Front-Tracking Techniques for Multiphase Viscous Flow

AM205 Final Project

Dylan Nelson[1⋆]

[1] *Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA, 02138, USA*

14 December 2012

ABSTRACT

We investigate the numerical simulation of multiphase fluid flow problems in two dimensions. In particular, we implement a front-tracking approach where a number of discrete points represent the free interface between two fluid phases. This boundary is advected in time, and at each timestep we calculate the surface curvature and include a model for surface tension effects. The Lagrangian surface is coupled to a fixed, Cartesian grid mapped to a square domain. The incompressible Navier Stokes equations are used to model continuum fluid flow of both phases, which have different densities and physical viscosities. We use the projection technique to split the second order time update into an advection-diffusion step following by a pressure correction step to enforce the divergence free constraint, while the spatial discretization uses the finite volume approach with staggered, rectangular control volumes for pressure and velocity. We investigate the numerical accuracy and convergence of the curvature calculation, area conservation of a high density drop surrounded by low density air, and the generation of spurious numerical velocities. The approach is then used to simulate the bounce of a water drop off of a rigid boundary. A proof of concept boundary merger algorithm is presented to handle the topological change of two colliding water drops, and extensions to more accurate numerical methods and physical models are discussed.

Key words: numerical methods – multiphase flow – front tracking – computational fluid dynamics

1 INTRODUCTION

Multiphase flows are fluid dynamics systems where different fluid phases – liquid-gas mixtures for instance – are simultaneously present. In theory these could be different phases of the same material, e.g. liquid and vapor water, although they could also be different materials, e.g. liquid water and oil. The formation and breakup of bubbles and droplets of liquid in air or air in liquid are common examples of multiphase fluid interaction. Bubbles can carry thermal energy away from hot boundaries, as in boiling. Liquid jets can atomize into a large number of droplets, as in fuel combustion. Droplet collision and droplets impacting surfaces result in splashing and disruption. Air bubbles can form during the phase change from liquid to gas and rapidly collapse, leading to cavitation. Liquid-solid interactions are also a category of multiphase flows, in which case the interface often remains rigid though perhaps moves. However, flexible solids can change their shape in response to fluid motion, as with blood flow through arteries, and solidification or dissolution can lead to mass exchange between phases similarly to evaporation or condensation. Clearly a wealth of both engineering and natural fluid problems are multiphase in nature (see Prosperetti & Tryggvason 2007, for more examples), and the difficulty in obtaining analytical solutions or setting up well controlled experiments has motivated strong interest in the numerical simulation of multiphase flow.

Several numerical methods have been developed to solve multiphase flow problems in various regimes. These can be either Lagrangian or ALE, when the computational elements follow the flow to some extent, or Eulerian, where the computational grid is fixed in space (Caboussat 2005). In the Lagrangian case the interface between two fluid phases is followed with edge elements of a mesh (Hansbo 1992), although typically only small displacements are possible (for instance, fluid-structure interactions or small amplitude waves) and *general* motion leads to topological difficulties and mesh tangling (e.g. an ocean wave breaking onto itself). For a fixed grid or mesh, there are two broad classes of following and modeling the interface between two fluid phases. The

⋆ E-mail: dnelson@cfa.harvard.edu

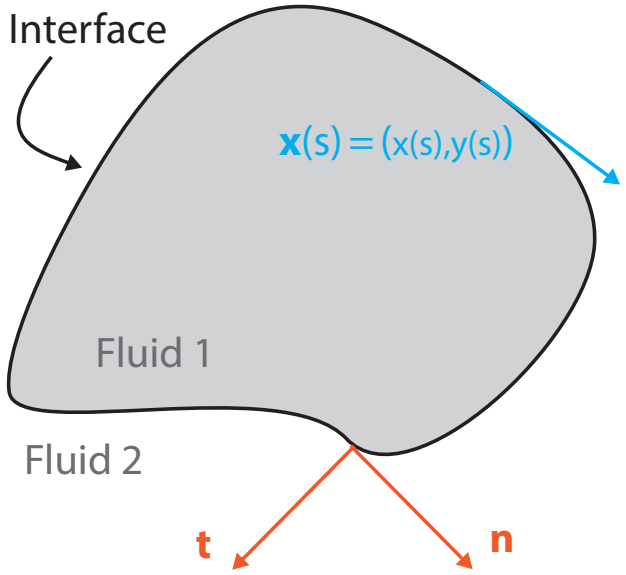

Figure 1. Schematic of the interface between two fluids. The coordinate s parameterizes distance along the front, while $\boldsymbol{n}$ is the normal and $\boldsymbol{t}$ the tangential vectors at any point $\boldsymbol{x}(s)$.

first is interface tracking, and the second is volume tracking. With interface tracking a fixed grid is coupled to Lagrangian tracer particles, which can either exist solely at the interface, or everywhere within one of the domains – e.g. the "markers and cells" (MAC) method (Harlow & Welch 1965) or the surface "front tracking" method (LeVeque & Shyue 1996). Two popular volume tracking techniques are the "level set" method (Chang et al. 1996), where the boundary is defined by the level line of some smooth function $\phi(\boldsymbol{x}, t)$ which is advected with the velocity field and can represent the signed distance of each point to the interface, and the "volume of fluid" (VOF) method (Hirt & Nichols 1981) which captures, rather than follows, the liquid and its interface. The largest difference between these methods arise during topological changes at the interface, which occur implicitly with volume tracking methods but must be handled explicitly with front tracking methods. Conversely, the interface curvature can be estimated directly from its representation in front tracking methods, whereas with volume tracking methods it must first be reconstructed from the ϕ function. In what follows we consider a moving surface front tracking algorithm coupled to a fixed grid incompressible Navier Stokes solver.

In §2 we present the mathematical foundations for the physical system we are trying to model. §3 discusses the numerical approach for the fluid solution in general, while §4 details the use of a tracked interface between two separate fluids. Next, §5 tests the implementation and verifies its accuracy, allowing us to explore a few more interesting simulations in §6. We conclude with a brief discussion of the results and future work in §7.

2 PHYSICAL MODEL

We consider the mathematical formulation of a domain consisting of two continuum fluids, such as that each is well described by the Navier Stokes equations. The interface between these two fluids is assumed to be infinitely sharp (zero thickness), such that the properties of the fluids change discontinuously across this surface. In reality, this transition occurs at molecular scales (e.g. van der Waals forces) and is unresolved in our simulations; we model this regime by including surface tension at the interfaces. The two fluids are treated as immiscible, that is, no mass exchange between them. Following (Tryggvason et al. 2001) we present the "one-fluid" approach, where the entire domain is modeled by the same set of equations while fluid properties such as density and viscosity can vary.

