An Adaptive Level Set Approach For Incompressible Two-Phase Flows[1]

Mark Sussman
Department of Mathematics
University of California Davis
Davis, CA 95616

Ann S. Almgren
John B. Bell
Phillip Colella
Louis H. Howell
Michael L. Welcome
Center for Computational Sciences and Engineering
Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Abstract

In Sussman, Smereka and Osher (1994), a numerical method using the level set approach was formulated for solving incompressible two-phase flow with surface tension. In the level set approach, the interface is represented as the zero level set of a smooth function; this has the effect of replacing the advection of density, which has steep gradients at the interface, with the advection of the level set function, which is smooth. In addition, the interface can merge or break up with no special treatment. We maintain the level set function as the signed distance from the interface in order to robustly compute flows with high density ratios and stiff surface tension effects. In this work, we couple the level set scheme to an adaptive projection method for the incompressible Navier-Stokes equations, in order to achieve higher resolution of the interface with a minimum of additional expense. We present two-dimensional axisymmetric and fully three-dimensional results of air bubble and water drop computations.

1 Introduction

The ability to compute incompressible two-phase flows has many applications. In a paper by Oguz and Prosperetti [23], the boundary integral method was used to study the cause of noise due to raindrops. A necessary capability of their method was the ability to compute air and water flow (density ratio of 1:816) and to compute with stiff surface tension effects. The problem of an air bubble rising up to the water surface and then bursting was studied by Boulton-Stone and Blake [9], also using the boundary integral method. This particular problem was studied in order to better understand the death of cells in a "bioreactor." The numerical simulation of the bubble growth and collapse in a thermal ink-jet print-head was described in a paper by Deshpande (1989) [14]. In that work, the volume-of-fluid method was used to track the interface between the ink and the air. Viscosity was modeled in their

Key words and phrases: Incompressible, Two Phase Flow, Level Sets, Adaptive Mesh Refinement.
1991 Mathematics Subject Classification: 65M06, 76D05, 76T05.
[1]Support for this work was provided by the Applied Mathematical Sciences Program and the HPCC Grand Challenge Program of the DOE Office of Mathematics, Information, and Computational Sciences under contract DE-AC03-76SF00098. The first author was also supported in part by contract DE-FG03-95ER25271 from the U.S. DOE Office of Energy Research Mathematical, Information and Computational Sciences (MICS) program and an Advanced Scientific Computing Initiative (ASCI) grant from the Los Alamos National Laboratory.

problems and the interface was allowed to break; both effects are difficult (and sometimes impossible) to model using the boundary integral method. In the work of Esmaeeli and Tryggvason ([15]), a combination front-tracking plus Eulerian scheme is used to compute the motion of many bubbles in a box. Like the volume-of-fluid scheme, their method [34] is capable of computing flows with viscous effects and changes in topology. In addition to applications involving bubbles and drops, there are many applications involving breaking waves. An example is the motion of a wave breaking at the bow of a moving ship ([17]). In the work of Longuet-Higgins and Cokelet [19], plunging breakers are studied using a Lagrangian scheme in which an integral equation is solved at each time step. The wave motion was created by an artificial pressure applied only at the surface. The pressure varied in space and time. In recent work by Marcus et al. [11], the incompressible level set formulation ([32]) was used to model wave growth due to wind. This formulation allowed one to have the waves "forced" by wind; the horizontal velocity of the air was faster than that of the water. Also, by using the level set formulation for computing the motion of the interface, the motion of a breaking wave can be simulated beyond the time in which the interface merges with itself.

In Figure 1, we display the adaptive computation of a 1cm water drop hitting the ground. The density ratio is 816:1 and surface tension effects are predominant in keeping the drop spherical until it hits the ground. With our method, we can compute beyond the point of break-up of the interface with no extra coding. We model the motion of the interface using the level set method. The velocity field is computed using an approximate projection formulation (see [3]) for variable density problems (see [6] and [25]). The projection formulation allows us to include effects such as wind and viscosity. The level set method enables robust interface "capturing" where interfacial effects such as surface tension, steep density gradients and the merger and break-up of fluid mass come into play (see, e.g., Sussman et al. [30] and Osher and Sethian [24]).

Other methods used to solve problems similar to our "drop" example include boundary integral methods (see e.g. [21], [20], and [9]), front tracking methods (see Unverdi and Tryggvason [34]), and volume-of-fluid methods (see Brackbill,Kothe and Zemach [10] and Puckett et al. [25]). An advantage of the boundary integral method is the fact that only grid points on the interface need to be discretized. A similar advantage exists for the front tracking method of Unverdi and Tryggvason. They discretize the interface in the Lagrangian frame of reference. Unlike the boundary integral scheme, they discretize the velocity field on an Eulerian grid. Volume-of-fluid schemes track both the velocity field and the interface on an Eulerian grid. Only those cells next to cells with a volume fraction between 0 and 1 come into play when advancing the interface. By the nature of most volume-of-fluid schemes, the volume is conserved for all time.

Here we describe the level set method as combined with an adaptive projection method. In the level set formulation, the interface is implicitly represented as the zero level set of a smooth function ϕ; thus merging and breaking up of the interface are handled naturally since the level set function does not become ill-defined during a topology change (Osher and Sethian, 1988). In our implementation of the level set method, ϕ is the signed distance from the interface for all time, thus quantities such as the gradient of ϕ and the curvature can be accurately computed using high-order schemes. Since the interface need never be explicitly reconstructed, implementations of level set schemes are relatively easy to program and easily generalizable from two to three dimensions. Problems involving gas bubbles, water drops, and wave growth due to wind have been computed using the level set scheme on a single grid (Sussman et al., [32, 31, 30]), (Marcus et al., [11]).

The level set method lends itself well to being computed on an adaptive grid. For problems with interfaces it is natural to refine around the interface; the refinement criterion can then easily be defined in terms of the level set function, which is also the distance from the interface. Since most of the vorticity in these flows is concentrated on or near the interface, refinement around the interface should yield not only finer resolution of the interface itself but also reduced numerical diffusion of the velocity field. In this work, we present results from calculations of bubbles and drops on an adaptive grid in order to demonstrate the fine-scale structure revealed by the improved resolution due to adaptivity. The adaptive algorithm uses a hierarchical structured grid approach first developed by Berger and Oliger [7] for hyperbolic partial differential equations. In particular, AMR is based on a sequence of nested grids with successively finer spacing in both time and space. Increasingly finer grids are recursively embedded in coarse grids until the solution is sufficiently resolved. An error estimation procedure based on user-specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change. Recently, Almgren et al. ([2]) developed a conservative, variable density incompressible adaptive mesh method based on the approximate projection algorithm developed in Almgren, Bell and Szymczak ([3]). In this paper we generalize the work of Almgren et al. to incompressible two-phase flows in which the jumps in density and viscosity between fluids are sharp and can be large. Also, surface tension is included in our formulation.

2 Numerical Formulation

2.1 Governing Equations

At each time step we solve the following dimensionless evolution equations for the velocity $\vec{u} = (u, v, w)$, pressure gradient ∇p and the level set function ϕ,

$$\begin{aligned} \phi_t &= -\nabla \cdot (\vec{u}\phi) && (1) \\ \vec{u}_t &= -\frac{1}{\rho}\nabla p + \vec{F}(\vec{u}, \phi) && (2) \\ \nabla \cdot \vec{u} &= 0 \end{aligned}$$

where

$$\begin{aligned} \vec{F}(\vec{u}, \phi) &= -\begin{pmatrix} \nabla \cdot (u\vec{u}) \\ \nabla \cdot (v\vec{u}) \\ \nabla \cdot (w\vec{u}) \end{pmatrix} - \frac{\hat{z}}{Fr} + \frac{\nabla \cdot (2\mu D)}{R\rho} - \frac{\kappa(\phi)\nabla H(\phi)}{W\rho} \\ \kappa(\phi) &= \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} \\ D &= 1/2((\nabla\vec{u}) + (\nabla\vec{u})^T) \\ H(\phi) &\equiv \begin{cases} 1 & \text{if } \phi > 0 \\ 0 & \text{if } \phi < 0 \\ 1/2 & \text{if } \phi = 0 \end{cases} \\ \rho(\phi) &= H(\phi) + (\rho_2/\rho_1)(1 - H(\phi)) && (3) \\ \mu(\phi) &= H(\phi) + (\mu_2/\mu_1)(1 - H(\phi)) \end{aligned}$$

The dimensionless parameters used are Reynolds number ($R = \frac{\rho_1 LU}{\mu_1}$), Froude number ($Fr = \frac{U^2}{gL}$) and Weber number ($W = \frac{\rho_1 LU^2}{\sigma}$).

