb{R}$, with the property that $\phi(\vec{x}, 0) = \psi(\vec{x})$ and $\phi(\vec{y}(t), t) = 0$ for all t. We want to know the differential equation that $\phi(\vec{y}(t), t)$ solves. This is given by taking the total derivative with respect to time

$$0 = \nabla\phi(\vec{y}(t), t) \cdot \frac{d}{dt}\vec{y}(t) + \partial_t\phi(\vec{y}(t), t) = \nabla\phi(\vec{y}(t), t) \cdot \vec{u}(\vec{y}(t), t) + \partial_t\phi(\vec{y}(t), t).$$

Since the point $\vec{y}$ is arbitrary, the above equation can be re-written as

$$\partial_t\phi(\vec{x}, t) + \vec{u}(\vec{x}) \cdot \nabla\phi(\vec{x}, t) = 0, \text{ for all } \vec{x} \in E\Omega \text{ and all } t \geq 0, \tag{2.6}$$

which is the level set equation. By finding solutions to (2.6), we are able to track the positions of the curves as functions of time in an implicit way.

Insert Local Diffusion

Equation (2.6) is linear and well-posed as long as the velocity $\vec{u}$ is smooth [47]. In order to have two boundaries (i.e. curves defined by the zero level set of ϕ) touch, it is necessary to have a velocity field that is not Lipschitz [3] (This follows from standard uniqueness theorems for ODE's). But in this case, the solvability of (2.6) is questionable, especially in the case of a topological change. To address this, we simply add a small diffusion term to the equations that is active only locally on R_{pinch} (i.e. the region where the topological change is happening),

$$\partial_t\phi + \vec{u} \cdot \nabla\phi = \nabla \cdot (\varepsilon(\vec{x})\nabla\phi). \tag{2.7}$$

This guarantees that the equation is well-posed since the only possible regions for a shock are in the covering region R_{pinch}. In effect, we obtain the 'viscosity' solution [23] of (2.6) which allows for pinching and reconnection of boundaries.

The local diffusion parameter $\varepsilon(\vec{x})$ is defined in the following way. Let θ be a function of one variable defined by

$$\theta(\xi) = \begin{cases} 1 - \frac{2}{d_{neck}}\xi, & 0 \le \xi \le \frac{d_{neck}}{2}, \\ 0, & \xi > \frac{d_{neck}}{2}. \end{cases} \tag{2.8}$$

Then we define the variable diffusion as

$$\varepsilon(\vec{x}) = \varepsilon_0 \theta(dist(R_{pinch}, \vec{x})), \ \vec{x} \in E\Omega, \tag{2.9}$$

with $\varepsilon_0 = \frac{d_{neck}^2}{\tau_{pinch}}$, where τ_{pinch} is the time-step used in going through the topological change (see the next section). Note that ε_0 has the correct units and is chosen by the following 'back-of-the-envelope' calculation.

Let us assume that the domain has a thin region (thickness less than d_{neck}) and is collapsing together (i.e. pinching). When we solve equation (2.7) in Section 2.6.1, we compute a distance function ϕ^0 (to the domain boundary Γ) for the initial condition. The maximum value of the distance function at a point $\vec{x}_0$ at time t_0 in the neck region is:

$$\phi^0(\vec{x}_0, t_0) = \frac{d_{neck}}{2},$$

a positive value because we assume the distance function is positive inside the domain. In order to guarantee that the neck will pinch using the level set update, the following inequality must be true

$$\frac{d_{neck}}{2} + \tau_{pinch}\partial_t \phi(\vec{x}_0, t_0) < 0,$$

which implies that the updated ϕ will be negative in the thin region, meaning the domain will pinch-off there. But by equation (2.7), this gives

$$\frac{d_{neck}}{2} + \tau_{pinch}\varepsilon_0 \Delta\phi(\vec{x}_0, t_0) - \tau_{pinch}\vec{u}(\vec{x}_0, t_0) \cdot \nabla\phi(\vec{x}_0, t_0) < 0, \tag{2.10}$$

where the ε_0 is pulled out of the divergence because $\varepsilon(\vec{x})$ is constant in the region R_{pinch}. Due to the shape of ϕ^0 around the necking region, we have that $\Delta\phi(\vec{x}_0, t_0) < 0$. Also, $\Delta\phi(\vec{x}_0, t_0)$ scales as $1/d_{neck}$. Plugging this into (2.10) gives

$$\frac{d_{neck}}{2} - \tau_{pinch}\frac{\varepsilon_0}{d_{neck}} - \tau_{pinch}\vec{u}(\vec{x}_0, t_0) \cdot \nabla\phi(\vec{x}_0, t_0) < 0. \tag{2.11}$$

Rearranging this gives

$$\varepsilon_0 > \frac{d_{neck}^2}{2\tau_{pinch}} - \vec{u}(\vec{x}_0, t_0) \cdot \nabla\phi(\vec{x}_0, t_0) d_{neck}, \tag{2.12}$$

where the velocity term varies over the necking region and could be quite small near the center of the neck. But also the term $\vec{u} \cdot \nabla\phi$, in the case of pinching, will be positive. Therefore, by choosing $\varepsilon_0 = \frac{d_{neck}^2}{\tau_{pinch}}$ (conservatively) we guarantee that any thin regions of thickness less than d_{neck} will pinch-off if the evolution time of the level set equation is τ_{pinch}. The same analysis holds for joining or reconnecting domains.

The addition of the diffusion term is directly analogous to the methods used in solving hyperbolic equations by 'up-winding', which adds a small amount of diffusion on the order of the mesh size h. In our computations, we take $\tau_{pinch} = d_{neck}$ (note: everything is non-dimensional). This means $\varepsilon_0 = d_{neck}$, which is the desired resolution of our method and is related to the mesh size, since there are only a small number of triangles present in the necking region. Hence, our method of adding artificial diffusion is not completely out of line from other techniques for solving convection equations.

Time-Discrete Level Set Equation

Next, we need to discretize the level set equation for use in our algorithm. The time-discrete version of (2.7) using Euler's method for one time step is

$$\frac{\phi^1 - \phi^0}{\tau_{pinch}} + \vec{u} \cdot \nabla \phi^0 - \nabla \cdot (\varepsilon(\vec{x}) \nabla \phi^1) = 0, \tag{2.13}$$

where the convective term is explicit, the diffusive term is implicit, and a time-step of τ_{pinch} is used. One could also use a higher-order time-stepping scheme here. But this is not critical since only one step of the level-set equation is used in updating the topology.

<u>Weak-Formulation</u>

The weak formulation of the level set update equation is now obtained by multiplying with a test function w, and integrating the diffusion term by parts. The variational formulation of the problem then reads: Given a velocity field $\vec{u}$ and initial data ϕ^0, find a new function $\phi^1 \in H^1(E\Omega)$ such that

$$\frac{1}{\tau_{pinch}} \int_{E\Omega} (\phi^1 - \phi^0) w + \int_{E\Omega} (\vec{u} \cdot \nabla \phi^0) w + \int_{E\Omega} \varepsilon(\vec{x}) \nabla \phi^1 \cdot \nabla w = 0, \text{ for all } w \in H^1(E\Omega), \tag{2.14}$$

where we have assumed zero Neumann data on the boundary of $E\Omega$ (i.e. Γ_E). Rearranging the formulation gives

$$\int_{E\Omega} \phi^1 w + \tau_{pinch} \int_{E\Omega} \varepsilon(x) \nabla \phi^1 \cdot \nabla w = \int_{E\Omega} \phi^0 w - \tau_{pinch} \int_{E\Omega} (\vec{u} \cdot \nabla \phi^0) w, \tag{2.15}$$

with the given data placed on the right hand side. This weak formulation is discretized using piecewise linear basis functions for ϕ^0, ϕ^1 and the vector velocity $\vec{u}$. The variable diffusion term $\varepsilon(\vec{x})$ is accounted for when computing the integrals through the use of formula 2.9 and quadrature. In addition, we use a different mesh of triangles $\top_\phi$ which is a locally refined version of $\top_{E\Omega}$ where the refinement is done in the region R_{pinch}. This is done for improved accuracy in the region where we need it.

Obtaining the New Domain Topology

The new domain $\tilde{\Omega}$ that corresponds to Ω after the topological change has occurred is given by the following procedure:

1. Let ϕ^0 be the signed distance function to the boundary Γ on the extended domain $E\Omega$; note that $\Omega = \{\vec{x} \in E\Omega : \phi^0(\vec{x}) \geq 0\}$. This embeds the domain boundary Γ as the zero level set of ϕ^0.

2. Let ϕ^0 be the initial condition for solving the level set equation (2.15) stated in the previous section. We use a time-step given by $\tau_{pinch} := d_{neck}$ for computing the update. This gives a new level set function ϕ^1 that defines the new topology.

3. Let $\tilde{\Omega} := \{\vec{x} \in E\Omega : \phi^1(\vec{x}) \geq 0\}$ and let $\tilde{\Gamma} := \partial\tilde{\Omega}$. Note that $\tilde{\Gamma} = \{\vec{x} \in E\Omega : \phi^1(\vec{x}) = 0\}$.

4. Let $\vec{x}_i$ denote the barycenter of triangle $T_i \in \top_{E\Omega}$.

5. Then the mesh of triangles for the new domain $\tilde{\Omega}$ is given by the set formula: $\top_{\tilde{\Omega}} = (\top_\Omega \bigcup \top_P) \setminus \top_N$, where $\top_P := \{T_i \in \top_{E\Omega} : \vec{x}_i \in \tilde{\Omega} \bigcap R_{pinch}\}$ and $\top_N := \{T_i \in \top_{E\Omega} : \vec{x}_i \in (\backslash\tilde{\Omega}) \bigcap R_{pinch}\}$. Hence we obtain the new triangulation by adding and subtracting triangles from the old mesh $\top_\Omega$ in the local region

R_{pinch}.

Now we define some notation that is used in the next section. Let $\mathsf{S}_{\tilde{\Omega}}$ be the set of triangle sides that make up the boundary of the mesh $\mathsf{T}_{\tilde{\Omega}}$. Let Γ_{S} denote the set of points in the plane defined by the union of all sides in $\mathsf{S}_{\tilde{\Omega}}$.

2.6.2 Active Contours for Mesh Reconstruction

The boundary of the mesh Γ_{S} will not necessarily conform to the zero level set of ϕ^1 (i.e. to the new domain boundary $\tilde{\Gamma}$ after the topological change). Therefore, it is necessary to adjust the new boundary mesh $\mathsf{S}_{\tilde{\Omega}}$ so that it does conform. In Section 2.6.2, we derive a minimization problem that gives us a method for adjusting the mesh boundary. For notational convenience, we take Γ_{S} to be synonymous with $\mathsf{S}_{\tilde{\Omega}}$ and let $\Psi = \phi^1$. In Section 2.6.2, we give the procedure for obtaining the final boundary and domain mesh.

Shape Minimization Problem

Since the updated level set function $\Psi : \mathbb{R}^3 \to \mathbb{R}$ is available, we can adjust the boundary mesh Γ_{S} by solving a minimization problem. This can also be done in 3-D for a 2-D surface, which makes this approach attractive. Hence, we will take a more general point of view and denote by d the ambient dimension. First we define the 'cost' functional that will be minimized,

$$J(\Gamma) = \int_{\Gamma} \Psi^2, \tag{2.16}$$

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

$$\Gamma^* = \arg\min_{\Gamma} J(\Gamma). \tag{2.17}$$

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

The surface that minimizes the functional (2.16) is computed by defining an L^2 gradient flow. This is basically a gradient descent method that seeks to move the surface in a direction that is guaranteed to minimize the cost J. We proceed first by deriving the shape derivative of the functional J to get the descent direction.

<u>Derive Shape Derivative</u>

Let $\vec{X} : U_i \to \mathbb{R}^d$ be a mapping, where U_i is a reference domain in $\mathbb{R}^{d-1}$ and i is a parameter in a finite set. Next, let $\vec{X}$ satisfy $\bigcup_i \vec{X}(U_i) = \Gamma$. Hence, $\vec{X}(\cdot) = \Gamma$ is a surface parameterization using local charts [12]. Next, let $\vec{\varphi} : \Gamma \to \mathbb{R}^d$ be a vector perturbation and define $\vec{X}_\epsilon := \vec{X} + \epsilon[\vec{\varphi} \circ \vec{X}]$ (with $\epsilon > 0$) to be a new mapping. This defines a new surface $\Gamma_\epsilon := \vec{X}_\epsilon(\cdot)$, which is a perturbation of Γ.

Let $\mu(\epsilon) := J(\Gamma_\epsilon)$ be a scalar function of one variable. By taking $\vec{\varphi}$ to be such that $[\vec{\varphi} \circ \vec{X}]$ has compact support in some local chart U_i, we can write the derivative of $\mu(\epsilon)$ in the following form by a change of variables

$$\frac{d\mu(\epsilon)}{d\epsilon} = \frac{d}{d\epsilon} \int_{\Gamma_\epsilon} \Psi^2 = \frac{d}{d\epsilon} \int_{U_i} (\Psi^2) \circ \vec{X}_\epsilon |\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon| ds_1 ds_2, \tag{2.18}$$

where s_1 and s_2 are the surface parameterization variables, $\partial_{s_1} \vec{X}_\epsilon = \partial_{s_1} \vec{X} + \epsilon \partial_{s_1} [\vec{\varphi} \circ \vec{X}]$ ($\partial_{s_2} \vec{X}_\epsilon$ is similar), and $(\Psi^2) \circ \vec{X}_\epsilon$ means composition of Ψ^2 with the mapping $\vec{X}_\epsilon$. The product rule then gives

$$\frac{d\mu(\epsilon)}{d\epsilon} = \int_{U_i} \frac{d}{d\epsilon}[(\Psi^2) \circ \vec{X}_\epsilon] |\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon| + [(\Psi^2) \circ \vec{X}_\epsilon] \frac{d}{d\epsilon} |\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon| ds_1 ds_2. \tag{2.19}$$

- *Compute Derivative Terms*

Next, we compute the first derivative appearing in the integrand of (2.19). Since $\frac{d}{d\epsilon}\vec{X}_\epsilon = \vec{\varphi} \circ \vec{X}$, we have by the chain rule

$$\frac{d}{d\epsilon}[(\Psi^2) \circ \vec{X}_\epsilon]|_{\epsilon=0} = [(\nabla(\Psi^2)) \circ \vec{X}_\epsilon] \cdot [\vec{\varphi} \circ \vec{X}]|_{\epsilon=0} = [(\nabla(\Psi^2)) \circ \vec{X}] \cdot [\vec{\varphi} \circ \vec{X}], \quad (2.20)$$

where $\cdot$ denotes the 'dot' product of the two vectors.

The other derivative is quite technical, so we will proceed with caution. First, the result we want is

$$\frac{d}{d\epsilon}|\partial_{s_1}\vec{X}_\epsilon \times \partial_{s_2}\vec{X}_\epsilon|_{\epsilon=0} = \{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\} |\partial_{s_1}\vec{X} \times \partial_{s_2}\vec{X}|. \quad (2.21)$$

To get (2.21), we start by defining some quantities from differential geometry [12]. The 1st fundamental form of differential geometry is given by a metric which, for a 2-D surface, is a 2x2 matrix:

$$[g_{ij}]_{1 \le i,j \le 2} = \begin{bmatrix} E & F \\ F & G \end{bmatrix}, \quad (2.22)$$

where E, F, G are given by the 'dot' products

$$E = \partial_{s_1}\vec{X} \cdot \partial_{s_1}\vec{X},\ F = \partial_{s_1}\vec{X} \cdot \partial_{s_2}\vec{X},\ G = \partial_{s_2}\vec{X} \cdot \partial_{s_2}\vec{X}, \quad (2.23)$$

where $\vec{X}$ is the parameterization of the surface Γ, and s_1, s_2 are the parameterization coordinates (i.e. $\vec{X} = \vec{X}(s_1, s_2)$). The inverse of the matrix is given by

$$[g^{ij}]_{1 \le i,j \le 2} = \frac{1}{EG - F^2} \begin{bmatrix} G & -F \\ -F & E \end{bmatrix}, \quad (2.24)$$

and of course we have this property,

$$\delta_i^j = \sum_{k=1}^{3} g_{ik} g^{kj} = \sum_{k=1}^{3} g_{ik} g^{jk}, \quad (2.25)$$

$$\delta_i^j = 1, i = j, \tag{2.26}$$

$$\delta_i^j = 0, i \neq j.$$

Let $\omega : \Gamma \to \mathbb{R}$ be a scalar function defined on the surface Γ. Then the surface gradient $\nabla_\Gamma(\cdot)$ of ω in local coordinates is defined by

$$[\nabla_\Gamma \omega] \circ \vec{X} = \sum_{i,j=1}^{2} g^{ij} \partial_{s_i} \tilde{\omega} \partial_{s_j} \vec{X}, \tag{2.27}$$

where $\tilde{\omega} = \omega \circ \vec{X}$ is in local coordinates.

Recall $\vec{\varphi}$ is a vector perturbation and let φ_k denote the coordinate functions of $\vec{\varphi}$ (i.e. $\vec{\varphi} = (\varphi_1, \varphi_2, \varphi_3)$). Let $\tilde{\vec{\varphi}} = \vec{\varphi} \circ \vec{X}$ and $\tilde{\varphi}_k = \varphi_k \circ \vec{X}$ denote the quantities in local coordinates. As a first step, we will compute the surface gradient of φ_k

$$[\nabla_\Gamma \varphi_k] \circ \vec{X} = \sum_{i,j=1}^{2} g^{ij} \partial_{s_i} \tilde{\varphi}_k \partial_{s_j} \vec{X}. \tag{2.28}$$

And we also want the following quantity as well

$$[\nabla_\Gamma (X_k \circ \vec{X}^{-1})] \circ \vec{X} = \sum_{i,j=1}^{2} g^{ij} \partial_{s_i} X_k \partial_{s_j} \vec{X}, \tag{2.29}$$

where $X_k : U_i \to \mathbb{R}$, for $k = 1, 2, 3$, are the coordinate functions of $\vec{X}$ (i.e. $\vec{X} = (X_1, X_2, X_3)$).