2.1 Incompressible Navier Stokes

We do not present a physically motivated derivation of the equations but present them as a mathematical model to which we seek a numerical solution. In conservative form, conservation of mass gives us

$$\frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0 \tag{1}$$

where $\rho(\boldsymbol{x}, t)$ is the fluid density, and $\boldsymbol{u}(\boldsymbol{x}, t)$ is the fluid velocity. We restrict our investigation to incompressible flow, in which case this reduces to

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2}$$

Momentum conservation gives us

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = -\nabla P + \rho \boldsymbol{g} + \nabla \cdot \mu(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}) + \boldsymbol{f} \tag{3}$$

where the stress tensor has been written in terms of the viscosity $\mu(\boldsymbol{x}, t)$. $P(\boldsymbol{x}, t)$ is the pressure, $\boldsymbol{g}$ is a gravitational acceleration and $\boldsymbol{f}$ any other body forces. The usual convention is to call the second $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ term the "advection term" and the second to last $\nabla \cdot \mu(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u})$ term the "diffusion term." A more compact form is then

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\frac{1}{\rho}\nabla P - \boldsymbol{A} + \frac{1}{\rho}\boldsymbol{D} + \boldsymbol{g} + \frac{1}{\rho}\boldsymbol{f} \tag{4}$$

2.2 Surface Tension

At small spatial scales the effect of surface tension becomes important. We model this phenomenon by adding a term to $\boldsymbol{f}$ corresponding to the usual model (Prosperetti & Tryggvason 2007) where

$$\boldsymbol{f}_\sigma = \sigma \kappa \boldsymbol{n} \delta(n) \tag{5}$$

where σ is the coefficient of surface tension (with units of force per length in two dimensions), and κ is the curvature such that

$$\kappa \boldsymbol{n} = \frac{\partial \boldsymbol{t}}{\partial s} \tag{6}$$

where $\boldsymbol{t}$ is the tangent vector to the interface and s parameterizes the curve representing the interface, shown schematically in Figure (1). The delta function $\delta(n)$ indicates that this body force is zero everywhere except at the interface, so it added as a "singular term". We compute surface tension along each segment of the interface Δs_k as

$$\boldsymbol{f}_\sigma^k = \sigma \int_{\Delta s_k} \frac{\partial \boldsymbol{t}}{\partial s} ds = \sigma \left(\boldsymbol{t}_{k+1/2} - \boldsymbol{t}_{k-1/2}\right). \tag{7}$$

This force, over each segment, is then smoothed onto the grid as discussed below in §4.2.

3 NUMERICAL SOLUTION

We describe a second order approach to solving the equations of §2 using an Eulerian, rectilinear grid and the finite volume technique. We consider only two dimensional problems.

3.1 Time Integration

We use the "projection" method and split the time stepping of the momentum equation into two parts. First, we solve for a velocity field $\boldsymbol{u}^\dagger$ neglecting the pressure term. From Equation (4) this means taking

$$\frac{\boldsymbol{u}^\dagger - \boldsymbol{u}^n}{\Delta t} = -\boldsymbol{A}^n + \frac{1}{\rho^n}\boldsymbol{D}^n + \boldsymbol{g} + \frac{1}{\rho^n}\boldsymbol{f} \tag{8}$$

where $\boldsymbol{u}^\dagger$ represents an intermediate velocity field which does not yet satisfy $\nabla \cdot \boldsymbol{u} = 0$, and the superscript n indicates that value at timestep number n. To obtain the final velocity we then add the pressure term

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^\dagger}{\Delta t} = -\frac{1}{\rho^n}\nabla P^n. \tag{9}$$

The pressure at each timestep P^n must be determined such that the resulting $\boldsymbol{u}^{n+1}$ is divergence free. This requires an iterative procedure; taking the divergence of Equation (9) we have

$$\frac{1}{\Delta t}\nabla \boldsymbol{u}^{n+1} - \frac{1}{\Delta t}\nabla \boldsymbol{u}^\dagger = -\nabla \frac{1}{\rho^n}\nabla P^n \tag{10}$$

and since we require $\nabla \cdot \boldsymbol{u}^{n+1} = 0$ then this reduces to

$$\frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^\dagger = \nabla \cdot \left(\frac{1}{\rho^n}\nabla P^n\right). \tag{11}$$

We will again follow Tryggvason et al. (2001) and use the "successive over relaxation" (SOR) technique to solve for P^n at each timestep. This is written out explicitly in the following subsection. Finally, note that Equation (8) as written is a first order in time, explicit Euler update. We modify this to be a second order, explicit, predictor-corrector type update by taking two substeps of $\Delta t/2$ (following the above procedure) and computing

$$\boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \frac{\Delta t}{2}\left((\frac{\partial \boldsymbol{q}}{\partial t})^n + (\frac{\partial \boldsymbol{q}}{\partial t})^{(n+1)}\right). \tag{12}$$

where $\boldsymbol{q}$ represents the velocity components, density, viscosity, or interface position. The interface position is updated (discussed in §4) and the density field is advected *indirectly*, by reconstructing the density over all of Ω from the location of the interface between the two fluids.

The only remaining aspect of the time integration is the allowed timestep. For two-dimensional flow (Tryggvason et al. 2011) we must have

$$\frac{\mu \Delta t}{\rho h^2} \leqslant \frac{1}{4} \quad ; \quad (\boldsymbol{u}\cdot\boldsymbol{u})\frac{\rho \Delta t}{\mu} \leqslant 2 \quad ; \quad \frac{u \Delta t}{h} \leqslant 1. \tag{13}$$

Where $h = \Delta x = \Delta y$ in our case. The first condition arises due to the diffusion term, while the second results from the advection term. We use both heuristically to set our timestep, but offer no error analysis for their derivation. Our tests will generally be sufficiently low velocity that these first two restrictions are more stringent than the third, which is the standard CFL condition.

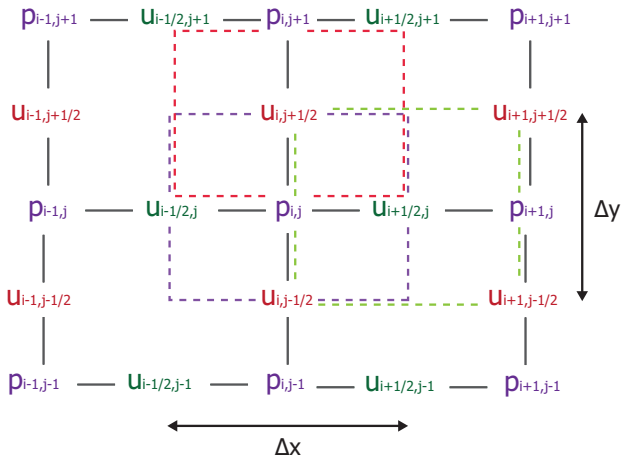

Figure 2. Schematic of the staggered grid. Pressure points (purple) are shown along with the horizontal velocity u_1 (colored green) and vertical velocity u_2 (colored red). The control volumes associated with each primitive variable are shown as dashed lines.

