 FDMP, vol.3, no.1, pp.65-95, 2007

An arbitrary Lagrangian-Eulerian (ALE) method for interfacial flows with insoluble surfactants

Xiaofeng Yang[1] **and Ashley J. James** [1 2]

(*Communicated by John Lowengrub and Mark Sussman*)

Abstract: An arbitrary Lagrangian-Eulerian (ALE) method for interfacial flows with insoluble surfactants is presented. The interface is captured using a coupled level set and volume of fluid method, which takes advantage of the strengths of both the level set method and the volume of fluid method. By directly tracking the surfactant mass, the method conserves surfactant mass, and prevents surfactant from diffusing off the interface. Interfacial area is also tracked. To accurately approximate the interfacial area, the fluid interface is reconstructed using piece-wise parabolas. The surfactant concentration, which determines the local surface tension through an equation of state, is then computed as surfactant mass per interfacial area. The evolution of the level set function, volume fraction, interfacial area, and surfactant mass is performed using an ALE method. The fluid flow is governed by the Stokes equations, which are solved using a finite element method. The surface tension force is included in the momentum equation using a continuum surface stress formulation. To efficiently resolve the complex interfacial dynamics, the grid is adapted at every time step so that the grid near the moving interface is always refined. The method is extendible to 3D, and can be generalized to other types of grids as well.

Keyword: Interfacial flow, Surfactant, Volume of fluid (VOF), Level set, Arbitrary Lagrangian-Eulerian (ALE), Unstructured grid.

[1] Department of Aerospace Engineering and Mechanics, University of Minnesota, 107 Akerman Hall, 110 Union St SE, Minneapolis, MN 55455, USA

[2] Corresponding author. Tel.: +1-612-625-6027; Fax: +1-612-626-1558; Email: ajames@aem.umn.edu.

1 Introduction

Interfacial flows, fluid flows involving two fluids that do not mix, are common in many natural and industrial processes, such as rain drop formation, polymer blending, fiber coating, emulsification, hydrodesulphurization, targeted drug delivery, food processing, and so on. The interfacial dynamics plays an important role in these processes. For example, in polymer blending, the interfacial dynamics between the component melts can strongly affect the development of the blend morphology, which is crucial in determining the material properties of the blend. Surfactants, molecules of a third material distributed in the flow system, can greatly affect the interfacial dynamics because the presence of surfactants can alter (in most cases lower) the local surface tension, and non-uniformly distributed surfactants can result in non-uniform surface tension, which consequently induces a Marangoni force. The Marangoni force is tangent to the interface, and causes interface motion in such a way that the surfactant concentration, and consequently the surface tension, tends to become uniform. In the present paper, the authors only consider insoluble surfactants. So, it is assumed that surfactants only reside on the interface, and are not soluble in either of the bulk fluids.

The numerical simulation of interfacial flows, especially with surfactants, is very complicated because one has to simultaneously solve the flow field, track/capture the interface position, and evolve the surfactant concentration. These three processes are coupled together in the sense that the interface motion and the surfactant evolution directly depend on the underlying fluid flow,

while the interface position and the surfactant distribution in turn affect the fluid motion through the surface tension force and the Marangoni force. Moreover, because the fluid properties are discontinuous across the interface, and the surface forces are singular at the interface, it is more difficult to solve a flow field containing an interface. In addition, evolving the surfactant concentration along a moving deformable surface is a very challenging task, and depending on the methodology being taken, tracking/capturing an interface also requires great expertise.

In the past decades, primary works have been performed in developing numerical methods for tracking/capturing an interface. Many methods have been developed. Examples include the interface tracking type methods such as moving grid methods [Hyman (1984)], front tracking methods [Peskin (1977); Glimm, Grove, Lindquist, McBryan, and Tryggvason (1988); Unverdi and Tryggvason (1992); Glimm, Grove, Li, Shyue, Zeng, and Zhang (1998); Tryggvason, Bunner, Esmaeeli, Juric, Al-Rawahi, Tauber, Han, Nas, and Jan (2001)], and (volume) marker particle methods [Harlow and Welch (1965)], and the interface capturing type methods such as level set methods [Osher and Sethian (1988); Sethian and Smereka (2003); Osher and Fedkiw (2001)], volume-of-fluid (VOF) methods [Hirt and Nichols (1981); Noh and Woodward (1976); Rider and Kothe (1998); Scardovelli and Zaleski (1999)], and phase field methods [Badalassi, Ceniceros, and Banerjee (2003); Kim, Kang, and Lowengrub (2004); Jacqmin (1999); Anderson, McFadden, and Wheeler (1998); Keestra, Puyvelde, Anderson, and Meijer (2003)]. Because the interface tracking methods track the interface motion directly, they are in general more accurate than interface capturing methods. However, in cases where interface topology changes it could be very complicated to explicitly track the interface. In contrast, interface capturing methods can handle such topology changes automatically because the interface is implicitly embedded in a special function field, such as the level set function and volume fraction, the evolution of which does not require a continuous delineation of the interface. Recently, methods that couple two different methods have also been developed. Examples are the coupled level-set/volume-of-fluid method (CLSVOF) [Sussman and Puckett (2000); Sussman (2003); Yang, James, Lowengrub, Zheng, and Cristini (2006)], the hybrid particle level set method [Enright, Fedkiw, Ferziger, and Mitchell (2002)], and the mixed markers and volume-of-fluid method [Aulisa, Manservisi, and Scardovelli (2003)]. A coupled method takes advantage of the strengths of each of the individual methods, and is therefore superior to either of the methods alone.

Significant works have also been performed in solving fluid flows containing an interface. As mentioned above, it is difficult to solve a flow field containing an interface because of the discontinuous fluid properties and the singular surface forces. One approach to overcome these difficulties is to smooth the discontinuity and the singularity, for example, by mollified Heaviside and delta functions, respectively. The mollified surface force can then be directly included in the momentum equation as a body/volume force, reducing the problem to the solution of a usual flow field of a single fluid phase with variable fluid properties and an applied body force. The idea of incorporating a singular force through a mollified delta function dates back to Peskin [Peskin (1977)]. It was originally developed for simulating blood flows in the heart. In the past decades, similar ideas have been introduced to solve interfacial flows in contexts of various interface tracking/capturing methods [Brackbill, Kothe, and Zemach (1992); Unverdi and Tryggvason (1992); Chang, Hou, Merriman, and Osher (1996)]. After [Brackbill, Kothe, and Zemach (1992)], this kind of method has been called the continuum surface force (CSF) method. Because they are easy to implement, CSF formulations have been widely used. However, because the surface force is mollified/spread, resulting in an error in modeling the original singular surface force, these methods are usually not so accurate as the sharp interface methods, which will be addressed next, and intuitively the spreading of the surface force can directly influence the fluid flows near the interface, and consequently the interface mo-

tion.

Another common type of method are the sharp interface methods, in which interface jump conditions are, more or less, directly handled, and thus the singular surface forces remain sharp. So, there is no modeling error in such methods, and consequently, high order accuracy can be achieved. However, directly handling the jump conditions can make the algorithm more complicated than a CSF formulation. To apply the jump conditions, the interface location has to be explicitly known, while in a CSF method this is not always necessary. Examples of sharp interface methods are moving-grid methods [Hyman (1984)], the immersed interface method [LeVeque and Li (1994, 1997); Li (1994)], the ghost-fluid/boundary-capturing method [Fedkiw, Aslam, Merriman, and Osher (1999); Liu, Fedkiw, and Kang (2000); Hou and Liu (2005)], the method of Helenbrook et al. [Helenbrook, Martinelli, and Law (1999)], and the boundary integral method [Hou, Lowengrub, and Shelley (2001); Pozrikidis (2001)].

Although there have been many methods developed for interface tracking/capturing, and the solution of the flow field containing an interface has been tackled since the commencement of the numerical simulation of interfacial flows, works that have incorporated the effects of surfactants are very few. Moreover, most such works have been performed in the context of boundary integral methods. Examples are [Stone and Leal (1990); Milliken, Stone, and Leal (1993); Pawar and Stebe (1996); Eggleton, Tsai, and Stebe (2001); Eggleton, Pawar, and Stebe (1999); Li and Pozrikidis (1997); Yon and Pozrikidis (1998); Pozrikidis (2001, 2004); DeBisschop, Miksis, and Eckmann (2002)]. The authors refer readers to Table I in the review article [Pozrikidis (2001)] for some applications. However, because the boundary integral method can not handle interface topology changes, for example when drop pinching occurs, this method is unable to resolve the dynamics of the interface beyond pinch-off. In addition, boundary integral methods are typically limited to inviscid and Stokes flows only. It is not easy to extend them to Navier-Stokes flows.

Works performed in the context of other methods are much fewer. In [Renardy, Renardy, and Cristini (2002)], Renardy et al. presented a method for interfacial flows with surfactant in the context of a volume-of-fluid method. This method is somewhat special because the formulation depends on the assumption of a linear equation of state. In [James and Lowengrub (2004)], James and Lowengrub presented a surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant, in which the surface tension could be related to the surfactant concentration using any equation of state. The evolution of the surfactant concentration was performed by tracking the interfacial area and the surfactant mass separately. The surfactant concentration was then recovered as the surfactant mass per interfacial area. By directly tracking the surfactant mass, surfactant mass was easily conserved. In [Xu, Li, Lowengrub, and Zhao (2006)], Xu et al. presented a level-set/immersed-interface method for two dimensional interfacial flows with insoluble surfactant. The evolution of the surfactant concentration was performed by solving a convection-diffusion equation using an Eulerian approach, first presented in [Xu and Zhao (2003)]. In this method, the surface diffusion operator was expressed in global derivatives, and the surfactant concentration, which was supposed to be only defined along the interface, was extended off the interface into a narrow band enclosing the interface [Adalsteinsson and Sethian (1999)]. The extended concentration was then evolved in this narrow band, similar to the local level set strategy. Thus, the evolution of the concentration along the interface (zero level set contour) was embedded in the evolution of the extended concentration in the narrow band. One advantage of this method is that the evolution of the extended concentration can be easily performed using standard numerical methods, such as the finite difference and finite volume methods, on the same Eulerian grid as used for the representation of the flow field, avoiding explicitly discretizing the interface. However, due to numerical diffusion, surfactant mass can diffuse off the interface, resulting in surfactant loss. An *ad hoc* method was introduced by the authors to conserve the surfactant mass. Similar works

on evolving quantities along a moving interface in the context of level set methods can be found in [Adalsteinsson and Sethian (2003); Bertalmio, Cheng, Osher, and Sapiro (2001); Harabetian, Osher, and Shu (1996)], although they are not specifically applied to evolve surfactant concentration. In [Ceniceros (2003)], Ceniceros presented an immersed-interface/hybrid-level-set-front-tracking method for the study of the effects of surfactants on capillary waves. A dynamically adaptive front-tracking method was used to represent interfacial quantities (surfactant concentration and surface tension) using surface markers, while a level set approach was employed to update the material properties (density and viscosity) of the flow. The evolution of the surfactant concentration was performed by solving a convection-diffusion equation along the interface using a finite difference approach. In [Lee and Pozrikidis (2006)], Lee & Pozrikidis presented a method which combines Peskin's immersed-interface method with a diffuse-interface approximation. The convection-diffusion equation governing surfactant evolution was solved using a finite volume method. In [Kruijt-Stegeman, van de Vosse, and Meijer (2004); Stegeman (2002)], Kruijt-Stegeman et al. presented a finite-element/marker method. The interface was represented and tracked by marker points. The convection-diffusion equation governing surfactant evolution was solved on the moving interface using a finite element approach.

In the present paper, the authors present an arbitrary Lagrangian-Eulerian (ALE) method to simulate interfacial flows with insoluble surfactants. The interface is captured using a coupled level-set/volume-of-fluid (CLSVOF) method, which combines the strengths of both the level set method and the VOF method. The evolution of the level set function, volume fraction, surfactant mass, and the interfacial area is performed using an arbitrary Lagrangian-Eulerian (ALE) method. The surfactant concentration, which determines the local surface tension through an equation of state, is then computed as surfactant mass per interfacial area. By directly tracking the surfactant mass, instead of the surfactant concentration, the method conserves surfactant mass, and prevents surfactant from diffusing off the interface. The idea of decomposing the surfactant concentration into surfactant mass and interfacial area was first introduced in [James and Lowengrub (2004)]. The big difference here is that James & Lowengrub solved the evolution equations for the surfactant mass and the interfacial area on a fixed rectangular grid using an Eulerian approach, while, in the present work, the solution of all the equations is performed on unstructured triangular grids using an ALE approach. The fluid flow is modeled by the Stokes equations for illustration, which are solved using a finite element method. The surface forces are included in the momentum equation using a continuum surface stress formulation. To efficiently resolve the complex interfacial dynamics, interfaces of complex configurations, regions of high surface curvature, and near contact regions between interacting interfaces, the grid is adapted at every time step so that the grid near the moving interface is always refined. The method is extendible to the axisymmetric and 3D cases, and can be generalized to other types of flows and grids.

The rest of the paper is organized as follows. In section 2, the governing equations are introduced. In section 3, the numerical method are presented. In section 4, test problems are performed to validate the method. Finally, in section 5, conclusions are drawn, and future work is addressed.

2 Governing equations

2.1 The dimensional form of the governing equations

The fluid flow is modeled by Stokes equations

$$-\nabla\cdot(\mu(\nabla\mathbf{u}+(\nabla\mathbf{u})^t))+\nabla p=\nabla\cdot F_s, \tag{1}$$

$$\nabla\cdot\mathbf{u}=0, \tag{2}$$

where μ is the dynamic viscosity of the local fluid, $\mathbf{u}$ is the velocity, the superscript t denotes a matrix transpose, p is the pressure, $F_s=\sigma\delta_\Sigma(I-\mathbf{n}\otimes\mathbf{n})$ is the capillary pressure tensor, σ is the surface tension, $\delta_\Sigma=\delta(\phi)|\nabla\phi|$ is the surface delta function, ϕ is the level set function, and $\mathbf{n}$ is the unit normal vector to the interface. Notice that the

surface forces have been incorporated in the momentum equation as a singular body force. The term $\nabla \cdot F_s$ models both the normal surface tension stress and the tangential Marangoni stress. It can be shown that [James and Lowengrub (2004)]

$$\nabla \cdot (\sigma \delta_\Sigma (I - \mathbf{n} \otimes \mathbf{n})) = \sigma \kappa \delta_\Sigma \mathbf{n} + (\nabla_s \sigma) \delta_\Sigma, \quad (3)$$

where κ is the surface curvature, and ∇_s is the surface gradient operator. Notice that the two terms on the right hand side of the equation account for the normal surface tension and the tangential Marangoni stress, respectively. The inclusion of the normal surface tension force as a singular body force in the momentum equation has been derived in [Chang, Hou, Merriman, and Osher (1996)]. One can show that the Marangoni force can be included in a similar way. Notice that the curvature does not explicitly appear in the above formulation. In addition, we could also extend this to solve Navier-Stokes equations, without complicating the overall algorithm. Finite element Navier-Stokes solvers with surface tension have been developed for both 2D and 3D cases (e.g., [Zheng, Lowengrub, Anderson, and Cristini (2005)]).