Next, we want the 'dot' product of the two previous quantities

$$\{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\} := \sum_{k=1}^{3} \{[\nabla_\Gamma(X_k \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \varphi_k] \circ \vec{X}\}, \tag{2.30}$$

$$= \sum_{k=1}^{3} \sum_{i,j=1}^{2} \sum_{p,q=1}^{2} g^{ij} \partial_{s_i} \tilde{\varphi}_k \partial_{s_j} \vec{X} \cdot \partial_{s_q} \vec{X} g^{pq} \partial_{s_p} X_k, \tag{2.31}$$

which is equal to

$$\{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\} = \sum_{k=1}^{3} \sum_{i,j=1}^{2} \sum_{p,q=1}^{2} g^{ij} \partial_{s_i} \tilde{\varphi}_k g_{jq} g^{pq} \partial_{s_p} X_k, \qquad (2.32)$$

by definition (2.22). Using properties (2.25) and (2.26), and noting that $\partial_{s_i} \tilde{\vec{\varphi}} \cdot \partial_{s_j} \vec{X} = \sum_{k=1}^{3} \partial_{s_i} \tilde{\varphi}_k \partial_{s_j} X_k$, we get

$$\{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\} = \sum_{i,j=1}^{2} g^{ij} \partial_{s_i} \tilde{\vec{\varphi}} \cdot \partial_{s_j} \vec{X}. \qquad (2.33)$$

We write this more explicitly for later use

$$\begin{aligned} \{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\} = \frac{1}{EG - F^2} \cdot \\ \cdot [G(\partial_{s_1} \tilde{\vec{\varphi}} \cdot \partial_{s_1} \vec{X}) - F(\partial_{s_1} \tilde{\vec{\varphi}} \cdot \partial_{s_2} \vec{X}) \qquad (2.34) \\ - F(\partial_{s_2} \tilde{\vec{\varphi}} \cdot \partial_{s_1} \vec{X}) + E(\partial_{s_2} \tilde{\vec{\varphi}} \cdot \partial_{s_2} \vec{X})]. \end{aligned}$$

Now, we compute the left hand side of equation (2.21)

$$\begin{aligned} \frac{d}{d\epsilon} |\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon|_{\epsilon=0} &= \frac{d}{d\epsilon} [(\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon) \cdot (\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon)]^{1/2}|_{\epsilon=0}, \qquad (2.35) \\ &= \frac{(\partial_{s_1} \vec{X} \times \partial_{s_2} \vec{X})}{|\partial_{s_1} \vec{X} \times \partial_{s_2} \vec{X}|} \cdot \frac{d}{d\epsilon} (\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon)|_{\epsilon=0}. \qquad (2.36) \end{aligned}$$

Since $\frac{d}{d\epsilon} \vec{X}_\epsilon|_{\epsilon=0} = \vec{\varphi} \circ \vec{X} = \tilde{\vec{\varphi}}$, we have that

$$\frac{d}{d\epsilon} \partial_{s_1} \vec{X}_\epsilon|_{\epsilon=0} = \partial_{s_1} \tilde{\vec{\varphi}} \qquad \frac{d}{d\epsilon} \partial_{s_2} \vec{X}_\epsilon|_{\epsilon=0} = \partial_{s_2} \tilde{\vec{\varphi}}.$$

Plugging this into (2.35) and applying the product rule gives

$$\frac{d}{d\epsilon} |\partial_{s_1} \vec{X}_\epsilon \times \partial_{s_2} \vec{X}_\epsilon|_{\epsilon=0} = \frac{(\partial_{s_1} \vec{X} \times \partial_{s_2} \vec{X})}{|\partial_{s_1} \vec{X} \times \partial_{s_2} \vec{X}|} \cdot [(\partial_{s_1} \tilde{\vec{\varphi}} \times \partial_{s_2} \vec{X}) + (\partial_{s_1} \vec{X} \times \partial_{s_2} \tilde{\vec{\varphi}})]. \qquad (2.37)$$

Before continuing, note that

$$|\partial_{s_1} \vec{X} \times \partial_{s_2} \vec{X}| = (EG - F^2)^{1/2}, \qquad (2.38)$$

and the following vector identity is valid pointwise

$$(\vec{a} \times \vec{b}) \cdot (\vec{c} \times \vec{d}) = (\vec{a} \cdot \vec{c})(\vec{b} \cdot \vec{d}) - (\vec{a} \cdot \vec{d})(\vec{b} \cdot \vec{c}), \tag{2.39}$$

where $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ are 3-D vectors.

Using relations (2.38) and (2.39), equation (2.37) can be simplified

$$\begin{aligned}\frac{d}{d\epsilon}|\partial_{s_1}\vec{X}_\epsilon \times \partial_{s_2}\vec{X}_\epsilon|_{\epsilon=0} = \frac{1}{(EG-F^2)^{1/2}}&[+(\partial_{s_2}\vec{X} \cdot \partial_{s_2}\vec{X})(\partial_{s_1}\tilde{\vec{\varphi}} \cdot \partial_{s_1}\vec{X}) \\ &- (\partial_{s_1}\vec{X} \cdot \partial_{s_2}\vec{X})(\partial_{s_1}\tilde{\vec{\varphi}} \cdot \partial_{s_2}\vec{X}) \\ &- (\partial_{s_1}\vec{X} \cdot \partial_{s_2}\vec{X})(\partial_{s_2}\tilde{\vec{\varphi}} \cdot \partial_{s_1}\vec{X}) \\ &+ (\partial_{s_1}\vec{X} \cdot \partial_{s_1}\vec{X})(\partial_{s_2}\tilde{\vec{\varphi}} \cdot \partial_{s_2}\vec{X})],\end{aligned}$$

where upon using the definition in (2.23) gives

$$\begin{aligned}\frac{d}{d\epsilon}|\partial_{s_1}\vec{X}_\epsilon \times \partial_{s_2}\vec{X}_\epsilon|_{\epsilon=0} = &\frac{1}{(EG-F^2)^{1/2}}\cdot \\ &\cdot [G(\partial_{s_1}\tilde{\vec{\varphi}} \cdot \partial_{s_1}\vec{X}) - F(\partial_{s_1}\tilde{\vec{\varphi}} \cdot \partial_{s_2}\vec{X}) \\ &- F(\partial_{s_2}\tilde{\vec{\varphi}} \cdot \partial_{s_1}\vec{X}) + E(\partial_{s_2}\tilde{\vec{\varphi}} \cdot \partial_{s_2}\vec{X})].\end{aligned} \tag{2.40}$$

Now notice that equation (2.40) is almost exactly (2.34). Thus, we get the result (2.21) that we wanted:

$$\begin{aligned}\frac{d}{d\epsilon}|\partial_{s_1}\vec{X}_\epsilon \times \partial_{s_2}\vec{X}_\epsilon|_{\epsilon=0} &= \{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\}(EG-F^2)^{1/2}, \\ &= \{[\nabla_\Gamma(\vec{X} \circ \vec{X}^{-1})] \circ \vec{X}\} \cdot \{[\nabla_\Gamma \vec{\varphi}] \circ \vec{X}\}|\partial_{s_1}\vec{X} \times \partial_{s_2}\vec{X}|.\end{aligned}$$

- *Define Shape Derivative*

The shape derivative is defined by

$$dJ(\Gamma, \vec{\varphi}) := \frac{d}{d\epsilon}J(\Gamma_\epsilon)\ |_{\epsilon=0} = \frac{d\mu(\epsilon)}{d\epsilon}\ |_{\epsilon=0}\ .$$

Combining this with the definition of $\mu(\epsilon)$ and equations (2.19), (2.20), (2.21), we get

$$dJ(\Gamma,\vec{\varphi}) = \int_{U_i} [(\nabla(\Psi^2))\circ\vec{X}]\cdot[\vec{\varphi}\circ\vec{X}]|\partial_{s_1}\vec{X}\times\partial_{s_2}\vec{X}|+ \tag{2.41}$$

$$+[(\Psi^2)\circ\vec{X}]\{[\nabla_\Gamma(\vec{X}\circ\vec{X}^{-1})]\circ\vec{X}\}\cdot\{[\nabla_\Gamma\vec{\varphi}]\circ\vec{X}\}|\partial_{s_1}\vec{X}\times\partial_{s_2}\vec{X}|ds_1ds_2,$$

on the local chart U_i. After mapping back to the surface Γ we get

$$dJ(\Gamma,\vec{\varphi}) = \int_\Gamma \nabla(\Psi^2)\cdot\vec{\varphi} + \Psi^2\nabla_\Gamma(\vec{X}\circ\vec{X}^{-1})\cdot\nabla_\Gamma\vec{\varphi}, \tag{2.42}$$

where the term $(\vec{X}\circ\vec{X}^{-1})$ is clearly the identity. However, it is common to adopt the following abuse of notation, and write (2.42) as

$$dJ(\Gamma,\vec{\varphi}) = \int_\Gamma \nabla(\Psi^2)\cdot\vec{\varphi} + \Psi^2\nabla_\Gamma\vec{X}\cdot\nabla_\Gamma\vec{\varphi}. \tag{2.43}$$

Define Gradient Flow

The shape derivative allows us to define a gradient flow that will minimize the cost. We do this by first defining a vector velocity $\vec{V}$ on the surface Γ by

$$\int_\Gamma \vec{V}\cdot\vec{\varphi} = -dJ(\Gamma,\vec{\varphi}), \tag{2.44}$$

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

$$\frac{d}{dt}\vec{X}(\cdot,t) = \vec{V}(\cdot), \quad \vec{X}(\cdot,t) = \Gamma(t), \tag{2.45}$$

which means the surface Γ will move with the velocity $\vec{V}$.

Semi-Implicit Time Discretization

We solve the gradient flow problem by using a semi-implicit time-discretization. This is done by setting $\vec{V}$ to $\vec{V}^{n+1}$ in (2.44) and using a backward Euler method for

(2.45). Combining with equation (2.42) gives

$$\int_{\Gamma} \vec{V}^{n+1} \cdot \vec{\varphi} = -\int_{\Gamma} \nabla(\Psi^2) \cdot \vec{\varphi} + \Psi^2 \nabla_{\Gamma} \vec{X}^{n+1} \cdot \nabla_{\Gamma} \vec{\varphi}, \tag{2.46}$$

$$\vec{X}^{n+1} = \vec{X}^{n} + \alpha \vec{V}^{n+1}, \tag{2.47}$$

where the superscript is the iteration index and α is the step size to use in updating Γ at each iteration.

Weak-Formulation

Rearranging slightly, gives the following variational formulation: given $\vec{X}^n$ and Ψ^2, find $\vec{V}^{n+1} \in H^1(\Gamma)$ such that

$$\int_{\Gamma} \vec{V}^{n+1} \cdot \vec{\varphi} + \alpha \int_{\Gamma} \Psi^2 \nabla_{\Gamma} \vec{V}^{n+1} \cdot \nabla_{\Gamma} \vec{\varphi} = -\int_{\Gamma} \nabla(\Psi^2) \cdot \vec{\varphi} + \Psi^2 \nabla_{\Gamma} \vec{X}^{n} \cdot \nabla_{\Gamma} \vec{\varphi}, \tag{2.48}$$

for all $\vec{\varphi} \in H^1(\Gamma)$. Given the solution $\vec{V}^{n+1}$, the new position of Γ is obtained by equation (2.47). This process is then iterated until the surface Γ reaches a minimum of $\int_{\Gamma} \Psi^2$.

This minimization process is quite general and can be applied to the polygonal curve Γ_{S}. The equations are exactly the same, except Γ is the 1-D curve Γ_{S}, and $\vec{X}$, $\vec{V}$, $\vec{\varphi}$ are 2-D vector fields (defined on Γ_{S}) instead of 3-D.

Adjusting the Boundary Mesh

For our computational purposes, equation (2.48) is discretized in space using piecewise linear 'hat' functions over the polygonal boundary Γ_{S}. See [14], [15] for examples of this kind. We then use the FEM implementation of equation (2.48), with Γ_{S} as the initial condition, to obtain a new polygonal boundary $\overline{\Gamma_{\mathsf{S}}}$ that better approximates the zero level set of Ψ.

This defines a new set of edges $\overline{\mathsf{S}_{\tilde{\Omega}}}$ (corresponding to $\overline{\Gamma_{\mathsf{S}}}$) that defines the boundary of the final domain mesh $\overline{\top_{\tilde{\Omega}}}$. This mesh is the final output of the overall algorithm, and is returned to the main simulation for further time-stepping.

Computing the Final Mesh

Re-distancing

In order to get a 'good' flow direction from $\nabla\Psi^2$, we need to replace Ψ by a distance function with the same zero level set as Ψ (i.e. we must re-distance Ψ). This is because the variable diffusion term in (2.15) causes the scalar function Ψ to be fairly flat in the topological change region R_{pinch}. This means $\nabla\Psi^2$ does not provide a good 'forcing' direction for the minimization process to follow. Re-distancing is not a problem, since this only needs to be done locally around R_{pinch}.

Continuous Approximation of $\nabla(\Psi^2)$

Also, in implementing $\nabla(\Psi^2)$, we actually compute it weakly before starting the minimization. This is defined by

$$\int_{E\Omega} \vec{G} \cdot \vec{w} = -\int_{E\Omega} \Psi^2 \nabla \cdot \vec{w}, \tag{2.49}$$

for all test functions $\vec{\varphi}$ with compact support in $E\Omega$ (recall that Ψ is defined over $E\Omega$). Hence, $\vec{G} = \nabla(\Psi^2)$ in the weak sense. For the discrete problem, $\vec{G}$ and $\vec{w}$ are piecewise linear functions, defined over $E\Omega$, with zero boundary data on Γ_E. In order to have a satisfactory approximation of $\nabla(\Psi^2)$ using $\vec{G}$, we use a mesh for $E\Omega$ that is locally refined in the region R_{pinch}. This is done by applying a fixed number of local refinements to the triangulation $\top_{E\Omega}$ of $E\Omega$, which gives a new triangulation denoted $\top_{\vec{G}}$. When computing the right hand side integral $\int_\Gamma \nabla(\Psi^2) \cdot \vec{\varphi}$ in equation (2.48), we first replace it by $\int_\Gamma \vec{G} \cdot \vec{\varphi}$. We then use quadrature to interpolate the piecewise linear function $\vec{G}$, defined on the triangulation $\top_{\vec{G}}$, onto the curve Γ which allows us to compute the integral. As long as the mesh $\top_{\vec{G}}$ is fine enough to resolve $\nabla(\Psi^2)$ around the local region R_{pinch}, this will not interfere with the minimization process too much. In other words, $\vec{G}$ is essentially just given data for the weak form (2.48).

Local Re-meshing

With the new mesh boundary given, we have to adjust the bulk interior mesh to account for any inverted elements. This only needs to be done in the region R_{pinch}, so is not too expensive. For 2-D meshes, this is done using Triangle.

Lastly, we emphasize that the adjustment only needs to be done in the local covering region R_{pinch}. The mesh vertices away from the topological change, which originally lie along the zero level set of ϕ^0, can be moved directly using the velocity $\vec{u}$. Hence, the updated positions of those vertices will lie along the zero level set of $\phi^1 = \Psi$. Therefore, this final mesh adjustment step does not impact the computational efficiency in a significant way.

An obvious question here is "Why not use the piecewise linear (polygonal) boundary given by finding the zero level contour (on the mesh $\top_\phi$) of the updated level set function ϕ^1 as the new boundary?". This would certainly give a minimum of the cost functional J. But there is a problem with this because after updating the level set function, the new zero level contour will not (necessarily) conform to the edges of the mesh given by the triangulation $\top_\phi$ on which we computed the level set update. This will produce many small edges in some places along the zero level set curve of ϕ^1. Thus, it is undesirable to use the updated zero level set as the new domain boundary $\overline{\Gamma_{\mathsf{S}}}$ because a post-process would need to be done to improve the mesh quality along the whole boundary, especially in the region R_{pinch} where we used a local refinement. Furthermore, a satisfactory mesh is already present around the mesh away from R_{pinch}. Hence, we decided to preserve the shape regularity of the mesh by using our optimization method.

Chapter 3

Numerical Experiments

We present three simulations to demonstrate the method described in Chapter 2. 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 [10], [11], [44], which consists of a Hele-Shaw cell [40], [26] with the ability to modify surface tension effects through electric fields. These devices are capable of splitting and merging droplets and have potential applications for 'lab-on-a-chip' devices [24], [27]. The third simulation demonstrates reconnection of droplets in a Hele-Shaw cell due solely to surface tension (no electro-forcing). These examples are rather extreme, and were chosen to push the limits of our method.

3.1 Rotating Vortices

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

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

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

which is a two-by-two array of counter-rotating vortices, and the divergence of $\vec{u}$ is zero. The initial domain shape is a rectangle inside a unit square, shown in Figure 3.1. 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 equation (2.1)). The rectangular 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 $d_{neck} = 4 \times 10^{-3}$, the topological change routine is executed. Figures 3.2 and 3.3, show time-frames of the pinching process.

In Figure 3.4, we show a closeup of the pinching region depicted in Figure 3.3. Of course, the dynamics of the flow after the pinch do not change since the velocity field is prescribed.

In Figure 3.5, we show a comparison of the before and after effects of our optimization method (from Section 2.6.2) for pinching in the rotating vortex case.

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.

3.2 EWOD Pinching

In our next experiment, 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. A 3x3 array of square electrodes is embedded in the bottom plate, which are used for applying voltages that can change the effective surface tension locally [38]. This allows for

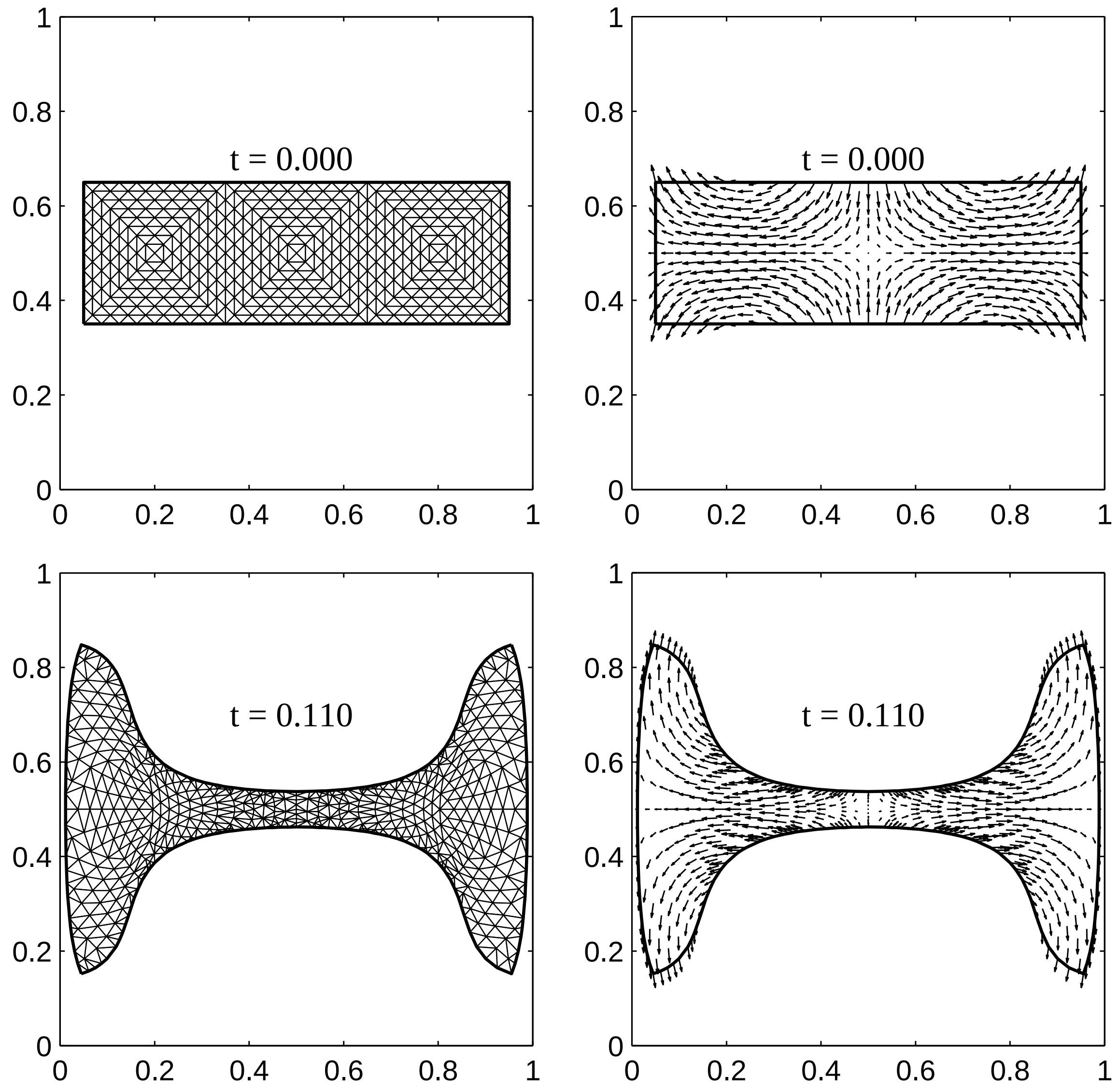


Figure 3.1: Initial rectangular domain (first row) and deformed version shown at a later time (second row). The first column shows a triangular mesh for the rectangle and the second column shows the domain boundary and velocity field. The rectangle undergoes extreme distortion under this flow field. Eventually, the domain becomes very thin in the center and a topological change is executed (shown in later figures).