In our computations, we will always assume that ϕ is less than zero in the gas. The dimensionless density and viscosity of the liquid will be identically one. The value for ρ_1 will be the actual liquid density (1.00g/cm^3 for water) and the value for ρ_2 will be the actual gas density (0.001225g/cm^3 for air). The density ratio for air/water problems is $\frac{\rho_2}{\rho_1}$ which is 1 : 816. For air/water problems the values for μ_1, μ_2 and the viscosity ratio are 0.01g/cm s, 0.00018 g/cm s and 1 : 56, respectively.

The formulation we present here is the "continuum" formulation. The free-surface boundary conditions due to viscosity and surface tension are implicitly enforced through the force $\vec{F}$. Further information on our "continuum" formulation can be found in the work of Brackbill, Zemach and Kothe [10] and in the work of Chang et al. [12].

2.2 Projection Methodology

The method used to update the velocity and pressure is a variable density extension of the approximate projection method described by Almgren et al. [3]. A projection method is a fractional step scheme in which a discretization of (2) is first used to approximate the velocity at the new time, then an elliptic equation for pressure, which results from taking the divergence of (2), is used to impose the divergence constraint on the new velocity and to update the pressure. Briefly, we may rewrite (2) as:

$$\vec{u}_t + \frac{1}{\rho}\nabla p = \vec{F}(\vec{u}, \phi).$$

The resulting equation for pressure is then

$$\nabla \cdot (\frac{1}{\rho}\nabla p) = \nabla \cdot \vec{F}$$

The divergence of the surface tension term as it appears in (2) is ill-defined; we rewrite this term as:

$$-\frac{1}{\rho}\kappa(\phi)\nabla H(\phi) \equiv -\frac{1}{\rho}\nabla(\kappa H) + \frac{1}{\rho}\nabla(\kappa)H \tag{4}$$

The first term on the right-hand-side can be incorporated in the pressure gradient term; the remaining term is now well-defined.

Note that the level set approach differs from a standard variable density projection method in that the quantity advected is ϕ, which is smooth at the air/water interface, rather than ρ, which initially is discontinuous across the interface. The density is then defined by (3).

2.3 Interface Thickness

In order to robustly compute flows with steep density gradients and surface tension, we need to give the interface a fixed thickness that is proportional to the spatial mesh size. The analytic Heaviside function $H(\phi)$ is replaced by a smoothed Heaviside function $H_\varepsilon(\phi)$:

$$H_\varepsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\varepsilon \\ \frac{1}{2}[1 + \frac{\phi}{\varepsilon} + \frac{1}{\pi}\sin(\pi\phi/\varepsilon)] & \text{if } |\phi| \le \varepsilon \\ 1 & \text{if } \phi > \varepsilon \end{cases} \tag{5}$$

It is clear from (5) that the thickness of the interface is approximately

$$\frac{2\varepsilon}{|\nabla\phi|} \tag{6}$$

In our algorithm the front will have a uniform thickness which means we require $|\nabla\phi| = 1$ when $|\phi| \leq \varepsilon$. A function that satisfies

$$|\nabla d| = 1 \quad \text{for} \quad \boldsymbol{x} \in \Omega \quad \text{with} \quad d = 0 \quad \text{for} \quad \boldsymbol{x} \in \Gamma \tag{7}$$

is called a distance function. This is because d is the signed normal distance to the interface, Γ.

If the level set function is equal to a distance function, then it follows from (5) that the thickness of the interface is 2ε. In our numerical calculation we shall take $\varepsilon = \alpha\Delta x$ where Δx is the grid size.

In the work by Sussman et al. [32], it was found necessary to maintain ϕ as a distance function in order to compute flows characterized by large density and viscosity jumps across the free-surface and also in order to compute flows in which surface tension effects are predominant. In that paper, an efficient redistance scheme was presented. In this paper, we will use the improved scheme as developed by Sussman and Fatemi [29]. In section 4.6.1, we will discuss some issues associated with including the redistance scheme as part of an adaptive mesh algorithm.

It is clear that we can choose $\phi(\boldsymbol{x}, 0)$ to be a distance function; however, under the evolution of (1) it will not necessarily remain one. This means we must be able to solve the following problem: given a level set function $\phi(\boldsymbol{x}, t)$, reinitialize it so that it is a distance function *without changing its zero level set.*

This is achieved by solving the following partial differential equation

$$\frac{\partial d}{\partial \tau} = \operatorname{sign}(\phi)(1 - |\nabla d|) \tag{8}$$

with initial conditions

$$d(\boldsymbol{x}, 0) = \phi(\boldsymbol{x}, t)$$

where

$$\operatorname{sign}(\phi) = \begin{cases} -1 & \text{if } \phi < 0 \\ 0 & \text{if } \phi = 0 \\ 1 & \text{if } \phi > 0 \end{cases} \tag{9}$$

and τ is an artificial time. The steady solutions of (8) are distance functions. Furthermore, since $\operatorname{sign}(0) = 0$, then $d(\boldsymbol{x}, \tau)$ has the same zero level set as $\phi(\boldsymbol{x}, t)$.

A nice feature of using this procedure to reinitialize is that the level set function is reinitialized near the front first. To see this we rewrite (8) as

$$d_\tau + \boldsymbol{w} \cdot \nabla d = \operatorname{sign}(\phi) \tag{10}$$

where

$$\boldsymbol{w} = \operatorname{sign}(\phi)\frac{\nabla d}{|\nabla d|}$$

It is evident that (10) is a nonlinear hyperbolic equation with the characteristic velocities pointing *outwards* from the interface in the direction of the normal. This means that d will be reinitialized to $|\nabla d| = 1$ near the interface first. Since we only need the level set function to be a distance function near the interface, it is only necessary to solve (10) for $\tau = 0$ to $\tau = \varepsilon$.

3 Single-grid Discretization

Given initial state values $\vec{S}^n = (\vec{u}^n, \phi^n)$ defined at cell centers and an initial pressure field $p^{n-\frac{1}{2}}$ defined at nodes, the following steps are taken to compute $\vec{S}^{n+1}$ and $p^{n+1/2}$:

1. Predictor step:
   - Use the source terms $\vec{F}(\vec{u}^n, \phi^n)$ and $\nabla p^{n-\frac{1}{2}}/\rho^n$ to compute face-centered quantities $u^{L,n+1/2}_{i+1/2,j,k}$, $u^{R,n+1/2}_{i+1/2,j,k}$, $v^{B,n+1/2}_{i,j+1/2,k}$, $v^{T,n+1/2}_{i,j+1/2,k}$ and $w^{D,n+1/2}_{i,j,k+1/2}$, $w^{U,n+1/2}_{i,j,k+1/2}$ by extrapolation from cell centers. Decide between the two states at each face by upwinding to define define $\tilde{u}^{n+1/2}_{i+1/2,j,k}$, $\tilde{v}^{n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{n+1/2}_{i,j,k+1/2}$.
   - Perform a MAC projection of $\tilde{\vec{u}}$ to construct time-centered face-centered discretely divergence-free advection velocities, $u^{ADV,n+1/2}_{i+1/2,j,k}$, $v^{ADV,n+1/2}_{i,j+1/2,k}$ and $w^{ADV,n+1/2}_{i,j,k+1/2}$.
   - Use the source terms $\vec{F}(\vec{u}^n, \phi^n)$, $\nabla p^{n-\frac{1}{2}}/\rho^n$ and $\nabla \cdot (\vec{u}^n \phi^n)$ to compute face-centered quantities $\vec{S}^{L,n+1/2}_{i+1/2,j,k}$, $\vec{S}^{R,n+1/2}_{i+1/2,j,k}$, $\vec{S}^{B,n+1/2}_{i,j+1/2,k}$, $\vec{S}^{T,n+1/2}_{i,j+1/2,k}$ and $\vec{S}^{D,n+1/2}_{i,j,k+1/2}$, $\vec{S}^{U,n+1/2}_{i,j,k+1/2}$ by extrapolation from cell centers. Decide between the two states at each face by upwinding (using $\vec{u}^{ADV}$) to define the time-centered face-centered state values $\vec{S}^{n+1/2}_{i+1/2,j,k}$, $\vec{S}^{n+1/2}_{i,j+1/2,k}$, $\vec{S}^{n+1/2}_{i,j,k+1/2}$.
   - Construct $\nabla \cdot (\vec{u}\phi)^{n+1/2}$ and $\nabla \cdot (\vec{u}\vec{u})^{n+1/2}$ using the MAC-projected advection velocity and the time-centered face-centered values of $\vec{u}$ and ϕ

2. Level set update:

$$\phi^{n+1} = \phi^n - \Delta t \nabla \cdot (\vec{u}\phi)^{n+1/2}$$

3. Semi-implicit solve for velocity incorporating $(\nabla \cdot (2\mu D))^{n+1/2}$.

4. Projection: decompose $\vec{F}^{n+1/2}$ into two components, $\vec{u}_t^{n+1/2}$ and $\nabla p^{n+1/2}/\rho^{n+1/2}$.

5. Velocity update:

$$\vec{u}^{n+1} = \vec{u}^n + \Delta t \vec{u}_t^{n+1/2}$$

6. Redistance operation: drive $|\nabla \phi^{n+1}|$ to one.

The steps described here are almost identical to the single-grid method found in Almgren et al. [2]. In our formulation, we include changes specific to the two-phase problems we wish to solve:

- Instead of advecting density ρ, we advect the smooth level set function ϕ.
- We assume the viscous coefficient μ is non-constant (1:63 jump for air/water). This entails the use of a tensor representation for the viscous force term $\vec{F}_{visc}(\vec{u})$.