3.2 Spatial Discretization

The domain Ω is decomposed into a set of disjoint control volumes V_i. The average of any fluid quantity $\boldsymbol{q}$ within such a "cell" is given by the volume integral

$$\boldsymbol{q} = \frac{1}{V}\int_V \boldsymbol{q}(\boldsymbol{x})dv. \tag{14}$$

Our goal is to rewrite the terms of the momentum equation (3) in terms of surface fluxes. To do this we need Gauss' theorem to transform the volume integral of a quantity $\boldsymbol{q}$ into a surface integral with normal $\boldsymbol{n}$, as

$$\int_V (\nabla \cdot \boldsymbol{q})\, dv = \oint_S (\boldsymbol{q}\cdot\boldsymbol{n})\, ds. \tag{15}$$

The diffusion term can be written

$$\begin{aligned} \boldsymbol{D} &= \frac{1}{V}\int_V \nabla \cdot \mu\left(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}\right) dv \\ &= \frac{1}{V}\oint \mu\left(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}\right)\cdot \boldsymbol{n} ds. \end{aligned} \tag{16}$$

Likewise the advection term can be written (since $\boldsymbol{u}\cdot\nabla\boldsymbol{u} = (\nabla\cdot\boldsymbol{u})\boldsymbol{u}$) as

$$\boldsymbol{A} = \frac{1}{V}\int_V \boldsymbol{u}\cdot\nabla\boldsymbol{u} dv = \frac{1}{V}\oint_S \boldsymbol{u}\,(\boldsymbol{u}\cdot\boldsymbol{n})\, ds. \tag{17}$$

For simplicity and computational efficiency we take control volumes aligned with the coordinate axes, and have $V = \Delta x \Delta y$. The approach of Tryggvason et al. (2001) is to use a grid where the pressure and velocity components are stored on different, staggered grids, and we adopt this approach. In particular, the horizontal velocity component u_1 is stored on a grid displaced $\Delta x/2$, while the vertical velocity component u_2 is similarly shifted by $\Delta y/2$. We label the nodes of P by

(i,j) and the velocity nodes as $(i+1/2,j)$ or $(i,j+1/2)$ for example. Note that the fluid density and viscosity, for instance, are stored at the pressure nodes.

This arrangement staggered grid is shown schematically in Figure (2). The goal is that the primitives required at the midpoints of the finite volume faces are stored there. For instance, the center of the u_1 (horizontal) velocity cell is placed at the midpoint of the vertical face of the pressure cell. The cell spacing (we take $\Delta x = \Delta y$ throughout) is the distance between adjacent cell centers of any given primitive variable.

Given these conventions we can write discretized forms of the relevant equations. Firstly, using Gauss' theorem with $\nabla \cdot \boldsymbol{u}^{n+1}$ requires the dot product of $\boldsymbol{u}^{n+1}$ with the normal of each of the four faces of the pressure cells, integrated over those faces, vanish. Using the midpoint approximation, this divergence free requirement becomes

$$\begin{aligned}&\Delta y \left((u_1)^{n+1}_{i+1/2,j} - (u_1)^{n+1}_{i-1/2,j}\right)\\ &+\Delta x \left((u_2)^{n+1}_{i,j+1/2} - (u_2)^{n+1}_{i,j-1/2}\right) = 0.\end{aligned} \tag{18}$$

Next, the two components for the velocity update in Equation (8) become

$$\begin{aligned}(u_1)^{\dagger}_{i+1/2,j} = (u_1)^n_{i+1/2,j} + \Delta t \Big((-A_1)^n_{i+1/2,j} + \\ (g_1)^n_{i+1/2,j} + \frac{(D_1)^n_{i+1/2,j}}{\rho^n_{i+1/2,j}} + \frac{(f_1)^n_{i+1/2,j}}{\rho^n_{i+1/2,j}}\Big)\end{aligned} \tag{19}$$

$$\begin{aligned}(u_2)^{\dagger}_{i,j+1/2} = (u_2)^n_{i,j+1/2} + \Delta t \Big((-A_2)^n_{i,j+1/2} + \\ (g_2)^n_{i,j+1/2} + \frac{(D_2)^n_{i,j+1/2}}{\rho^n_{i,j+1/2}} + \frac{(f_2)^n_{i,j+1/2}}{\rho^n_{i,j+1/2}}\Big).\end{aligned} \tag{20}$$

Similarly, the two components for the velocity update from Equation (9) become

$$(u_1)^{n+1}_{i+1/2,j} = (u_1)^{\dagger}_{i+1/2,j} - \frac{\Delta t}{\rho^n_{i+1/2,j}} \frac{P^n_{i+1,j} - P^n_{i,j}}{\Delta x} \tag{21}$$

$$(u_2)^{n+1}_{i,j+1/2} = (u_2)^{\dagger}_{i,j+1/2} - \frac{\Delta t}{\rho^n_{i,j+1/2}} \frac{P^n_{i,j+1} - P^n_{i,j}}{\Delta y} \tag{22}$$

We can also write down the discretized advection term $\boldsymbol{A}^n$ following Equation (17) for each component

$$\begin{aligned}&(A_1)^n_{i+1/2,j} = \tfrac{1}{\Delta x}\left((u_1u_1)^n_{i+1,j} - (u_1u_1)^n_{i,j}\right)\\ &+\tfrac{1}{\Delta y}\left((u_1u_2)^n_{i+1/2,j+1/2} - (u_1u_2)^n_{i+1/2,j-1/2}\right)\end{aligned} \tag{23}$$

$$\begin{aligned}&(A_2)^n_{i,j+1/2} = \tfrac{1}{\Delta y}\left((u_2u_2)^n_{i,j+1} - (u_2u_2)^n_{i,j}\right)\\ &+\tfrac{1}{\Delta x}\left((u_1u_2)^n_{i+1/2,j+1/2} - (u_1u_2)^n_{i-1/2,j+1/2}\right)\end{aligned} \tag{24}$$

Similarly, the discretized diffusion term $\boldsymbol{D}^n$ following Equation (16) for each component gives

$$\begin{aligned}(D_1)^n_{i+1/2,j} = \tfrac{1}{\Delta x}\left(2(\mu\tfrac{\partial u_1}{\partial x})^n_{i+1,j} - 2(\mu\tfrac{\partial u_1}{\partial x})^n_{i,j}\right)\\ +\tfrac{1}{\Delta y}\Big(\left[\mu(\tfrac{\partial u_1}{\partial y} + \tfrac{\partial u_2}{\partial x})\right]^n_{i+1/2,j+1/2}\\ - \left[\mu(\tfrac{\partial u_1}{\partial y} + \tfrac{\partial u_2}{\partial x})\right]^n_{i+1/2,j-1/2}\Big).\end{aligned} \tag{25}$$

$$\begin{aligned}(D_2)^n_{i,j+1/2} = \tfrac{1}{\Delta y}\left(2(\mu\tfrac{\partial u_2}{\partial y})^n_{i,j+1} - 2(\mu\tfrac{\partial u_2}{\partial y})^n_{i,j}\right)\\ +\tfrac{1}{\Delta x}\Big(\left[\mu(\tfrac{\partial u_1}{\partial y} + \tfrac{\partial u_2}{\partial x})\right]^n_{i+1/2,j+1/2}\\ - \left[\mu(\tfrac{\partial u_1}{\partial y} + \tfrac{\partial u_2}{\partial x})\right]^n_{i-1/2,j+1/2}\Big).\end{aligned} \tag{26}$$