Surface tension depends on the surfactant concentration, Γ, according to an equation of state. Many equations of state have been proposed. In the present work, the Langmuir equation of state, one of the most popularly used, will be used for illustration, though other types of equations of state can be used without any further complication. In dimensional form, the Langmuir equation of state is given as

$$\sigma(\Gamma) = \sigma_0 + RT\Gamma_\infty \ln(1 - \Gamma/\Gamma_\infty), \quad (4)$$

where σ_0 is the surface tension of a clean drop ($\Gamma = 0$), R is the universal gas constant, T is the absolute (Kelvin) temperature, Γ_∞ is the surfactant concentration in the maximum packing limit, which exists because each surfactant molecule occupies a finite surface area, putting a limit on the maximum possible surface concentration. For polymeric surfactants, $\Gamma_\infty \approx 0.1 \sim 0.5\,\text{chain}/\text{nm}^2$ [Hu and Lips (2003)].

The evolution of the surfactant concentration is governed by

$$\frac{\partial \Gamma}{\partial t} + \nabla_s \cdot (\Gamma \mathbf{u}_s) + \Gamma(\nabla_s \cdot \mathbf{n})(\mathbf{u} \cdot \mathbf{n}) = D_s \nabla_s^2 \Gamma, \quad (5)$$

where $\mathbf{u}_s$ is the velocity tangent to the interface, and D_s is the surface diffusivity of surfactant. In this equation, the second term accounts for the convection of Γ along the interface, the third term accounts for the change in Γ due to surface expansion or contraction in the surface normal direction, and the term on the right hand side accounts for surfactant diffusion along the interface. See [Stone (1990); Wong, Rumschitzki, and Maldarelli (1996); James and Lowengrub (2004)] for a derivation.

To facilitate the ALE formulation and the conservation of the surfactant mass, Eq. (5) can be integrated over an arbitrary material surface element $S(t)$ on the interface. Let $M = \int_{S(t)} \Gamma dS$ be the mass of the surfactant residing on the surface element $S(t)$. One can show that [Stone (1990); James and Lowengrub (2004)]

$$\frac{DM}{Dt} = D_s \int_{\partial S(t)} \nabla_s \Gamma d\mathbf{s}, \quad (6)$$

where $\partial S(t)$ is the boundary of $S(t)$. Eq. (6) is equivalent to saying that the time rate of change of the surfactant mass in a material surface element is due to surfactant diffusion along the interface. The right hand side has been written as a line integral for easy finite volume discretization. One can also write the right hand side as $D_s A \nabla_s^2 \Gamma$ with A being the surface area of $S(t)$ (see [James and Lowengrub (2004)]), which is convenient for finite difference discretization. By solving Eq. (6), instead of Eq. (5), the surfactant mass, instead of the surfactant concentration, is directly handled, and thus surfactant mass conservation can be more easily enforced in a numerical algorithm. By definition, the surfactant concentration is recovered from the surfactant mass as

$$\Gamma = \frac{M}{A}, \quad (7)$$

and therefore, to determine Γ one has to also track A.

The time rate of change of the interfacial area of a material surface element is governed by

$$\frac{DA}{Dt} = -A(\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n}), \tag{8}$$

which is derived by James et al. in [James and Lowengrub (2004)], and by Batchelor in [Batchelor (1967)], and says that the area of a material surface element may change by surface stretching. Notice that, although the components of $\nabla\mathbf{u}$ are in general discontinuous across the interface, $\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n}$ is continuous (see appendix A in [James and Lowengrub (2004)] for a proof).

The idea of tracking M and A, instead of Γ, has been introduced in [James and Lowengrub (2004)]. In the current work, however, rewriting Eq. (5) as two integral equations for M and A also facilitates the ALE formulation. In addition, A will be used together with the volume fraction in the interface reconstruction. This will enable reconstructing a 'self-consistent' interface, which will not only enclose a prescribed amount of fluid, but also span a prescribed interfacial area.

The evolution of the interface is captured using a CLSVOF method. So, both the level set function, ϕ, and the volume fraction, f, are tracked. The evolution of the level set function and volume fraction are, respectively, governed by

$$\frac{D\phi}{Dt} = \phi_t + \mathbf{u}\cdot\nabla\phi = 0, \tag{9}$$

$$\frac{Df}{Dt} = f_t + \mathbf{u}\cdot\nabla f = 0. \tag{10}$$

We refer readers to [Yang, James, Lowengrub, Zheng, and Cristini (2006)] for more information about the method. One difference is that in [Yang, James, Lowengrub, Zheng, and Cristini (2006)] the evolution of the level set function was performed by solving Eq. 9 using a discontinuous Galerkin method on an Eulerian grid. In the present work, however, both the level set function and volume fraction are evolved using an ALE approach, which will be described in detail in section 3.

Lastly, because in an ALE formulation, the variables are first evolved in time using a Lagrangian approach, for which each grid cell is treated as a material element and transports with the local fluid velocity, we need the following kinematic condition

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}, \tag{11}$$

where $\mathbf{x}$ is the position of an arbitrary point on the border of an element. This equation will be solved to transport the material fluid elements during each Lagrangian step.

2.2 The non-dimensional form of the governing equations

The equations are non-dimensionalized by an arbitrary velocity scale U, a length scale L, time scale L/U, and pressure scale $\mu_2 U/L$, where μ_2 is the dynamic viscosity of fluid 2 (for convenience, let us call one fluid fluid 1 and the other fluid fluid 2). The viscosity is scaled by μ_2. The surfactant concentration is scaled by an 'equilibrium' concentration Γ_{eq}. The surface tension is scaled by σ_{eq}, the surface tension corresponding to the equilibrium concentration Γ_{eq}. With this scaling, the non-dimensional parameters are the viscosity ratio λ, the capillary number Ca, the surface Peclet number Pe_s, the surfactant elasticity E, and the surfactant coverage x, which are respectively defined as

$$\begin{aligned} &\lambda = \frac{\mu_1}{\mu_2}; \quad Ca = \frac{\mu_2 U}{\sigma_{eq}}; \quad Pe_s = \frac{UL}{D_s}; \\ &E = \frac{RT\Gamma_\infty}{\sigma_0}; \quad x = \frac{\Gamma_{\mathrm{eq}}}{\Gamma_\infty}. \end{aligned} \tag{12}$$

For simplicity, the non-dimensional variables will not be distinguished from the original variables. From here on, all variables are referred to as non-dimensional, unless otherwise noted.

The resulting non-dimensional Stokes equations are

$$-\nabla\cdot(\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^t)) + \nabla p = \frac{1}{Ca}\nabla\cdot F_s, \tag{13}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{14}$$

where F_s remains in the same form, except that all the terms in it should be evaluated as non-dimensional, and μ is the non-dimensional dynamic viscosity

$$\mu = \begin{cases} \lambda & \text{in fluid 1,} \\ 1 & \text{in fluid 2.} \end{cases} \tag{15}$$

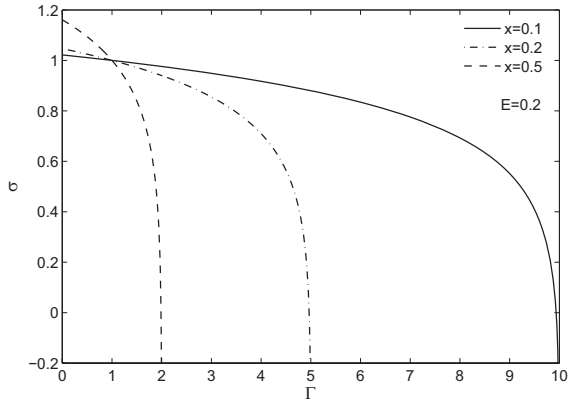

Figure 1: Dependence of surface tension on surfactant concentration.

The dimensionless Langmuir equation of state is

$$\sigma = \frac{1 + E\ln(1 - x\Gamma)}{1 + E\ln(1 - x)}. \tag{16}$$

Fig. 1 shows the dependence of σ on Γ for typical values of E and x. It is seen that the relation between σ and Γ is highly nonlinear. Especially, as Γ approaches the packing limit, the change in σ is very steep. Eq. (16) adequately describes the behavior of a wide range of surfactants. However, one should keep in mind that it gives unrealistic negative surface tension as Γ gets very close to the packing limit. In real systems, solubility of surfactant in one or both of the bulk fluids provides a natural cut off. In the current work, surfactant is constrained to the interface. Solubility will be considered in future work.

For cases where E and x are small (for example, for a polymer E is about $0.1 \sim 0.5$; and x is small for dilute surfactant coverage), Eq. (16) can be approximated using a linear equation of state

$$\sigma = 1 + \beta(1 - \Gamma), \tag{17}$$

where $\beta = Ex$. However, as the surfactant accumulates at the tips of a drop, for example, Γ gets large at the tips, and this approximation fails. The linear equation of state should be used only for cases with small β and with reasonably mild value of Γ. However, because it is very easy to cope with a linear equation of state, and the essential physics of the surfactant effects are mostly captured in a linear equation of state, linear equations of state have been used by many earlier researchers [Stone and Leal (1990); Li and Pozrikidis (1997); Ceniceros (2003); Renardy, Renardy, and Cristini (2002)].

By non-dimensionalization, Eq. (6) becomes

$$\frac{DM}{Dt} = \frac{1}{Pe_s}\int_{\partial S(t)} \nabla_s \Gamma d\mathbf{s}, \tag{18}$$

Eq. (7), (8), (9), (10), and (11) remain the same.

3 Computational method

In the present paper, the governing equations are discretized on a 2D adaptive unstructured triangular mesh for illustration, though the method can be, in general, extended to the axisymmetric and 3D cases and can be generalized to other types of grids. For example, a 3D finite element Navier-Stokes flow solver for interfacial flows can be found in [Zheng, Lowengrub, Anderson, and Cristini (2005)], and a 3D VOF method with parabolic interface reconstruction can be found in [Khismatullin, Renardy, and Renardy (2006)].

The time evolution of the interface and surfactant variables is performed using an ALE approach, which consists of four sub-steps: Lagrangian integration, reconstruction, grid adaptation, and mapping. During the Lagrangian integration, grid cells are considered material elements, and all dependent variables are evolved using a Lagrangian approach. Thus, each grid cell or fluid element transports with the local fluid velocity. See Fig. 2 for an example. In a time step, cell ABC travels to a new position $A'B'C'$. The distance that a grid node travels in a time step is simply the integration of the local fluid velocity over time. By moving/projecting all grid cells this way, the projected grid cells form a new grid, which is called a Lagrangian grid. In contrast, the original grid is called an Eulerian grid. The values of dependent variables on the Lagrangian grid can then be obtained by time-integrating the governing equations using a Lagrangian approach. For example, according to Eq. (10), the volume fraction in a material element or a grid cell remains a constant during a Lagrangian integration. In the

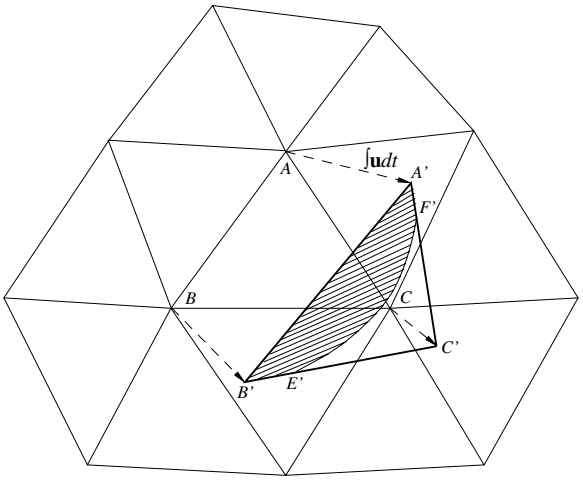

Figure 2: An illustration of the ALE algorithm. The interface in cell $A'B'C'$ is approximated using quadratic segment $E'F'$.

second step, we reconstruct the interface on the Lagrangian grid, based on the level set function, volume fraction, and the interfacial area. By reconstructing the interface, we know where the interface is, and consequently where fluids 1 and 2 are. See Fig. 2 for an example. The reconstructed interface $E'F'$ separates the two fluids in the Lagrangian cell $A'B'C'$. Fluid 1 entirely occupies the region $A'B'E'F'$, and fluid 2 occupies the rest of the cell, region $E'F'C'$. Based on this information, the new updated values of all the variables on the original grid can then be obtained by mapping the variables from the Lagrangian grid back to the original grid. Let us consider the mapping of volume fraction for example (see Fig. 2 for illustration). We see that by mapping, part of fluid 1 in region $A'B'E'F'$ maps back into the original cell ABC, and that the rest of fluid 1 in region $A'B'E'F'$ maps into some cells adjacent to ABC. The new volume fraction in the original cell ABC is then the total volume of fluid 1 that maps into this cell divided by the volume of the cell. Similarly, the new interfacial area in cell ABC is just the length of the portion of the interface that maps into the cell, and the surfactant mass in the cell is the amount of surfactant that resides on this portion of the interface. The new surfactant concentration in the cell is simply the surfactant mass divided by the interfacial area. The new level set function at a grid node in the original grid is directly calculated as a signed distance function to the reconstructed interface. This also reinitializes the level set function to a signed distance function. The new velocity in the original grid is obtained by interpolation. In general, however, we can map the variables on the Lagrangian grid to any arbitrary grid, and in practice, we adapt the original grid before mapping so that the grid near the moving interface is always refined. The variables on the adapted grid can be obtained through a similar mapping.

In summary, we first evolve all the variables using a Lagrangian approach. Then we reconstruct the interface on the Lagrangian grid, and adapt the original grid. The new variables on the adapted grid are then obtained by mapping. Details about each of the sub-steps are given as follows.

3.1 The RK Lagrangian integration

The Lagrangian integration is performed using a two-stage Runge-Kutta (RK) method, consisting of a half time step forward Euler integration followed by a full time step midpoint integration.

Since the grid cells are considered material elements, each grid cell transports with the local fluid velocity. Let us assume that as a grid cell transports, a triangle remains a triangle, although it may deform. This is true for linear velocity fields, and it will be assumed that during every grid transportation the fluid velocity in a cell is locally linear. Under this assumption, the new position of a grid cell after transportation can be determined from the new position of its three vertices. From Eq. (11), the new position of a vertex, i, after a half time step transportation can be computed as

$$\tilde{\mathbf{x}}_i^{n+\frac{1}{2}} = \mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{u}(\mathbf{x}_i^n, t^n), \quad i = 1, ..., MT, \qquad (19)$$

where the symbol˜denotes a Lagrangian variable, and MT is the total number of grid nodes. The transported grid cells form a new mesh, which is called the intermediate Lagrangian mesh.

From Eq. (9), (10), and (8), the intermediate Lagrangian level set function, volume fraction, and

the interfacial area are computed as follows

$$\tilde{\phi}_i^{n+1/2} = \phi_i^n, \quad i = 1, ..., MT, \tag{20}$$

$$\tilde{f}_K^{n+1/2} = f_K^n, \quad \forall K \in \mathscr{T}_h, \tag{21}$$

$$\tilde{A}_K^{n+1/2} = A_K^n - \frac{\Delta t}{2} A_K^n (\mathbf{n} \cdot \nabla \mathbf{u} \cdot \mathbf{n})_K^n, \quad \forall K \in \mathscr{T}_h, \tag{22}$$

where K denotes an arbitrary grid cell in the domain, and $\mathscr{T}_h$ is a triangulation of the domain. Notice that the level set function is defined at the vertices, while volume fraction, interfacial area, surfactant mass, and concentration are associated with the grid cells. In Eq. (22), $\nabla \mathbf{u}$ is computed by taking the derivative of the velocity approximated by the finite elements, and $\mathbf{n}$ is computed from the level set function using a least squares fitting. The method for computing $\mathbf{n}$ is the same as the one introduced in [Yang, James, Lowengrub, Zheng, and Cristini (2006)], which is described below for completeness.