- We include effects of surface tension. We denote the surface tension forcing term as $\vec{F}_{surf}(\phi)$.
- Our only body force is due to gravity: $\vec{F}_{grav}(\rho) = (0, 0, \rho/(Fr))$.
- We perform a redistance step at the end of every timestep.

3.1 Predictor Step

Our goal in this step is to construct the advective update terms, $\nabla\cdot(\vec{u}\phi)^{n+1/2}$ and $\nabla\cdot(\vec{u}\vec{u})^{n+1/2}$. In the predictor we first extrapolate the normal velocities to cell faces at $t^{n+1/2}$ using a second-order Taylor series expansion in space and time. The time derivative is replaced using Eq. (2). For face $(i+\frac{1}{2}, j, k)$ this gives

$$\tilde{u}^{L,n+1/2}_{i+1/2,j,k} \approx u^n_{i,j,k} + \frac{\Delta x}{2}u_x + \frac{\Delta t}{2}u_t$$

$$= u^n_{i,j,k} + \left(\frac{\Delta x}{2} - u^n_{i,j,k}\frac{\Delta t}{2}\right)(u_x^{n,lim})_{i,j,k} + \frac{\Delta t}{2}\left(-(\widehat{vu_y})_{i,j,k} - (\widehat{wu_z})_{i,j,k} + \right. \tag{11}$$

$$\left.\frac{1}{\rho^n_{i,j,k}}(-(Gp)^{n-\frac{1}{2}}_{i,j,k} + (\vec{F}_{visc}(\vec{u}^n))_{i,j,k} + (\vec{F}_{surf}(\phi^n))_{i,j,k} + (\vec{F}_{grav}(\rho(\phi^n)))_{i,j,k}\right).$$

extrapolated from (i, j, k), and

$$\tilde{u}^{R,n+1/2}_{i+1/2,j,k} \approx u^n_{i+1,j,k} - \frac{\Delta x}{2}u_x + \frac{\Delta t}{2}u_t$$

$$= u^n_{i+1,j,k} - \left(\frac{\Delta x}{2} + u^n_{i+1,j,k}\frac{\Delta t}{2}\right)(u_x^{n,lim})_{i+1,j,k} + \frac{\Delta t}{2}\left(-(\widehat{vu_y})_{i+1,j,k} - (\widehat{wu_z})_{i+1,j,k} + \right. \tag{12}$$

$$\left.\frac{1}{\rho^n_{i+1,j,k}}(-(Gp)^{n-\frac{1}{2}}_{i+1,j,k} + (\vec{F}_{visc}(\vec{u}^n))_{i+1,j,k} + (\vec{F}_{surf}(\phi^n))_{i+1,j,k} + (\vec{F}_{grav}(\rho(\phi^n)))_{i+1,j,k}\right)$$

extrapolated from $(i+1, j, k)$.

Here, G is a discretization of the gradient operator. $F_{visc}(\vec{u})_{i,j}$ for the two-dimensional case is discretized as:

$$\begin{aligned}
&\frac{2}{R\Delta x^2}(\mu_{i+1/2,j}(u_{i+1,j} - u_{i,j}) - \mu_{i-1/2,j}(u_{i,j} - u_{i-1,j})) + \\
&\quad \frac{1}{R\Delta y^2}(\mu_{i,j+1/2}(u_{i,j+1} - u_{i,j}) - \mu_{i,j-1/2}(u_{i,j} - u_{i,j-1})) + \\
&\quad \frac{1}{R\Delta x\Delta y}(\mu_{i,j+1/2}(v_{i+1,j+1} - v_{i-1,j+1} + v_{i+1,j} - v_{i-1,j}) - \\
&\quad \mu_{i,j-1/2}(v_{i+1,j} - v_{i-1,j} + v_{i+1,j-1} - v_{i-1,j-1})),
\end{aligned}$$

where

$$\mu_{i+1/2,j} = \frac{1}{2}(\mu(\phi_{i,j}) + \mu(\phi_{i+1,j})).$$

(We have not yet implemented the three-dimensional case with variable viscosity.) The surface tension term is given by the second term on the right-hand-side of (4):

$$\vec{F}_{surf}(\phi) = \nabla\kappa(\phi)H(\phi)/W \tag{13}$$

The curvature $\kappa(\phi) = \nabla \cdot (\frac{\nabla\phi}{|\nabla\phi|})$ is discretized using central differencing as described in [30]. The gradient of the curvature is also computed with centered differences. Our interface has a thickness ε thus we discretize $H(\phi)$ as $H_\varepsilon(\phi)$. In all of our computations, ϕ is negative in the gas making $H(\phi)/\rho(\phi)$ less than one. If ϕ were positive in the gas, then $H(\phi)/\rho(\phi)$ would have a large jump at the interface. In preliminary tests, this caused the algorithm to fail. Another solution to the latter case is to replace H by $(H-1)$ in (13). This is allowed since the term $-\nabla\kappa(\phi)(-1)$ can also be incorporated into the pressure term.

Analogous formulae are used to predict values for $\tilde{v}^{F/B,n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{U/D,n+1/2}_{i,j,k+1/2}$ at the other faces of the cell. In evaluating these terms the first derivatives normal to the face (in this case $u^{n,lim}_x$) are evaluated using a monotonicity-limited fourth-order slope approximation [13]. The limiting is done on each component of the velocity at time n individually.

The transverse derivative terms ($\widehat{vu_y}$ and $\widehat{wu_z}$ in this case) are evaluated by first extrapolating all velocity components to the transverse faces from the cell centers on either side, then choosing between these states using the upwinding procedure as described in detail by Almgren et al. ([2]).

The normal velocity at each face (preliminary face-centered advection velocities) is then determined by an upwinding procedure based on the states predicted from the cell centers on either side. The procedure is similar to that described in [2], i.e. (suppressing the $(i+1/2, j, k)$ indices)

$$\tilde{u}_{i+\frac{1}{2},j,k} = \begin{cases} \tilde{u}^{L,n+1/2} & \text{if } \tilde{u}^{L,n+1/2} > 0 \text{ and } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} > 0 \\ 0 & \text{if } \tilde{u}^{L,n+1/2} \leq 0, \tilde{u}^{R,n+1/2} \geq 0 \text{ or } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} = 0 \\ \tilde{u}^{R,n+1/2} & \text{if } \tilde{u}^{R,n+1/2} < 0 \text{ and } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} < 0 \end{cases}$$

We follow a similar procedure to construct $\tilde{v}^{n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{n+1/2}_{i,j,k+1/2}$.

The normal velocities on cell faces are now centered in time and second-order accurate, but do not, in general, satisfy the divergence constraint. In order to enforce the constraint at this intermediate time, we apply the MAC projection (see [5]) to the face-based velocity field before construction of the conservative updates. The equation

$$D^{MAC}(\frac{1}{\rho^n} G^{MAC} p^{MAC}) = D^{MAC}(\tilde{\vec{u}}^{n+1/2}) \tag{14}$$

is solved for p^{MAC}, where

$$D^{MAC}(\tilde{\vec{u}}^{n+1/2}) \equiv \frac{\tilde{u}^{n+1/2}_{i+\frac{1}{2},j,k} - \tilde{u}^{n+1/2}_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{\tilde{v}^{n+1/2}_{i,j+\frac{1}{2},k} - \tilde{v}^{n+1/2}_{i,j-\frac{1}{2},k}}{\Delta y} + \frac{\tilde{w}^{n+1/2}_{i,j,k+\frac{1}{2}} - \tilde{w}^{n+1/2}_{i,j,k-\frac{1}{2}}}{\Delta z}$$

and $G^{MAC} = -(D^{MAC})^T$ so that

$$(G^{MAC}_x p^{MAC})_{i+1/2,j,k} = \frac{(p^{MAC}_{i+1,j,k} - p^{MAC}_{i,j,k})}{\Delta x}$$

with G^{MAC}_y and G^{MAC}_z defined analogously.

Equation (14) represents a matrix system which is solved using the multigrid-preconditioned conjugate gradient method [33]. This elliptic solve is described in more detail in section 4.4.

The face-based advection velocities are then defined by

$$u^{ADV}_{i+\frac{1}{2},j,k} = \tilde{u}^{n+1/2}_{i+\frac{1}{2},j,k} - \frac{1}{\rho^n_{i+\frac{1}{2},j,k}}(G^{MAC}_x \phi^{MAC})_{i+1/2,j,k}$$

with $v^{ADV}_{i,j+1/2,k}$ and $w^{ADV}_{i,j,k+1/2}$ defined analogously. Here and in (14), ρ on the faces is averaged geometrically from the cell centers at time n.