Finally we consider the pressure equation. Substituting Equations (21) and (22) into the velocity components in Equation (18) we have

$$\begin{aligned}&\tfrac{1}{\Delta x^2}\left(\tfrac{P^n_{i+1,j} - P^n_{i,j}}{\rho^n_{i+1/2,j}} - \tfrac{P^n_{i,j} - P^n_{i-1,j}}{\rho^n_{i-1/2,j}}\right)\\ &+\tfrac{1}{\Delta y^2}\left(\tfrac{P^n_{i,j+1} - P^n_{i,j}}{\rho^n_{i,j+1/2}} - \tfrac{P^n_{i,j} - P^n_{i,j-1}}{\rho^n_{i,j-1/2}}\right)\\ &-\tfrac{1}{2\Delta t}\Bigg(\tfrac{(u_1)^{\dagger}_{i+1/2,j} - (u_1)^{\dagger}_{i-1/2,j}}{\Delta x}\\ &+\tfrac{(u_2)^{\dagger}_{i,j+1/2} - (u_2)^{\dagger}_{i,j-1/2}}{\Delta y}\Bigg) = 0.\end{aligned} \tag{27}$$

If we then let $\mathcal{G}$ be the last two terms (involving $\boldsymbol{u}^{\dagger}$), $\mathcal{F}$ the four terms containing $P^n_{\alpha,\beta}$ for $\alpha \neq \beta$, and $\mathcal{H}$ the four terms containing $P^n_{i,j}$ though with $P^n_{i,j}$ factored out, we then have $P^n_{i,j} = \mathcal{H}^{-1}(\mathcal{F} - \mathcal{G})$. Because $P^n_{i,j}$ is calculated with a stencil over neighbors, this is implicit, and we seek an iterative solution for $k = 1, 2, ...$ by writing

$$P^{(k+1)}_{i,j} = \mathcal{H}^{-1(k)}\left(\mathcal{F}^{(k)} - \mathcal{G}^{(k)}\right). \tag{28}$$

Introducing ω the "relaxation factor" we write a modified version of the iteration step as a weighted sum of the previous and successive values

$$P^{(k+1)}_{i,j} = \omega\mathcal{H}^{-1(k)}\left(\mathcal{F}^{(k)} - \mathcal{G}^{(k)}\right) + (1-\omega)P^{(k)}_{i,j}. \tag{29}$$

Here ω is a free parameter of the SOR method, we generally take $\omega = 3/2$ ($\omega > 1$ accelerates the convergence rate) and terminate the iterations when the infinity norm between successive approximations drops below some tolerance, $||P^{(k+1)}_{i,j} - P^{(k)}_{i,j}||_{\text{inf}} < \epsilon_{\text{tol}}$.

The discretized forms of the velocity update equations have one problem, showing where the elegance of the staggered grid fails. In particular, several fluid quantities are required at grid points where they are not stored. In particular, the density and viscosity on the velocity cell centers, and some velocity components on the pressure cell centers. In all cases we use a linear interpolation (centered differencing) of the neighboring values. For instance,

$$\rho^n_{i+1/2,j} = \frac{1}{2}\left(\rho^n_{i+1,j} + \rho^n_{i,j}\right) \tag{30}$$

$$(u_\alpha)^n_{i+1,j} = \frac{1}{2}\left((u_\alpha)^n_{i+3/2,j} + (u_\alpha)^n_{u+1/2,j}\right) \tag{31}$$

$$\mu^n_{i+1/2,j+1/2} = \frac{1}{4}\left(\mu^n_{i+1,j} + \mu^n_{i+1,j+1} + \mu^n_{i,j+1} + \mu^n_{i,j}\right). \tag{32}$$

The final numerical consideration is the boundaries of the domain.

3.3 Boundary Conditions

We set the staggered grid such that the edges of the pressure control volumes coincide with the edges of the domain. For simplicity we take a rectangular domain Ω with four boundaries $\delta\Omega_{\text{left}}$, $\delta\Omega_{\text{right}}$, $\delta\Omega_{\text{top}}$, $\delta\Omega_{\text{bottom}}$. This means that the normal velocity (u_1 on the left/right, u_2 on the top/bottom) is known directly on the boundaries. For a rigid wall we set it directly to zero. To set the tangential velocity (and the pressure) we make use of ghost cells, requiring a single width ghost cell buffer for velocity, and likewise for pressure. We use the same linear interpolate (centered differencing) across

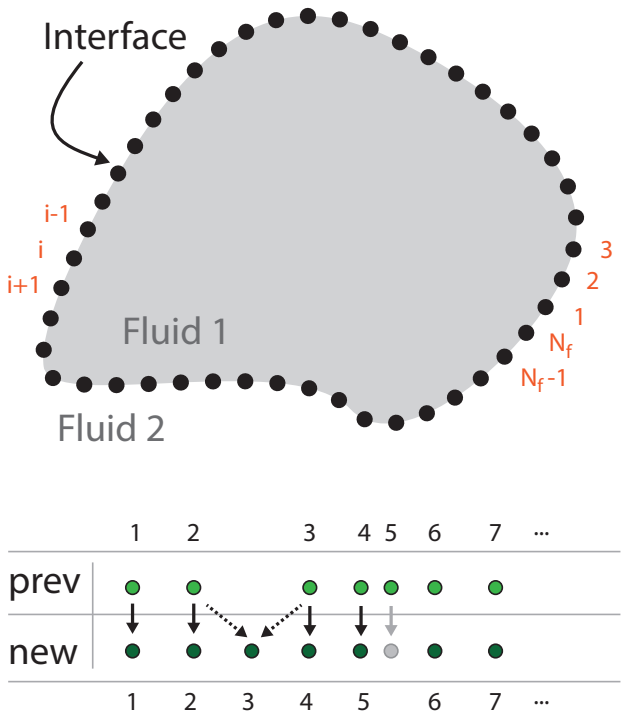

Figure 3. *(Top)* Schematic of the front representation by a series of linked points. *(Bottom)* Maintanence operations on the discrete front points to maintain roughly equal spacing.

the boundary as above. This means that, for tangential velocity zero ($u_1 = 0$ on $\partial\Omega_{\text{top}}$ for instance) we set the ghost cell velocity equal to negative of the first interior cell velocity, such that the sum is zero. In order to set the pressure at the boundary we take Equation (27) and set the term corresponding to the ghost pressure to zero.

4 THE INTERFACE

We represent the interface between the two fluids by a series of linked points $\boldsymbol{x}_i(s)$ for $i \in [1, ..., N_f]$, shown schematically in Figure (3). These points do not necessarily coincide with the fixed grid points, and we must interpolate quantities back and forth between the two types of points. In particular, the fluid velocity must be interpolated to the position of the front points so that they can be advected, while the density and viscosity discontinuity, as well as the surface tension force, defined at the position of the front points, must be interpolated back to the fixed grid.