To calculate $\mathbf{n}$ in a cell, a quadratic function for ϕ is fitted using a least squares method over a stencil that consists of the vertices of the cell and of all the cells that share at least a vertex with the cell, as shown in Fig. 3. To simplify the calculation, a local Cartesian coordinate system $x' - y'$ is defined for each normal calculation (see Fig. 3). The origin of the local coordinate system is located at the center of the cell under consideration, at which the normal is located. The local coordinate axes x' and y' are parallel to the global coordinate axes x and y, respectively. Let (x_c, y_c) be the global coordinates of the cell center. Then, the local and global coordinates of a point are related to each other via $x' = x - x_c$ and $y' = y - y_c$.

In each local coordinate system, the quadratic function for ϕ has the generic form

$$\phi = a_1 x'^2 + a_2 x' y' + a_3 y'^2 + a_4 x' + a_5 y' + a_6, \tag{23}$$

where a_1, a_2, a_3, a_4, a_5, and a_6 are coefficients to be determined using the least squares method. In particular, a_1, a_2, a_3, a_4, a_5, and a_6 are the least squares solution of the over-constrained linear system

$$Qs = r, \tag{24}$$

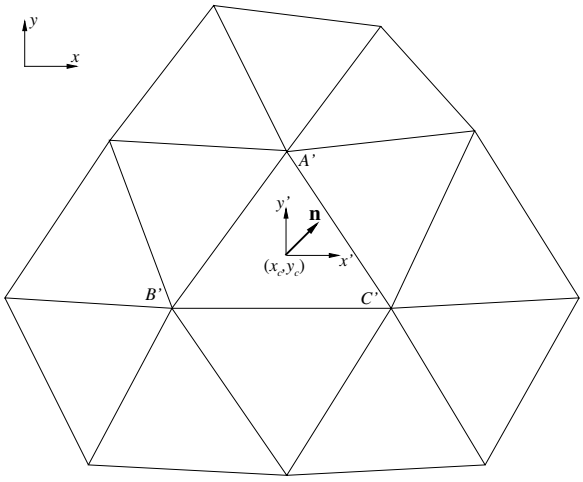

Figure 3: The stencil for calculating the interface normal vector in element $A'B'C'$. The normal vector and the origin of the local coordinate system $x' - y'$ are located at the center of element $A'B'C'$.

where

$$Q = \begin{pmatrix} x_1'^2 & x_1' y_1' & y_1'^2 & x_1' & y_1' & 1 \\ x_2'^2 & x_2' y_2' & y_2'^2 & x_2' & y_2' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N'^2 & x_N' y_N' & y_N'^2 & x_N' & y_N' & 1 \end{pmatrix},$$

$$s = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix}, \quad r = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{pmatrix},$$

(x_i', y_i'), $i = 1, \cdots, N$, are the local coordinates of the ith node in the stencil, N is the total number of nodes in the stencil, and ϕ_i, $i = 1, \cdots, N$, is the level set value at the ith node. The least squares solution of this linear system is $s = (Q^t Q)^{-1} Q^t r$, where the superscript t denotes matrix transpose and -1 denotes matrix inversion. Since $Q^t Q$ is symmetric positive definite, it can be efficiently inverted using Cholesky decomposition. Once the coefficients are known, the unit normal vector, which is located at the origin of the local coor-

dinate system, is simply

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} = \left(\frac{a_4}{\sqrt{a_4^2+a_5^2}}, \frac{a_5}{\sqrt{a_4^2+a_5^2}} \right).$$

From Eq. (18), the surfactant mass on the intermediate Lagrangian mesh is computed as

$$\tilde{M}_K^{n+1/2} = M_K^n + \frac{\Delta t}{2Pe_s} \sum_{j=1}^{N_{c0}} \frac{\tilde{\Gamma}_{Kj}^{n+1/2} - \tilde{\Gamma}_K^{n+1/2}}{\tilde{L}_{Kj}^{n+1/2}}, \quad \forall K \in \mathscr{T}_h, \quad (25)$$

where N_{c0} is the number of adjacent grid cells that contain an interface segment, and $\tilde{L}_{Kj}^{n+1/2}$ is the surface distance between the midpoints of the interface segment in cell K and the jth adjacent interface segment. It is apparent that the gradient of the surfactant concentration in Eq. (18) has been discretized using a central finite difference. Normally, in 2D, $N_{c0} = 2$, i.e., there are exactly two interface segments adjacent to the interface segment under consideration, as shown in Fig. 4(a). However, in the cases shown in Fig. 4(b), 4(c) and 4(d), N_{c0} may be different from 2. Special treatment of these three cases will be addressed in detail later. For now, it is assumed that $N_{c0} = 2$. In addition, in order to locate the midpoint of an interface segment, in general one needs first to reconstruct the interface segment in the cell, which will be described in section 3.2. However, in 2D the interfacial area is the same as the arc-length of the interface. Thus, $\tilde{L}_{Kj}^{n+1/2}$ can be alternatively computed as

$$\tilde{L}_{Kj}^{n+1/2} = \frac{\tilde{A}_K^{n+1/2} + \tilde{A}_{Kj}^{n+1/2}}{2}.$$

Notice that all equations have been discretized explicitly, except that the surfactant diffusion term in Eq. (18) is discretized implicitly. The surfactant diffuses along the interface, which is approximated in each interfacial cell by a facet (in 3D) or segment (in 2D), which could be linear or parabolic, for example. The size of the facet or segment ranges from just bigger than zero to the size of the cell itself, depending on how the interface intersects the cell. When a very small facet

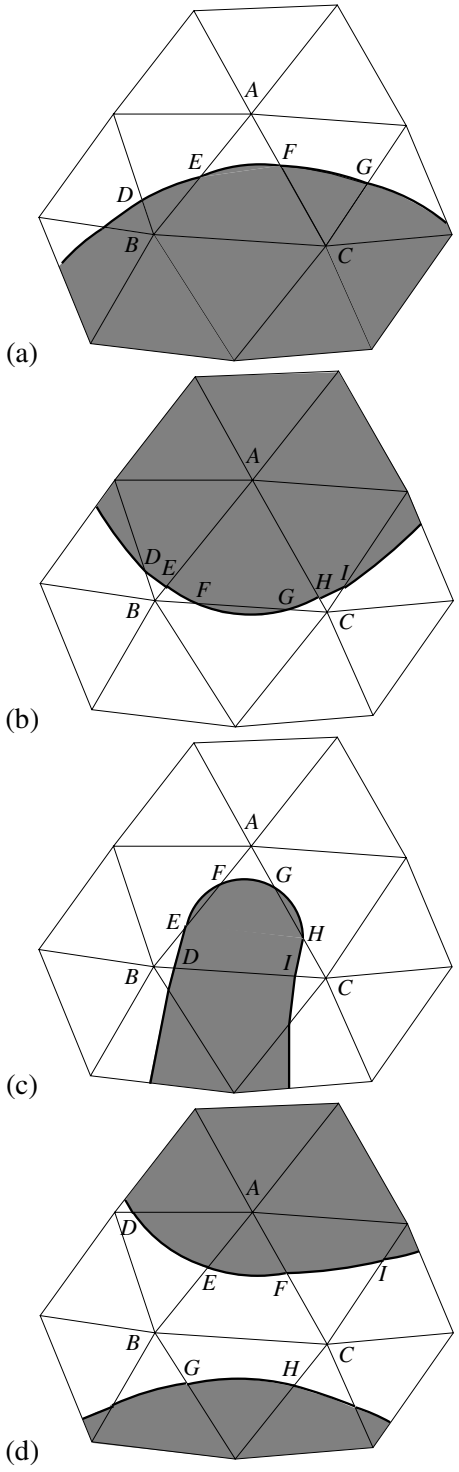

Figure 4: Interface complexity. (a) typical case; (b) multiple intersections; (c) high curvature; (d) near-contact interfaces.

or segment is produced, an explicit method becomes impractical due to the time step restriction for stability, and an implicit discretization must be used. On the other hand, due to the implicit discretization, Eq. (25) involves both $\tilde{M}_K^{n+1/2}$ and $\tilde{\Gamma}_K^{n+1/2}$, which makes it impossible to update M directly based solely on Eq. (25). Thus, following the same procedure as in [James and Lowengrub (2004)], and utilizing Eq. (7), Eq. (25) can be rewritten as an implicit equation solely for $\tilde{\Gamma}_K^{n+1/2}$, i.e.,

$$\tilde{A}_K^{n+1/2}\tilde{\Gamma}_K^{n+1/2} = M_K^n + \frac{\Delta t}{2Pe_s}\sum_{j=1}^{N_{c0}} \frac{\tilde{\Gamma}_{Kj}^{n+1/2} - \tilde{\Gamma}_K^{n+1/2}}{\tilde{L}_{Kj}^{n+1/2}}, \quad \forall K \in \mathscr{T}_h, \quad (26)$$

which is solved iteratively for $\tilde{\Gamma}_K^{n+1/2}$ using the point Gauss-Seidel method. The surfactant mass is recovered as

$$\tilde{M}_K^{n+1/2} = \tilde{A}_K^{n+1/2}\tilde{\Gamma}_K^{n+1/2}. \quad (27)$$

The intermediate Lagrangian velocity $\tilde{\mathbf{u}}(\tilde{\mathbf{x}}_i^{n+1/2}, t^{n+\frac{1}{2}})$ and pressure are obtained by solving the Stokes equations on the intermediate Lagrangian grid. Notice that the Stokes equations do not explicitly involve time derivatives. They can be solved as a quasi-static problem at any time instance to provide the instantaneous velocity field. Details of the flow solver will be described in section 3.5. For time dependent flows, for example the unsteady Navier-Stokes flows, the flow equations could also be solved using an ALE approach. See [Hirt, Amsden, and Cook (1974)] for a finite volume implementation and [W. Dettmer (2003); Zhang, Gerstenberger, Wang, and Liu (2004)] for finite element implementations. Alternatively, one could also employ a flow solver written in an Eulerian way, while still evolving the interface position, interfacial area and surfactant mass and concentration using an ALE approach. However, this would require some communication between the Eulerian and Lagrangian meshes at every sub-step of the RK integration. That is, one needs mapping (and thus reconstruction) or interpolating the variables on the Lagrangian mesh to the Eulerian mesh, or vice versa.

Knowing the values of all the variables on the intermediate Lagrangian mesh, the final Lagrangian variables are then computed using a midpoint time integration. Specifically, the position of each final Lagrangian grid node is computed as

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t\tilde{\mathbf{u}}(\tilde{\mathbf{x}}_i^{n+1/2}, t^{n+\frac{1}{2}}), \quad i = 1, ..., MT. \quad (28)$$

The time dependent variables on the final Lagrangian mesh are computed as

$$\tilde{\phi}_i^{n+1} = \phi_i^n, \quad i = 1, ..., MT, \quad (29)$$

$$\tilde{f}_K^{n+1} = f_K^n, \quad \forall K \in \mathscr{T}_h, \quad (30)$$

$$\tilde{A}_K^{n+1} = A_K^n - \Delta t\tilde{A}_K^{n+1/2}(\tilde{\mathbf{n}}\cdot\nabla\tilde{\mathbf{u}}\cdot\tilde{\mathbf{n}})_K^{n+1/2}, \quad \forall K \in \mathscr{T}_h, \quad (31)$$

$$\tilde{M}_K^{n+1} = M_K^n + \frac{\Delta t}{Pe_s}\sum_{j=1}^{N_{c0}} \frac{\tilde{\Gamma}_{Kj}^{n+1/2} - \tilde{\Gamma}_K^{n+1/2}}{\tilde{L}_{Kj}^{n+1/2}}, \quad \forall K \in \mathscr{T}_h, \quad (32)$$

$$\tilde{\Gamma}_K^{n+1} = \tilde{M}_K^{n+1}/\tilde{A}_K^{n+1}, \quad \forall K \in \mathscr{T}_h. \quad (33)$$

The implementations of these equations are similar to those in the first RK step. The velocity on the final Lagrangian grid is computed by the Stokes flow solver.

3.2 Interface reconstruction

Knowing all the variables on the final Lagrangian grid, an interface is reconstructed on the final Lagrangian grid based on the level set function, volume fraction, and the interfacial area. Previous research indicated that interfaces reconstructed using piecewise linear segments were not adequate to accurately represent the interfacial area [James and Lowengrub (2004)]. In the present work, interfaces are approximated using piecewise parabolic segments. Previous research, for example [Price, Reader, Rowe, and Bugg (1998)] and [Renardy and Renardy (2002)], has shown that reconstructing the interface using parabolic instead of linear segments can improve the accuracy greatly. In the present work,

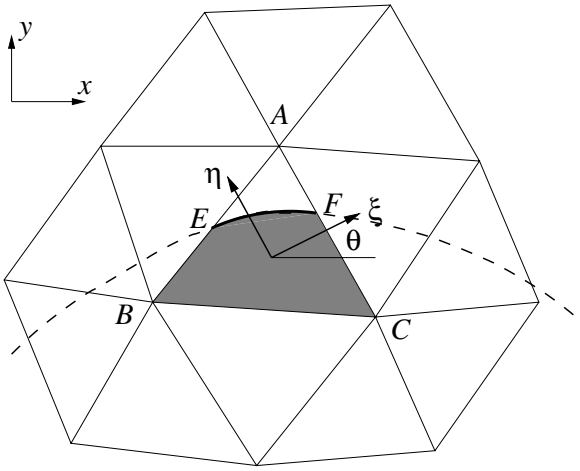

Figure 5: Reconstructing a piecewise parabolic interface. The dashed line is an extension of the parabolic interface segment EF in cell ABC.

a self-consistent piecewise parabolic interface reconstruction (PPIC) method has been developed, in which the parabolic interface segment in a cell not only truncates the cell by the given volume fraction, but also spans a given interfacial area.

The method can be considered an extension of the PPIC method developed in [Price, Reader, Rowe, and Bugg (1998)]. As is illustrated in Fig. 5, the parabolic interface segment in cell ABC is defined in a local Cartesian coordinate system $\xi-\eta$. The origin of the local coordinate system is located at the cell center (x_c, y_c). The orientation of the local coordinate system is measured by angle θ, the angle formed between the ξ-axis and the x-axis. The local and global coordinates of a point are related to each other as follows

$$\xi = (x-x_c)\cos\theta + (y-y_c)\sin\theta, \quad (34)$$

$$\eta = (y-y_c)\cos\theta - (x-x_c)\sin\theta. \quad (35)$$

In the local coordinate system, a parabolic curve has the generic form

$$\eta = a\xi^2 + b\xi + c. \quad (36)$$

Our interface reconstruction algorithm thus must determine the set of θ, a, b, and c that minimize

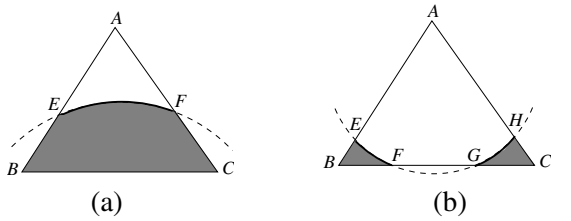

Figure 6: Clipping a triangular cell by a parabola. The dashed lines represent the extended parabolic interface segments.

the following quadratic error function

$$E_s = \sum_{i=1}^{N_s} (m_i(\hat{f}_i - f_i))^2 + (m_{N_s+1}(\hat{A}-A))^2, \quad (37)$$

where N_s is the number of cells in the stencil, as shown in Fig. 5, which consists of the cell under consideration, i.e. ABC, and all the adjacent cells that share at least a vertex with ABC, m_i is a weight on the corresponding term, f_i is the given volume fraction in the ith cell in the stencil, A is the given interfacial area in the cell under consideration, and $\hat{f}_i$ and $\hat{A}$ are the corresponding volume fractions and the interfacial area obtained by truncating the cells in the stencil by an extended parabolic curve defined by a set of θ, a, b, and c. Thus, minimizing the error function E_s results in the set of θ, a, b, and c that define a parabola that best fits the volume fraction in every cell in the stencil and the interfacial area in the cell under consideration in a least squares sense.