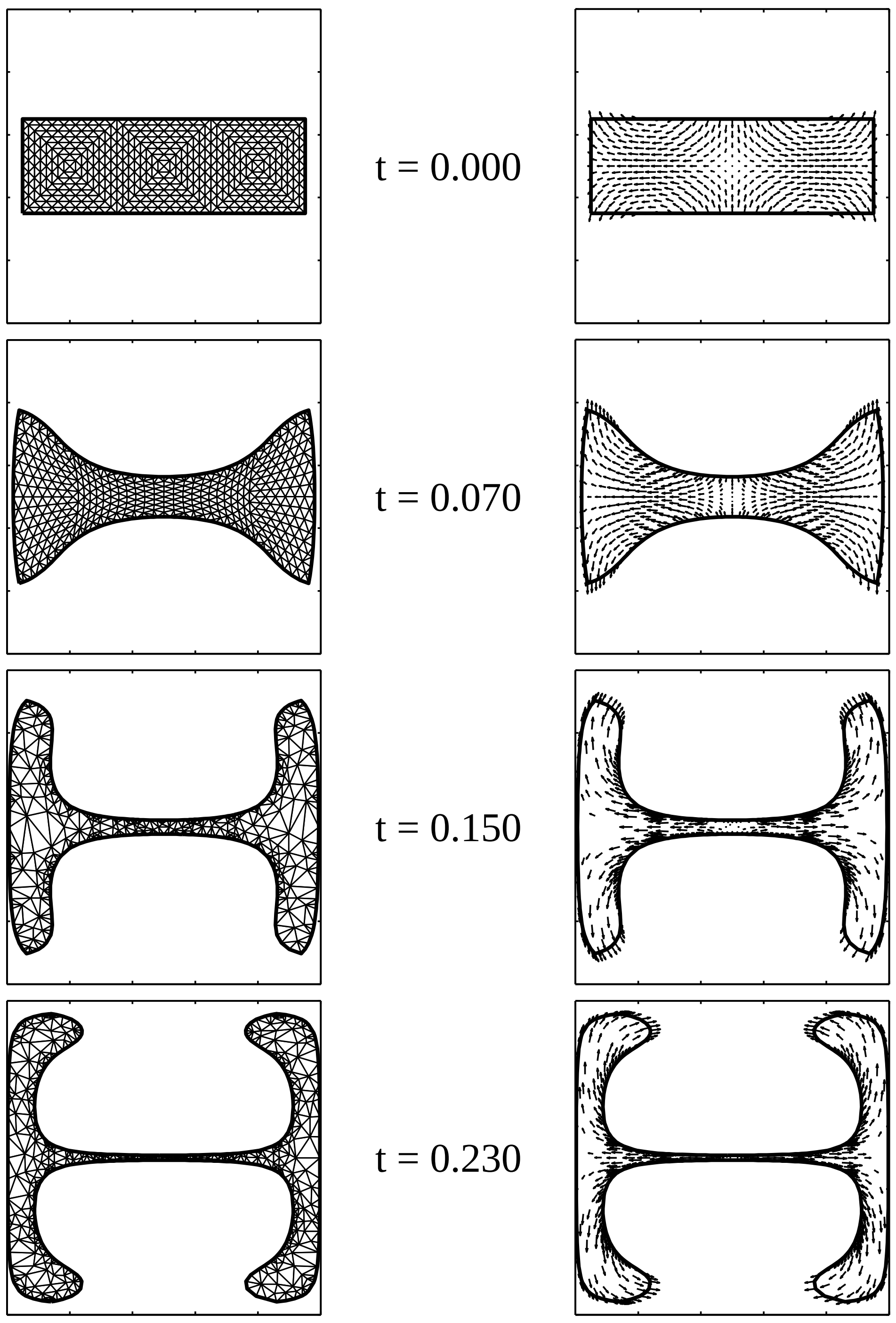


Figure 3.2: Sequences of simulation snapshots for the example given in Figure 3.1. Left column shows the mesh; right column shows the velocity field. Rows correspond to instants in time. Because of the vortex flow field, a long neck develops in the central region. See Figure 3.3 for a continuation.

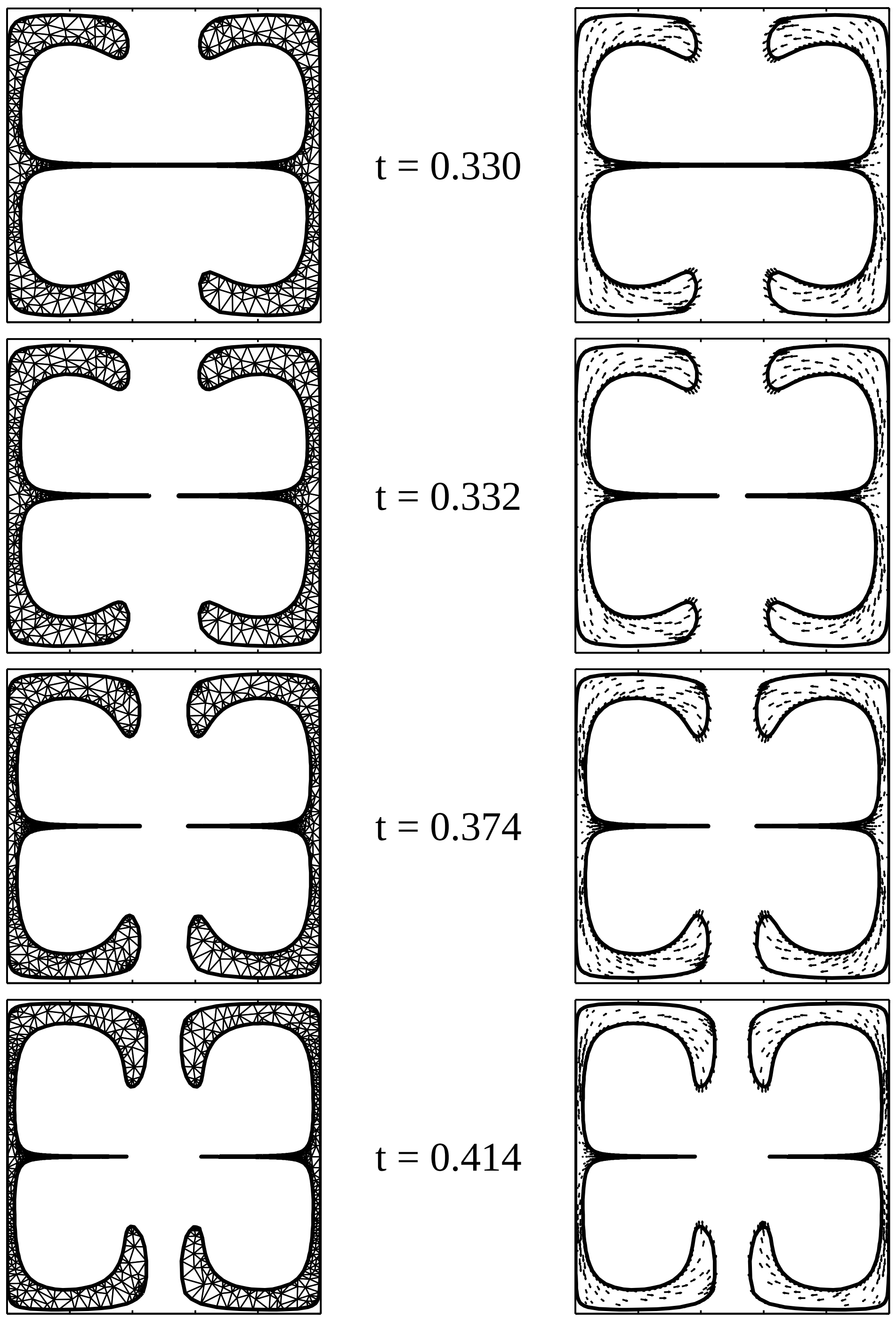


Figure 3.3: Continuation of the simulation shown in Figure 3.2 ($d_{neck} = 4 \times 10^{-3}$) with same format. First and second frames are immediately before and after the time of pinch-off. Afterwards, the prescribed flow field continues to convect the vertices of the mesh.

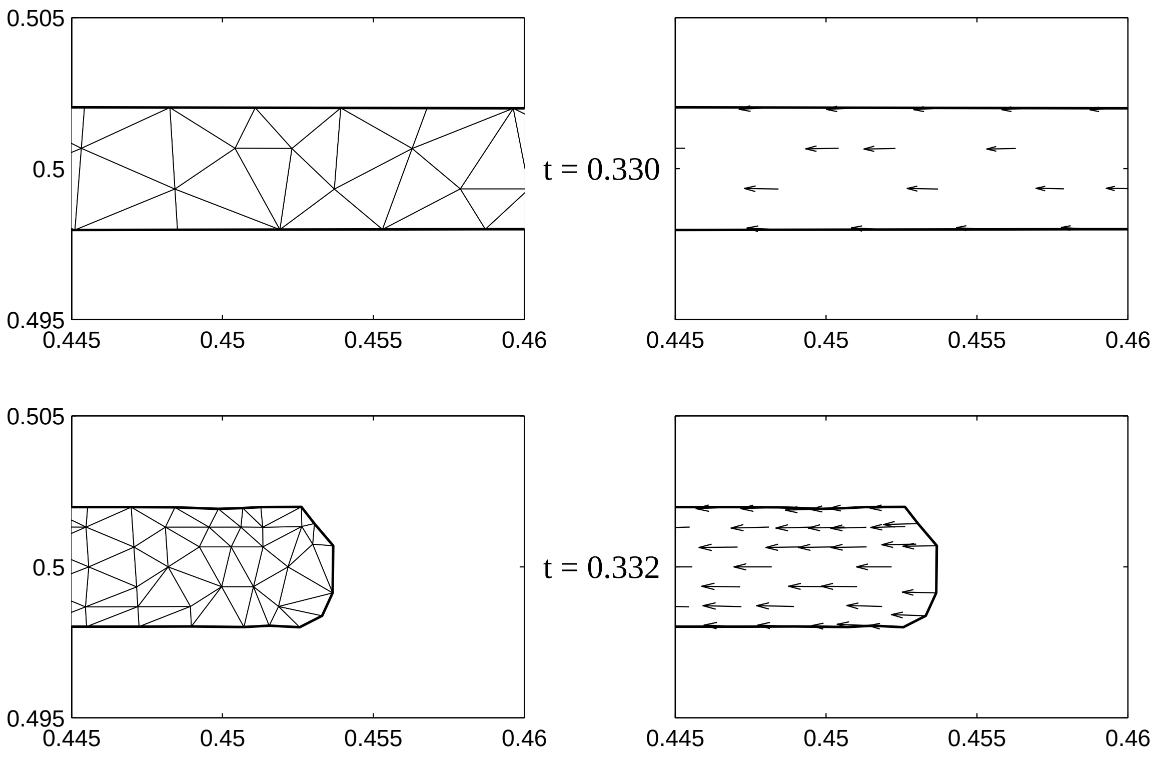
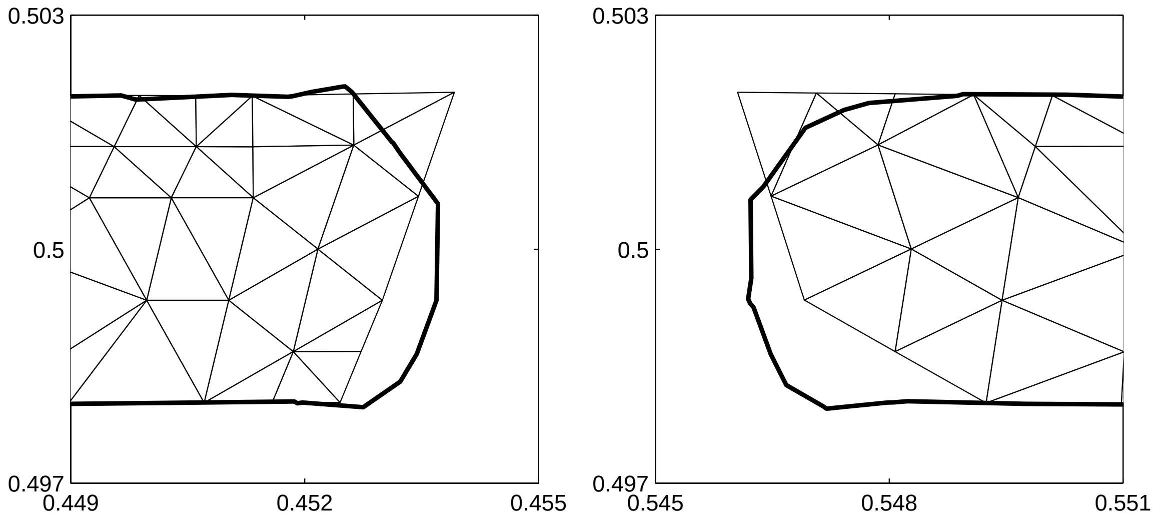
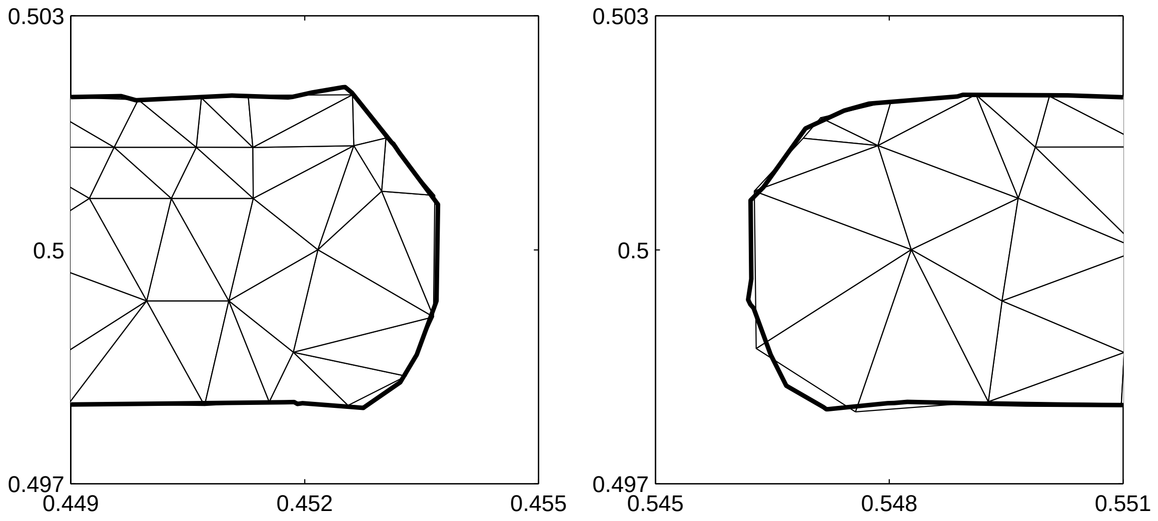


Figure 3.4: Zoom-in of the pinching region in Figure 3.3 ($d_{neck} = 4 \times 10^{-3}$). All of the triangles that were in the pinching region R_{pinch} that have a negative level set value have been deleted. The second row includes the boundary smoothing step shown in Figure 3.5.



(a) Pinched regions before boundary mesh adjustment.



(b) Pinched regions after boundary mesh adjustment.

Figure 3.5: Comparison of the effect of using our optimization method to smooth the boundary ($d_{neck} = 4 \times 10^{-3}$ case). The zero level contour of the level set function (after the topological change) is shown as a thick black curve. The optimization algorithm acts to smooth or compress the boundary mesh towards the zero level contour. Here, only five steps of the optimization algorithm were used.

the ability to force a circular droplet to pinch-off. The initial mesh and electrode layout is given in Figure 3.6.

Figure 3.7 shows six time-frames of a finite element simulation for the electro-wetting example given in Figure 3.6. These time-frames show the motion of the droplet before any topological change take place.

In Sections 3.2.1, 3.2.2, and 3.2.3, we give several snapshots of the simulation shown in Figure 3.7 at later times, which show the behavior of the topological change for three different choices of d_{neck}. In all three cases, the initial droplet configuration and voltage actuation are exactly the same. Yet in each case, we see a change in the dynamics of the topological change. This allows us to see the effect of the choice of d_{neck} on the simulated physics.

In Section 3.2.4, we discuss the effects of the different minimum neck sizes d_{neck} on the velocity field away from the necking region.

3.2.1 EWOD Pinching - $d_{neck} = 2 \times 10^{-2}$

Figure 3.8 shows a sequence of snapshots that is a continuation of the simulation shown in Figure 3.7 with $d_{neck} = 2 \times 10^{-2}$. The dynamics of the topological change in this example are relatively mild with just a single pinch and no satellite droplets.

In Figure 3.9, we show a more detailed (zoom-in) view of the dynamics of the topological change depicted in Figure 3.8.

Figure 3.10 shows the effects of our optimization smoothing algorithm for

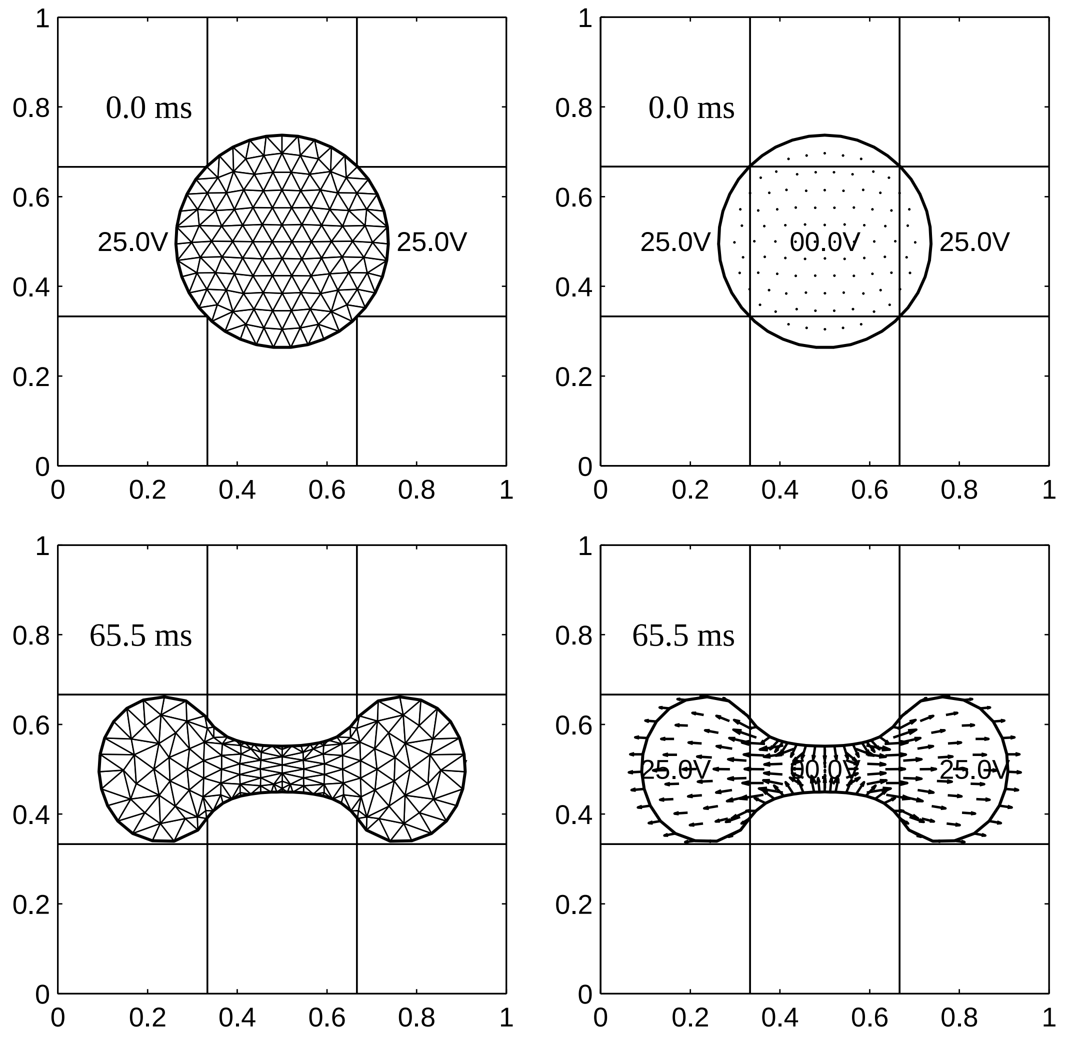


Figure 3.6: Initial circular droplet (first row) in an EWOD device with a ‘necking’ version shown at a later time (second row). The first column shows the finite element mesh for the domain of the droplet and the second column shows the droplet boundary and velocity field. The edges of the 3x3 grid of electrodes are depicted by the light solid lines. The voltage actuation consists of applying 25 V on the left and right electrodes, and 0 V everywhere else continuously throughout the simulation. This causes the pressure applied at the left and right sides of the droplet’s liquid-gas interface to decrease significantly, with a relatively high pressure remaining on the top and bottom. This causes the droplet to be squeezed in the middle until there are two ‘bulbs’ of fluid with a neck in between.