4.1 Tracking and Maintaince

Consider a single front point with coordinates (x_f, y_f). Using a equi-spaced Cartesian grid makes it easy to locate the index of the grid point nearest a front point (simply dividing the position x_f by the grid spacing Δx for instance). We continue the same (bi)linear interpolation used previously, as with the viscosity. Then, a fluid quantity q at the front is given by the weighted sum

$$q_f = \sum_{k=1}^{4} w_k q_k. \tag{33}$$

where the four points $(i, j), (i, j+1), (i+1, j+1), (i+1, j)$ are those closest to the front point, and the weights w_k are the one dimensional distances to each grid point, normalized by the spacings Δx or Δy. Each velocity component u_i is interpolated to the front positions $\boldsymbol{x}_i$ which are then moved using an explicit Euler step

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \Delta t\, \boldsymbol{u}_i^n \tag{34}$$

although we substep this process, as with the velocity field update, to make it second order in time. Now, as the front is advected by the velocity field, its finite number of points may either spread out or bunch up. To efficiently calculate an accurate curvature for the front it is desirable to maintain a roughly constant spacing between the front points. To do this we adopt the "copy while inserting or deleting" approach. Each timestep, the linked point list is walked through and the distance between successive points is calculated. If this distance falls below some threshold (high linear density) we delete the successor point by omission in the new point list. If this distance is above some threshold (low linear density) we insert a new point at the midpoint of the line bisecting the two old points, thereby increasing our sampling of the interface. This process is shown in the bottom of Figure (3), where at the end of the maintenance a new point #3 has been added, while the old point #5 has been deleted.

4.2 Density Reconstruction

As indicated earlier, in this approach the fluid density $\rho(\boldsymbol{x}, t)$ is not advected directly via an upwind scheme, or similar methods. Rather, it is *reconstructed* at each timestep from the position of the interface. In particular, if fluid one has ρ_1 and fluid two has ρ_2 then the density everywhere away from the interface will be one of these two values. Near the interface, the zero width density discontinuity is mapped to a steep density gradient on the fixed grid. This is similar in spirit to smoothing out, and therefore capturing, a hydrodynamic shock. The density gradient can be related to the jump, which should be constant everywhere except for a single interface between the same two fluids, as

$$\nabla\rho = (\rho_2 - \rho_1)\boldsymbol{n}\delta(n). \tag{35}$$

As we expect the gradient is in the direction normal to the front. It is important in constructing this gradient that we conserve the total value. For some quantity $q(s)$ defined on the interface, what we require is that the integral of $q(s)$ along a segment equals the area integral of the smoothed quantity $q(\boldsymbol{x})$ on the grid, or

$$\int_A q_{i,j}(\boldsymbol{x})dA = \int_s q_f(s)ds \Rightarrow q_{i,j} = \sum_k q_f^k w_{i,j}^k \frac{A_k}{\Delta x \Delta y}. \tag{36}$$

We store the density, along with the viscosity, at the centers of the pressure control volumes. This means that the density gradients $(\partial\rho/\partial x)$ can be estimated at $\pm\Delta x/2$ from this point, and likewise $(\partial\rho/\partial y)$ can be estimated at $\pm\Delta y/2$ from this point (via central differencing). For example,

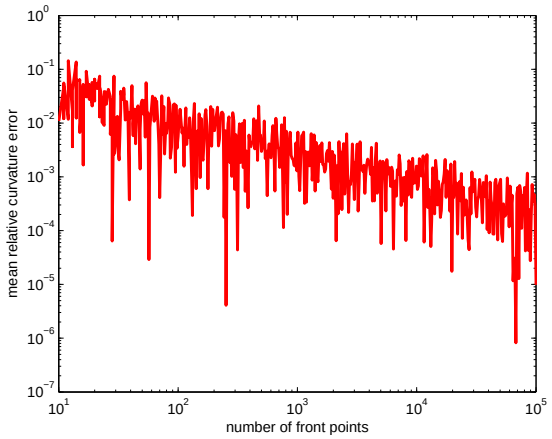

Figure 4. Measuring the relative error in the curvature for a interface point distribution placed in a circular arrangement with known radius.

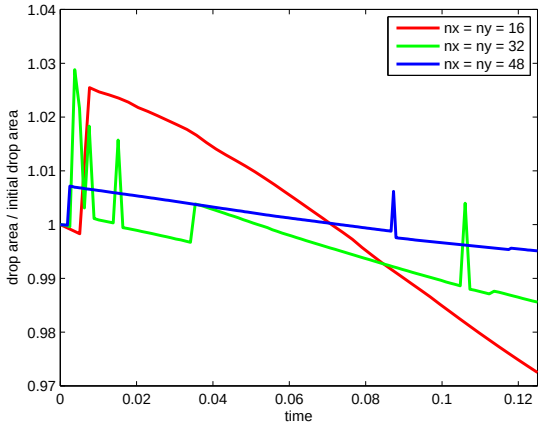

Figure 5. Time evolution of the ratio of the drop area to its initial area for three different numerical resolutions.

$$\left(\frac{\partial \rho}{\partial x}\right)_{i+1/2,j} = \frac{1}{\Delta x}\left(\rho_{i,j} - \rho_{i+1,j}\right). \tag{37}$$

For each point $\boldsymbol{x} = (x_i, x_j)$ we take the density gradients calculated in this manner from each of the four neighbors, sum them and divide by four, obtaining an estimate for $\rho_{i,j}$. Just as when determining the pressure $P_{i,j}$ this leads to an iterative solution for $\rho_{i,j}$, since the density is calculated on a stencil. We use the same SOR technique and iterate for $k = 1, 2, ...$ until some tolerance on

$$\begin{aligned}\rho_{i,j}^{(k+1)} = \omega\tfrac{1}{4}\left[\rho_{i,j-1} + \rho_{i,j+1} + \rho_{i-1,j} + \rho_{i+1,j}\right. \\ \Delta x\left((\tfrac{\partial\rho}{\partial x})_{i-1/2,j} - (\tfrac{\partial\rho}{\partial x})_{i+1/2,j}\right) \\ \left.+\Delta y\left((\tfrac{\partial\rho}{\partial y})_{i,j-1/2} - (\tfrac{\partial\rho}{\partial y})_{i,j+1/2}\right)\right] + (1-\omega)\rho_{i,j}^{(k)}.\end{aligned} \tag{38}$$

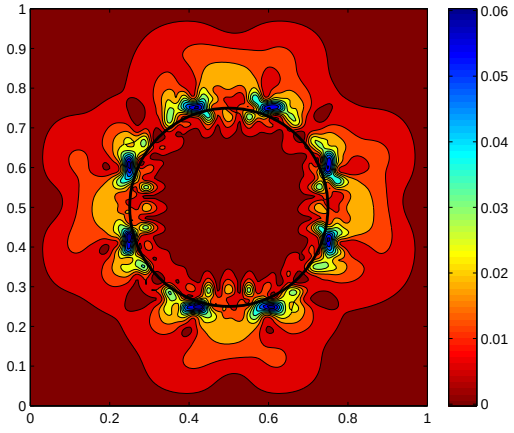

Figure 6. An example of the numerical growth of "parsitic currents" in the fluid velocity field.

5 NUMERICAL TESTS

Before moving on to more complex examples we explore the accuracy, stability and convergence behavior of the approach in simple test setups.

5.1 Accuracy of Curvature Calculation

The surface tension force depends on the accuracy with which the curvature of the interface is calculated. We attempt to recreate a test similar to that of Figure (4) of Tryggvason et al. (2001), by setting a number N_f of points in a circular arrangement with randomized angles. We compute the curvature κ and the radius of curvature $R = 1/\kappa$ at each point, and calculate the mean absolute value of the relative error between this value and the known radius. The convergence of this quantity with increasing N_f is shown in Figure (4). As the number of points in the front increases their linear density also increases, leading to better estimates of the circular shape, as expected. We see that we can control the error of the curvature estimate as desired by increasing N_f, although we generally choose just $N_f = 100$ (per drop) which achieves an order of $\simeq 10^{-2}$.