At this point the tracing is performed for the tangential velocity components and level set function from cell centers to all cell faces. Let $\vec{S} = (\vec{u}, \phi)$. The extrapolation of the normal velocity components has been described above; the tracing of the level set function and tangential velocity components is analogous, with the time derivatives replaced using (1) and (2).

Time-centered values $\tilde{S}^{n+1/2}$ at each face (i.e. $\tilde{\phi}^{n+1/2}$ and $\tilde{\vec{u}}^{n+1/2}$ including the normal velocity component) are now determined by upwinding, as follows:

$$\tilde{S}_{i+\frac{1}{2},j,k} = \begin{cases} \tilde{S}^L & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} > 0 \\ 1/2(\tilde{S}^L + \tilde{S}^R) & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} = 0 \\ \tilde{S}^R & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} < 0 \end{cases}$$

The conservative update terms can now be defined by

$$[\nabla \cdot (S\vec{u})]^{n+1/2}_{i,j,k} = [\nabla \cdot (\vec{u}^{ADV}\tilde{S}^{n+1/2})]_{i,j,k} = \frac{1}{\Delta x}(\tilde{S}_{i+\frac{1}{2},j,k}u^{ADV}_{i+\frac{1}{2},j,k} - \tilde{S}_{i-\frac{1}{2},j,k}u^{ADV}_{i-\frac{1}{2},j,k}) +$$

$$\frac{1}{\Delta y}(\tilde{S}_{i,j+\frac{1}{2},k}v^{ADV}_{i,j+\frac{1}{2},k} - \tilde{S}_{i,j-\frac{1}{2},k}v^{ADV}_{i,j-\frac{1}{2},k}) + \frac{1}{\Delta z}(\tilde{S}_{i,j,k+\frac{1}{2}}w^{ADV}_{i,j,k+\frac{1}{2}} - \tilde{S}_{i,j,k-\frac{1}{2}}w^{ADV}_{i,j,k-\frac{1}{2}})$$

3.2 Level set update

We now update the level set ϕ using the advective term constructed above:

$$\phi^{n+1} = \phi^n - \Delta t[\nabla \cdot (\phi\vec{u})]^{n+1/2}$$

We define $\rho^{n+1/2}$ and $\phi^{n+1/2}$ to be:

$$\begin{aligned} \rho^{n+1/2} &= 1/2(\rho(\phi^{n+1}) + \rho(\phi^n)) \\ \phi^{n+1/2} &= 1/2(\phi^{n+1} + \phi^n) \end{aligned}$$

3.3 Viscous Solve

We solve the following equation for $\vec{u}^*$:

$$\frac{\vec{u}^* - \vec{u}^n}{\Delta t} = -[\nabla \cdot (\vec{u}\vec{u})]^{n+1/2} + \frac{1}{\rho^{n+1/2}}(-\nabla p^{n-1/2} + \vec{F}_{visc}(\frac{(\vec{u}^n + \vec{u}^*)}{2}) + \vec{F}_{surf}(\phi^{n+1/2}) + \vec{F}_{grav}(\rho^{n+1/2})) \quad (15)$$

Equation (15) in n dimensions is a coupled parabolic system of n equations and n unknowns. The forcing terms and advective update terms are treated as explicit source terms (computed in previous steps). These coupled linear equations are solved using multigrid on all components simultaneously; this solve is described in more detail in Section 4.5.

3.4 Discretization of the Projection

In the projection step, a vector field decomposition is applied to $(\vec{u}^* - \vec{u}^n)/\Delta t$ to obtain an update to the velocity field, $(\vec{u}^{n+1} - \vec{u}^n)/\Delta t$, and an update for the pressure. In particular, if $\mathbf{P}$ represents the projection then

$$\frac{\vec{u}^{n+1} - \vec{u}^n}{\Delta t} = \mathbf{P}\left(\frac{\vec{u}^* - \vec{u}^n}{\Delta t}\right) \tag{16}$$

$$\frac{1}{\rho^{n+1/2}}\nabla p^{n+1/2} = \frac{1}{\rho^{n+1/2}}\nabla p^{n-\frac{1}{2}} + (\mathbf{I} - \mathbf{P})\left(\frac{\vec{u}^* - \vec{u}^n}{\Delta t}\right).$$

Note that the vector field we project is not $\vec{u}^*$, it is an approximation to $\vec{u}_t$. This distinction is significant when the projection is not exact. Discretely, the projection is computed by solving for the appropriately weighted gradient component of $(\vec{u}^* - \vec{u}^n)/\Delta t$ which we denote by $(1/\rho)Gp$. We determine p by solving

$$L_\rho^{n+1/2} p = D\left(\frac{\vec{u}^* - \vec{u}^n}{\Delta t}\right)$$

where D is a discrete nodal approximation to the divergence operator and $L_\rho^{n+1/2} p$ is a second-order accurate nodal approximation to $\nabla \cdot \left(\frac{1}{\rho^{n+1/2}}\nabla p\right)$.

In two dimensions the projection discretization can be derived directly from the variational form

$$\int \frac{1}{\rho}\nabla p(\mathbf{x}) \cdot \nabla\psi(\mathbf{x})\ d\mathbf{x} = \int \frac{\vec{u}^* - \vec{u}^n}{\Delta t} \cdot \nabla\psi(\mathbf{x})\ d\mathbf{x}, \qquad \forall\psi(\mathbf{x}) \tag{17}$$

where $d\mathbf{x}$ is the volume element $dx\ dy$, $r\ dr\ d\theta$ or $dx\ dy\ dz$ as appropriate. If this variational form is used in conjunction with standard piecewise bilinear or piecewise linear (on a standard triangulation of a mesh) finite element basis functions, the resulting discrete problem corresponds to standard nine-point and five-point discretizations of $L_\rho^{n+1/2}$, respectively. (In this paper we use the nine-point discretization for all two-dimensional problems.) We then define

$$\frac{\vec{u}^{n+1} - \vec{u}^n}{\Delta t} = \frac{\vec{u}^* - \vec{u}^n}{\Delta t} - \frac{1}{\rho^{n+1/2}}\overline{Gp}, \tag{18}$$

where $\overline{Gp}$ is the cell average of ∇p, and define

$$p^{n+1/2} = p^{n-1/2} + p.$$

In two dimensions, the discrete representations for $D\vec{u}$, $\overline{Gp}$ and $L_\rho p$ are:

$$\begin{aligned}
(D\vec{u})_{i+1/2,j+1/2} &= \frac{u_{i+1,j} + u_{i+1,j+1} - u_{i,j} - u_{i,j+1}}{\Delta x} + \\
&\quad \frac{v_{i+1,j+1} - v_{i+1,j} - v_{i,j} + v_{i,j+1}}{\Delta y} \\
\overline{(Gp)}_{ij} &\equiv \begin{pmatrix} \frac{p_{i+1/2,j+1/2} + p_{i+1/2,j-1/2} - p_{i-1/2,j+1/2} - p_{i-1/2,j-1/2}}{2\Delta x} \\ \frac{p_{i+1/2,j+1/2} - p_{i+1/2,j-1/2} + p_{i-1/2,j+1/2} - p_{i-1/2,j-1/2}}{2\Delta y} \end{pmatrix}
\end{aligned}$$

$$
\begin{aligned}
(L_\rho p)_{i+1/2,j+1/2} = \frac{1}{6h^2}(\quad & \tfrac{1}{\rho_{i,j}}(2p_{i-1/2,j-1/2} + p_{i+1/2,j-1/2} + p_{i-1/2,j+1/2} - 4p_{i+1/2,j+1/2}) + \\
& \tfrac{1}{\rho_{i,j+1}}(2p_{i-1/2,j+3/2} + p_{i+1/2,j+3/2} + p_{i-1/2,j+1/2} - 4p_{i+1/2,j+1/2}) + \\
& \tfrac{1}{\rho_{i+1,j}}(2p_{i+3/2,j-1/2} + p_{i+1/2,j-1/2} + p_{i+3/2,j+1/2} - 4p_{i+1/2,j+1/2}) + \\
& \tfrac{1}{\rho_{i+1,j+1}}(2p_{i+3/2,j+3/2} + p_{i+1/2,j+3/2} + p_{i+3/2,j+1/2} - 4p_{i+1/2,j+1/2}))
\end{aligned}
$$

We note that this is not a discrete orthogonal projection; in fact, $D\vec{u}^{n+1} \neq 0$. However, the projection as defined by Eqs. (16) and (17) is a discrete orthogonal projection onto a larger velocity space (in the finite element sense) which is then averaged onto the grid. The resulting approximate projection satisfies the divergence constraint to second-order accuracy and the overall algorithm is stable. The reader is referred to Almgren et al. [3] for a detailed discussion of this approximate projection.