Notice that $\hat{f}_i$, $\hat{A}$, and thus E_s are nonlinear functions of θ, a, b, and c. Given a set of θ, a, b, and c, $\hat{f}_i$ and $\hat{A}$ can be found by clipping each grid cell in the stencil by the extended parabolic curve defined by θ, a, b, and c, as illustrated in Fig. 6. Algorithm 1 describes how to clip a general polygon $p_1p_2\cdots p_n$ by a parabola $\mathscr{P}$. The algorithm always returns the region under the parabola, and it is assumed that fluid 1 always occupies this region. If fluid 1 were above the parabola, the minimization process would automatically re-orient the local coordinate system by modifying θ so that fluid 1 lies under the parabola in the new coordinate system. Simple modification could be made to the algorithm to return the region above

the parabola as well. For convenience, the vertices of the returned region are named $q_1 q_2 \cdots q_m$, and it is assumed that $p_n = p_1$ and $q_m = q_1$. Because the interface segment is parabolic, the returned region consists of both linear and parabolic edges. For convenience, we call such a region a generalized polygon. For example, in Fig. 6(a) it is $BEFC$, and in Fig. 6(b) it is the sum of the two generalized polygons BEF and GHC. In practice, in the second example the algorithm will return a single generalized polygon $BEHCGF$, of which the parabolic edge GF is coincident with part of the other parabolic edge EH. So, $BEHCGF$ is a connection of BEF and GHC by the infinitely thin parabolic channel FG. Let (ξ_i, η_i), $i = 0, \cdots, n$, be the coordinates of the vertices of a generalized polygon with $\xi_n = \xi_0$ and $\eta_n = \eta_0$. Then, the volume of fluid in a generalized polygonal fluid element, i.e. the area of the generalized polygon, can be computed as [Price, Reader, Rowe, and Bugg (1998)]

$$S = \frac{1}{2}\sum_{i=0}^{n-1}\left(\xi_i\eta_{i+1} - \xi_{i+1}\eta_i + \frac{a_i'(\xi_{i+1}-\xi_i)^3}{3}\right), \tag{38}$$

where $a_i' = a$ if the edge between vertices i and $(i+1)$ is parabolic, and $a_i' = 0$ if it is linear, and S is positive if the vertices are in counterclockwise order, and negative otherwise. Following the idea of Sunday [Sunday (2002)], one more efficient formula can be obtained, i.e.,

$$S = \frac{1}{2}\sum_{i=1}^{n}\left(\xi_i(\eta_{i+1} - \eta_{i-1}) + \frac{a_i'(\xi_{i+1}-\xi_i)^3}{3}\right), \tag{39}$$

where $\xi_{n+1} = \xi_1$ and $\eta_{n+1} = \eta_1$. Notice that the second formula reduces the number of multiplications in the first two terms by a half.

The interfacial area in a cell is the total length of the parabolic interface segments, which are, in most cases, the parabolic edges of the generalized polygon that contains fluid 1. One exception is the case illustrated in Fig. 6(b), in which the generalized polygon containing fluid 1 has overlapping parabolic edges, i.e., the parabolic edge GF overlaps part of the parabolic edge EH. Notice that

Algorithm 1 Clipping a polygon by a parabola.

Set $k = 0$.
for $j = 1, \cdots, n-1$ **do**
  Calculate the intersection(s) w_1 (and w_2) between the linear edge $p_j p_{j+1}$ and the parabola $\mathscr{P}$ (note: a parabola could have 0, 1, or 2 intersections with a linear segment; if there are two intersections, it is assumed that w_1 is closer to p_j than w_2 is; and a tangent point is not considered an intersection in this algorithm).
  if 0 intersections **then**
    if p_j and p_{j+1} are under $\mathscr{P}$ **then**
      $k = k+1$; $q_k = p_j$; tag edge $q_k q_{k+1}$ as linear.
    else
      Do nothing.
    end if
  else if 1 intersection **then**
    if p_j is under $\mathscr{P}$ and p_{j+1} is above $\mathscr{P}$ **then**
      $k = k+1$; $q_k = p_j$; tag edge $q_k q_{k+1}$ as linear.
      $k = k+1$; $q_k = w_1$; tag edge $q_k q_{k+1}$ as parabolic.
    else
      $k = k+1$; $q_k = w_1$; tag edge $q_k q_{k+1}$ as linear.
    end if
  else if 2 intersections **then**
    if the parabola opens upward **then**
      $k = k+1$; $q_k = p_j$; tag edge $q_k q_{k+1}$ as linear.
      $k = k+1$; $q_k = w_1$; tag edge $q_k q_{k+1}$ as parabolic.
      $k = k+1$; $q_k = w_2$; tag edge $q_k q_{k+1}$ as linear.
    else
      $k = k+1$; $q_k = w_1$; tag edge $q_k q_{k+1}$ as linear.
      $k = k+1$; $q_k = w_2$; tag edge $q_k q_{k+1}$ as parabolic.
    end if
  end if
end for

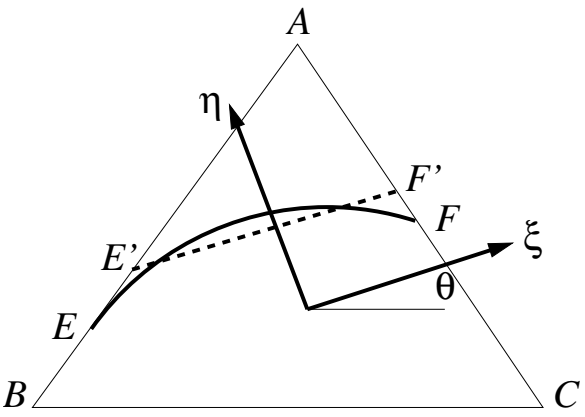

Figure 7: Initial guess of the parabolic interface segment EF. $E'F'$ is a linear approximation of EF.

this happens only when the two parabolic edges follow opposite directions along the parabola. More specifically, for the generalized polygon $BEHCGF$ the parabolic edge EH is from E to H, i.e. from a vertex of smaller ξ-coordinate to a vertex of larger ξ-coordinate, while the parabolic edge GF is from G to F, i.e. from a vertex of larger ξ-coordinate to a vertex of smaller ξ-coordinate. For this case, the real parabolic interface segments must be the part of the longest parabolic edge (e.g., EH) obtained by removing the parts that are coincident with the edges in the opposite direction (e.g., GF).

The minimization of E_s is performed using the Levenberg-Marquardt method, for which Fortran source codes are available to download from http://www.netlib.org. The method requires initial guesses for θ, a, b, and c, which are determined from a piecewise linear interface reconstruction (PLIC), as illustrated in Fig. 7. Specifically, θ is determined by requiring that the η-axis points in the same direction as the interface normal vector obtained using the least squares method described in section 3.1. a is determined by requiring that the peak curvature of the parabola is the same as the one estimated from taking the second order derivative of a quadratic function of the level set function obtained using the same least squares method. Assuming that the quadratic function of ϕ has the generic form defined by Eq. (23), the curvature of the interface can be computed as

$$\kappa = -\nabla\cdot(\frac{\nabla\phi}{|\nabla\phi|}) = \frac{2(a_2a_4a_5 - a_2a_5^2 - a_3a_4^2)}{(a_4^2+a_5^2)^{3/2}}. \tag{40}$$

b is determined by requiring that the symmetry axis of the parabola is also the perpendicular bisector of the linear interface segment obtained from the piecewise linear interface reconstruction. The linear interface is obtained using the analytic PLIC method developed in [Yang, James, Lowengrub, Zheng, and Cristini (2006)]. We refer readers to [Yang, James, Lowengrub, Zheng, and Cristini (2006)] for details. Lastly, c is determined by enforcing volume conservation.

3.3 Grid adaptation

To efficiently resolve the complex interfacial dynamics, after we obtain the Lagrangian grid, the original Eulerian grid is adapted so that the grid near the moving interface is always refined. In addition, by adapting/re-zoning the grid, grid tangling, which often occurs in a pure Lagrangian method due to large deformation of fluid elements, can be successfully prevented. We use the adaptive volume re-meshing algorithm developed in [Anderson, Zheng, and Cristini (2005); Zheng, Lowengrub, Anderson, and Cristini (2005)], which is an adaptation to flat domains of the adaptive surface triangulated mesh of [Cristini, Blawzdziewicz, and Loewenberg (2001)]. For brevity, we refer readers to [Anderson, Zheng, and Cristini (2005); Zheng, Lowengrub, Anderson, and Cristini (2005); Cristini, Blawzdziewicz, and Loewenberg (2001)] for details. An application of the method in the context of an adaptive CLSVOF interface capturing can be found in [Yang, James, Lowengrub, Zheng, and Cristini (2006)].

3.4 Mapping

The variables on the adapted grid are obtained by mapping. The mapping is, in nature, equivalent to the convection in a pure Eulerian method, which accounts for the relative motion between the Lagrangian fluid elements and the Eulerian

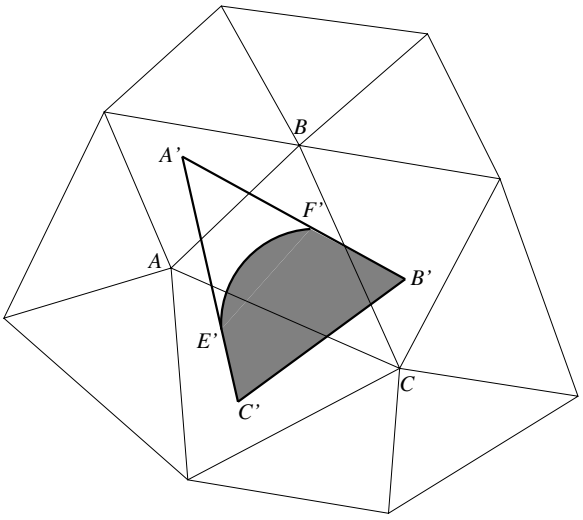

Figure 8: Mapping the fluid on the Lagrangian grid onto the Eulerian grid.

grid cells [Hirt, Amsden, and Cook (1974)]. The details are as follows.

3.4.1 Mapping volume fraction

The volume fraction f^{n+1} on the adapted Eulerian grid is obtained by mapping each fluid element on the Lagrangian grid onto the Eulerian grid. For example, in Fig. 8, the two fluids in the Lagrangian cell $A'B'C'$ are separated by the parabolic interface segment $E'F'$, which is obtained through the interface reconstruction process. Fluid 1 occupies the entire volume $B'C'E'F'$. By mapping, one sees that part of fluid 1 in $B'C'E'F'$ maps into cell ABC on the Eulerian grid, and that the rest of fluid 1 in $B'C'E'F'$ maps into the adjacent cells of ABC.

Geometrically, the mapping of volume fraction is equivalent to performing a sequence of polygon-polygon clippings, in which fluid elements and grid cells are treated as generalized polygons, and each fluid region filled with fluid 1, such as region $B'C'E'F'$ in Fig. 8, is clipped against every grid cell in the Eulerian mesh. The clipping algorithm returns the polygonal intersection between a Lagrangian fluid region and an Eulerian grid cell, the volume of which can then be computed via Eq. (39). The new volume fraction in a cell on the Eulerian mesh is then the total volume of fluid 1 that maps into this cell divided by the volume of the cell.

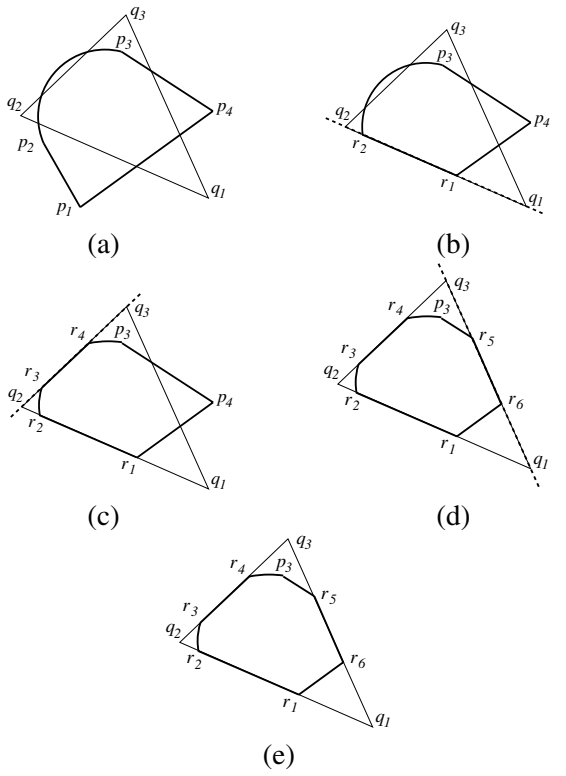

Figure 9: Clipping of a four-sided generalized polygonal fluid element against a triangular grid cell. (a) The polygons before clipping. (b) Clipping of $p_1p_2p_3p_4$ against edge q_1q_2. (c) Clipping of $r_1r_2p_3p_4$ against edge q_2q_3. (d) Clipping of $r_1r_2r_3r_4p_3p_4$ against edge q_3q_1. (e) The polygons after clipping.

As an example, Fig. 9 illustrates clipping of a four-sided generalized polygonal fluid element $p_1p_2p_3p_4$ against a triangular grid cell $q_1q_2q_3$. In [Yang, James, Lowengrub, Zheng, and Cristini (2006)], Yang et al. employed the Sutherland-Hodgeman algorithm [Sutherland and Hodgeman (1974)] for polygon clippings. It is worth pointing out that in previous works the Sutherland-Hodgeman algorithm was often used to clip polygons only consisting of linear edges. In the present work, because the interface is approximated using piecewise parabolic segments, the fluid elements consist of both linear and parabolic

edges (notice that in Fig. 9 edge p_2p_3 of the fluid element is parabolic). We have generalized the standard Sutherland-Hodgeman algorithm to clip generalized polygons consisting of both linear and parabolic edges.

The basic idea of the clipping algorithm is to "trim" a polygonal fluid region by each of the edges of a grid cell, as illustrated in Fig. 9. To illustrate the "trimming" process, let us consider the clipping of $p_1p_2p_3p_4$ against the first edge q_1q_2 of the grid cell for example. For convenience, let us name the infinitely-long line obtained by infinitely extending edge q_1q_2, $\mathscr{L}$, and define that a vertex p_j is on the "right" side of $\mathscr{L}$ if p_j is on the same side of $\mathscr{L}$ as the other vertex q_3 is, and is on the "left" side otherwise. Then, the trimming process can be described as in algorithm 2, where, to be general, the vertices of the fluid region are denoted by $p_1p_2\cdots p_n$, and it is assumed that $p_n = p_1$. The vertices of the clipped polygon that lie on the right side of $\mathscr{L}$ are denoted by $r_1r_2\cdots r_n$. In this example, the list of the vertices produced from the trimming process define the clipped polygon $r_1r_2p_3p_4$ (see Fig. 9(b)). In general, the trimming process is reentered to clip the previously clipped polygon against the subsequent edges of the grid cell (Fig. 9(c) and (d)), and the overall clipping completes as soon as the polygonal fluid element has been trimmed by all the extended edges of the grid cell (Fig. 9(e)).