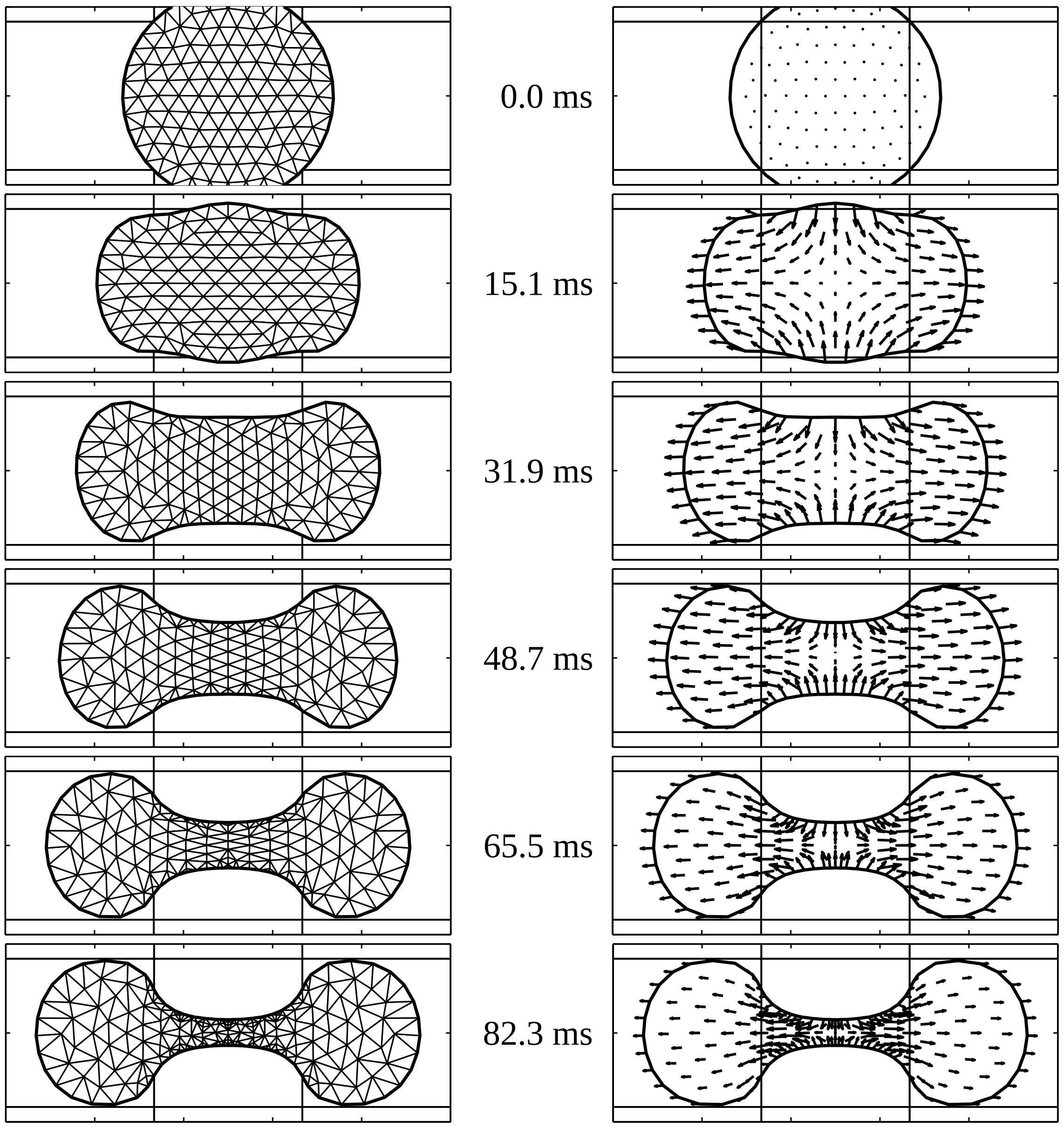


Figure 3.7: Sequence of simulation snapshots for the example given in Figure 3.6. Left column shows the mesh; right column shows the velocity field. Rows correspond to instants in time. Low pressure regions on the left and right sides of the droplet (induced by the voltage actuation) cause the droplet to pull itself apart, because fluid moves from regions of high pressure to low pressure. As a result, a long neck develops between two 'bulbs' of fluid with the neck becoming thinner. The results of the topological change are shown in sections 3.2.1, 3.2.2, and 3.2.3 for three different choices of d_{neck}.

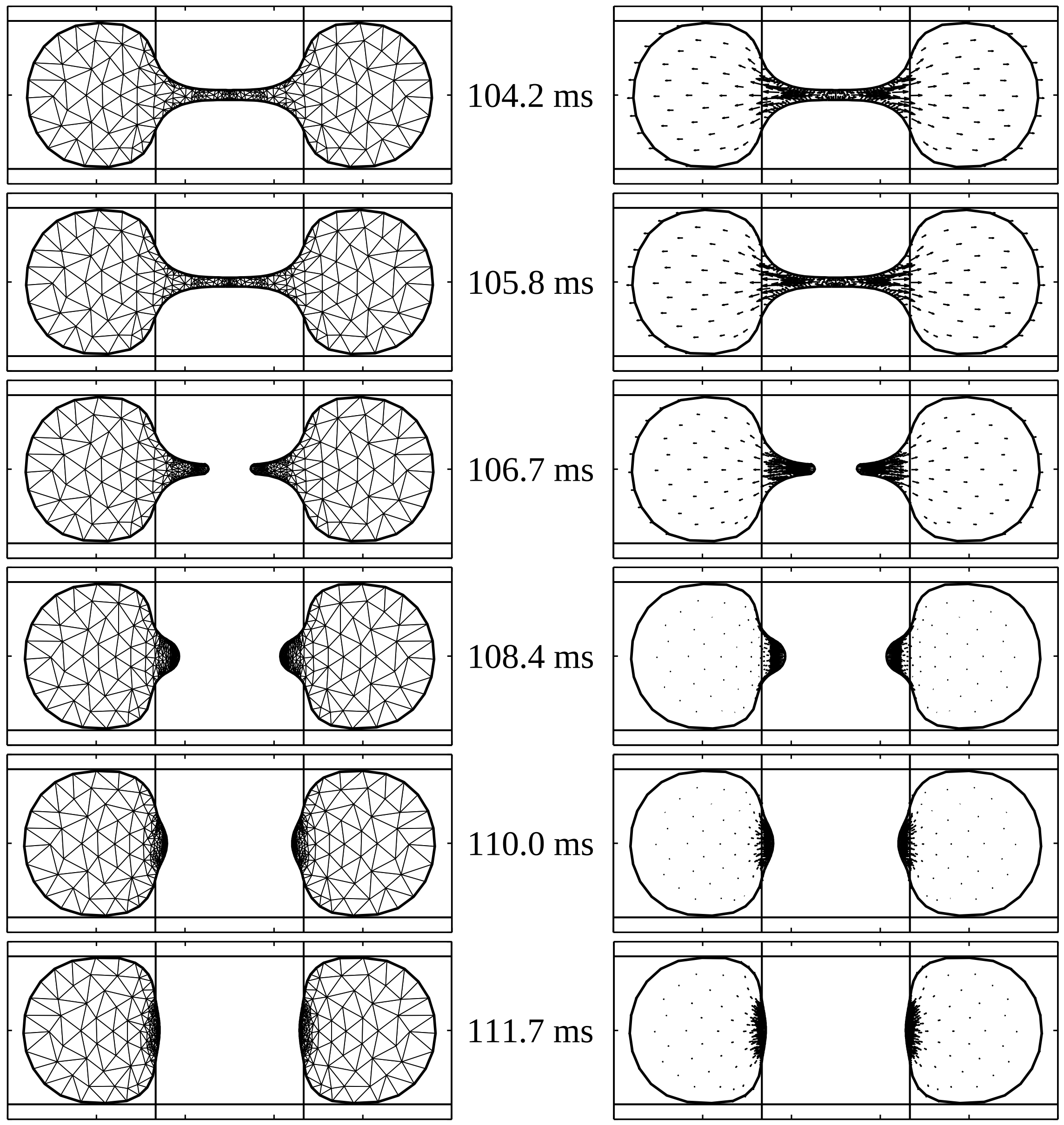


Figure 3.8: Sequence of simulation snapshots for the case of $d_{neck} = 2 \times 10^{-2}$. Snapshots are a continuation of those shown in Figure 3.7. Left column shows the mesh; right column shows the velocity field. Rows correspond to instants in time. The topological change is rather coarse because of the large value of d_{neck}.

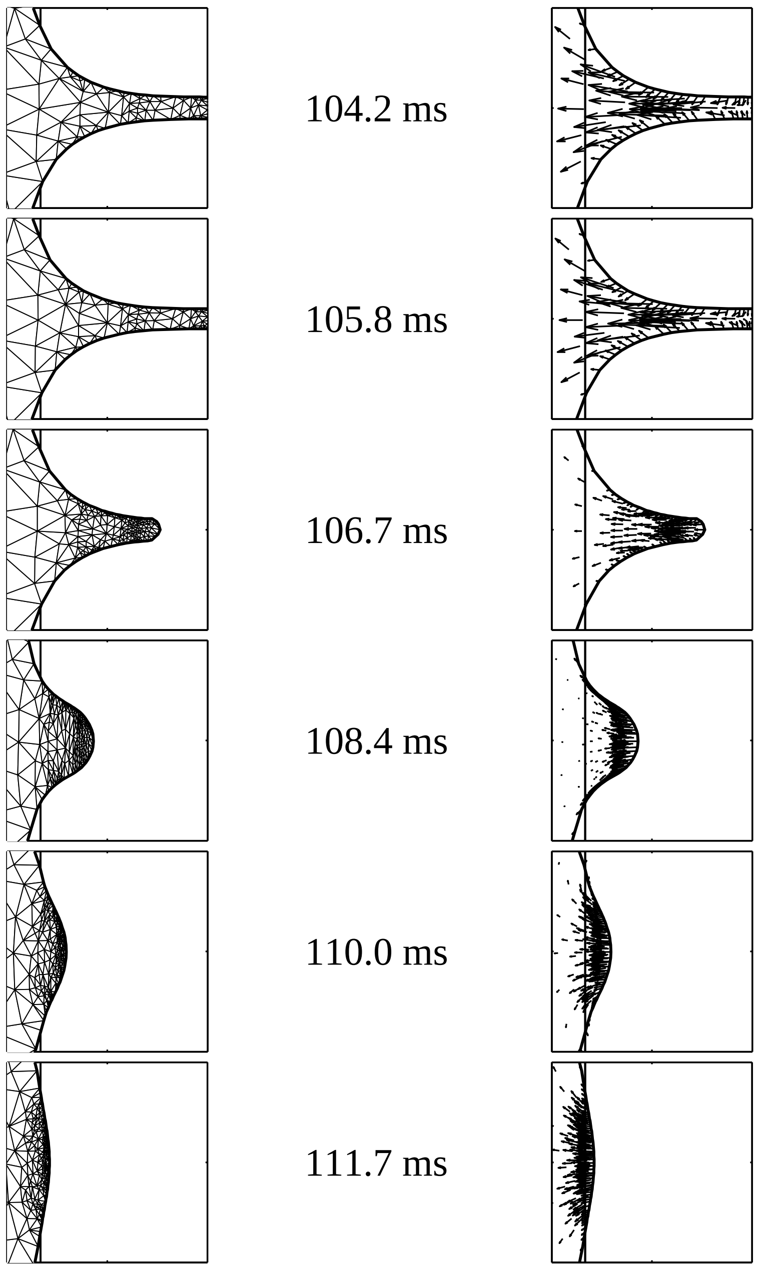


Figure 3.9: Zoom-in (left-side) of the simulation shown in Figure 3.8 ($d_{neck} = 2 \times 10^{-2}$) with same format. The neck becomes thinner until its thickness drops below d_{neck} and triggers the topological change routine of Chapter 2. Immediately following the pinch, the surface tension forces of the droplet act to smooth out the high curvature regions of the pinched neck. Eventually, the two droplets come to rest on the left and right electrode pads. The other pinched region on the right side is similar.

adjusting the boundary mesh after the topological change.

3.2.2 EWOD Pinching - $d_{neck} = 10^{-3}$

Figure 3.11 shows a sequence of snapshots that is a continuation of the simulation shown in Figure 3.7 with $d_{neck} = 10^{-3}$. The dynamics of the topological change in this example are different from the previous section. Here we obtain a pinch in two places and a remaining satellite drop in between the two larger droplets.

In Figure 3.12, we show a more detailed (zoom-in) view of the dynamics of the topological change depicted in Figure 3.11.

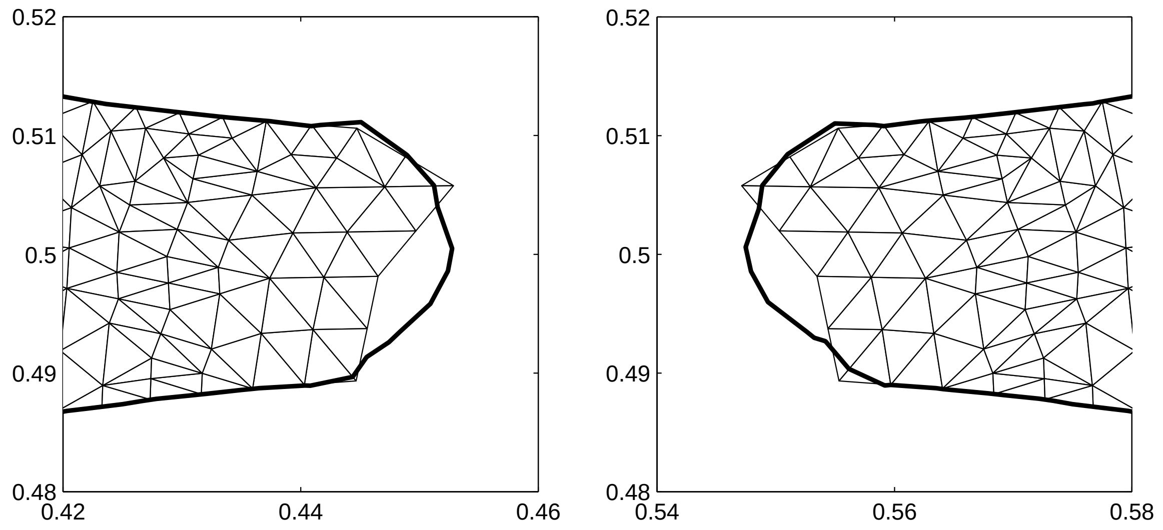
In Figures 3.13 and 3.14, we show an even more detailed (ultra zoom-in) view of the dynamics of the satellite drop depicted in Figures 3.11 and 3.12.

Figure 3.15 shows the effects of our optimization smoothing algorithm for adjusting the boundary mesh after the symmetric double pinch.

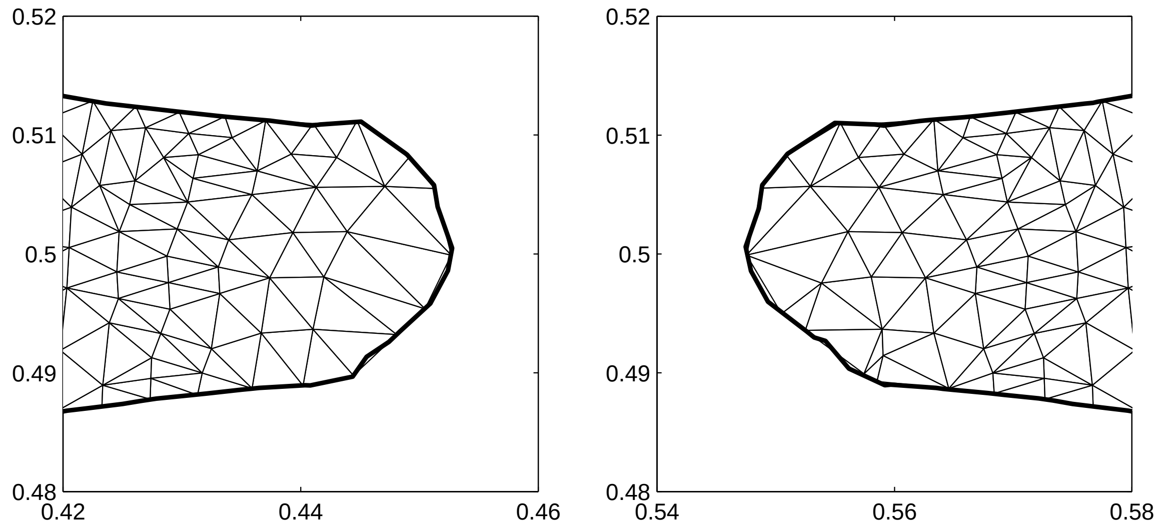
3.2.3 EWOD Pinching - $d_{neck} = 8 \times 10^{-4}$

For this experiment, we lower the minimum thickness just slightly from the case in the previous section. This causes the curvature in the pinched region to be higher, thereby inducing an even larger velocity, which causes the satellite drop to slam together and pinch again! In this section, we only focus on the dynamics of the satellite drop (see Figures 3.16 and 3.17) since the rest of the flow is essentially the same as in Section 3.2.2.

Figure 3.18 shows the effects of our optimization smoothing algorithm for



(a) Pinched regions (connected to the 'bulbs' of fluid) before boundary mesh adjustment.



(b) Pinched regions after boundary mesh adjustment.

Figure 3.10: Comparison of the effect of using our optimization method to smooth the boundary ($d_{neck} = 2 \times 10^{-2}$ case). The zero level contour of the level set function (after the topological change) is shown as a thick black curve. The optimization algorithm acts to smooth or compress the boundary mesh towards the zero level contour. Here, only five steps of the optimization algorithm were used.