5.2 Area Conservation of a Drop

In various static situations it is useful to check the behavior of the interface and the interaction between the two fluids. In fact, in general we should always conserve the area of a drop since the two fluids are incompressible. We use the result that for any two dimensional simple polygon with n vertices the area is given in terms of the vertex coordinates as

$$A = \frac{1}{2}\sum_{i=0}^{n-1}\left(x_i y_{i+1} - x_{i+1} y_i\right). \tag{39}$$

The most basic test case is a circular drop in equilibrium in the center of the domain, which is $(x_c, y_c) = (0.5, 0.5)$ within $\Omega = [0,1] \times [0,1]$. With a radius $r = 0.25$ and the same fluid parameters as in §6.1 we measure the area conservation as a function of the fixed grid resolution $h = n_x = n_y \in [16, 32, 64]$. We scale the timestep as $1/h$ and evolve until the same final time, such that the total number of timesteps scales as h. The ratio of the area to the initial as a function of time in shown in Figure (5). We see that

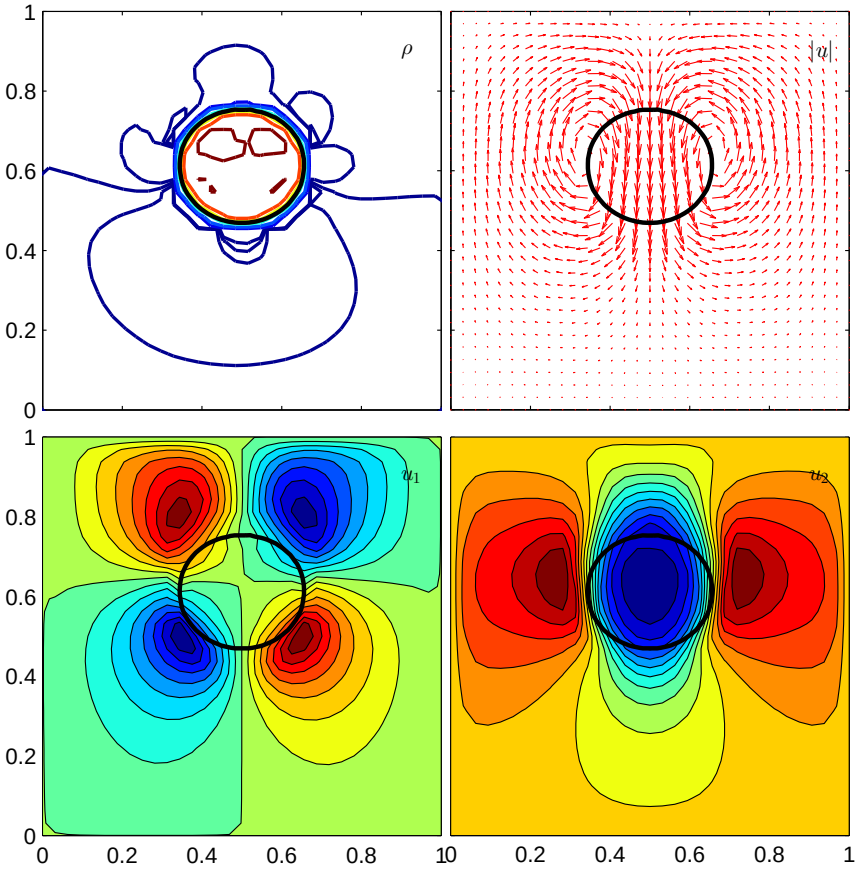

Figure 7. The density, velocity field, and individual u_1 and u_2 velocity components in the domain at $t = 0.075$, when the drop is in free-fall prior to its first bounce.

the area conservation improves with increasing fixed grid resolution, as expected. However, all resolutions exhibit a systematic trend with time – there is no mechanism to prevent the growth of numerical artifacts, of the type discussed below. The jumps in the area calculation arise when points are inserted or deleted from the interface. We use just the linear midpoint to place a new point, which could be improved by using a quadratic or spline fit to local interface points (a higher order calculation of the curvature could be done in the same spirit). An issue of points immediately being added and then removed is also apparent in the narrow spikes, which could be prevented. Overall though the equilibrium case seems to work well, and reasonable error levels are achieved at moderate fixed grid resolution.

5.3 Parasitic Current Generation

As discussed in Tryggvason et al. (2001), any anisotropies in the surface forces of such an equilibrium setup can lead to the generation of unphysical velocities called "parasitic currents". Indeed this appears to be the main mechanism which leads to a lack of area conservation in the previous example. In such a static case, the velocity field of the fluid should remain everywhere zero, but small pressure fluctuations lead to recirculation near the drop. These depend on both numerical and physical parameters in the system. We take $n_x = n_y = 50$ and for $r = 0.25$ after 250 timesteps plot the magnitude of the fluid velocity field in Figure (6). Their magnitude is actually rather significant after this period of time. Again it is likely our first order treatment of the curvature calculation that leads to this problem, although it could also be the smoothing and density reconstruction phases.

6 WATER DROP EXAMPLES

We present a few more interesting simulations using this approach, which could be thought of in analogy to a water drop moving through air. The domain is $\Omega = [0, 1] \times [0, 1]$.

6.1 Surface Tension: Drop Bounce

One drop is created at $(x_c, y_c) = (0.5, 0.75)$ with radius $r = 0.15$ with surface tension coefficient $\sigma = 10$, densities and viscosities $\rho_1 = 1$, $\mu_1 = 0.02$ (air) and $\rho_2 = 2$, $\mu_2 = 0.1$ (water). Gravity is included downward as $\boldsymbol{g} = -200\hat{\boldsymbol{y}}$. For