As is in [Yang, James, Lowengrub, Zheng, and Cristini (2006)], in order to save the computational time, in practice we only need to clip a grid cell against the fluid regions which are in the vicinity of the cell. Since the Lagrangian fluid elements are connected consecutively without any gap or intersections between them, the new volume fraction on the Eulerian mesh must be between 0 and 1, and the total volume of fluid is conserved exactly during mapping.

3.4.2 Mapping the interfacial area

Mapping the interfacial area is equivalent to clipping each parabolic interface segment on the Lagrangian grid against each grid cell on the Eulerian grid. For example, in Fig. 8, to compute how much of the interfacial area in the Lagrangian cell

Algorithm 2 Trimming a generalized polygon by a line.

Set $k = 0$.
for $j = 1, \cdots, n-1$ **do**
  Calculate the intersection(s) w_1 (and w_2) between edge p_jp_{j+1} and the extended line $\mathscr{L}$ (note: if there are two intersections, it is assumed that w_1 is closer to p_j than w_2 is; and a tangent point is not considered an intersection in this algorithm).
  if edge p_jp_{j+1} is parabolic **then**
    if 0 intersections **then**
      if p_j and p_{j+1} are on the right side of $\mathscr{L}$ **then**
        $k = k+1$; $r_k = p_j$; tag r_kr_{k+1} as parabolic.
      else
        Do nothing.
      end if
    else if 1 intersection **then**
      if p_j is on the right and p_{j+1} on the left **then**
        $k = k+1$; $r_k = p_j$; tag r_kr_{k+1} as parabolic.
        $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as linear.
      else if p_j is on the left and p_{j+1} on the right **then**
        $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as parabolic.
      end if
    else if 2 intersections **then**
      if p_j and p_{j+1} are on the right side of $\mathscr{L}$ **then**
        $k = k+1$; $r_k = p_j$; tag r_kr_{k+1} as parabolic.
        $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as linear.
        $k = k+1$; $r_k = w_2$; tag r_kr_{k+1} as parabolic.
      else if p_j and p_{j+1} are on the left side of $\mathscr{L}$ **then**
        $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as parabolic.
        $k = k+1$; $r_k = w_2$; tag r_kr_{k+1} as linear.
      end if
    end if
  else if edge p_jp_{j+1} is linear **then**
    if both p_j and p_{j+1} are on the right side of $\mathscr{L}$ **then**
      $k = k+1$; $r_k = p_j$; tag r_kr_{k+1} as linear.
    else if p_j is on the right and p_{j+1} on the left **then**
      $k = k+1$; $r_k = p_j$; tag r_kr_{k+1} as linear.
      $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as linear.
    else if p_j is on the left and p_{j+1} on the right **then**
      $k = k+1$; $r_k = w_1$; tag r_kr_{k+1} as linear.
    else
      Do nothing.
    end if
  end if
end for

$A'B'C'$ maps into the Eulerian cell ABC, we need to determine the portion of the parabolic interface segment $E'F'$ that overlaps cell ABC, the length of which is then the interfacial area that maps from cell $A'B'C'$ into cell ABC.

The clipping of a parabolic segment against a polygonal cell can be performed by utilizing either algorithm 1 or algorithm 2. The computational effort is about the same. When using algorithm 1, for example, one first clips the polygonal cell by an extended parabola obtained by infinitely extending the parabolic interface segment under consideration. Then, the portion of the parabolic interface segment that maps into the cell is just the common part of the clipped parabolic segment resulted from clipping and the parabolic interface segment itself.

3.4.3 Mapping the surfactant

The new surfactant mass in an Eulerian grid cell is simply the amount of surfactant residing on the part of the interface that maps into the cell. It is assumed that the surfactant within a cell is linearly distributed along the interface.

In most cases, there is only one interface segment in a grid cell, which is adjacent to exactly two interface segments in its adjacent cells, as illustrated in Fig. 4(a). Let us define a one dimensional curvilinear local coordinate, s, along an interface segment EF, with its origin located at E and pointing toward F. Then, the linear distribution of the surfactant along the interface segment EF can be expressed as

$$\Gamma = \overline{\nabla_s \Gamma}(s-\overline{s}) + \Gamma_K, \tag{41}$$

where $\overline{\nabla_s \Gamma}$ is the slope, $\overline{s}$ is the s-coordinate of the midpoint of the interface segment, and Γ_K is the average concentration in cell ABC.

To preserve monotonicity, the van Leer limiter is applied to the calculation of the slope. So, the slope of the distribution is computed as

$$\overline{\nabla_s \Gamma} = \begin{cases} \mathrm{sgn}(s1)\min\{|s1|,|s2|,|s3|\} \\ \qquad \text{if } \mathrm{sgn}(s1)=\mathrm{sgn}(s2)=\mathrm{sgn}(s3) \\ 0 \qquad \text{otherwise,} \end{cases} \tag{42}$$

where

$$\begin{aligned} s1 &= 2(\Gamma_K - \Gamma_{K1})/A_K, \\ s2 &= (\Gamma_{K2} - \Gamma_{K1})/(A_K + A_{K1}/2 + A_{K2}/2), \\ s3 &= 2(\Gamma_{K2} - \Gamma_K)/A_K, \end{aligned}$$

$K1$ and $K2$ denote the two adjacent cells containing interface segments DE and FG, respectively, and 'sgn' is the sign function. This distribution guarantees that the linear distribution in cell ABC will not take values beyond the average values in the neighboring cells, and thus, it also preserves the positivity of Γ automatically. More complex cases as illustrated in Fig. 4(b), (c), and (d) will be discussed separately in section 3.4.5.

After the slope is determined, the surfactant mass on an interface segment is obtained by integrating Eq. (41) over the segment area. Surfactant mass is conserved during mapping.

3.4.4 Mapping the velocity, pressure, and the level set function

The velocity and pressure on the new Eulerian grid are obtained by interpolation. Specifically, to find the value of a function at a node P_E on the Eulerian grid, we first locate the element K_L on the Lagrangian mesh such that $P_E \in K_L$. Then, the value of the function at node P_E is obtained from the local representation of the function on K_L. To efficiently locate the element K_L, a steepest descent type algorithm is used, in which the search proceeds in the direction of the most negative barycentric coordinate of P_E with respect to the current element on the Lagrangian mesh. See also [Zheng, Lowengrub, Anderson, and Cristini (2005)].

The level set function at a nodal point on the new Eulerian grid is directly calculated as the signed distance of the point to the piecewise parabolic interface reconstructed previously on the Lagrangian grid. The distance of a point to the interface is the minimum of the distances between the point and each parabolic interface segment. Notice that after mapping the new level set function has been reinitialized to a signed distance function, as is desired.

3.4.5 Surfactant diffusion and distribution for complex cases

In sections 3.1 and 3.4.3, we have assumed that there is only one interface segment in a grid cell, and that the single interface is adjacent to exactly two other interface segments, as illustrated in Fig. 4(a). In practice, however, there are extreme cases where these assumptions do not hold. See Fig. 4(b), (c) and (d) for an illustration of these cases.

In Fig. 4(b), the interface intersects cell *ABC* through one its edges, resulting in two interface segments in the cell. Consequently, all three cells that share an edge with cell *ABC* contain an interface, and for the cell containing the parabolic segment FG only one of its adjacent cells, i.e. *ABC*, contains the interface. When this occurs, it can be assumed that the surfactant distribution has the same slope on FG, EF and GH, and that the net surfactant diffusion between FG, EF and GH is zero. That is, we treat FG, EF and GH together as a single interface segment. The surfactant mass and interfacial area of the 'merged' segment are, respectively, the sum of the surfactant mass and interfacial area in cell *ABC* and the cell containing segment FG. We are also implementing a mechanism in the grid adaptation for preventing the grid nodes from being too close to an interface, which should greatly, if not completely, reduce the chance of the case happening.

In Fig. 4(c), cell *ABC* contains three interface segments. This happens only when the radius of curvature of the interface is roughly less than the size of the grid cell, indicating inadequate resolution. Because the interface is not well resolved, it is of little importance to determine how the surfactant is distributed or diffused. When this happens, the diffusion in the cell is simply neglected, and the slope is set to zero. This case will not occur if the high curvature is well resolved.

When two interfaces come very close, there may still be one interface segment in cell *ABC*, as illustrated in Fig. 4(d), but all three cells that share an edge with cell *ABC* contain an interface. For this case, the segment in cell *ABC* is considered to be connected to the interface segments in the adjacent cells that themselves have only two adjacent cells containing an interface segment. According to this criterion, segment EF will be considered to be connected to DE and FI, but not GH, and the surfactant diffusion will occur between these connected segments and the surfactant distribution slope will be computed from the concentrations in these cells. Notice that this case happens just because the grid resolution is too low relative to the thickness of the film formed between the two interfaces. If the film is well resolved, the case will not occur.

As discussed above, the cases shown in Fig. 4(c) and (d) occur solely because of inadequate resolution. So, in practice, it is of little importance to develop sophisticated algorithms to handle these cases because the overall solution is not well resolved anyway. One can simply ignore the surfactant diffusion in these cases, and set the surfactant distribution slope to zero. The case shown in Fig. 4(b) deserves more carefully handling. It is of great benefit to align the grid points away from the interface so that the interface intersects a grid cell roughly through the middle of the cell, which will prevent the case shown in Fig. 4(b) from occurring, eliminate the difficulty in reconstructing the interfacial area (when the interface almost coincides with an edge, slight changes in the orientation of the reconstructed interface can result in large errors in the interfacial area), and avoid tiny interface segments and/or extremely small fluid volumes when the interface intersects a grid cell near a corner.

3.5 The Stokes flow solver

The Stokes equations are solved using a Galerkin finite element method. The velocity and pressure are approximated using continuous piecewise quadratic and piecewise linear functions, respectively. In particular, the finite element spaces for velocity and pressure are, respectively, defined as

$$V_h = \{\mathbf{v} \in (H^1(\Omega))^2 : \mathbf{v}|_T \in (P^2(T))^2, \forall T \in \mathscr{T}_h, \mathbf{v}|_{\partial\Omega_D} = 0, \mathbf{v}\cdot\overline{\mathbf{n}}|_{\partial\Omega_N} = 0\} \quad (43)$$

$$Q_h = \{q \in C^0(\Omega) \cap L^2(\Omega) : q|_T \in P^1(T), \forall T \in \mathscr{T}_h, \int_\Omega q = 0\}, \quad (44)$$

where $\partial\Omega_D$ denotes Dirichlet boundary, $\partial\Omega_N$ denotes Neumann-type boundary, and $\overline{\mathbf{n}}$ is the normal to $\partial\Omega_N$.

Multiplying Eq. (13) by $\mathbf{v_h} \in V_h$ and Eq. (14) by $q_h \in Q_h$, we get the weak formulation for the unknowns $(\mathbf{u_h}, p_h) \in V_h \times Q_h$:

$$\int_\Omega \mu(\nabla \mathbf{u_h} + (\nabla \mathbf{u_h})^t) : \nabla \mathbf{v_h} - \int_\Omega p_h \nabla \cdot \mathbf{v_h} = -\frac{1}{Ca}\int_\Omega F_s : \nabla \mathbf{v_h}, \quad (45)$$

$$\int_\Omega q_h \nabla \cdot \mathbf{u_h} = 0, \quad (46)$$

for all test functions $(\mathbf{v_h}, q_h) \in V_h \times Q_h$. See [Zheng, Lowengrub, Anderson, and Cristini (2005)] for a derivation.

To evaluate the surface stress in the above weak formulation, the normal vector $\mathbf{n}$ at a cell vertex is calculated from the level set function using a least squares fit similar to that described in section 3.1. The only differences are that here the origin of the local coordinate system is located at the cell vertex, and the fitting is over a stencil that consists of the vertex under consideration and all its nearest neighbors. The surfactant concentration at a vertex is obtained by extending the concentration off the interface through a linear interpolation. As in [Zheng, Lowengrub, Anderson, and Cristini (2005); Yang, James, Lowengrub, Zheng, and Cristini (2006)], in practice the Dirac delta function $\delta(\phi)$ is replaced by a smoothed version $\delta_\varepsilon(\phi)$, where $\delta_\varepsilon(\phi) = dH_\varepsilon/d\phi$ and H_ε is a smoothed Heaviside function, defined as

$$H_\varepsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\varepsilon, \\ \frac{1}{2}(1 + \frac{\phi}{\varepsilon} + \frac{1}{\pi}\sin(\frac{\pi\phi}{\varepsilon})) & \text{if } |\phi| \le \varepsilon, \\ 1 & \text{if } \phi > \varepsilon, \end{cases} \quad (47)$$

where the parameter ε is taken to be $2 \sim 4$ times the smallest mesh size. The local dynamic viscosity μ is also smoothed using the smoothed Heaviside function. Thus,

$$\mu = \lambda + (1-\lambda)H_\varepsilon(\phi). \quad (48)$$

The linear system resulting from the discretization is solved using an inexact Uzawa method [Elman and Golub (1994)]. Matrix inversions are performed using an SSOR preconditioned conjugate gradient method. See also [Zheng, Lowengrub, Anderson, and Cristini (2005)].

4 Code validation

A number of test problems are performed to validate the method. For convergence studies, we define errors in volume fraction, interfacial area, surfactant mass, and concentration as

$$E^f = \sum_{K\in\mathscr{T}_h} |f_K^{\text{computed}} - f_K^{\text{exact}}|V_K, \quad (49)$$

$$E^A = \sum_{K\in\mathscr{T}_h} |A_K^{\text{computed}} - A_K^{\text{exact}}|, \quad (50)$$

$$E^M = \sum_{K\in\mathscr{T}_h} |M_K^{\text{computed}} - M_K^{\text{exact}}|, \quad (51)$$

$$E^\Gamma = \sum_{K\in\mathscr{T}_h} |\Gamma_K^{\text{computed}} - \Gamma_K^{\text{exact}}|A_K^{\text{exact}}, \quad (52)$$

where V_K is the volume of the Kth grid cell. To quantify the mass error in each fluid phase, we define a time-averaged mass error as

$$\overline{E}^V = \frac{\sum_{n=1}^N |\sum_{K\in\mathscr{T}_h} f_K^n V_K^n - \sum_{K\in\mathscr{T}_h} f_K^0 V_K^0|\Delta t^n}{\sum_{n=1}^N \Delta t^n}, \quad (53)$$

where N is the number of time steps. Recall that our method conserves the total surfactant mass exactly. So, we do not tabulate the errors in total surfactant mass. Indeed, all tests we have performed show that the total surfactant mass is conserved.