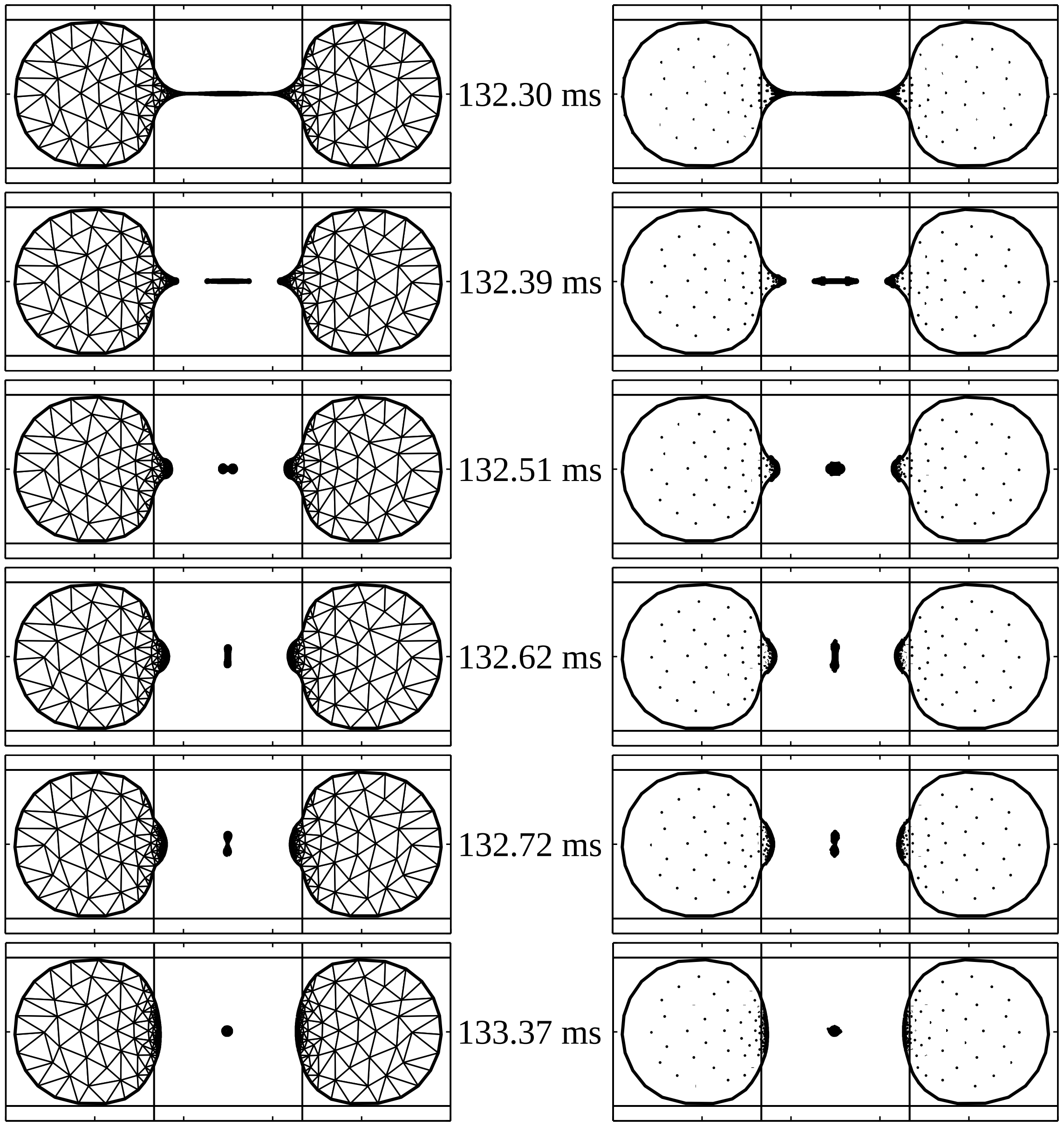


Figure 3.11: Sequence of simulation snapshots for the case of $d_{neck} = 10^{-3}$. Snapshots are a continuation of those shown in Figure 3.7. Format is the same. The topological change here is different than for the case with $d_{neck} = 2 \times 10^{-2}$. The neck undergoes what seems to be an instability similar to that shown in [40] for a Hele-Shaw cell with forcing due to gravity (i.e. this instability has two necking regions). This causes the neck to pinch in two places, which leaves a long thin satellite drop in the center, which then snaps together. A closeup of the pinching region is given in Figure 3.12.

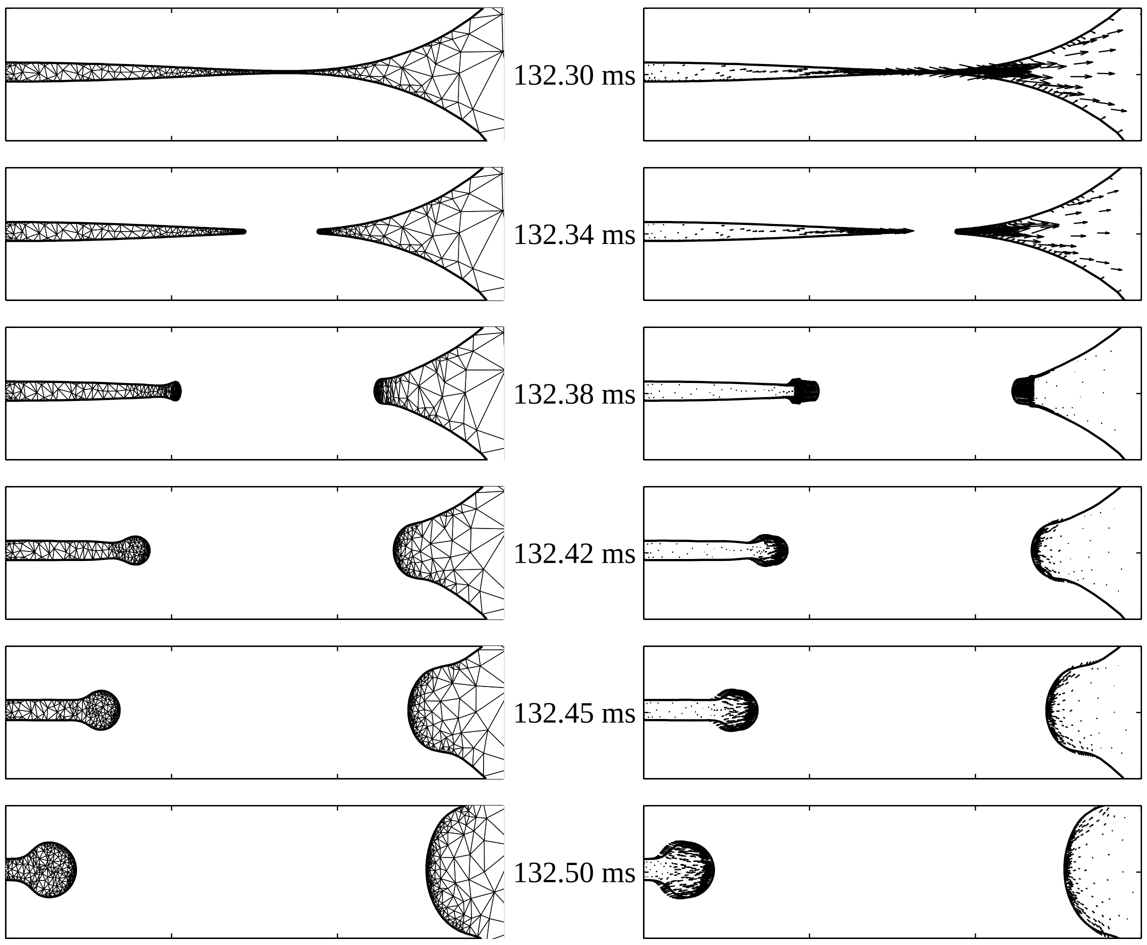


Figure 3.12: Zoom-in (right-side) of the simulation shown in Figure 3.11 ($d_{neck} = 10^{-3}$) with same format. The pinching behavior is symmetric with a similar thinning region on the left side (not shown; see Figure 3.11). After the pinch-off, the surface tension forces of the droplet act to smooth out the high curvature regions of the larger and smaller satellite droplets. The evolution of the larger droplet is similar to that shown in Section 3.2.1. The satellite drop starts to bulge at the ends because of the high velocities induced by the high curvature there. Figures 3.13 and 3.14 focus on the evolution of the satellite drop.

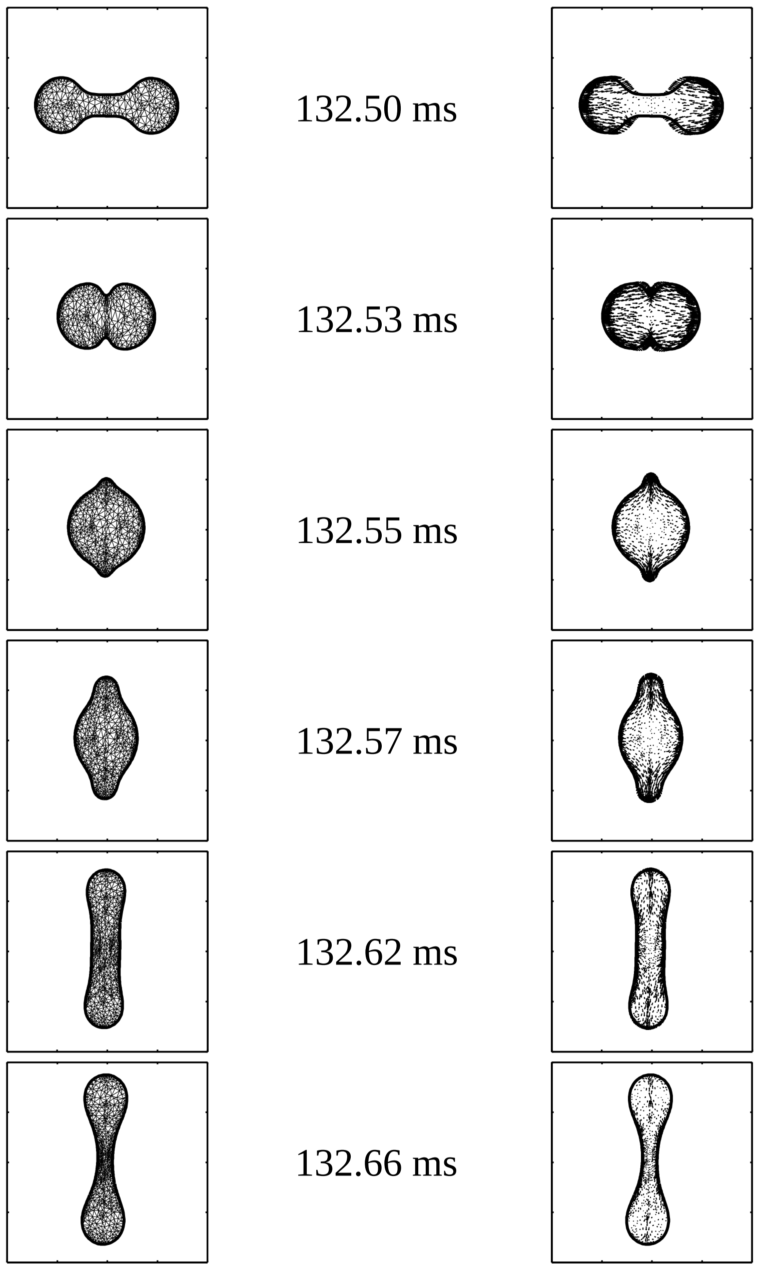


Figure 3.13: Ultra zoom-in of the satellite drop shown in Figure 3.11 ($d_{neck} = 10^{-3}$) with same format. Because of the high velocities induced by the high curvature after the pinch-off, the satellite drop rapidly contracts and slams together. This causes it to deform into a dumbbell shape because of an inertial term in the model that governs EWOD flow (i.e. there is a $\partial_t \vec{u}$ term in the model). Simulation frames continue in Figure 3.14.

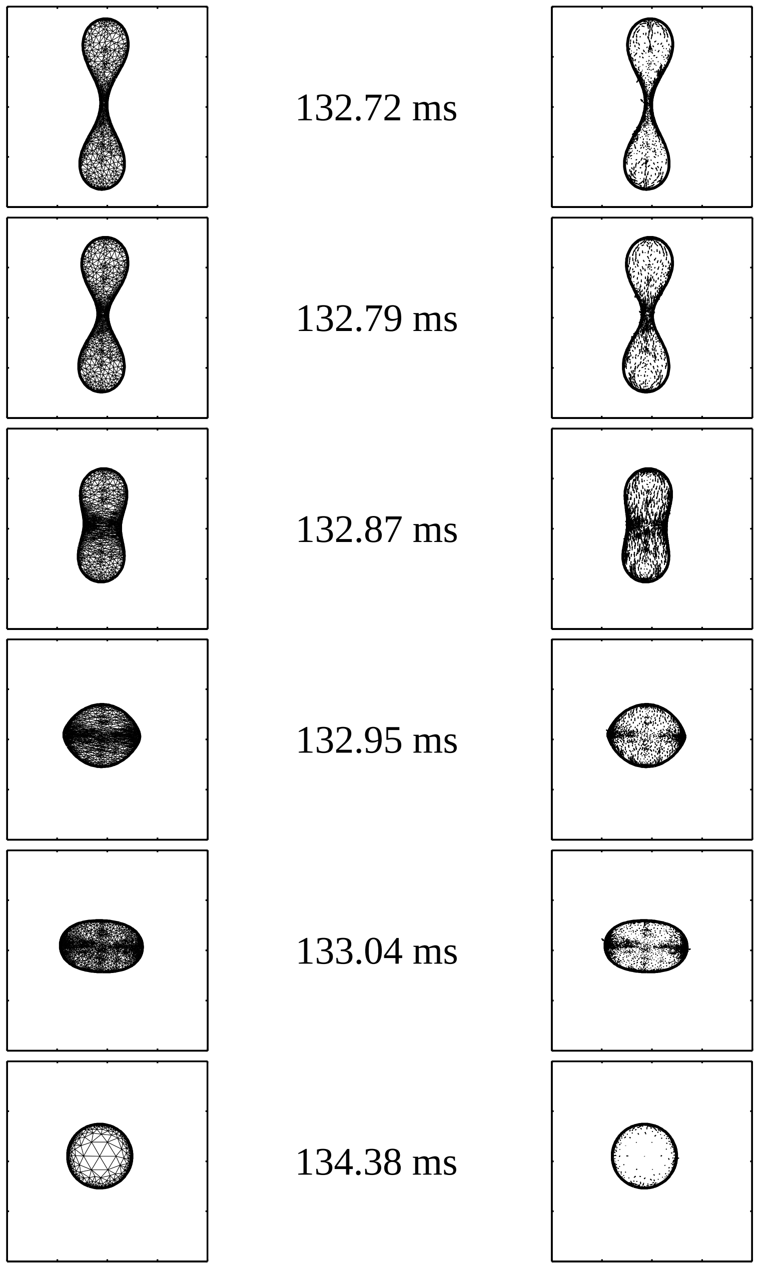
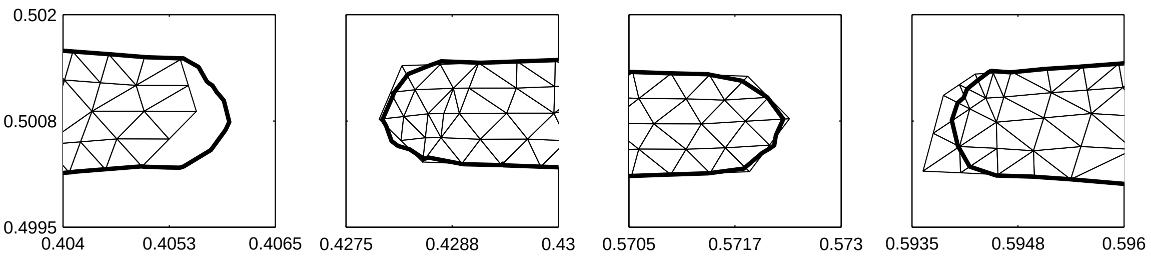
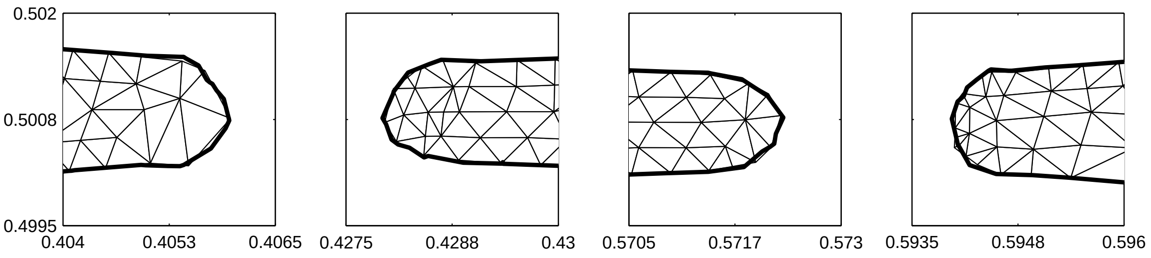


Figure 3.14: Ultra zoom-in of the satellite drop shown in Figure 3.11 ($d_{neck} = 10^{-3}$); continuation from Figure 3.13. The dumbbell eventually relaxes into a circular shape.



(a) Pinched regions before boundary mesh adjustment.



(b) Pinched regions after boundary mesh adjustment.

Figure 3.15: Comparison of the effect of using our optimization method to smooth the boundary ($d_{neck} = 10^{-3}$ case). The left (right) half of the figure coincides with the left (right) pinch region (see Figure 3.11). The zero level contour of the level set function (after the topological change) is shown as a thick black line. The optimization algorithm acts to smooth or compress the boundary mesh towards the zero level contour. Here, only five steps of the optimization algorithm were used.

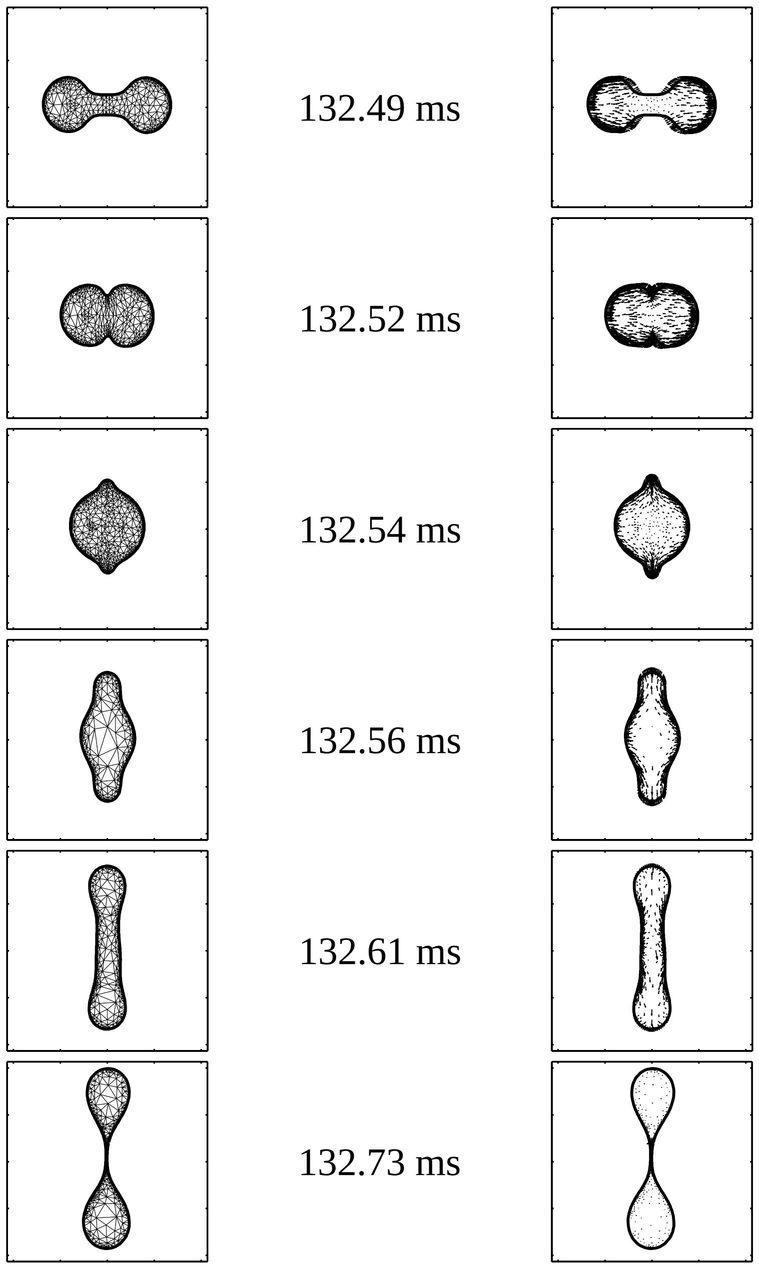


Figure 3.16: Zoom-in of the satellite drop (same figure format) shown in Figure 3.11, except the minimum thickness is $d_{neck} = 8 \times 10^{-4}$. High velocities induced by the high curvature after pinch-off cause the satellite drop to rapidly contract and slam together. The drop continues to deform and pinch again, splitting into two satellite drops. The sequence of snapshots continues in Figure 3.17.