In three dimensions, instead of using the 27-point analog to the nine-point stencil presented here, we use a seven-point stencil as discussed in [2].

3.5 Redistance Operation

In section 2.3, we stated the importance of maintaining our level set function as the signed distance from the interface. We describe in this section the two-dimensional details for discretizing (8). The three-dimensional analog case is a natural extension.

Analytically, equation (8) does not change the position of the zero level set of ϕ. Unfortunately in numerical computation this may not be true. In recent work [29], a "constraint" was developed that significantly improves the accuracy of solving (8). We use the fact that

$$\partial_\tau \int_{\Omega_{ij}} H(d) = 0 \tag{19}$$

in every grid cell $\Omega_{ij} = ((x,y)|x_{i-1/2} < x < x_{i+1/2} \text{ and } y_{j-1/2} < y < y_{j+1/2})$. That is, since the interface should not move, the volume should not change either. We modify (8):

$$
\begin{aligned}
\frac{\partial d}{\partial \tau} &= \operatorname{sign}(\phi)(1 - |\nabla d|) + \lambda_{ij} f(\phi) \equiv L(\phi, d) + \lambda_{ij} f(\phi) && (20) \\
d(\boldsymbol{x}, 0) &= \phi(\boldsymbol{x}, t) && (21)
\end{aligned}
$$

λ_{ij} is *constant* in each cell Ω_{ij} and is determined using,

$$\partial_\tau \int_{\Omega_{ij}} H(d) = \int_{\Omega_{ij}} H'(d) d_\tau \approx \int_{\Omega_{ij}} H'(\phi) d_\tau = \int_{\Omega_{ij}} H'(\phi)(L(\phi, d) + \lambda_{ij} f(\phi)) = 0.$$

λ_{ij} is calculated to be

$$\lambda_{ij} = \frac{-\int_{\Omega_{ij}} H'(\phi) L(\phi, d)}{\int_{\Omega_{ij}} H'(\phi) f(\phi)} \tag{22}$$

In our calculations we choose

$$f(\phi) \equiv H'(\phi)|\nabla\phi|.$$

This ensures that we only correct at the interface without disturbing the distance function property away from the interface.

For ease of notation, we will denote $\phi^{n+1,(0)}$ as ϕ, $\phi^{n+1,(k)}$ as d_k and Δx as h. If we want to recover the distance function at a distance αh from the zero level set of ϕ, we need to solve (8) for $\tau = 0$ to $\tau = \alpha h$. As noted in section 2.3, we can rewrite (8) in the form of an advection equation with velocity $\boldsymbol{w}$ (see (10)). In light of this, we shall use a second-order ENO upwind scheme (see [28, 16]) for solving (10). The discretization presented here is a second-order generalization of a first-order scheme presented in [26].

Given d_k we solve for $\tilde{d}_{k+1}$ as follows:

1. Compute an approximation to $\mid \nabla d_k \mid$ using second-order ENO as described below. Let $L(d_k) = \mathrm{sign}_h(\phi)(1- \mid \nabla d_k \mid)$. We use a sign function with thickness h:

$$\mathrm{sign}_h(\phi) \quad \equiv \quad 2(H_h(\phi) - 1/2)$$

2. Let $d_{k+1/2} = d_k + \Delta\tau(L(d_k))$.

3. Compute $\mid \nabla d_{k+1/2} \mid$.

4. Let $\tilde{d}_{k+1} = d_k + (\Delta\tau/2)(L(d_k) + L(d_{k+1/2}))$

In order to compute $\mid \nabla d_k \mid = \sqrt{d_x^2 + d_y^2 + d_z^2}$ (or $\sqrt{d_x^2 + d_y^2}$ in two dimensions) we use the following process:
For $(d_x)_{ij}$ we have:

1. Approximate d_x^-

$$\begin{aligned} a &= \frac{d_{i+1,j} - 2d_{i,j} + d_{i-1,j}}{h^2} \\ b &= \frac{d_{i,j} - 2d_{i-1,j} + d_{i-2,j}}{h^2} \\ c &= \begin{cases} a & \text{if } |a| \le |b| \\ b & \text{otherwise} \end{cases} \\ d_x^- &= \frac{d_{i,j} - d_{i-1,j}}{h} + ch/2 \end{aligned}$$

2. Approximate d_x^+

$$\begin{aligned} a &= \frac{d_{i+1,j} - 2d_{i,j} + d_{i-1,j}}{h^2} \\ b &= \frac{d_{i+2,j} - 2d_{i+1,j} + d_{i,j}}{h^2} \\ c &= \begin{cases} a & \text{if } |a| \le |b| \\ b & \text{otherwise} \end{cases} \\ d_x^+ &= \frac{d_{i+1,j} - d_{i,j}}{h} - ch/2 \end{aligned}$$

3. Upwind

$$d_x \approx \begin{cases} d_x^+ & \text{if } d_x^+ \mathrm{sign}(\phi) < 0 \text{ and } d_x^- \mathrm{sign}(\phi) < -d_x^+ \mathrm{sign}(\phi) \\ d_x^- & \text{if } d_x^- \mathrm{sign}(\phi) > 0 \text{ and } d_x^+ \mathrm{sign}(\phi) > -d_x^- \mathrm{sign}(\phi) \\ 0 & \text{if } d_x^- \mathrm{sign}(\phi) < 0 \text{ and } d_x^+ \mathrm{sign}(\phi) > 0 \end{cases}$$

In two or three dimensions, one computes d_y and d_z in a similar manner.

We modify the above discretization in order to include the constraint as described in equations (20) through (22). Given d_k, we compute $\tilde{d}_{k+1}$ as described above. Then we compute λ_{ij} (see (22)) where:

$$\begin{aligned} L(\tilde{d}_{k+1}) &\approx (\tilde{d}_{k+1} - \phi)/\tau_{k+1} \\ H'(\phi) &\approx \partial H_\epsilon(\phi)/\partial\phi \\ f(\phi) &\approx H'(\phi) \mid \nabla\phi \mid \\ \lambda(\tilde{d}_{k+1}) &\approx \frac{-\int_{\Omega_{ij}} H'(\phi) L(\tilde{d}_{k+1})}{\int_{\Omega_{ij}} H'(\phi) f(\phi)} \end{aligned}$$

The numerical integration over the domain

$$\Omega_{ij} = ((x,y)|x_{i-1/2} < x < x_{i+1/2} \text{ and } y_{j-1/2} < y < y_{j+1/2})$$

is computed using a nine-point stencil:

$$\int_{\Omega_{ij}} f \approx ((\sum_{m=-1}^{1} \sum_{n=-1}^{1} f_{i+m,j+n}) + 15 f_{ij}) h^2/24.$$

Our new updated d_{k+1} is

$$d_{k+1} = \tilde{d}_{k+1} + \tau_{k+1}\lambda_{ij} f(\phi).$$

In [29] we showed that the constraint eliminated the lower order term in the mass error:

$$\int_{\Omega_{ij}} H_\epsilon(d_{k+1}) - H_\epsilon(\phi).$$

As stated before, we only solve up to $\tau = \alpha h$ where α is the number of grid points away from the interface at which we wish d to represent a signed distance from the interface. For an interfacial thickness of four points, we would require only four iterations, each with a time step of $h/2$.

3.6 Initialization of the Data

Specification of the problem must include values for U and ϕ at time $t^0 = 0$ and values for the initial pressure p at time $t^{1/2}$. The pressure is not initially prescribed, and must be calculated in an initial iterative step.

To begin the calculation, the initial velocity field is first projected to ensure that it satisfies the divergence constraint at t=0. Then an initial iteration is performed to calculate an approximation to the pressure at $t = \frac{\Delta t}{2}$. If this process were iterated to convergence and the projection were exact, then $U^1 \equiv U^*$ in the first step, because the pressure used in Eq. (15) would in fact be $p^{1/2}$, not $p^{-1/2}$. However, in practice we typically perform only a few iterations, since what is needed for second-order accuracy in Eq. (15) is only a first-order accurate approximation to $p^{n+1/2}$, which in a standard time step is approximated by $p^{n-\frac{1}{2}}$.

In each step of the iteration we follow the procedure described in the above subsections. In the first iteration we use $p^{-1/2} = 0$. At the end of each iteration we have calculated a value of U^1 and a pressure $p^{1/2}$. During the iteration procedure, we discard the value of U^1, but define $p^{-1/2} = p^{1/2}$. Once the iteration is completed, we use the value of $p^{-1/2}$ in Eq. (15) along with the values of U^0 and ϕ^0.