4.1 Simple translation test

An initially circular fluid body of radius 0.25 centered in a $(-0.6, 0.6) \times (-0.6, 0.6)$ computational domain is translated by a uniform velocity field along the 45° domain diagonal. The flow velocity is specified and remains constant with unit components, but changes sign at times 0.25 and 0.75 respectively. Therefore, the fluid body should return to its initial position after 1 time unit without any interface deformation, allowing errors in volume fraction and interfacial area to be quantified with Eq. (49) and (50), respectively. The

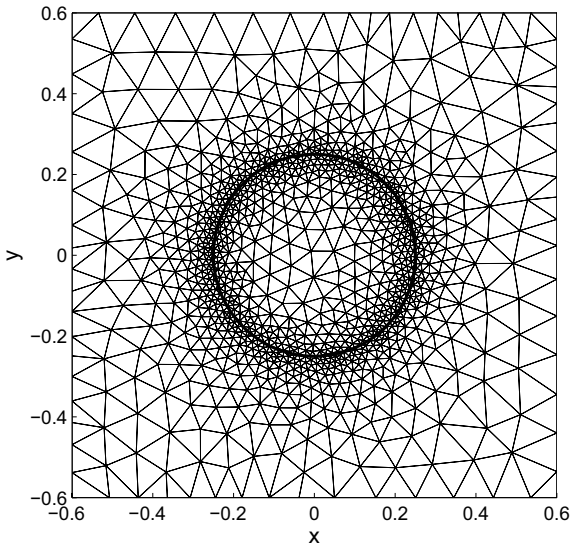

Figure 10: The final reconstructed interface and the computational grid for a simple translation test. $h_{\min} = 0.016$.

surfactant concentration is initially uniform, and it should remain uniform during a simple translation of the interface. Thus, errors in surfactant mass and concentration can be quantified with Eq. (51) and (52), respectively. To prevent surfactant diffusion from damping out any error in surfactant mass or concentration, surfactant diffusion is turned off in this test by setting Pe_s to be a very large number. In particular, we use $Pe_s = 10^{10}$. Although there is no interface deformation, surface area stretching, or surfactant diffusion, and the flow velocity is simply specified, this test provides a simple, direct evaluation of the overall algorithm (except the particular components listed above), including the Lagrangian advection, interface reconstruction, and mapping. Validation of the other specific components will be presented in subsequent sections.

Fig. 10 shows the final reconstructed interface, and the corresponding computational grid. Notice that the grid is refined near the interface. The grid cells at the interface have minimum edge lengths of roughly $h_{\min} = 0.016$. Various errors obtained on grids of different $h_{\min}$ and the corresponding convergence rates are tabulated in Table 1. It is shown that the interface capturing (measured by the E^f error) is third order accurate and the evolution of the interfacial area and surfactant mass is second order accurate. It is interesting to notice that errors in interfacial area and surfactant mass are exactly identical, and consequently the error in surfactant concentration is approximately zero. This is because the error in surfactant mass is solely due to the inaccuracy in representing/reconstructing the interface, which is also the only cause of the error in interfacial area in this test. Notice that because the velocity field is simply constant, the RK Lagrangian time integration and the calculation of the interfacial area stretching term are exact; that because surfactant diffusion is negligible, surfactant mass is locally conserved in each Lagrangian grid cell; and that the mapping processes themselves are exact. So, the error in interfacial area is solely due to a position mismatch caused by interface reconstruction. Finally, we see that the total mass error is identically zero, again because the time integration is exact for this test. So, in this test, the non-zero errors are essentially a measure of the accuracy in interface reconstruction.

4.2 Single vortex flow test

The single vortex flow test has been widely used by many researchers and has become one of the standard test problems for assessing the integrity and capability of an interface tracking/capturing method [Rider and Kothe (1995, 1998)]. In the present work, however, we use this test to assess not only the capability of the interface capturing part of our method, but also the capability of our method for evolving the interfacial area and the surfactant concentration.

The flow field is a time-reversing single vortex centered in a $(0,1)\times(0,1)$ computational domain (see Fig. 11(a)), and is defined by the stream function

$$\Psi = \frac{1}{\pi}\cos\left(\frac{\pi t}{T_f}\right)\sin^2(\pi x)\sin^2(\pi y), \tag{54}$$

where the velocity vector is defined by $(-\partial\Psi/\partial y, \partial\Psi/\partial x)$, and T_f determines the flow period. An initially circular fluid body of

Table 1: Errors and convergence rates for the simple translation test.

$h_{\min}$	E^f	order	E^A	order	E^M	order	E^Γ	$\overline{E}^V$
0.032	1.29E-6	-	2.08E-4	-	2.08E-4	-	<1.E-15	0
0.016	8.98E-8	3.84	3.33E-5	2.64	3.33E-5	2.64	<1.E-15	0
0.008	6.65E-9	3.76	5.97E-6	2.48	5.97E-6	2.48	<1.E-15	0

radius 0.15, centered at $(0.5, 0.75)$, is evolved in this vortex flow field. The surfactant is initially uniformly distributed along the interface.

In the first $T_f/2$ time units, the vortex spins the fluid elements in the clockwise direction, stretching the circular fluid body into a filament (see Fig. 11(b)). Thus, each fluid element can undergo very large deformation. The interface not only translates and rotates, but also deforms, stretches and compresses. Due to the time-reversing cosine term in the stream function, the flow reverses at time $T_f/2$. Consequently, any fluid element should return to its initial position at time T_f, and the final interface should exactly coincide with the initial one, allowing errors in volume fraction and interfacial area to be quantified with Eq. (49) and (50), respectively. Clearly, all the convection terms and the interfacial area stretching term are time-reversible. However, the surfactant diffusion term is not time-reversible. So, in this test, surfactant diffusion is turned off by setting $Pe_s = 10^{10}$. Fig. 11(c) shows the surfactant concentration along the interface shown in Fig. 11(b). Because the interface undergoes very complicated evolution, high surfactant concentrations develop at the nose and tail of the fluid body. Especially, the concentration gradient at the tail is remarkably large, and it will be larger when the fluid body deforms more under larger T_f, requiring more resolution at the tail to accurately describe the surfactant concentration. Without diffusion, the surfactant distribution, which has undergone very complicated evolution, should become uniform as it was initially, when the flow field reverses to its initial state, allowing errors in surfactant mass and concentration to be quantified with Eq. (51) and (52), respectively. Therefore, this test provides rigorous tests on volume fraction advection, interfacial area advection and stretching, and surfactant convection and dilution.

The errors and convergence rates are shown in Table 2 and Table 3 for $T_f = 0.5$ and $T_f = 1$, respectively. It is seen that the convection of volume fraction is second order, that the evolution of the interfacial area and surfactant mass is first order, and that the evolution of the surfactant concentration is second order. Because the time integration is performed using second order RK method, we can only achieve second order convergence in volume tracking. However, the error levels are much smaller than those obtained using piecewise linear interface reconstruction methods (see [Yang, James, Lowengrub, Zheng, and Cristini (2006)] for a comparison). In addition, notice that, similar to the translation test, the errors in interfacial area and surfactant mass are of similar orders, resulting in higher (second) order convergence in surfactant concentration. It is important that the concentration converges with second order accuracy because it is the concentration that is finally used to compute the surface tension.

4.3 Surfactant diffusion test

This test is specifically performed for assessing the discretization of the surfactant diffusion term. Surfactant diffusion along a circular interface of unit radius, centered in a $(-2,2)\times(-2,2)$ computational domain, is considered. The flow velocity is fixed at zero everywhere so that the interface is stationary. Thus, there is no convection of the volume fraction, interfacial area, and surfactant, and there is no interfacial area stretching and surfactant dilution. Only surfactant diffusion is present in this test problem. An analytic solution for this test problem is

$$\Gamma(\psi,t) = 1 + \exp\left(\frac{-t}{Pe_s}\right)\cos\psi, \tag{55}$$

Table 2: Errors and convergence rates for the single vortex flow test. $T_f = 0.5$.

$h_{\min}$	E^f	order	E^A	order	E^M	order	E^Γ	order	$\overline{E}^V$
0.032	2.39E-5	-	2.95E-3	-	3.20E-3	-	1.23E-3	-	5.88E-6
0.016	7.27E-6	1.72	1.79E-3	0.72	1.80E-3	0.83	2.95E-4	2.06	1.43E-7
0.008	1.36E-6	2.42	6.25E-4	1.52	6.27E-4	1.52	7.68E-5	1.94	4.75E-8
0.004	2.47E-7	2.46	2.25E-4	1.47	2.28E-4	1.46	1.73E-5	2.15	2.67E-8

Table 3: Errors and convergence rates for the single vortex flow test. $T_f = 1.0$.

$h_{\min}$	E^f	order	E^A	order	E^M	order	E^Γ	order	$\overline{E}^V$
0.032	6.48E-5	-	8.33E-3	-	1.09E-2	-	6.01E-3	-	1.33E-5
0.016	1.61E-5	2.00	3.45E-3	1.27	4.15E-3	1.39	1.76E-3	1.77	1.83E-6
0.008	3.76E-6	2.10	1.76E-3	0.97	1.89E-3	1.13	4.63E-4	1.93	1.64E-7
0.004	8.66E-7	2.12	8.20E-4	1.10	8.21E-4	1.20	8.37E-5	2.47	8.09E-8

Table 4: Errors and convergence rates for surfactant diffusion along a circular interface. Errors are computed at $t = 1$.

$h_{\min}$	E^Γ	order
0.2	1.14E-3	-
0.1	3.74E-4	1.61
0.05	8.03E-5	2.22
0.025	2.02E-5	1.99

where ψ is the angle formed by the position vector of a point on the interface and the y coordinate axis. The surfactant concentration is initialized with the exact solution and then allowed to diffuse in time numerically.

Fig. 12 shows the surfactant concentration profiles at various time instances. The result is obtained using an adaptive grid with $h_{\min} = 0.2$, a time step of 0.005, and $Pe_s = 1$. It is seen that the numerical solution agrees very well with the analytic solution. Table 4 shows the concentration errors at $t = 1$ on grids of different resolutions, and the corresponding convergence rates. It is seen that the discretization of surfactant diffusion is second order accurate.

Notice that, according to the exact solution given by Eq. (55), for any given Pe_s if one allows the surfactant to diffuse for a time of Pe_s units, one gets the same surfactant distribution at time Pe_s, that is $1 + \exp(-1)\sin\psi$. Furthermore, from Eq. (25) and (32) it is seen that if one chooses the time step such that $\Delta t/Pe_s$ is a constant, then one gets exactly the same numerical solution at time Pe_s for any Pe_s, when other conditions are the same. Therefore, the test for $Pe_s = 1$ applies to any Pe_s.

4.4 Surfactant redistribution through Marangoni convection

This test is performed to assess the implementation of the Marangoni force. Surfactant evolution along an initially circular interface of unit radius, centered in a $(-10,10)\times(-10,10)$ computational domain, is considered. Different from the previous diffusion test, the flow velocity at the boundary of the computational domain is fixed at zero, while the flow inside the domain is allowed to evolve in time. The surfactant concentration is initialized as

$$\Gamma(\psi) = 1 + \Gamma' \cos(2\psi), \tag{56}$$

where Γ' is a constant defining the amplitude of the nonuniformity. Since the concentration is not uniform, there is a Marangoni force along the interface. The Marangoni force causes interface

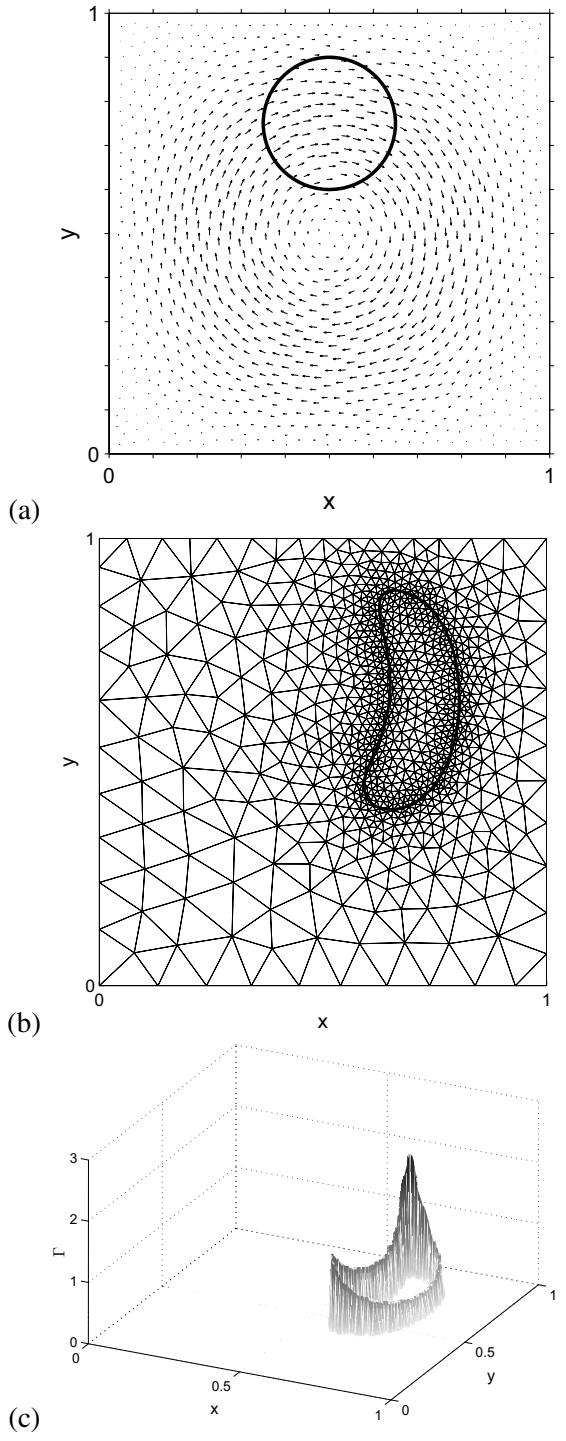

Figure 11: Evolution of a circular interface in a time-reversing single vortex flow field. (a) The initial interface and the single vortex flow field; (b) The reconstructed interface and the computational grid at $t = T_f/2$ for $T_f = 1$. $h_{\min} = 0.016$. (c) The surfactant concentration along the interface shown in (b).

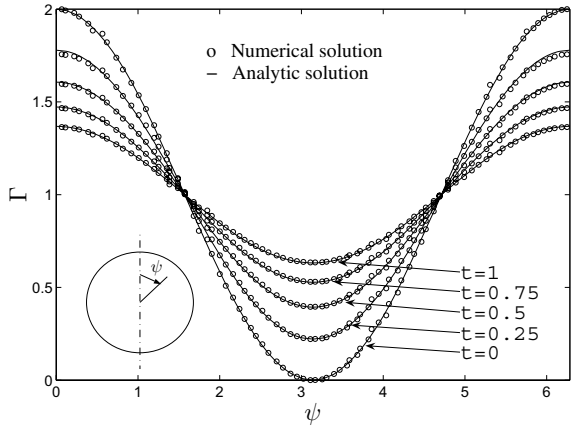

Figure 12: Surfactant concentration profiles at various time levels for surfactant diffusion test. $Pe_s = 1$. $h_{\min} = 0.2$.

motion and redistributes the surfactant in a way such that the surfactant distribution eventually becomes uniform.

For qualitative comparison, an analytic solution is developed. First, notice that, since the concentration defined by Eq. (56) is symmetric about both the x and y axes, the induced Marangoni force will also be symmetric about both axes. Consequently the concentration and the flow field will remain symmetric about the two axes as they evolve in time, and the drop will remain centered at the origin. Then, for simplicity, we assume a linear equation of state, a unit viscosity ratio, that the flow is stationary far away from the drop, and that drop deformation is negligible, which is a valid assumption under Stokes flow conditions (see [Chen and Stebe (1997)]). To isolate the effect of surfactant diffusion, the term that accounts for surfactant diffusion is eliminated from the governing equation for surfactant evolution. The length scale for the problem is the drop radius. The velocity scale is chosen to be σ_{eq}/μ_2, with which $Ca = 1$. Under these conditions, a Fourier series solution can be found (see Appendix A:). With the first three leading order terms in Γ', it is

$$\begin{aligned}\Gamma(\psi,t) &= 1+\Gamma' \exp(-\frac{1}{2}\beta t)\cos(2\psi) \\ &\quad -\frac{1}{2}\Gamma'^2 \beta t \exp(-\beta t)\cos(4\psi). \quad (57)\end{aligned}$$

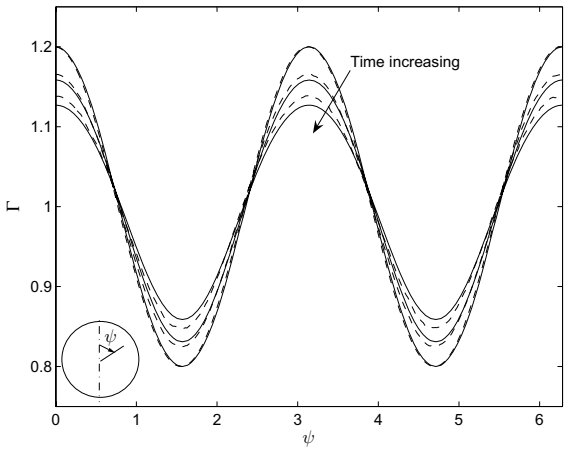

Figure 13: Surfactant concentration profiles at various time levels for the Marangoni test. Solid lines represent analytic solution. Dashed lines represent numerical solution. The curves correspond to $t = 0$, 2, and 4, respectively. Linear equation of state. $h_{\min} = 0.05$, $Pe_s = 10^{10}$, $\lambda = 1$, $Ca = 1$, $\beta = 0.2$, and $\Gamma' = 0.2$.