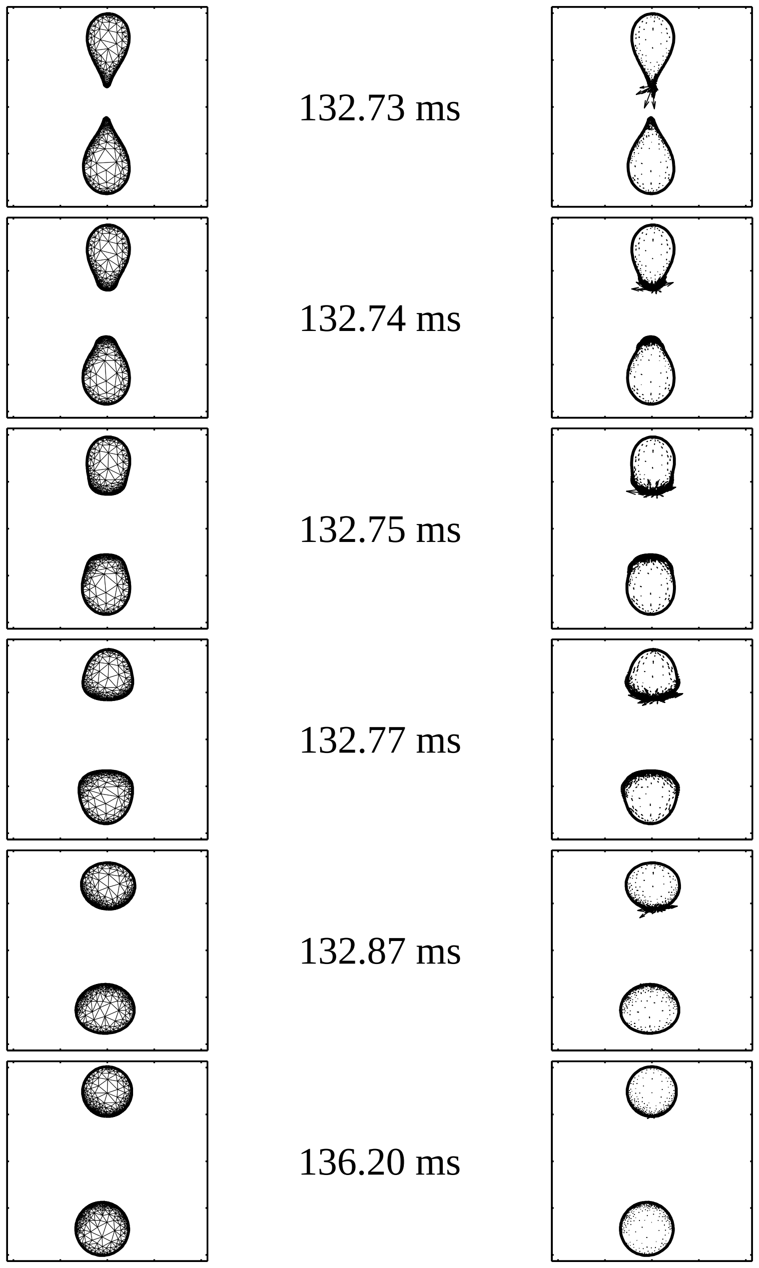


Figure 3.17: Continuation of the simulation shown in Figure 3.16 ($d_{neck} = 8 \times 10^{-4}$). After the second pinch event, the two smaller droplets deform and move away from each other. This pinching experiment demonstrates the ability of our method to track the topology of the evolving droplet through multiple pinches.

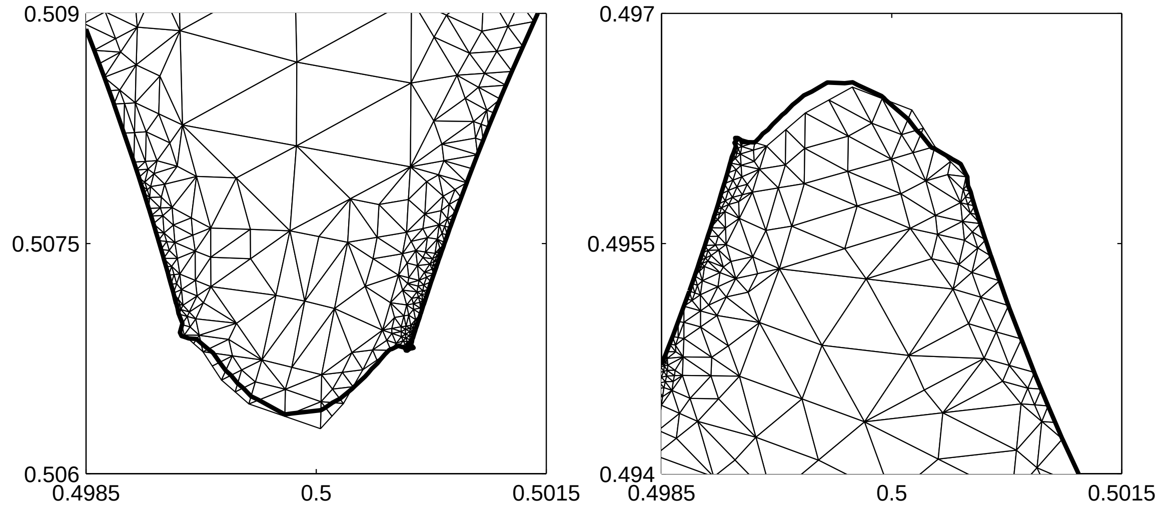
adjusting the boundary mesh after the second topological change (i.e. the second pinch of the dumbbell).

3.2.4 Bulk Fluid Flow Versus d_{neck}

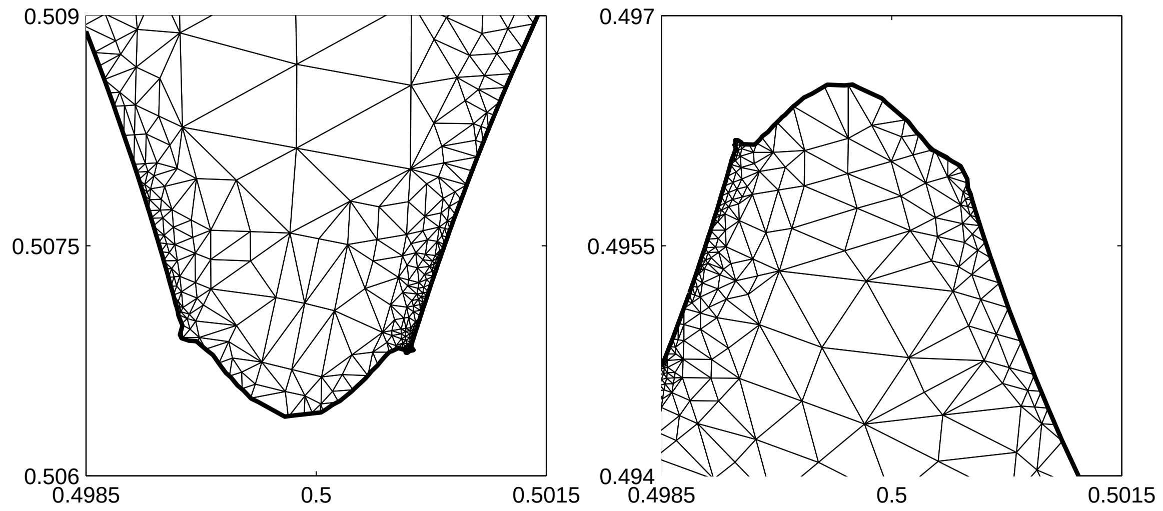
There is some concern over the effect that d_{neck} may have on the bulk fluid flow away from the pinch. In this section, we compare velocities in the two main 'bulbs' of fluid with respect to varying d_{neck}.

First, we define our *region of interest* to be the droplet domain that overlaps the left and right electrodes of the EWOD device (i.e. the electrodes with 25 V applied; see Figure 3.6). In Figure 3.19, we have plotted the maximum velocity and average velocity in the region of interest as a function of time, for all three cases of $d_{neck} = 2 \times 10^{-2}, 10^{-3}, 8 \times 10^{-4}$. Our goal is to check if there is a significant difference in these quantities when d_{neck} is varied. This comparison is rather approximate, but is certainly reasonable for a first glance.

From the data in Figure 3.19, the main discrepancy in the measured velocity quantities occurs because of the different times of pinch-off for the different minimum neck sizes d_{neck}. But this discrepancy disappears after about 10-15 ms, after which all three data plots follow the same trend. This implies that the size of d_{neck} can cause momentary discrepancies in the droplet evolution at times of topological change, but that the discrepancies eventually decay. This is reasonable given the sensitive nature of pinching events, and the significant damping factor that is present in EWOD driven flow.

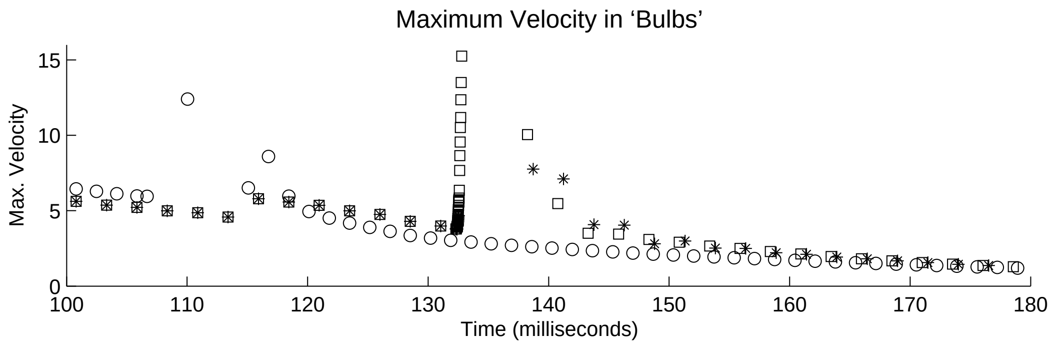


(a) Pinched regions before boundary mesh adjustment.



(b) Pinched regions after boundary mesh adjustment.

Figure 3.18: Comparison of the effect of using our optimization method to smooth the boundary ($d_{neck} = 8 \times 10^{-4}$ case) for the second pinch event. The zero level contour of the level set function (after the topological change) is shown as a thick black line. The optimization algorithm acts to smooth or compress the boundary mesh towards the zero level contour. Here, only five steps of the optimization algorithm were used.



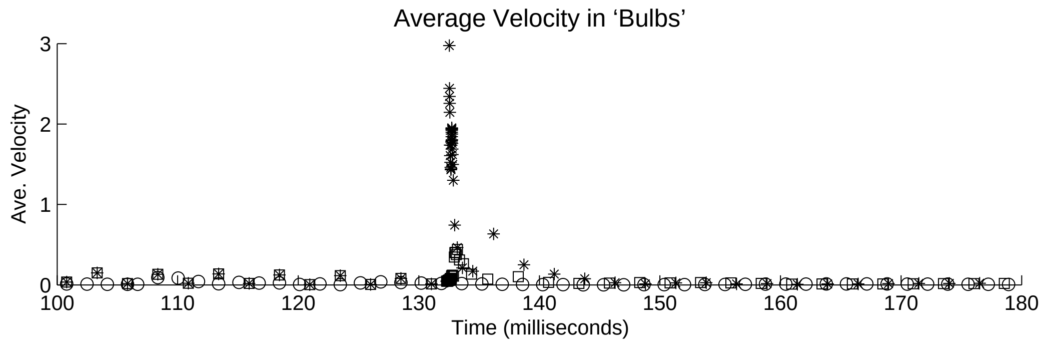


Figure 3.19: Comparison of flow in the 'bulbs' (i.e. away from the necking region) for three different values of d_{neck}. ($\circ$) denotes data points for the $d_{neck} = 2 \times 10^{-2}$ case, ($\Box$) denotes data points for the $d_{neck} = 10^{-3}$ case, ($*$) denotes data points for the $d_{neck} = 8 \times 10^{-4}$ case. The maximum velocity for $d_{neck} = 2 \times 10^{-2}$ spikes at about 108 ms which is about the time of pinch-off in that case (there is no spike for the average velocity). Both max and average velocity in the other two cases spikes later (at 133 ms), which corresponds to their delayed pinch-off time. Other than at the pinch-off times, these velocity quantities appear to follow the same trend.

But this analysis is very rudimentary and depends on the particular physical problem being simulated. It is certainly possible to envision a physical situation in which the size of d_{neck} can drastically affect the results of the simulation. But, again, this is expected given the (violent) nature of fluid pinching.

3.3 Joining of Droplets by Surface Tension

In this last experiment, we use the EWOD simulation without any electrical forcing. Hence, the flow is purely due to surface tension. This example shows how our method deals with connecting or joining droplets. The initial domain shape (and mesh) is given in Figure 3.20.

Figures 3.21, 3.22, and 3.23 show several time-frames of a finite element simulation for the surface tension example given in Figure 3.20. As the donut shaped droplet approaches the smaller drop, the boundary mesh begins to refine because of our adaptive meshing routine. This is because the mesh is extended at every time-step using the program Triangle. A thin neck of 'air' develops between the two droplets. Eventually, the thickness of the neck drops below the minimum thickness of $d_{neck} = 10^{-3}$ that we set and a topological change is executed.

In Figures 3.24, 3.25, 3.26, and 3.27, we show a closeup of the evolution of the thin neck of 'air'. Note how the high curvature regions get smoothed out by the surface tension effect.

Figure 3.28 shows the effects of our optimization smoothing algorithm for adjusting the boundary mesh around the cusp region.

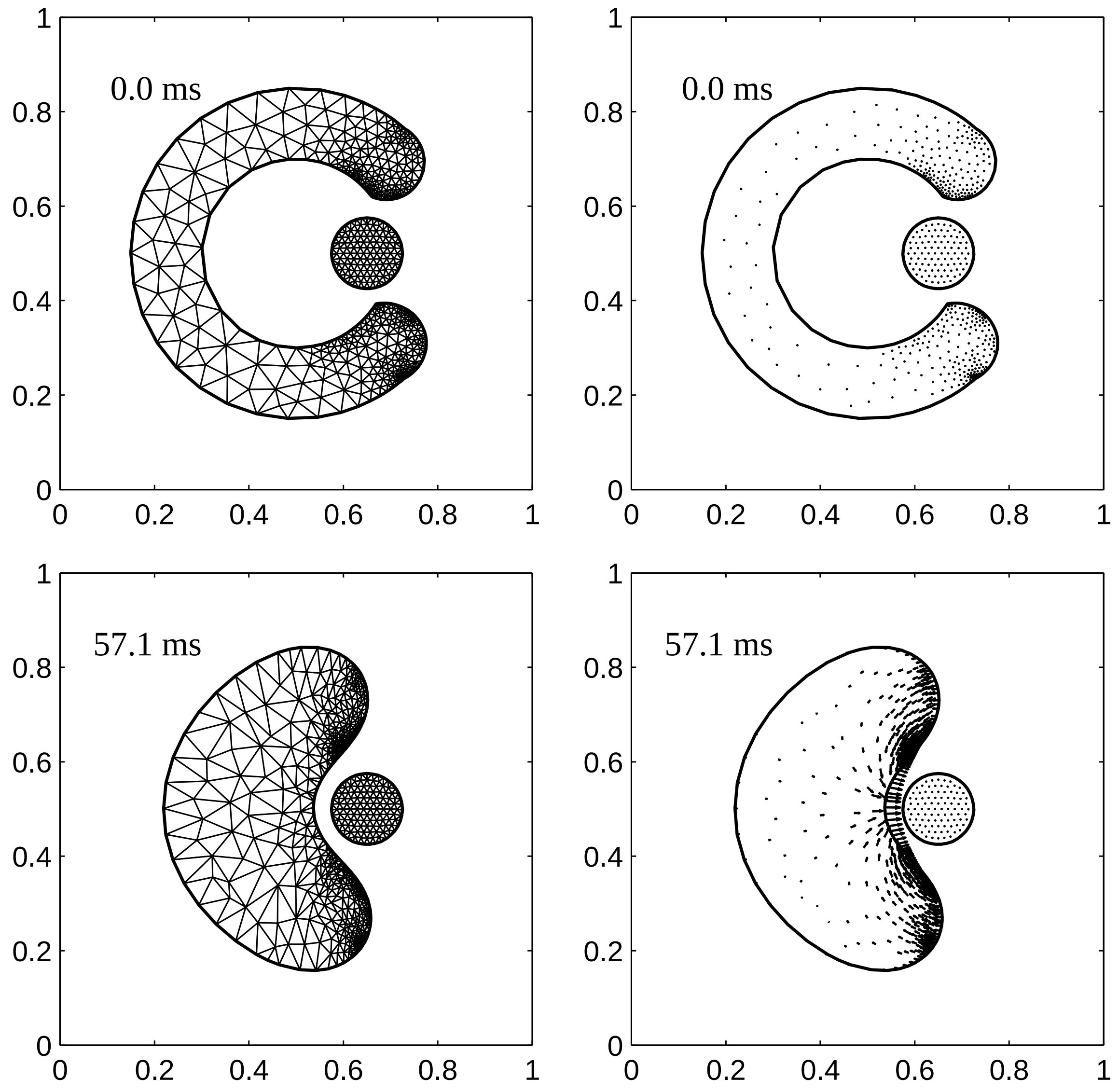


Figure 3.20: Initial 'donut' droplet surrounding another smaller droplet (first row), with a deformed version shown at a later time (second row). The first column shows the finite element mesh for the domain of the droplet and the second column shows the droplet boundary and velocity field. Since the larger droplet is not in a circular shape, the surface tension effect causes it to move such that it minimizes its boundary length. This eventually forces the donut to come into contact with the smaller droplet, which causes a topological change (a joining event).