3.7 Timestep

The timestep Δt is determined by restrictions due to CFL condition, gravity, viscosity and surface tension ([32, 10]):

$$
\begin{aligned}
\Delta t_s &\equiv \sqrt{\frac{(\rho_1 + \rho_2)W}{8\pi}} \Delta x^{3/2} \\
\Delta t_v &\equiv \min_{\Omega}((\frac{3}{14})\frac{\rho R \Delta x^2}{\mu}) \\
\Delta t_c &\equiv \min_{\Omega}(\frac{\Delta x}{\mid \boldsymbol{u} \mid}, \Delta x Fr) \\
\Delta t^{n+1} &= 1/2 \min(\Delta t_v, \Delta t_s, \Delta t_c)
\end{aligned}
$$

We note here that even though we handle the viscous terms semi-implicitly, we have still found a need for the stringent timestep constraint when it comes to problems with large density ratios. One reason for this, as pointed out in [2], is the fact that viscous terms are not included in defining the states used in the transverse derivatives. In a paper by Minion ([22]), the issues dealing with stability in the presence of stiff viscous effects are covered.

4 Adaptive Mesh Refinement

In this section we present the extension of the single-grid algorithm described above to an adaptive hierarchy of nested rectangular grids. The basic adaptive framework is as described by Almgren et al. [2]; here we focus on the changes necessary for the level set formulation. These changes include:

- The curvature of the interface along with vorticity are used as criteria for determining grid placement at the finest level. The location of the interface determines grid locations at coarser levels.
- A redistance synchronization step is necessary to account for errors resulting from doing the redistance operation separately on coarse and fine levels. This allows for robust computation even if part of the interface crosses a coarse/fine grid boundary. Other synchronization steps related to the level set function are also necessary for maintaining the regularity of not only the level set function ϕ across a coarse/fine grid boundary, but also the regularity of the curvature $\kappa(\phi)$.
- Because of the large density ratios, the multigrid solver is replaced by a multigrid-preconditioned conjugate gradient solver [33] for all of the "level" and "sync" projections. (The viscous solve is still done with multigrid.)

In the first subsection we describe the creation of the grid hierarchy and the regridding procedure used to adjust the hierarchy during the computation. Then we present a brief description of the time step algorithm for the grid system that subcycles in time, and describe the initialization procedure used to begin a multilevel calculation. Next we describe the multigrid-preconditioned conjugate gradient algorithm, both for the MAC projections and the nodal projections, giving the details necessary for efficient convergence for large density ratios. Finally, we discuss the details of the adaptive time step procedure, focusing on the synchronization between different levels of refinement.

4.1 Creating and Managing the Grid Hierarchy

The grid hierarchy is composed of different levels of refinement ranging from coarsest ($\ell = 0$) to finest ($\ell = \ell_{max}$). The coarsest level ($\ell = 0$) covers the whole computational domain while successively higher levels ($\ell + 1$) lie on top of the level underneath them (level ℓ). Each level is represented as the union of rectangular grid patches of a given resolution. In our computations the refinement ratio r between levels can be either 2 or 4. Thus we have $\Delta x^{\ell+1} = \Delta y^{\ell+1} = \Delta z^{\ell+1} = \frac{1}{r}\Delta x^{\ell}$. The grids are properly nested, in the sense that the union of grids at level $\ell + 1$ is contained in the union of grids at level ℓ for $0 \le \ell < \ell_{max}$. Furthermore, the containment is strict in the sense that, except at physical boundaries, the level ℓ grids are large enough to guarantee that there is a border at least one level ℓ cell wide surrounding each level $\ell + 1$ grid. (Grids at all levels are allowed to extend to the physical boundaries so the proper nesting is not strict there.)

The initial creation of the grid hierarchy and the subsequent regridding operations in which the grids are dynamically changed to reflect changing flow conditions use the same procedures as were used by Bell et al. [4] for hyperbolic conservation laws. In the problems we compute here, we shall "tag" cells which contain part of the air/water interface, i.e. those in which the level set function changes sign. For some problems, in order to generate the finest level of refinement, we require not only that that cells contain the interface, but also that the curvature or vorticity in those cells exceed a preset threshold in order to be "tagged." Once cells on a specified level are "tagged" for refinement, the grids at the next higher level can be constructed. The tagged cells are grouped into rectangular patches using the clustering algorithm given in Berger and Rigoustsos [8]. These rectangular patches are refined to form the grids at the next level. The process is repeated until either the error tolerance criteria are satisfied or a specified maximum level is reached.

At $t = 0$ the initial data is used to create grids at level 0 through ℓ_{max}. (Grids have a user-specified maximum size, therefore more than one grid may be needed to cover the physical domain.) As the solution advances in time, the regridding algorithm is called every k_ℓ (k_ℓ is also user-specified) level ℓ steps to redefine grids at levels $\ell + 1$ to ℓ_{max}. Level 0 grids remain unchanged throughout the calculation. Grids at level $\ell + 1$ are only modified at the end of level ℓ time steps, but because we subcycle in time, i.e. $\Delta t_{\ell+1} = \frac{1}{r}\Delta t_\ell$, level $\ell + 2$ grids can be created and/or modified in the middle of a level ℓ time step if $k_{\ell+1} < r$.

When new grids are created at level $\ell+1$, the data on these new grids are copied from the previous grids at level $\ell+1$ if possible, otherwise interpolated in space from the underlying level ℓ grids.

We note here that while there is a user-specified limit to the number of levels allowed, at any given time in the calculation there may not be that many levels in the hierarchy, i.e. ℓ_{max} can change dynamically as the calculation proceeds, as long as it does not exceed the user-specified limit.

4.2 Overview of Time-Stepping Procedure

The adaptive projection algorithm can most easily be thought of as a recursive procedure, in which to advance level ℓ, $0 \le \ell \le \ell_{max}$ the following steps are taken:

- Advance level ℓ in time as if it is the only level. Supply boundary conditions for the velocity, level set function and pressure from level $\ell - 1$ if level $\ell > 0$, and from the physical domain boundaries.

- If $\ell < \ell_{max}$
  - Advance level $(\ell + 1)$ r times with time step $\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$. Use boundary conditions for the velocity, level set function and pressure from level ℓ, and from the physical domain boundaries.
  - Synchronize the data between levels ℓ and $\ell + 1$, and interpolate corrections to higher levels if $\ell + 1 < \ell_{max}$.

As in the adaptive mesh technique for hyperbolic systems, the evaluation of the conservative derivatives in (1) and (2) can be performed one grid at a time, with boundary data copied from other fine grids, interpolated from underlying coarse grids, or supplied from physical boundary conditions. These correspond to steps found in section 3 for the single grid scheme. Since the redistance operation is discretized as an advection equation, the same procedure as for handling the conservative derivatives may be applied to handling these equations too. The parabolic and single-level elliptic solves require that the solution be computed on all grids at a level at one time, since these are no longer explicit operations. Boundary data for these solves are interpolated from underlying coarse grids or supplied from physical boundary conditions. The interpolation and solution procedure for these equations are discussed in Sections 4.4 and 4.5.

Once the level $\ell + 1$ data have been advanced to the same point in time as the level ℓ data using grid-by-grid or level operations, synchronization of the data between levels is required. This synchronization attempts to correct for the difference in the solution generated by performing the update operations sequentially on coarse then fine grids, rather than simultaneously on a composite hierarchy.

The synchronization has several components. In order to correct for the flux mismatches resulting from the hyperbolic and parabolic components of the updates, the data on the fine grids is averaged onto the coarse grids beneath them, and the data on the coarse grids immediately outside the fine grids is updated using corrected advective and viscous fluxes at the coarse/fine interface (i.e., "refluxing"). (In the case of variable viscosity, the viscous solve is a tensor solve rather than decoupled equations for each component of velocity, but this can still be written in flux form.)

Because of the incompressibility constraint, steps must also be taken to account for the mismatch in the MAC-projected advection velocity ($\vec{u}^{ADV}$) and the mismatch in the projected new-time velocity ($\vec{u}^{n+1}$) which result from not having satisfied the full elliptic matching conditions at the coarse/fine interface. When performing the MAC and nodal projections we impose Dirichlet boundary conditions from level ℓ for the level $\ell + 1$ grids. Consequently, the fields computed in the elliptic solves match in value at coarse/fine interfaces but do not, in general, match normal derivatives. It is this mismatch in normal derivatives which defines the source for the correction solves ("sync projections"). One "sync projection" is done for each type of "level projection" (i.e. MAC and nodal).

Finally, when using our level set formulation, we must also include synchronization steps in order to maintain the regularity of the level set function ϕ and curvature $\kappa(\phi)$ across a coarse-fine grid interface. Typically, the position of the zero level set on the fine level will be different from that on the coarse level. Thus, after an averaging down of the fine data onto the coarse level, the distance function ϕ on the coarse level *outside* of the fine level will no longer represent the distance from the zero level set. We thus include a sync-redistance step in order to correct the level set function on the coarse level. We have found that the sync-redistance procedure is not enough to ensure smoothness of curvature $\kappa(\phi)$ across a coarse-fine interface. In order to avoid computing the curvature across a coarse-fine interface, we only compute the curvature at points where $|\phi| < \alpha\Delta x$ (i.e. only compute curvature when within the thickness of the interface). The curvature is then extended in an iterative fashion from the interface. The iteration procedure, like the redistance procedure, is only α steps long.