A case with $\beta = 0.2$ and $\Gamma' = 0.2$ is simulated and compared to the analytic solution. To suppress the surfactant diffusion effect in the numerical simulations, we set $Pe_s = 10^{10}$. Fig. 13 shows the evolution of the concentration profiles along the interface. The result is obtained on an adaptive grid with $h_{\min} = 0.05$. It is seen that with Marangoni convection the surfactant concentration tends to become uniform.

Fig. 14 shows the reconstructed interface and the velocity vectors at $t = 4$. It is clearly seen that the Marangoni force induces fluid circulation near the interface. In addition, the numerical result verifies the assumption that drop deformation is negligible. In all simulations, the radius of the drop differs from unity by less than 0.001.

Fig. 15 shows the concentration profiles at $t = 4$ obtained using grids of different resolutions. It is seen that the numerical solution monotonically converges towards the analytic solution. Concentration errors and convergence rates are quantified in Table 5. Again, the result indicates the convergence of the numerical solution.

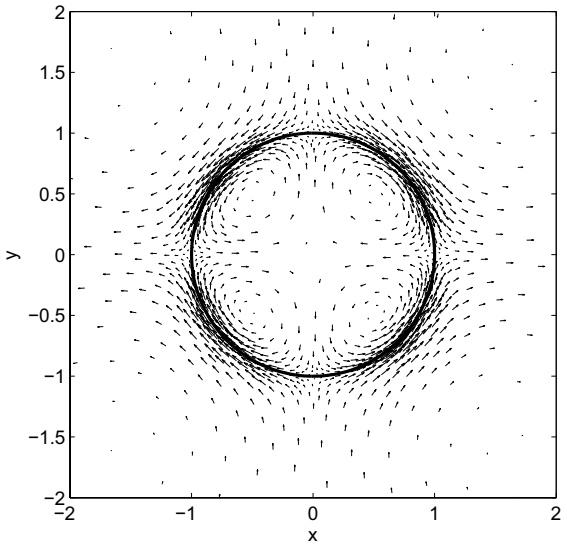

Figure 14: Reconstructed interface and velocity vectors at $t = 4$ for the Marangoni test. Linear equation of state. $h_{\min} = 0.05$, $Pe_s = 10^{10}$, $\lambda = 1$, $Ca = 1$, $\beta = 0.2$, and $\Gamma' = 0.2$.

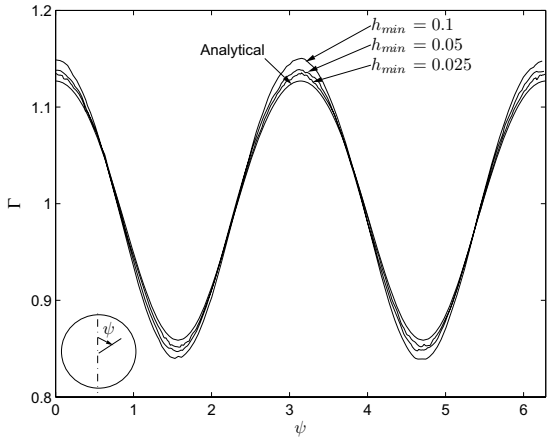

Figure 15: Surfactant concentration profiles at $t = 4$ for the Marangoni test. Linear equation of state. $Pe_s = 10^{10}$, $\lambda = 1$, $Ca = 1$, $\beta = 0.2$, and $\Gamma' = 0.2$.

Table 5: Errors and convergence rates for the Marangoni test. Errors are computed at $t = 4$.

$h_{\min}$	E^{Γ}	order
0.1	7.65E-2	-
0.05	4.40E-2	0.80
0.025	2.35E-2	0.90

5 Conclusions and future work

An ALE method has been developed for simulating interfacial flows with insoluble surfactant. By directly tracking the surfactant mass, the method conserves surfactant to machine accuracy, and prevents surfactant from diffusing off the interface. The method can be, in general, used with any equation of state, coupled to any flow solver, and can be extended to other types of grids. In the context of the ALE algorithm, the implementation of grid adaptation is convenient. The code can be extended in several ways by: incorporating surfactant solubility; coupling the interface capturing method and the surfactant evolution method to a Navier-Stokes solver; and extending the code to the axisymmetric or 3D cases. The code will be applied to study tip-streaming phenomenon in an extensional flow.

Acknowledgement: The authors thank the National Science Foundation (NSF-CTS-0448078) for support, and John Lowengrub, Vittorio Cristini and Xiaoming Zheng for the use of their adaptive mesh generator.

References

Adalsteinsson, D.; Sethian, J. A. (1999): The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, vol. 148, pp. 2–22.

Adalsteinsson, D.; Sethian, J. A. (2003): Transport and diffusion of material quantities on propagating interfaces via level set method. *J. Comput. Phys.*, vol. 185, pp. 271–288.

Anderson, A.; Zheng, X.; Cristini, V. (2005): Adaptive unstructured volume remeshing-I: The method. *J. Comput. Phys.*, vol. 208, pp. 616–625.

Anderson, D. M.; McFadden, G. B.; Wheeler, A. A. (1998): Diffuse-interface methods in fluid mechanics. *Annu. Rev. Fluid Mech.*, vol. 30, pp. 139–165.

Aulisa, E.; Manservisi, S.; Scardovelli, R. (2003): A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *J. Comput. Phys.*, vol. 188, pp. 611–639.

Badalassi, V. E.; Ceniceros, H. D.; Banerjee, S. (2003): Computation of multiphase systems with phase field models. *J. Comput. Phys.*, vol. 190, pp. 371–397.

Batchelor, G. K. (1967): *An introduction to fluid dynamics*. Cambridge University Press, New York.

Bertalmio, M.; Cheng, L. T.; Osher, S.; Sapiro, G. (2001): Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, vol. 174, no. 2, pp. 759–780.

Brackbill, J. U.; Kothe, D. B.; Zemach, C. (1992): A continuum method for modeling surface tension. *J. Comput. Phys.*, vol. 100, pp. 335–354.

Ceniceros, H. D. (2003): The effects of surfactants on the formation and evolution of capillary waves. *Phys. Fluids*, vol. 15, no. 1, pp. 245–256.

Chang, Y. C.; Hou, T. Y.; Merriman, B.; Osher, S. (1996): A level set formulation of Eulerian interface capturing methods for incompressible fluid flows. *J. Comput. Phys.*, vol. 124, pp. 449–464.

Chen, J.; Stebe, K. J. (1997): Surfactant-induced retardation of the thermocapillary migration of a droplet. *J. Fluid Mech.*, vol. 340, pp. 35–59.

Cristini, V.; Blawzdziewicz, J.; Loewenberg, M. (2001): An adaptive mesh algorithm for evolving surfaces: simulations of drop breakup and coalescence of interfaces in two-phase and free-boundary flows. *J. Comput. Phys.*, vol. 168, pp. 445–463.

DeBisschop, K. M.; Miksis, M. M.; Eckmann, D. M. (2002): Bubble rising in an inclined channel. *Phys. Fluids*, vol. 14, no. 1, pp. 93–106.

Eggleton, C. D.; Pawar, Y. P.; Stebe, K. J. (1999): Insoluble surfactants on a drop in an extensional flow: a generalization of the stagnated

surface limit to deforming interfaces. *J. Fluid Mech.*, vol. 385, pp. 79–99.

Eggleton, C. D.; Tsai, T.-M.; Stebe, K. (2001): Tip streaming from a drop in the presence of surfactants. *Phys. Rev. Lett.*, vol. 87, no. 048302.

Elman, H. C.; Golub, G. H. (1994): Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM J. Numer. Anal.*, vol. 31, no. 6, pp. 1645–1661.

Enright, D.; Fedkiw, R.; Ferziger, J.; Mitchell, I. (2002): A hybrid particle level set methods for improved interface capturing. *J. Comput. Phys.*, vol. 183, pp. 83–116.

Fedkiw, R. P.; Aslam, T.; Merriman, B.; Osher, S. (1999): A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, vol. 152, no. 2, pp. 457–492.

Glimm, J.; Grove, J.; Lindquist, B.; McBryan, O.; Tryggvason, G. (1988): The bifurcation of tracked scalar waves. *SIAM J. Sci. Stat. Comput.*, vol. 9, no. 1, pp. 61–79.

Glimm, J.; Grove, J. W.; Li, X.-L.; Shyue, K.-M.; Zeng, Y.; Zhang, Q. (1998): Three-dimensional front tracking. *SIAM J. Sci. Comput.*, vol. 19, no. 3, pp. 703–727.

Harabetian, E.; Osher, S.; Shu, C.-W. (1996): An Eulerian approach for vortex motion suing a level set regularization procedure. *J. Comput. Phys.*, vol. 127, no. 1, pp. 15–26.

Harlow, F. H.; Welch, J. E. (1965): Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, vol. 8, no. 12, pp. 2182–2189.

Helenbrook, B. T.; Martinelli, L.; Law, C. K. (1999): A numerical method for solving incompressible flow problems with a surface of discontinuity. *J. Comput. Phys.*, vol. 148, no. 2, pp. 366–396.

Hirt, C. W.; Amsden, A. A.; Cook, J. L. (1974): An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.*, vol. 14, no. 3, pp. 227–253. Reprinted in Vol. 135, Issue 2, 1997, pages 203-216.

Hirt, C. W.; Nichols, B. D. (1981): Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.*, vol. 39, pp. 201–225.

Hou, S.; Liu, X.-D. (2005): A numerical method for solving variable coefficient elliptic equation with interfaces. *J. Comput. Phys.*, vol. 202, no. 2, pp. 411–445.

Hou, T. Y.; Lowengrub, J. S.; Shelley, M. J. (2001): Boundary integral methods for multicomponent fluids and multiphase materials. *J. Comput. Phys.*, vol. 169, no. 2, pp. 302–362.

Hu, Y. T.; Lips, A. (2003): Estimating surfactant surface coverage and decomposing its effect on drop deformation. *Phys. Rev. Lett.*, vol. 91, no. 4, pp. 044501.

Hyman, J. M. (1984): Numerical methods for tracking interfaces. *Physica D*, vol. 12, pp. 396–407.

Jacqmin, D. (1999): Calculation of two-phase Navier-Stokes flows using phase-field modeling. *J. Comput. Phys.*, vol. 155, pp. 96–127.

James, A. J.; Lowengrub, J. (2004): A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant. *J. Comput. Phys.*, vol. 201, no. 2, pp. 685–722.

Keestra, B. J.; Puyvelde, P. C. J. V.; Anderson, P. D.; Meijer, H. E. H. (2003): Diffuse interface modeling of the morphology and rheology of immiscible polymer blends. *Phys. Fluids*, vol. 15, no. 9, pp. 2567–2575.

Khismatullin, D. B.; Renardy, Y.; Renardy, M. (2006): Development and implementation of VOF-PROST for 3D viscoelastic liquid-liquid simulations. *J. Non-Newtonian Fluid Mech.*, vol. 140, no. 1-3, pp. 120–131.

Kim, J. S.; Kang, K.; Lowengrub, J. (2004): Conservative multigrid methods for Cahn-Hilliard fluids. *J. Comput. Phys.*, vol. 193, pp. 511–543.

Kruijt-Stegeman, Y. W.; van de Vosse, F. N.; Meijer, H. E. H. (2004): Droplet behavior in the presence of insoluble surfactants. *Phys. Fluids*, vol. 16, no. 8, pp. 2785–2796.

Lavrenteva, O. M.; Tsemakh, D.; Nir, A. (2005): Locomotion of a viscous drop, induced by the internal secretion of surfactant: Boundary effects. *FDMP: Fluid Dynamics & Materials Processing*, vol. 1, no. 2, pp. 131–152.

Lee, J.; Pozrikidis, C. (2006): Effect of surfactants on the deformation of drops and bubbles in Navier-Stokes flow. *Comput. Fluids*, vol. 35, pp. 43–60.

LeVeque, R.; Li, Z. (1994): The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, vol. 31, no. 4, pp. 1019–1044.

LeVeque, R.; Li, Z. (1997): Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, vol. 18, no. 3, pp. 709–735.

Li, X.; Pozrikidis, C. (1997): Effect of surfactants on drop deformation and on the rheology of dilute emulsions in Stokes flow. *J. Fluid Mech.*, vol. 341, pp. 165–194.

Li, Z. (1994): *The immersed interface method – a numerical approach for partial differential equations with interfaces*. PhD thesis, University of Washington, Seattle, USA, 1994.

Liu, X.-D.; Fedkiw, R. P.; Kang, M. (2000): A boundary condition capturing method for Poisson's equation on irregular domains. *J. Comput. Phys.*, vol. 160, no. 1, pp. 151–178.

Meleshko, V. (2003): Selected topics in the history of the two-dimensional biharmonic problem. *Appl. Mech. Rev.*, vol. 56, no. 1, pp. 33–85.

Milliken, W. J.; Stone, H. A.; Leal, L. G. (1993): The effect of surfactant on the transient motion of Newtonian drops. *Phys. Fluids*, vol. 5, no. 1, pp. 69–79.

Noh, W.; Woodward, P. (1976): SLIC (simple line interface calculation). *Lect. Notes Phys.*, vol. 59, pp. 330–340. in Proc. 5th Int. Conf. Fluid Dyn.

Osher, S.; Fedkiw, R. P. (2001): Level set methods: An overview and some recent results. *J. Comput. Phys.*, vol. 169, pp. 463–502.

Osher, S. J.; Sethian, J. A. (1988): Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, vol. 79, pp. 12–49.

Pawar, Y.; Stebe, K. J. (1996): Marangoni effects on drop deformation in an extensional flow: The role of surfactant physical chemistry. I. Insoluble surfactants. *Phys. Fluids*, vol. 8, no. 7, pp. 1738–1751.

Peskin, C. S. (1977): Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, vol. 25, no. 1, pp. 220–252.

Pozrikidis, C. (2001): Interfacial dynamics for Stokes flow. *J. Comput. Phys.*, vol. 169, pp. 250–301.

Pozrikidis, C. (2004): A finite-element method for interfacial surfactant transport, with application to the flow-induced deformation of a viscous drop. *J. Eng. Math.*, vol. 49, no. 2, pp. 163–180.

Price, G. R.; Reader, G. T.; Rowe, R. D.; Bugg, J. D. (1998): A piecewise parabolic interface calculation for volume tracking. In *Proceedings of the Sixth Annual Conference of the Computational Fluid Dynamics Society of Canada*, volume VIII, pp. 71–77.