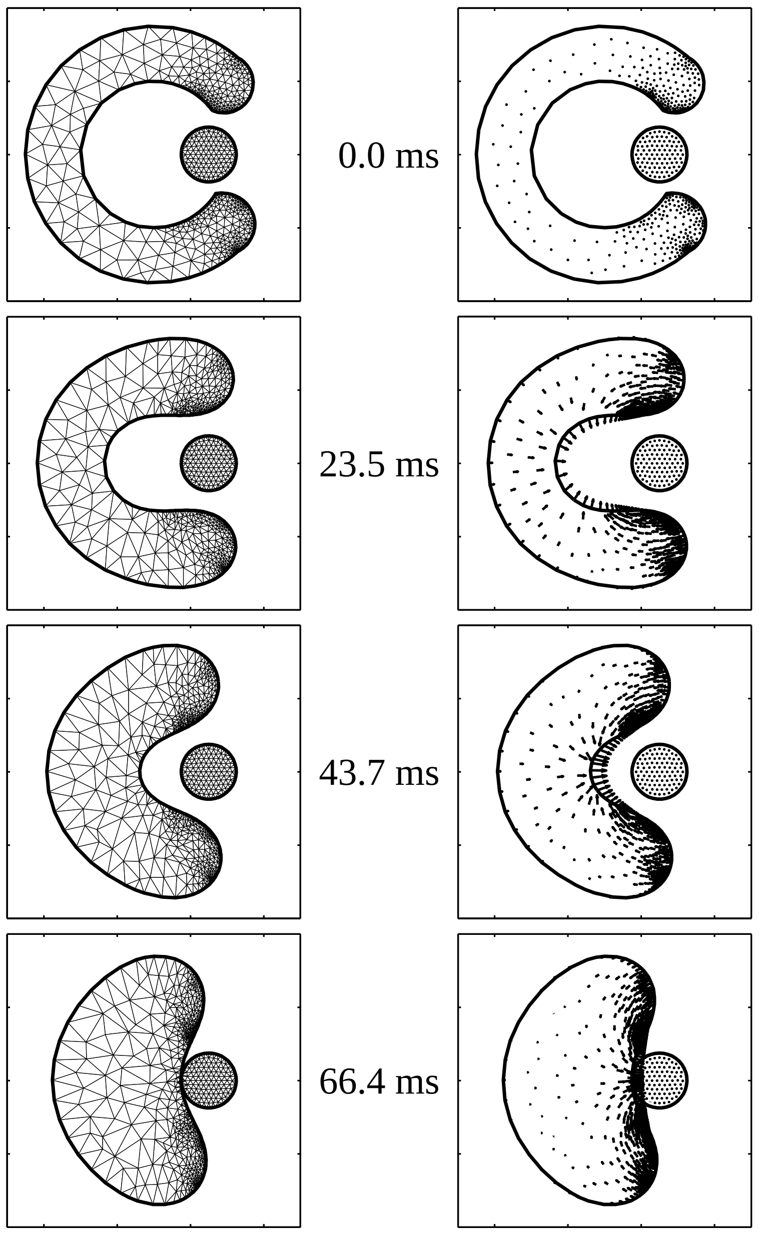


Figure 3.21: Sequence of snapshots for the example in Figure 3.20. Left column shows the mesh; right column shows the velocity field. Rows correspond to instants in time. The outer droplet slowly deforms because it is only driven by surface tension (i.e. no electrical forcing is present). As the two droplets come closer, the boundary mesh in those regions adapts because of the meshing program ‘Triangle’. The last frame is just before the droplets connect ($d_{neck} = 10^{-3}$). See Figure 3.22 for a continuation.

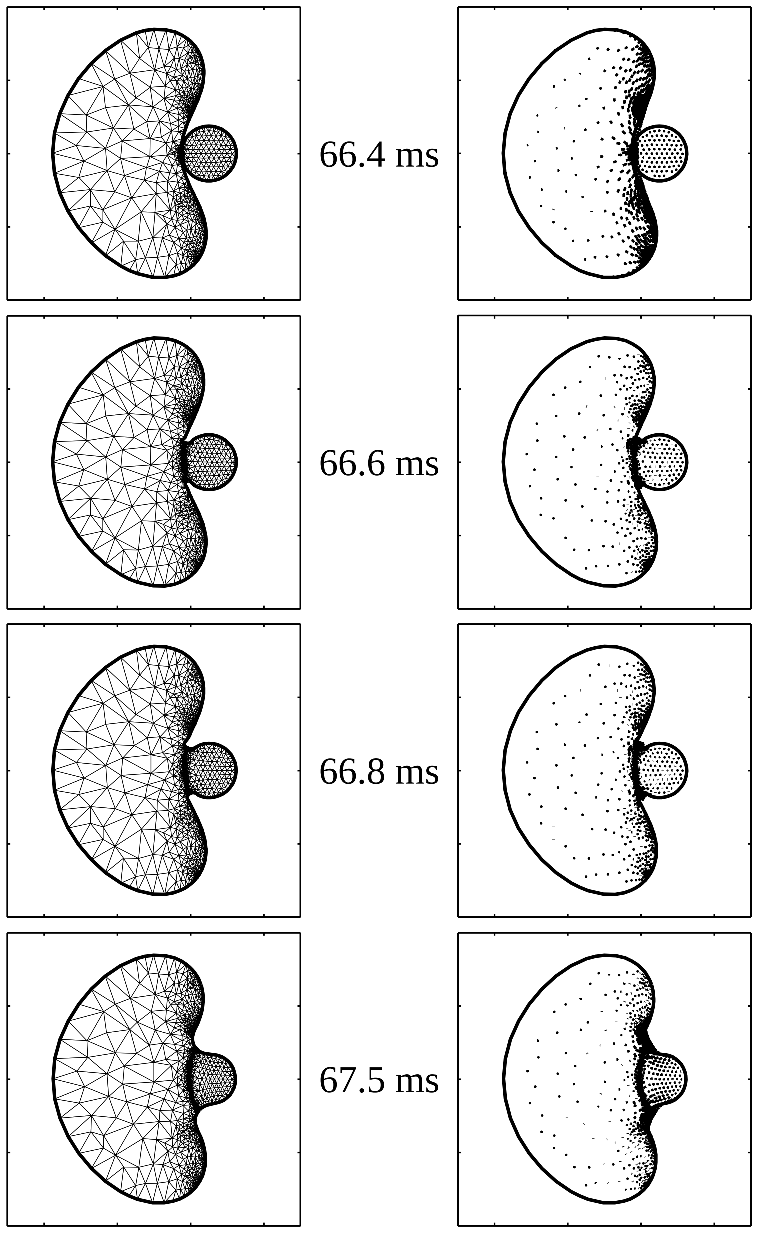


Figure 3.22: Sequence of simulation snapshots for the example given in Figure 3.20 (continuation from Figure 3.21 with same format). The first frame is immediately after the connection. Eventually, the high curvature region gets smoothed out. See Figure 3.23 for a continuation.

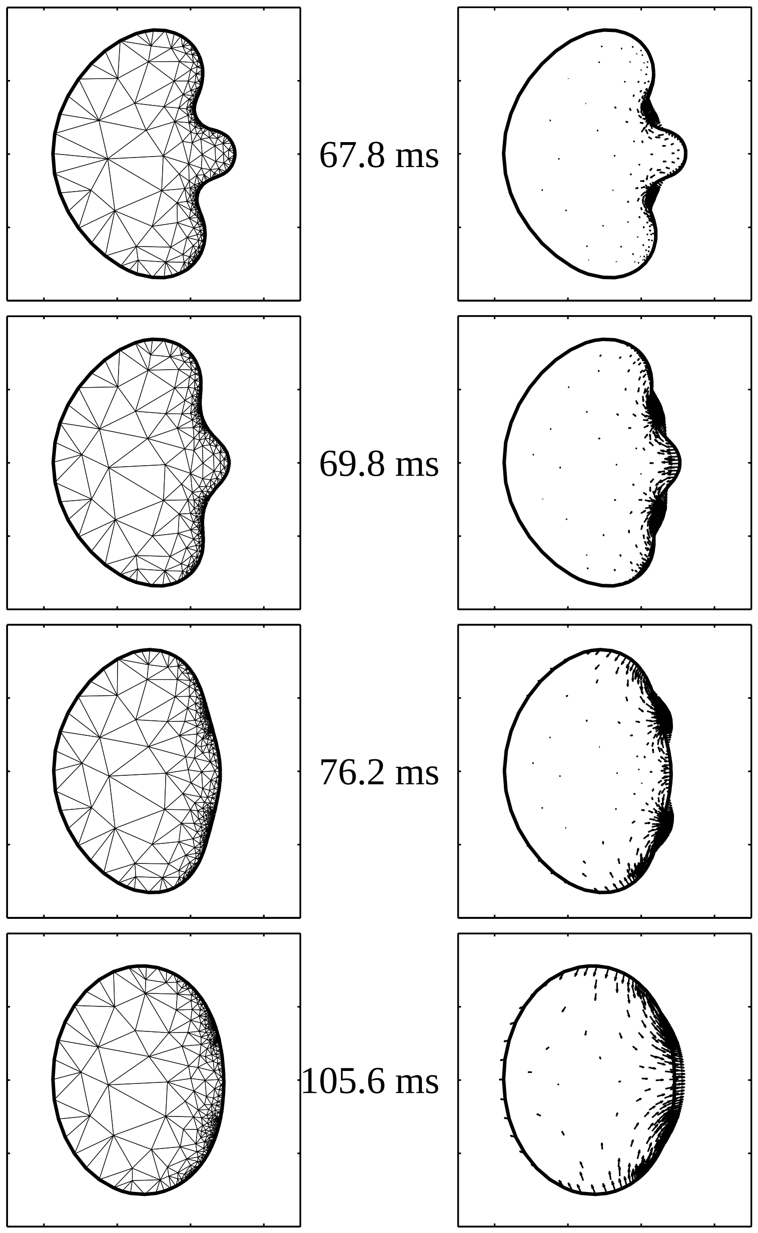


Figure 3.23: Sequence of simulation snapshots for the example given in Figure 3.20 (continuation from Figure 3.22 with same format). The final state of the joined droplet is a circular shape (not shown).

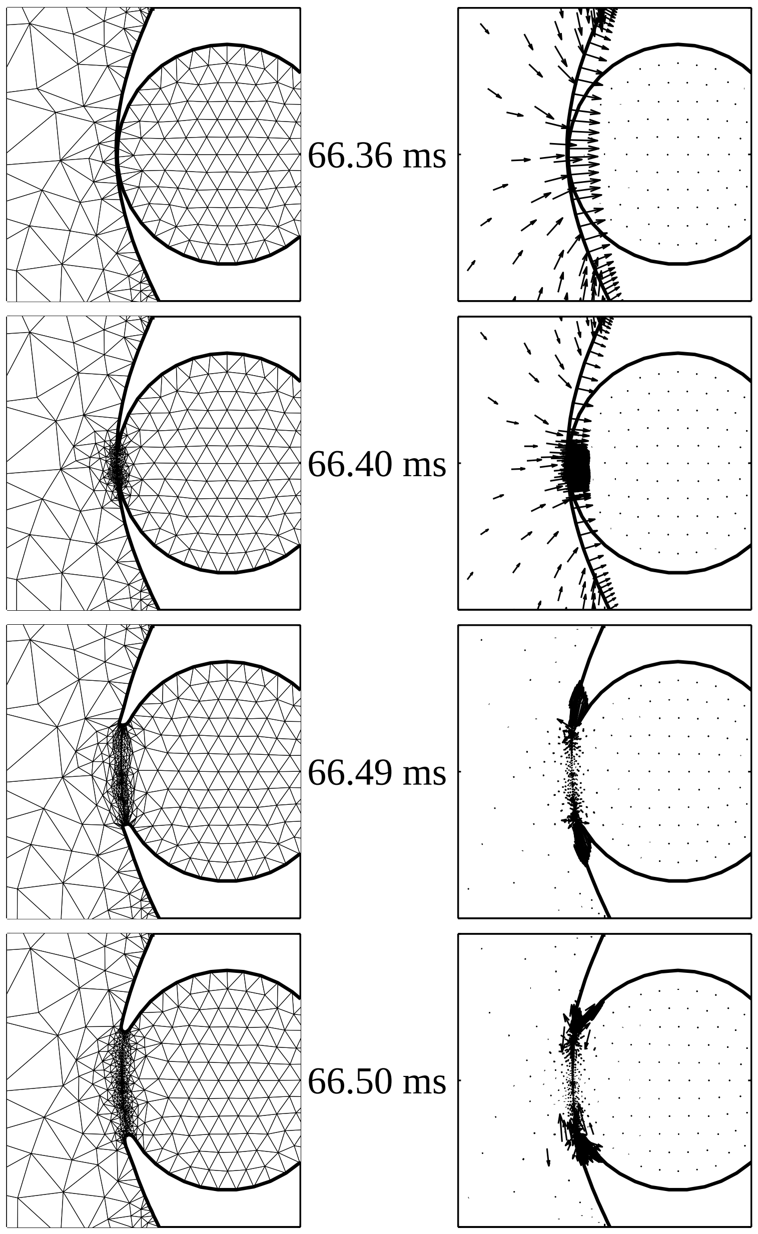


Figure 3.24: Closeup of droplet evolution for the example given in Figure 3.20. First frame is immediately before topological change. When the 'donut' shaped droplet forms an 'air' neck with a thickness less than $d_{neck} = 10^{-3}$, the topological change routine is executed. After joining, the sharp cusp that is formed proceeds to smooth itself out because of the surface tension effect. See Figure 3.25 for a continuation.

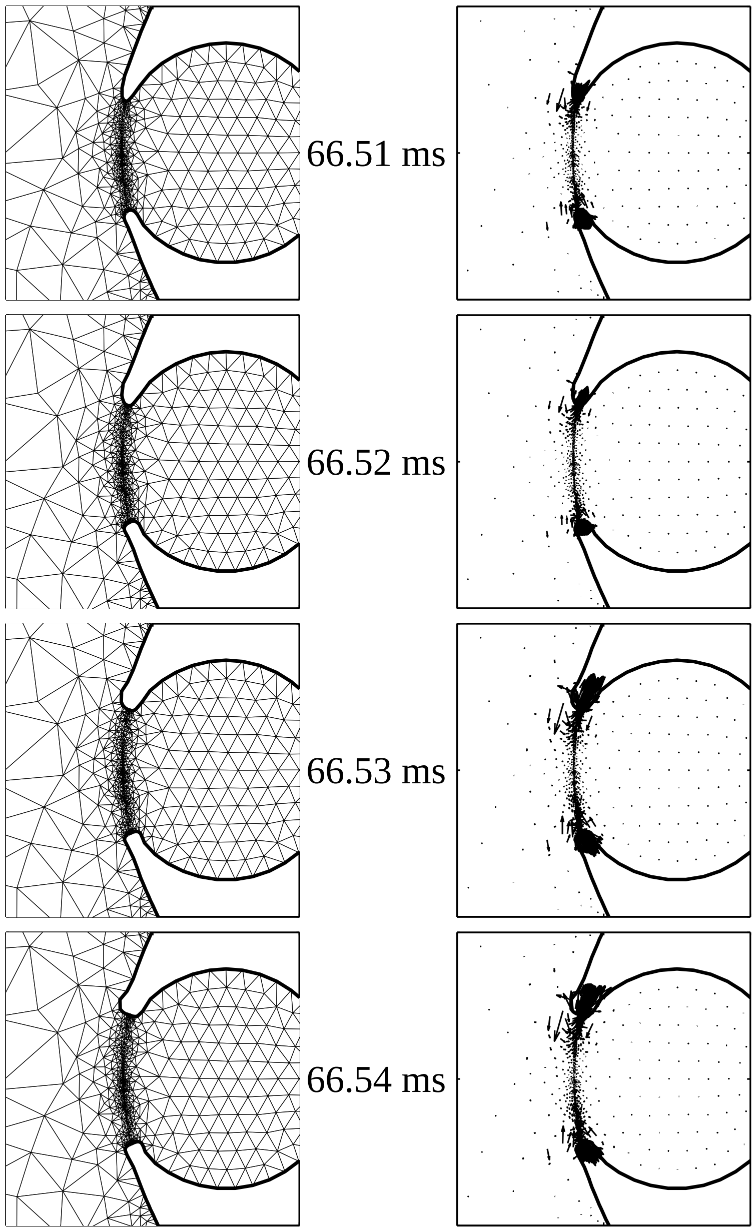


Figure 3.25: Continuation from Figure 3.24 for the 'Donut' example. The sharpness of the cusp begins to be smoothed out. See Figure 3.26 for a continuation.

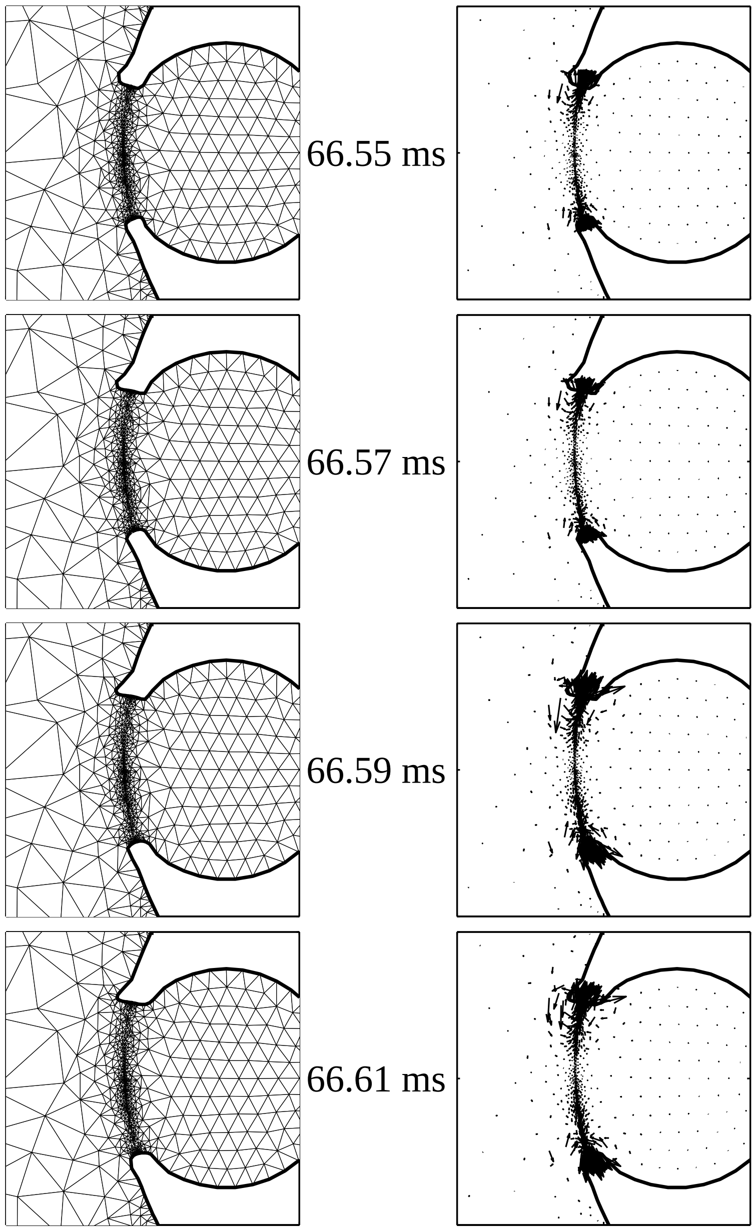


Figure 3.26: Continuation from Figure 3.25 for the 'Donut' example. Here, you can see a small numerical artifact where the boundary 'bunches' up on the top part of the connection region. This is because of inadequate meshing in the cusp region, which we discuss in Chapter 4. See Figure 3.27 for a continuation.