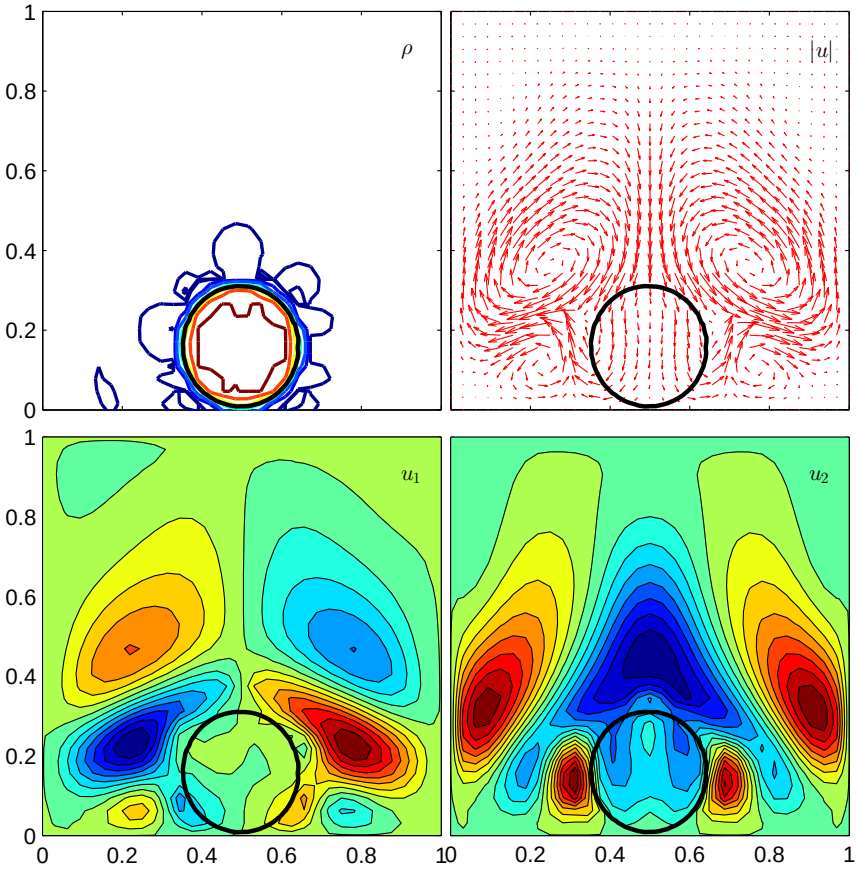

Figure 8. The density, velocity field, and individual u_1 and u_2 velocity components in the domain at the final time $t = 0.31125$, after 250 timesteps, when the drop has reached its maximum post-bounce height.

the fixed grid we use $n_x = n_y = 32$ and a timestep of $\Delta t = 0.00125$ for $n_{\text{step}} = 250$, which ends just as the drop reaches its maximal height after bouncing off the bottom domain boundary. Figure (9) shows the time evolution of the interface position and shape. As the downward velocity grows larger the drop accelerates due to the gravity, which causes an ellipsoidal distortion in the horizontal direction. This distortion is maximal when the velocity reaches zero at the middle of the bounce, at which point the ratio of semi-major to semi-minor axes of the drop is nearly 3:1. Increasing either the gravitational acceleration or the density contrast allows us to smash the drop into the bottom boundary with enough force that it reaches the left and right walls. Figure (7) shows the density, velocity field, and individual velocity components during the free fall phase, prior to the bounce. We can see that the velocity field exhibits excellent $\hat{x}$ symmetry. Errors in the density reconstruction at this spatial resolution are clear – we show contours at $\rho_1 \pm 0.001$ which demonstrate the level of error incurred throughout the domain. One approach that is mentioned in the literature is only reconstructed the density in the local neighborhood of the interface (we do it over the whole domain), and/or "cleaning" these spurious oscillations before they are allowed to modify the dynamics. Figure (8) shows the same four panels at the final time, after the drop has completed its first bounce. Counter-rotating vortices develop on either side of the drop. It is reassuring that the interface shape still appears symmetric in the horizontal direction. Figure (9) shows the time evolution of the interface shape and position for the drop bounce problem.

6.2 Topology Change: Drop Collision

There are two general types of topology changes: thin film rupture, and thread snapping. Physical models for how thin films "drain" are still a fairly modern topic, whereas the disconnection of a thin fluid thread is better described by the same Navier Stokes equations (Eggers 1995). Unfortunately, the collision (or coalescence) of two drops or a drop and a surface of the same fluid both fall into the thin film case. In particular, a thin film of air must evacuate from between the two water surfaces. If we simply neglect this difficulty, while also handling the necessary merging and splitting of the numerical front representation then, due to our finite timesteps, two drops should naturally merge when they ap-

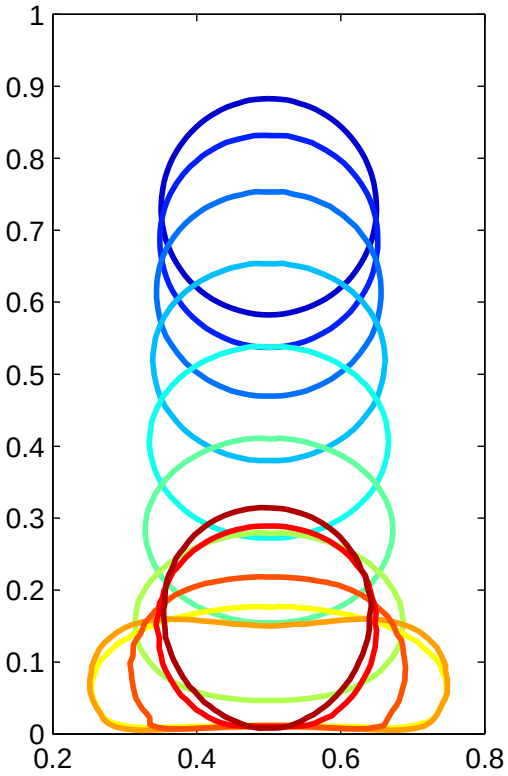

Figure 9. Time evolution of the shape and position of the interface between the water drop and surrounding air for the physical and numerical parameters as described in §6.1. The color indicates the time for each droplet, from t_0 (blue) to t_f (red).

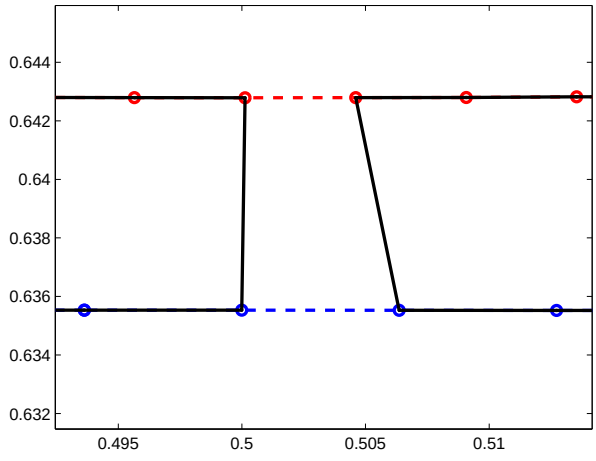

Figure 10. Demonstration of the binary front merger algorithm immediately after its application. The resulting unified interface is shown in black, while the two separate interfaces prior to the merge are shown in red and blue.

proach each other, although the timescale or resulting impact on the fluids may be incorrect.

We generalize our code to better handle two or more simultaneous fronts. At the end of each timestep we consider whether or not to merge two fronts, based on the condition that the minimum pairwise distance between points on either front drops to less than half the fixed grid spacing. We implement this search just as $\mathcal{O}(N^2)$ which works fine for two fronts in 2D. If the merger condition is satisfied, the closest pair of points between the two fronts is "bridged" as is the next sequential pair, while the connection between each successive point on a single front is cut.

This process is shown in Figure (10) which shows the new unified front (black) together with the prior two fronts (red and blue) immediately after the merge procedure. Clearly the front shape near the merge point is not captured with high accuracy, and this could be improved by pre-emptively refining the front in that neighborhood. Additionally, our algorithm handles only the simplest case, where the two fronts are merged into one after the collision. For instance, we cannot handle two near simultaneous merger points, nor a single front which gets deformed such that it would merge with itself (forming an air cavity in the center). Nor do we consider the opposite direction, i.e. drop breakup, where one interface breaks into two. The explicit consideration of all these different cases of topological change is one of the inherent complexities of the front tracking technique.