Renardy, Y.; Renardy, M. (2002): PROST: A parabolic reconstruction of surface tension for the volume-of-fluid method. *J. Comput. Phys.*, vol. 183, pp. 400–421.

Renardy, Y. Y.; Renardy, M.; Cristini, V. (2002): A new volume-of-fluid formulation for surfactants and simulations of drop deformation under shear at a low viscosity ratio. *Eur. J. Mech. B/Fluids*, vol. 21, pp. 49–59.

Rider, W. J.; Kothe, D. B. (1995): Stretching and tearing interface tracking methods. Technical Report AIAA 95-1717, AIAA, 1995. The 12th AIAA CFD conference, San Diego.

Rider, W. J.; Kothe, D. B. (1998): Reconstructing volume tracking. *J. Comput. Phys.*, vol. 141, pp. 112–152.

Scardovelli, R.; Zaleski, S. (1999): Direct numerical simulation of free-surface and interfacial flow. *Annu. Rev. Fluid Mech.*, vol. 31, pp. 567–603.

Sethian, J. A.; Smereka, P. (2003): Level set methods for fluid interfaces. *Annu. Rev. Fluid Mech.*, vol. 35, pp. 341–372.

Stegeman, Y. W. (2002): *Time dependent behavior of droplets in elongational flow*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.

Stone, H. A. (1990): A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface. *Phys. Fluids A*, vol. 2, no. 1, pp. 111–112.

Stone, H. A.; Leal, L. G. (1990): The effects of surfactants on drop deformation and breakup. *J. Fluid Mech.*, vol. 220, pp. 161–186.

Sunday, D. (2002): Fast polygon area and Newell normal computation. *J. Graphics Tools: JGT*, vol. 7, no. 2, pp. 9–13.

Sussman, M. (2003): A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.*, vol. 187, pp. 110–136.

Sussman, M.; Puckett, E. G. (2000): A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, vol. 162, pp. 301–337.

Sutherland, I. E.; Hodgeman, G. W. (1974): Reentrant polygon clipping. *ACM Transactions on Graphics*, vol. 27, no. 1, pp. 32–42.

Tryggvason, G.; Bunner, B.; Esmaeeli, A.; Juric, D.; Al-Rawahi, N.; Tauber, W.; Han, J.; Nas, S.; Jan, Y.-J. (2001): A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, vol. 169, pp. 708–759.

Unverdi, S. O.; Tryggvason, G. (1992): A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, vol. 100, pp. 25–37.

W. Dettmer, P. H. Saksono, D. P. (2003): On a finite element formulation for incompressible Newtonian fluid flows on moving domains in the presence of surface tension. *Commun. Numer. Meth. En.*, vol. 19, no. 9, pp. 659–668.

Wong, H.; Rumschitzki, D.; Maldarelli, C. (1996): On the surfactant mass balance at a deforming fluid interface. *Phys. Fluids*, vol. 8, no. 11, pp. 3203–3204.

Xu, J.-J.; Li, Z.; Lowengrub, J.; Zhao, H.-K. (2006): A level set method for interfacial flows with surfactant. *J. Comput. Phys.*, vol. 212, no. 2, pp. 590–616.

Xu, J.-J.; Zhao, H.-K. (2003): An Eulerian formulation for solving partial differential equations along a moving interface. *J. Sci. Comput.*, vol. 19, pp. 573–594.

Yang, X.; James, A. J.; Lowengrub, J.; Zheng, X.; Cristini, V. (2006): An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids. *J. Comput. Phys.*, vol. 217, pp. 364–394.

Yon, S.; Pozrikidis, C. (1998): A finite-volume/boundary-element method for flow past interfaces in the presence of surfactants, with application to shear flow past a viscous drop. *Comput. Fluids*, vol. 27, pp. 879–902.

Zhang, L.; Gerstenberger, A.; Wang, X.; Liu, W. K. (2004): Immersed finite element method. *Comput. Methods Appl. Mech. Engrg.*, vol. 193, no. 21, pp. 2051–2067.

Zheng, X.; Lowengrub, J.; Anderson, A.; Cristini, V. (2005): Adaptive unstructured vol-

ume remeshing-II: Application to two- and three-dimensional level-set simulations of multiphase flow. *J. Comput. Phys.*, vol. 208, pp. 626–650.

Appendix A: An analytic solution for the Marangoni convection about a circular interface

An analytic solution is developed for the Marangoni convection test problem performed in section 4.4. For convenience, the equations and variables are recast in the standard polar coordinates (r,θ). We use the conditions and assumptions described in section 4.4. All variables are nondimensionalized as described in section 4.4. The Stokes equations are rewritten in a stream-function form so that in the drop fluid and the surrounding fluid the following biharmonic equation holds

$$\nabla^4\Psi^{(i)} = 0, \tag{58}$$

where $\nabla^4 = \nabla^2\nabla^2$ is the biharmonic operator, Ψ is the stream-function, and $i = 1$ or 2 denotes the solution in drop fluid or in the surrounding fluid, respectively. By definition, the radial and azimuthal velocities are $u = \frac{1}{r}\frac{\partial\Psi}{\partial\theta}$ and $v = -\frac{\partial\Psi}{\partial r}$, respectively. There is a general solution to the biharmonic equation [Meleshko (2003)]. We start from the following 2π-periodic general solution

$$\begin{aligned}\Psi = &\\ &A_0r^2 + B_0r^2(\ln r - 1) + C_0\ln r\\ &+ (A_1r + B_1r^{-1} + C_1r^3 + D_1r\ln r)\cos(\theta)\\ &+ (E_1r + F_1r^{-1} + G_1r^3 + H_1r\ln r)\sin(\theta)\\ &+ \sum_{n=2}^{\infty}(A_nr^n + B_nr^{-n} + C_nr^{n+2} + D_nr^{-n+2})\cos(n\theta)\\ &+ \sum_{n=2}^{\infty}(E_nr^n + F_nr^{-n} + G_nr^{n+2} + H_nr^{-n+2})\sin(n\theta),\end{aligned} \tag{59}$$

where $A_0,\cdots,H_n$ are time-dependent coefficients to be determined. This general solution applies to both $\Psi^{(1)}$ and $\Psi^{(2)}$. As it will be seen next, the terms that appear in solution $\Psi^{(1)}$ will not appear in solution $\Psi^{(2)}$, and vice versa. So, it is not necessary to use two sets of coefficients for $\Psi^{(1)}$ and $\Psi^{(2)}$, respectively.

The next step is to determine the coefficients by applying boundary conditions. The following conditions are applied in turn:

1. At $r = 0$, the velocity and pressure or the viscous stress must be finite. So, $u^{(1)}|_{r=0} = \frac{1}{r}\frac{\partial\Psi^{(1)}}{\partial\theta}|_{r\to0}$, $v^{(1)}|_{r=0} = -\frac{\partial\Psi^{(1)}}{\partial r}|_{r\to0}$, and $\tau^{(1)}_{r\theta}|_{r=0} = \left(\frac{\partial v^1}{\partial r} - \frac{v^1}{r} + \frac{1}{r}\frac{\partial u^1}{\partial\theta}\right)|_{r\to0} = \left(-\frac{\partial^2\Psi^1}{\partial r^2} + \frac{1}{r}\frac{\partial\Psi^1}{\partial r} + \frac{1}{r^2}\frac{\partial^2\Psi^1}{\partial\theta^2}\right)|_{r\to0}$ must be finite, where $\tau^i_{r\theta} = \frac{\partial v^i}{\partial r} - \frac{v^i}{r} + \frac{1}{r}\frac{\partial u^i}{\partial\theta}$ is the viscous shear stress in the azimuthal direction.

2. Far away from the drop, the velocity of the surrounding fluid approaches zero. So, $u^{(2)}|_{r\to\infty} = \frac{1}{r}\frac{\partial\Psi^{(2)}}{\partial\theta}|_{r\to\infty} = 0$ and $v^{(2)}|_{r\to\infty} = -\frac{\partial\Psi^{(2)}}{\partial r}|_{r\to\infty} = 0$.

3. Because drop deformation is negligible and the velocity is continuous across the interface, $u^{(1)}|_{r=1} = u^{(2)}|_{r=1} = 0$, or $\frac{1}{r}\frac{\partial\Psi^{(1)}}{\partial\theta}|_{r=1} = \frac{1}{r}\frac{\partial\Psi^{(2)}}{\partial\theta}|_{r=1} = 0$.

4. Also, by velocity continuity, $v^{(1)}|_{r=1} = v^{(2)}|_{r=1}$, or $-\frac{\partial\Psi^{(1)}}{\partial r}|_{r=1} = -\frac{\partial\Psi^{(2)}}{\partial r}|_{r=1}$.

5. Lastly, the jump in the viscous shear stress in the azimuthal direction across the interface balances the Marangoni force. So, $\tau^{(2)}_{r\theta}|_{r=1} - \tau^{(1)}_{r\theta}|_{r=1} = -\nabla_s\sigma = \beta\frac{\partial\Gamma}{\partial\theta}$ for linear equation of state. Because we have assumed no drop deformation, the conventional normal force balance is not applied.

To successfully apply condition 5, we assume that Γ has the following general Fourier series solution

$$\Gamma(t,\theta) = 1 + \sum_{n=1}^{\infty}(I_n\cos(n\theta) + J_n\sin(n\theta)), \tag{60}$$

where I_n and J_n are time-dependent coefficients. By applying all these conditions, one ends up with

$$\begin{aligned}\Psi^1 \;=\; &-\frac{\beta}{8}\sum_{n=2}^{\infty}J_n(r^n - r^{n+2})\cos(n\theta)\\ &+\frac{\beta}{8}\sum_{n=2}^{\infty}I_n(r^n - r^{n+2})\sin(n\theta),\end{aligned} \tag{61}$$

$$\Psi^2 = -\frac{\beta}{8}\sum_{n=2}^{\infty} J_n(r^{-n}-r^{-n+2})\cos(n\theta) + \frac{\beta}{8}\sum_{n=2}^{\infty} I_n(r^{-n}-r^{-n+2})\sin(n\theta), \quad (62)$$

and from which the tangential velocity at the interface is

$$\begin{aligned} v_s &= v^{(1)}|_{r=1} \\ &= v^{(2)}|_{r=1} \\ &= -\frac{\beta}{4}\sum_{n=2}^{\infty} J_n\cos(n\theta) + \frac{\beta}{4}\sum_{n=2}^{\infty} I_n\sin(n\theta) \end{aligned} \quad (63)$$

In addition, I_1 and J_1 must be zero. Physically, if I_1 or J_1 is not zero, then the concentration, and consequently the induced Marangoni force and the flow field, will not be symmetric about both the x and y axes, and the drop will migrate towards where the concentration is higher or the surface tension is lower (see [Chen and Stebe (1997); Lavrenteva, Tsemakh, and Nir (2005)] for example). This violates our boundary condition that the flow velocity far away from the drop is zero. One can incorporate the case where I_1 or J_1 is not zero simply by modifying the boundary condition so that far away from the drop the fluid moves relative to the drop with some time-dependent velocity, which will be naturally determined by applying the above conditions with the far field velocity condition modified. Also, notice that by applying condition 4 the coefficients for Ψ^1 and Ψ^2 have been related, which are further related to the coefficients of the Fourier series solution to Γ (Eq. (60)) by applying condition 5.

Lastly, the coefficients I_n and J_n are determined by solving the equation that governs the evolution of the surfactant concentration. Without surfactant diffusion and normal surface expansion (recall that the drop is assumed not to deform), the dimensionless surfactant evolution equation becomes

$$\frac{\partial\Gamma}{\partial t} + v_s\frac{\partial\Gamma}{\partial\theta} + \Gamma\frac{\partial v_s}{\partial\theta} = 0. \quad (64)$$

By substituting Eq. (60) and (63) into Eq. (64), after doing some algebra and collecting terms, one obtains

$$\begin{aligned} &\sum_{n=2}^{\infty}\left(\frac{dI_n}{dt}\cos(n\theta) + \frac{dJ_n}{dt}\sin(n\theta)\right) \\ &+ \sum_{n=2}^{\infty}\frac{n\beta}{4}\left(I_n\cos(n\theta) + J_n\sin(n\theta)\right) \\ &+ \sum_{n=2}^{\infty}\frac{n\beta}{4}\left((I_n^2 - J_n^2)\cos(2n\theta) + 2I_nJ_n\sin(2n\theta)\right) \\ &+ \sum_{n=2}^{\infty}\sum_{m>n}^{\infty}\frac{(m+n)\beta}{4}\left((I_nI_m - J_nJ_m)\cos((m+n)\theta)\right. \\ &\left.+ (I_nJ_m + I_mJ_n)\sin((m+n)\theta)\right) \\ &= 0. \end{aligned} \quad (65)$$

By equating the coefficients of the Fourier modes to zero, Eq. (65) can be transformed into a set of ordinary differential equations for I_n and J_n, which can be solved with the initial condition given by Eq. (56), for which $I_2|_{t=0} = -\Gamma'$ and coefficients of all other modes are initially zero (notice that $\theta = \psi - \pi/2$). The solution for Γ, including four leading terms in Γ', is

$$\begin{aligned} &\Gamma(t,\theta) = \\ &1 - \Gamma'\exp(-\frac{1}{2}\beta t)\cos(2\theta) \\ &- \frac{1}{2}\beta t(\Gamma')^2\exp(-\beta t)\cos(4\theta) \\ &- \frac{3}{8}(\beta t)^2(\Gamma')^3\exp(-\frac{3}{2}\beta t)\cos(6\theta). \end{aligned} \quad (66)$$

As mentioned above, if I_1 or J_1 is not zero, the drop will migrate towards where the concentration is higher or the surface tension is lower. Let us consider an initial concentration profile defined as

$$\Gamma(\theta) = 1 + \Gamma'\sin(\theta). \quad (67)$$

For this profile, the concentration is the highest at $\theta = \pi/2$ and the lowest at $\theta = (3/2)\pi$. The drop will migrate in the positive y direction. Let the migration speed be U_0, and define the polar coordinate system with its origin located at the drop center. Then, as $r \to \infty$, the surrounding fluid moves with a relative speed U_0 in the negative y direction. So, $u^{(2)}|_{r\to\infty} = \frac{1}{r}\frac{\partial\Psi^{(2)}}{\partial\theta}|_{r\to\infty} = -U_0\sin(\theta)$ and

$v^{(2)}|_{r\to\infty} = -\frac{\partial \Psi^{(2)}}{\partial r}|_{r\to\infty} = -U_0\cos(\theta)$. With other conditions the same as before, and going through the same procedure as above, one can find that the solution for Γ with four leading terms is

$$\begin{aligned}&\Gamma(t,\theta) = \\ &1+\Gamma' \exp(-\frac{1}{4}\beta t)\sin(\theta) \\ &+\frac{1}{4}\beta t(\Gamma')^2\exp(-\frac{1}{2}\beta t)\cos(2\theta) \\ &-\frac{3}{32}(\beta t)^2(\Gamma')^3\exp(-\frac{3}{4}\beta t)\sin(3\theta),\end{aligned} \tag{68}$$

and that the migration velocity is

$$U_0 = \frac{\beta}{8}J_1 = \frac{\beta}{8}\Gamma'\exp(-\frac{1}{4}\beta t). \tag{69}$$

The distance that the drop migrates in time t is

$$\int_0^t U_0(t')dt' = \frac{\Gamma'}{2}(1-\exp(-\frac{1}{4}\beta t)). \tag{70}$$

Letting $t\to\infty$, we can find the maximum distance that the drop can migrate is $\frac{\Gamma'}{2}$.