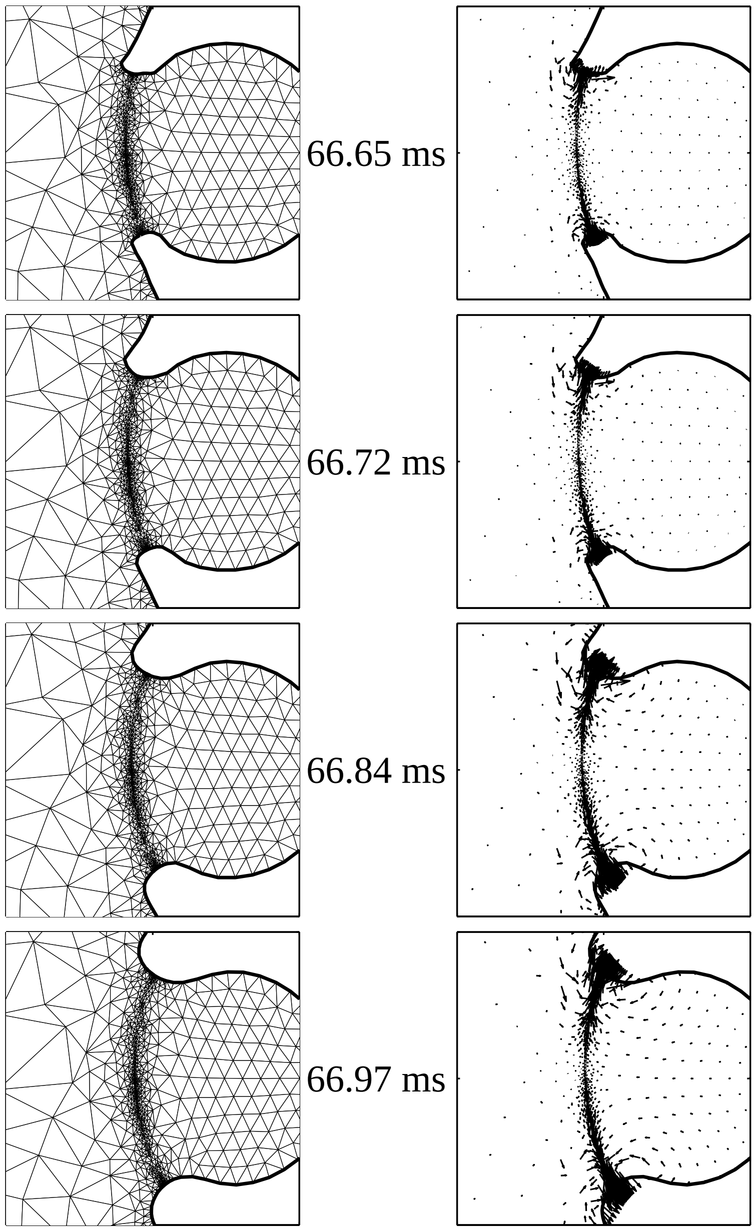
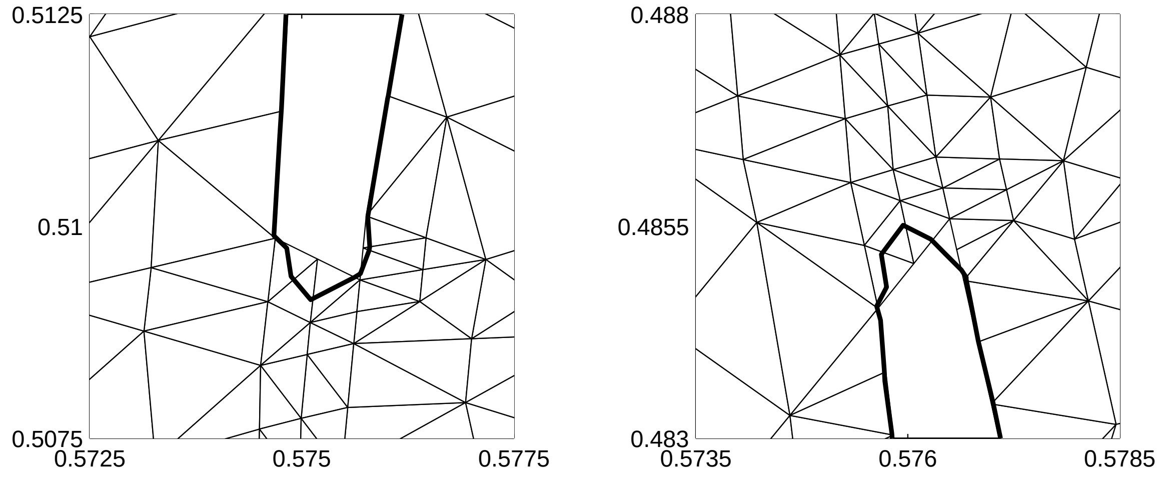
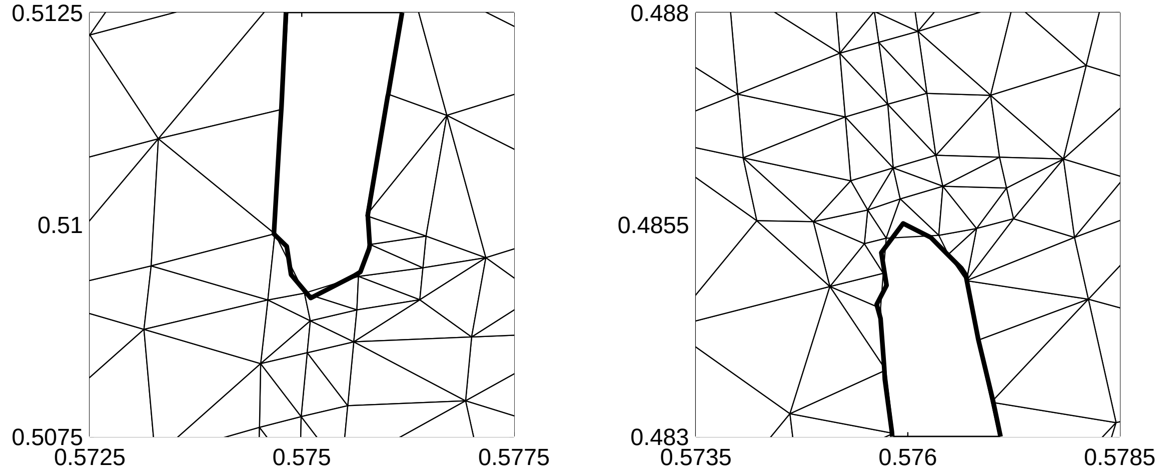


Figure 3.27: Continuation from Figure 3.26 for the 'Donut' example. The high curvature regions continue to be smoothed out. As can be seen, the method we propose is able to handle this severe topological change.



(a) Zoom-in of connecting regions before boundary mesh adjustment.



(b) Zoom-in of connecting regions after boundary mesh adjustment.

Figure 3.28: Comparison of the effect of using our optimization method to smooth the boundary ($d_{neck} = 10^{-3}$). The top cusp of the connecting region is shown on the left; the bottom cusp is on the right. The zero level contour of the level set function (after the topological change) is shown as a thick black line. The optimization algorithm acts to smooth or compress the boundary mesh towards the zero level contour. Here, only five steps of the optimization algorithm were used.

Chapter 4

Conclusions

We have presented a method for enabling meshes deforming in a Lagrangian way to undergo 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, a mesh smoothing step using a shape differential optimization technique is used to improve boundary mesh conformity to the zero level contour of the level set function.

One issue with our method is the need to detect when a topological change is happening. In some cases, the pre-cursor to the topological change can be quite smooth and gentle. This can be hard to detect if the velocity field were used to indicate a topological change. In contrast, our method of looking for 'thin' regions is not ambiguous, but can lead to spurious topological changes if the physical problem being simulated exhibits a lot of 'thin' features. This can be compensated for by choosing a minimum neck thickness d_{neck} that is smaller than the expected thin features.

Another issue has to do with resolving the dynamics of topological changes. For example, in Figure 3.26 one can see the boundary of the droplet start to bunch up in the region around the cusp because of the high velocities (and gradients) present. This is also due to the sudden change in mesh size around parts of the cusp. A solution for this would be to have an adaptive mesh algorithm that can accommodate geometric information from the boundary and the PDE solution in the interior (i.e. the velocity).

More improvements could be made, including having a method to adapt the mesh boundary (in some sense) when doing the 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. Other improvements include better methods for solving the level set equation, which we understated in our exposition. This was not critical for our demonstrations, since we only use one time-step in updating the level set function. However, this may be more critical for implementing our method in 3-D.

In comparison to the literature, we first look at the work by John Strain [3]. His method is mostly a level set method, with an explicit contouring algorithm for extracting the interface shape at each time-step. He uses the explicit construction to improve the calculation of certain geometric quantities and as a better way to capture the overall geometry of the interface motion. But his method still handles geometric terms, such as curvature, in an explicit way (which we avoid). And his method requires an explicit reconstruction at every simulation step, as opposed to our method which only requires mesh reconstruction when there is a topological change. Despite this, his method is an interesting option, especially given that he has shown it to work in 3-D.

Next, we look at Ron Fedkiw's work in [7]. The application here does not involve topological changes, but a way to construct tetrahedral meshes of implicit surfaces that are represented by the zero level set of a scalar function. Their application involves smoothing (or compressing as they say) of an explicit mesh onto the zero level set (analogous to our smoothing step). The main difference here is that their method for moving the mesh boundary is not a variational one like ours. They use the level set function directly, and its gradient, and compute explicitly the direction for moving the boundary vertices, which is followed by a method for moving the interior vertices by using elasticity equations or a mass-spring system. Another related aspect of Fedkiw's work is in [34], 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.

Lastly, it is our hope that our method may be plausible in 3-D simulations. The level set update and mesh smoothing, in principle, generalize to 3-D. But our method is dependent on having adequate tools for mesh refinement and manipulation, which are open questions for 3-D problems. Of course, any problem that has large deformations of an explicit mesh will require good meshing tools anyway. Hence, our method may be used in those cases.

It should be noted that doing topological changes in 3-D can be quite overwhelming. It is certainly possible that there are unforseen drawbacks to our method due to the highly complicated nature of topological changes in 3-D. But this is an issue for *any* method when solving a problem where the geometry affects the solution in a significant way.

Appendix A

Symbol Definitions

On the following page is a list of symbols and definitions used in this thesis.

Table A.1: Symbol Definitions

Symbol	**Definition**	**Symbol**	**Definition**
Ω	computational domain	Γ	$\Gamma := \partial\Omega$
$\Omega_{outside}$	outside extension of Ω	$E\Omega$	$E\Omega = \Omega \bigcup \Omega_{outside}$
Γ_E	$\Gamma := \partial E\Omega$	$\Gamma_{outside}$	$\Gamma_{outside} := \Gamma \bigcup \Gamma_E$
$\top_\Omega$	triangulation of Ω	$\top_{E\Omega}$	triangulation of $E\Omega$
$\top_{pinch}$	set of triangles in thin region	R_{pinch}	subset of $E\Omega$ covering thin region
$\tilde{\Omega}$	domain after top. change	$\tilde{\Gamma}$	$\tilde{\Gamma} := \partial\tilde{\Omega}$
$\top_{\tilde{\Omega}}$	triangulation of $\tilde{\Omega}$	T	refers to a triangle in a mesh
$\mathsf{S}_{\tilde{\Omega}}$	bdy. triangle sides of $\top_{\tilde{\Omega}}$	Γ_{S}	set of points defined by $\mathsf{S}_{\tilde{\Omega}}$
$\overline{\mathsf{S}_{\tilde{\Omega}}}$	mesh bdy. after adjustment	$\overline{\top_{\tilde{\Omega}}}$	domain mesh after adjustment
$\vec{x}$	position coordinate	t	time
$\vec{u}$	vector velocity	(u, v)	$\vec{u} = (u, v)$
$\vec{u}_{smooth}$	smooth velocity extension	$\vec{u}_\nu$	normal vector extension
τ	time-step size	ν	outward normal vector
τ_{min}	minimum time-step	τ_{max}	maximum time-step
τ_{pinch}	time-step of level set update	W_τ	wait time after topological change
d_{neck}	minimum neck thickness	ε_0	artificial diffusion, $\varepsilon_0 = \frac{d_{neck}^2}{\tau_{pinch}}$
ϕ	level set function	ϕ^0, ϕ^1	l.s. before and after top. change
$\top_\phi$	triangulation for level set eqn.	$\Psi := \phi^1$	for notational convenience
$\vec{X}$	surface parameterization	$\vec{\varphi}$	vector perturbation of surface
$J(\Gamma)$	cost function $J(\Gamma) := \int_\Gamma \Psi^2$	$dJ(\Gamma, \vec{\varphi})$	shape derivative
$\vec{G}$	weak gradient of Ψ^2	$\top_{\vec{G}}$	triangulation for computing $\vec{G}$

BIBLIOGRAPHY

[1] S. Alben and M. J. Shelley. Coherent locomotion as an attracting state for a free flapping body. In *Proceedings of the National Academy of Sciences USA 102*, pages 11163–11166, 2005.

[2] E. Baensch, P. Morin, and R. H. Nochetto. A finite element method for surface diffusion: the parametric case. *Journal of Computational Physics*, 203:321–343, 2005.

[3] A. W. Bargteil, T. G. Goktekin, J. F. O'Brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1), 2006.

[4] T. Baumgart, S. T. Hess, and W. W. Webb. Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension. *Nature*, 425:821–824, October 2003.

[5] T. D. Blake, A. Clarke, and E. H. Stattersfield. An investigation of electrostatic assist in dynamic wetting. *Langmuir*, 16(6):2928–2935, 2000.

[6] D. E. Breen, S. Mauch, and R. T. Whitaker. 3d scan conversion of csg models into distance volumes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 7–14, New York, NY, USA, 1998. ACM Press.

[7] R. Bridson, J. Teran, N. Molino, and R. Fedkiw. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers*, 21:2–18, 2005.

[8] L. Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *13th International Meshing Roundtable, Sandia National Laboratories*, pages 109–120, 2004.

[9] L. Chen and J. Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, 22(2):299308, 2004.

[10] S. Cho, H. Moon, J. Fowler, S.-K. Fan, and C.-J. Kim. Splitting a liquid droplet for electrowetting-based microfluidics. In *International Mechanical Engineering Congress and Exposition*, New York, NY, 2001.

[11] S. K. Cho, H. Moon, and C.-J. Kim. Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits. *Journal of Microelectromechanical Systems*, 12(1):70–80, 2003.

[12] M. P. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, Upper Saddle River, New Jersey, 1976.

[13] G. Doğan. Topological changes and adaptivity for curve evolution. *in preparation*, 0000.

[14] G. Doğan. *A variational shape optimization framework for image segmentation.* PhD thesis, University of Maryland at College Park, 2006.

[15] G. Doğan, P. Morin, R. H. Nochetto, and M. Verani. Discrete gradient flows for shape optimization and applications. *Computational Methods and Applications in Mechanical Engineering*, to appear.

[16] M. V. Dyke. *An Album of Fluid Motion.* Parabolic Press, May 1982.

[17] G. Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58:603–611, 1991.

[18] J. Eggers. Universal pinching of 3d axisymmetric free-surface flow. *Phys. Rev. Lett.*, 71(21):3458–3460, Nov 1993.

[19] J. Eggers. Singularities in droplet pinching with vanishing viscosity. *SIAM J. Appl. Math.*, 60:1997, 2000.

[20] D. Enright, R. P. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183:83–116, 2002.

[21] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 736–744, 2002.

[22] J. M. Escobar, G. Montero, R. Montenegro, and Rodrguez. An algebraic method for smoothing surface triangulations on a local parametric space. *International Journal for Numerical Methods in Engineering*, 66(4):740–760, 2006.

[23] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, 1998.

[24] R. B. Fair, V. Srinivasan, H. Ren, P. Paik, V. K. Pamula, and M. G. Pollack. Electrowetting-based on-chip sample processing for integrated microfluidics. In *IEEE International Electron Devices Meeting (IEDM)*, 2003.

[25] P. Fast, L. Kondic, M. J. Shelley, and P. Palffy-Muhoray. Pattern formation in non-newtonian hele-shaw flow. *Physics of Fluids*, 13(5):1191–1212, 2001.

[26] R. E. Goldstein, A. I. Pesci, and M. J. Shelley. Instabilities and singularities in hele-shaw flow. *Physics of Fluids*, 10(11):2701–2723, 1998.

[27] J. Gong, S. K. Fan, and C. J. Kim. Portable digital microfluidics platform with active but disposable lab-on-chip. In *Proc. IEEE Conf MEMS*, pages 355–358, Maastricht, The Netherlands, Jan. 2004.

[28] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley. Boundary integral methods for multicomponent fluids and multiphase materials. *Journal of Computational Physics*, 169:302–362, 2001.

[29] W. Kang and U. Landman. Breakup of liquid nanobridges: Molecular dynamics simulations and stochastic hydrodynamics. *Physical Review Letters (Accepted)*, 2006.

[30] W. Kang and U. Landman. Universality crossover of the pinch-off shape profiles of collapsing liquid nanobridges in vacuum and gaseous environments. *Physical Review Letters*, 98(6):064504, 2007.

[31] P. M. Knupp. Algebraic mesh quality metrics. *SIAM Journal of Scientific Computing*, 23:193–218, 2001.

[32] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 457–462, Los Angeles, 2004.

[33] S. Mauch. A fast algorithm for computing the closest point and distance function, 2000.

[34] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *SIGGRAPH, ACM TOG*, 23:385–392, 2004.

[35] R. Montenegro, G. Montero, J. M. Escobar, and E. Rodrguez. Efficient strategies for adaptive 3-d mesh generation over complex orography. *Neural, Parallel Sci. Comput.*, 10(1):57–76, 2002.

[36] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, NY, 2003.

[37] S. A. Sethian. *Level Set Methods and Fast Marching Methods, 2nd Edition*. Cambridge University Press, New York, NY, 1999.

[38] B. Shapiro, H. Moon, R. Garrell, and C. J. Kim. Equilibrium behavior of sessile drops under surface tension, applied external fields, and material variations. *Journal of Applied Physics*, 93, 2003.

[39] M. J. Shelley. Elasticity driven models with time-dependent preferred curvature in modeling locomotion. *Unpublished*, 2005.

[40] M. J. Shelley, R. E. Goldstein, and A. I. Pesci. Topological transitions in hele-shaw flow. *Singularities in fluids, plasmas and optics (Heraklion, 1992), NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, 404:167–188, 1993.

[41] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[42] Y. Sun and C. Beckermann. Diffuse interface modeling of two-phase flows based on averaging: mass and momentum equations. *Physica D*, 198:281, 2004.

[43] A.-K. Tornberg and M. J. Shelley. Simulating the dynamics and interactions of flexible fibers in stokes flows. *Journal of Computational Physics*, 196:8–40, 2004.

[44] S. W. Walker and B. Shapiro. Modeling the fluid dynamics of electrowetting on dielectric (ewod). *Journal of Microelectromechanical Systems*, 15(4):986–1000, August 2006.

[45] Y. Yao, C. S. Koh, and D. 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, 2003.*, volume 2, pages 732–735, 2003.

[46] P. Yue, J. J. Feng, C. Liu, and J. Shen. A diffuse-interface method for simulating two-phase flows of complex fluids. *Journal of Fluid Mechanics*, 515:293, 2004.

[47] E. C. Zachmanoglou and D. W. Thoe. *Introduction to Partial Differential Equations with Applications (Paperback)*. Dover Publications, 1987.

[48] O.-Y. Zhong-can. Elastic theory of biomembranes. *Thin Solid Films*, 393:19–23, 2001.