Using this simple algorithm we set up a two drop collision problem as follows. The first drop has $(x_{c1}, y_{c1}) = (0.5, 0.75)$ while the second is below it at $(x_{c2}, y_{c2}) = (0.5, 0.4)$. Both have the same radii of $r_1 = r_2 = 0.15$. With the same gravity we add an upward velocity of $u_{2,t=0} = 10$ in the initial conditions, only for grid points corresponding to the interior of the second (lower) drop. This kicks it upward as the upper drop slowly falls, and the two collide at $t = 0.03$. The corresponding time evolution of the front shapes is shown in Figure (11), where the dashed lines correspond to the lower drop and the solid lines to the upper. Just prior to the collision (light blue line) the lower drop has forced the upper into a mushroom shape. Immediately post-collision (green line) the combined drop rapidly evacuates the remaining air in the gap and oscillates as surface tension drives it towards a circular shape (orange and red lines).

7 DISCUSSION AND CONCLUSIONS

In this project we have implemented a simple, two dimensional compressible Navier Stokes solver using the projection method and the finite volume approach. On top of this we added an interface between two different fluids, represented by a moving set of discrete points. This allowed us to add the effects of surface tension to a number of test problems. We investigated the accuracy of the front related calculations as well as the smoothing of quantities from the front onto the fixed grid. Both suffer from our relatively "first order" choices in e.g. the curvature calculation, but are sufficient for testing purposes. We consider the case of a water drop bouncing off a rigid boundary, as well as the collisions of two water drops, showing the ability of the numerical method to handle basic multiphase flow scenarios.

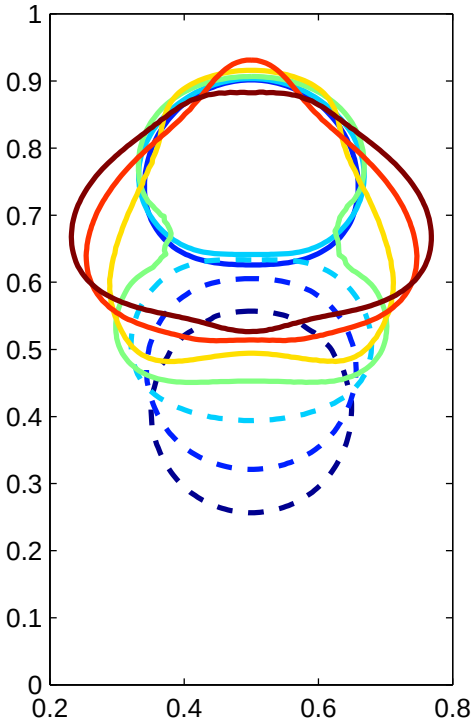

Figure 11. Time evolution of the interface shapes of two colliding drops. The lower drop (dashed) is kicked upwards, while the upper drop (solid) falls downward. The light blue and green lines indicate immediately before and after the collision, respectively.

We would like to ultimately include an explicit front tracking technique in the moving mesh code AREPO (described in Springel 2009). This code is a finite volume scheme where the control volumes are defined by a Voronoi tessellation of space. Euler's equations are solved using Godunov's method with the MUSCL-Hancock scheme to compute numerical fluxes and obtain second order accuracy. The tessellation is obtained by a set of mesh generating points which are allowed to move arbitrarily, though in our simulations follow the flow in a quasi-Lagrangian fashion. A recent extension solves the compressible Navier Stokes equations on this dynamic mesh (Muñoz et al. 2012). A natural approach to multiphase flow in this context would be to ensure that the interface is represented at all times by existing faces in the tessellation. An example of such tightly controlled interface motion is shown in Figure (12) from Springel (2009), where a rigid, curved "spoon" stirs a cup of coffee.

The moving mesh would make mapping between the fluid grid and interface position, as in the approach described in this paper, unnecessary. Reconstructing the density from the interface position would also be unnecessary. Surface tension "fluxes" could be added directly to the conservative equations. Alternatively, a (variable γ) Riemann type solution at the interface could be explored, and mass exchange due to phase change could be incorporated. The greatest strength of this approach would likely be the robust handling of topological changes in the fluid interface(s). The greatest difficulty would likely be accurately advecting the interface

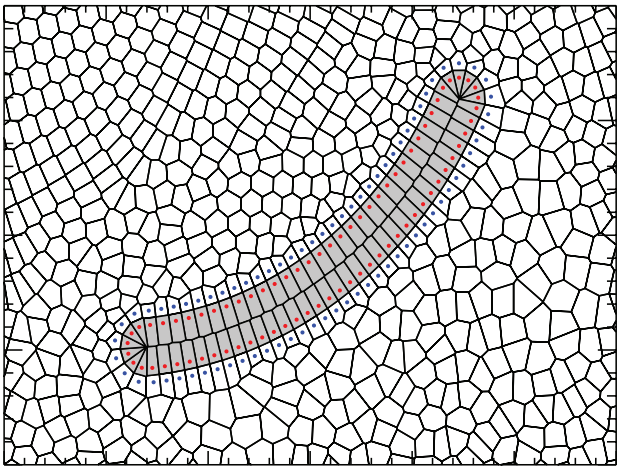

Figure 12. The "coffee spoon" example – a reflective, curved boundary condition with perscribed motion. Formed by two adjacent strings of 60 mesh-generating points (blue on fluid side, red 'outside'). Adapted from Springel (2009).

position via the cell center motions. Such an implementation would allow us to efficiently explore three dimensional problems, as well as multi-scale physics. The hierarchical adaptive timestepping scheme in AREPO would in principle enable a global simulation of a water drop while resolving the molecular scales ($\simeq$1 cm to $\simeq$1 $\mathring{A}$) which has a similar dynamic range as cosmological "zoom" simulations for which the code was originally designed.

REFERENCES

Caboussat A., 2005, Archives of Computational Methods in Engineering, 12, 165
Chang Y., Hou T., Merriman B., Osher S., 1996, Journal of Computational Physics, 124, 449
Eggers J., 1995, Physics of Fluids, 7, 941
Hansbo P., 1992, Computer Methods in Applied Mechanics and Engineering, 99, 171
Harlow F., Welch J., 1965, Physics of fluids, 8, 2182
Hirt C., Nichols B., 1981, Journal of computational physics, 39, 201
LeVeque R., Shyue K., 1996, Journal of Computational Physics, 123, 354
Muñoz D. J., Springel V., Marcus R., Vogelsberger M., Hernquist L., 2012, MNRAS, p. 63
Prosperetti A., Tryggvason G., 2007, Computational methods for multiphase flow. Cambridge University Press
Springel V., 2009, Monthly Notices of the Royal Astronomical Society, 401, 791
Tryggvason G., Bunner B., Esmaeeli A., Juric D., Al-Rawahi N., Tauber W., Han J., Nas S., Jan Y., 2001, Journal of Computational Physics, 169, 708
Tryggvason G., Scardovelli R., Zaleski S., 2011, Direct numerical simulations of gas-liquid multiphase flows. Cambridge University Press