4.3 Initialization of the Multilevel Data

As in the single grid projection method, we must first project the given velocity field to enforce the divergence constraint, and iterate with the initial data in order to define an initial pressure field. For accuracy, the initial projection is done as a full multilevel composite solve over all levels, using the stencils given in Section 3.4, as well as coarse-fine interface stencils where appropriate. As a result, the velocity resulting from this projection satisfies the divergence constraint not only at each level but also at all the coarse/fine interfaces. After the projection all quantities other than pressure are averaged down from fine grids onto the coarser cells underlying them, to ensure that any level ℓ data, $0 \leq \ell < \ell_{max}$, is the average of the finer values overlying it.

For the iteration used to define the initial pressure, we compute the time step on the finest level currently defined, and iterate all levels with that time step ($\Delta t^{\ell_{max}}$), i.e. without sub-cycling. Here, however, the velocity is advanced on each level without being projected at that level, i.e. $\vec{u}^{*,\ell}$ but not $\vec{u}^{1,\ell}$ is defined for $0 \leq \ell \leq \ell_{max}$. One multilevel composite solve is then done on the field $(\vec{u}^{*} - \vec{u}^{0})/\Delta t^{\ell_{max}}$ to compute the pressure update on all levels simultaneously. Here again the constraint is satisfied not only on each level but also at all the coarse/fine interfaces. As in the single grid case, during the iteration procedure the values of $\vec{u}^{1}$ computed by the projection are discarded, and the new value of pressure is used for the next iteration. When the iteration is complete, the regular time-stepping procedure (i.e. with sub-cycling) is begun.

4.4 The Multigrid-Preconditioned Conjugate Gradient Solvers

As described above, to advance a single level requires both a MAC level projection (to define divergence-free intermediate advection velocities) and a nodal level projection (to approximately enforce incompressibility of the new-time velocity field). In a multilevel calculation, we also require sync projections at the end of each coarse time step, both a MAC sync projection and a nodal sync projection.

The right-hand-sides for the level projections are created using single-level operations; they do not "see" data at another level. The right-hand-sides for the sync projections, however, are accumulated using multilevel operators, i.e. the right-hand-side "sees" both the coarse and the fine data. However, for the sake of computational efficiency, in the form of the algorithm presented here, the elliptic equations resulting from the MAC sync projection and the nodal sync projection are solved only on the coarse level. Both of these projections are solved using the multigrid-preconditioned conjugate gradient method [33].

In each case we solve an equation of the form

$$\nabla \cdot \frac{1}{\rho} \nabla p = RHS \tag{23}$$

If we have an initial guess p_0 to the solution p, we put this in "residual-correction" form for the solver. Thus we solve

$$\nabla \cdot \frac{1}{\rho} \nabla p_{diff} = RHS - \nabla \cdot \frac{1}{\rho} \nabla p_0 \tag{24}$$

The boundary conditions for the resulting system on the union of grids at a given level will be homogeneous Neumann (such as at a solid wall or axis of symmetry) or homogeneous Dirichlet (such as at a coarse-fine grid boundary or at outflow). At the coarse/fine interface, the solution is specified by linear interpolation from the coarse grids for the nodal solves, and quadratic or higher interpolation from the coarser level for the cell-centered solves.

In preliminary development, we attempted to use standard multigrid techniques to solve the resulting linear systems. These standard multigrid techniques used the coefficients in defining the interpolation operator (see [2] for the details or [1] for background information on the weighting), but would not converge for many problems with high density ratios. For an axisymmetric bubble rise problem with two levels of adaptivity, for example, we could not compute with density ratios greater than 10:1.

As a result, we have implemented the multigrid-preconditioned conjugate gradient method (MGPCG, [33]) to solve the linear systems. This allows us to run the bubble and drop problems that previously failed at the proper density ratio (816:1).

The preconditioner is a single multigrid V-cycle (see [35] for an introduction to multigrid) with the following properties, motivated by the need for the preconditioner to a conjugate gradient solve to be symmetric:

- The interpolation and restriction operators have no coefficient-weighting, and satisfy $cR^T = I$ (i.e. the restriction operator equals the transpose of the interpolant)

- Symmetric multicolor Gauss-Seidel relaxation is used as the smoother at each level of the V-cycle. For the nodal nine-point stencil in two dimensions, we use a four-color Gauss-Seidel relaxation step. On the way down the V-cycle, the order is RBGW. On the way up, the order is WGBR. Likewise, for the three-dimensional seven-point stencil, we use a multi-colored relaxation scheme in which the ordering is again reversed on the way up the V-cycle. For the MAC projections, red-black Gauss-Seidel relaxation is used, with the ordering RB on the way down the V-cycle and BR on the way back up.

- At the coarsest level of the V-cycle, the "bottom solver" is a preconditioned conjugate gradient solver. The preconditioner for this bottom solver is again symmetric multicolor Gauss-Seidel relaxation as described above (i.e. RBGW then WGBR for nodal, or RB then BR for cell-centered). The equation at the coarsest level must be solved to a tolerance two orders of magnitude smaller than the tolerance of the overall conjugate gradient solver, or the multigrid as preconditioner will not be sufficiently symmetric.

- If the boundary conditions are all homogeneous Neumann (only the case if the grids cover the entire domain and the physical boundary conditions are all Neumann), discrete solvability is enforced by ensuring that the sum of the right-hand-side is discretely zero.

- The elliptic operator at each level of the V-cycle is identical in form but with coarsened coefficients from the finer levels. The coefficients $\sigma = \frac{1}{\rho}$ are each associated with a directional flux and are coarsened by doing an arithmetic average transverse to each "flux" and a harmonic average parallel to the flux (see [2] for the details).

4.5 Viscous Solve

Standard multigrid V-cycles are used to solve the parabolic linear equations resulting from the Crank-Nicolson discretization of the viscous terms in the momentum equation. In the case of variable viscosity (here implemented only in two dimensions), the equations are coupled, and the stencil is a standard nine-point cell-centered discretization of the operator which can be written in flux form, where the fluxes now contain not only normal but tangential derivatives. For constant viscosity, the equations are decoupled, and standard five-point in two dimensions, seven-point in three-dimensions, cell-centered discretizations are used exactly as in [2].

The issues in solving a five-point in two dimensions, seven-point in three dimensions, cell-centered parabolic or elliptic equation on a union of fine rectangles embedded in a union of coarser grids (which provide Dirichlet boundary conditions) are described in detail in Section 3.5 of [2]. Here we address the only additional issue, that of how to provide boundary conditions for a nine-point (in two dimensions) rather than five-point stencil. The difficulty here is merely in how to define the value at each corner point of the stencil when that point lies outside the fine grid.

There are three possibilities for the corner ghost cell: 1) it lies in another fine grid, in which case the value is supplied by the other fine grid; 2) it lies outside the physical boundary, in which case the value is supplied by the physical boundary conditions, or 3) the value must be interpolated from the coarse grids. The interpolation scheme for such ghost cells is described in detail in [2] when the ghost cell is aligned with a row of fine grid cells, such as is always the case with a five-point stencil, and is the case for the nine-point stencil except at the corner of the grid. At the fine grid corners, rather than interpolate between fine grid and coarse grid points along a diagonal, the ghost cell value is filled by extrapolating from ghost cell values along one of the edges intersecting that corner. This dependence then must be captured in the relaxation coefficients within the Gauss-Seidel relaxation.

4.6 Details of Time-Stepping Procedure

The details of the adaptive time-stepping and synchronization procedures for incompressible variable-density flow are given in detail in [2]. Here we focus on the modifications necessary for the level set formulation and give details just of the *additional* operations.

First, as noted earlier, the level set function rather than density is advected; the density can be defined at any point as a function of ϕ. As a result, advective flux registers ($\delta F_{\phi,adv}$)

are defined for the level set function as well as for velocity (but no longer for density). Viscous flux registers ($\delta F_{\vec{u},visc}$) are defined for velocity only.

Assume now that we are advancing level ℓ, $0 \le \ell \le \ell_{max}$, one level ℓ time step. Let $U^{n,\ell}$ and $\phi^{n,\ell}$, be the velocity and level set function at time $n\Delta t^\ell$ on the level ℓ grid, where Δt^ℓ is the time step of the level ℓ grid. Let A^ℓ be the area of a face (assumed the same in each coordinate direction) at level ℓ, and let Vol^ℓ be the volume of a grid cell at level ℓ.

Once we complete the predictor step at level ℓ, if $\ell < \ell_{max}$, we initialize the level ℓ advective flux registers (defined only at the faces on the ℓ / $\ell+1$ interface and indexed by level ℓ indices) by

$$\delta F^\ell_{\phi,adv} := -\Delta t^\ell A^\ell (\vec{u}^{ADV,\ell} \tilde{\phi}^{n+1/2,\ell}).$$

If $\ell > 0$, we then update the level $\ell - 1$ advective flux registers (defined only at the faces on the $\ell - 1$ / ℓ interface and indexed by level $\ell - 1$ indices) by

$$\delta F^{\ell-1}_{\phi,adv} := \delta F^{\ell-1}_{\phi,adv} + A^{\ell-1} \sum_{faces} \Delta t^\ell (\vec{u}^{ADV,\ell} \tilde{\phi}^{n+1/2,\ell}).$$

Note that one level $\ell - 1$ face contains r^2 (in two dimensions it would be r) level ℓ faces; the sums above should be interpreted as summing over all level ℓ faces which are contained in the level $\ell - 1$ face.

If $\ell < \ell_{max}$ and r level $\ell + 1$ time steps have just been completed, the level ℓ and $\ell + 1$ data must now be synchronized. We average $\phi^{n+1,\ell+1}$ down onto the level ℓ grids wherever possible. This is a simple cell-centered averaging procedure, where for $r = 2$ in three dimensions, e.g., the level ℓ value becomes the average of the eight level $\ell + 1$ values occupying that volume. Pressure is also coarsened appropriately.

In order to do the refluxing of ϕ, we now define the cell-centered

$$S^\ell_{sync} = -\frac{1}{\Delta t^\ell \, Vol^\ell} \delta F^\ell_{\phi,adv}$$

on cells in the rows of level ℓ cells immediately *outside* the level $\ell+1$ grids and set $S_{sync} = 0$ elsewhere. This part of the correction to $\phi^{n+1,\ell}$ will make the scheme conservative again.

A MAC "sync projection" is now performed to correct for the fact that the MAC-projected velocities on the coarse and fine grids do not satisfy an effective composite divergence constraint. This results in a correction velocity field $\vec{u}^\ell_{corr}$ which is used to re-advect $\vec{u}$ and ϕ. S_{sync} is then updated:

$$S^\ell_{sync} := S^\ell_{sync} + \nabla \cdot (\vec{u}^\ell_{corr} \, \tilde{\phi}^{n+1/2,\ell}).$$

If $\ell > 0$, we must also modify the level $\ell - 1$ flux registers to account for the fact that we will be adding S^ℓ_{sync} to level ϕ^ℓ. To do this, we set

$$\delta F^{\ell-1}_{\phi,adv} := \delta F^{\ell-1}_{\phi,adv} + A^{\ell-1} \sum_{faces} \Delta t^\ell (\vec{u}^\ell_{corr} \tilde{\phi}^{n+1/2,\ell}),$$

using the same summing convention.

We can now add the corrections to the level set function:

$$\phi^{n+1,\ell} := \phi^{n+1,\ell} + \Delta t^\ell S^\ell_{sync}$$

and if $\ell < \ell_{max}$, we interpolate the correction onto the fine grids at *all* finer levels, q, $\ell < q \leq \ell_{max}$ using conservative interpolation:

$$\phi^{n+1,q} := \phi^{n+1,q} + \Delta t^{\ell} Interp_{cons}(S^{\ell}_{sync})$$

At this point the nodal sync projection is performed to correct the new-time velocity and pressure. The sync-projection described here differs from that in [2] in that this is a single-level operation rather than a two-level solve as in [2]. After one coarse level time step on level ℓ and r fine level time steps on level $\ell + 1$, the velocity field on the coarse level will be out of sync with the velocity field on the fine level. That is to say, the discrete divergence of the composite velocity field

$$\vec{u}^{\ell,\ell+1} \tag{25}$$

across coarse-fine grid interfaces will not be divergence-free. A two-level sync-project is described in [2] in which this composite velocity field $\vec{u}^{\ell,\ell+1}$ is projected onto the space of composite velocity fields that are discretely divergence-free across coarse-fine grid interfaces. Unfortunately, implementing a composite multigrid preconditioned conjugate gradient method turns out to be quite a challenge (the solver implemented in [2] used just multigrid). We implement a single level sync-project similar in nature to the single level MAC sync-project. Consider the composite velocity field in (25). If we average down the fine level velocity $\vec{u}^{\ell+1}$ (where the fine level exists) onto the coarse level, then the resulting coarse level "composite" velocity field

$$\vec{u}^{\ell,avgdown} \tag{26}$$

will undoubtedly not be divergence-free with respect to the discrete coarse level divergence at cells neighboring coarse-fine grid interfaces. A single level sync-project is used to project the coarse level "composite" velocity field onto a divergence-free velocity with respect to the coarse level. The resulting changes to the coarse level velocity field lying underneath the fine level are interpolated onto the fine level in order to correct the fine level velocity field.

4.6.1 Level Set Synchronization

When using the level set formulation, we must include synchronization steps in order to maintain the regularity of the level set function ϕ and curvature $\kappa(\phi)$ across a coarse-fine grid interface. Typically, the position of the zero level set on the fine level will be different from that on the coarse level. Thus, after an average down step of the fine level to the coarse level, the distance function ϕ on the coarse level *outside* the fine level will no longer represent the distance from the zero level set.

The "sync-redistance" step is performed after the level $\ell + 1$ data has been advanced in time to reach the current time of the level ℓ data. At this time the level $\ell + 1$ data is averaged down onto the level ℓ data where possible. The following iteration is performed until $\tau = \alpha \Delta x^{\ell_{max}}$:

1. perform a single-level redistance step at level ℓ;

2. perform a single-level redistance step at level $\ell + 1$;

3. average the level $\ell+1$ level set function onto the level ℓ grids where possible.

4. Let $\tau = \tau + \Delta x^{\ell+1}/2$, return back to step 1.

As seen from the above iteration, the redistance equation is solved on two levels simultaneously without sub-cycling. Thus, the time-step $\Delta\tau$ will be $\Delta x^{\ell+1}/2$. The redistance step on level ℓ acts to propagate information from the fine level (where the interface lies) to the coarse level. The redistance step on level $\ell+1$ is performed using improved boundary conditions at the coarse-fine interface due to the redistance-step on level ℓ.

We have found that the "sync-redistance" step does not necessarily maintain sufficient smoothness of the curvature $\kappa(\phi)$ across a coarse-fine grid interface. The divergence of the surface tension term (13) can have unphysical jumps at coarse-fine grid boundaries since the surface tension term contains the curvature gradient. An example of the effects of these jumps are shown in Figure 2 (bottom row). The divergence of the surface tension term (which contains the curvature gradient) has an unphysical jump across the grid boundary causing an unphysical velocity field even after the first time step. In order to ameliorate this problem, we only directly compute the curvature at points within the thickness of the interface $|\phi| < \alpha\Delta x^{\ell_{max}}$. For curvature outside the thickness, we extend the curvature away from the interface using an iteration scheme. Given the current iterate for curvature $\kappa^{(m)}$ we compute $\kappa^{(m+1)}$ as follows:

1. for all grids cells (i,j,k) in which $|\phi_{i,j,k}| \geq \alpha\Delta x^{\ell_{max}}$ do the following:

$$\kappa_{i,j,k}^{(m+1)} = \frac{\sum_{i'=-1,j'=-1,k'=-1,|\phi_{i+i',j+j',k+k'}|<|\phi_{i,j,k}|}^{i'=1,j'=1,k'=1} \kappa_{i+i',j+j',k+k'}^{(m)}}{\sum_{i'=-1,j'=-1,k'=-1,|\phi_{i+i',j+j',k+k'}|<|\phi_{i,j,k}|}^{i'=1,j'=1,k'=1} 1}.$$

2. repeat step 1 for $\alpha\Delta x^{\ell}/\Delta x^{\ell_{max}}$ times.

The iteration, as with the "sync-redistance" step, acts to propagate information on the interface (under finer levels) out onto the coarse levels, thus maintaining smoothness across coarse-fine grid boundaries. A few important notes:

- We store the curvature as a separate variable from the level set function. At $t=0$, the curvature $\kappa_{i,j,k}$ is initialized as the curvature of the point on the interface (the zero level set) that is closest to $\vec{x}_{i,j,k}$. At later times, the curvature variable is updated as described in the iteration above. That is, we recompute the curvature near the zero level set, and iteratively extend the new changes away from the zero level set.

- The curvature near the interface is computed as:

$$\kappa(\phi) = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$$

  For 2-d problems, we use the following expression for the curvature $\hat{\kappa}$:

$$\hat{\kappa}(\phi) = \kappa(\phi)/(1 - \phi\kappa(\phi))$$

  If ϕ is a distance function, this method of computing curvature (see [36]) will give the curvature for the point on the zero level set closest to the point in question.

- In (13), we have the following expression for the surface tension force:

$$\vec{F}_{surf}(\phi) = \nabla\kappa(\phi)H(\phi)/W$$

We replace $\kappa(\phi)$ in the above equation with $\tilde{\kappa}_{i,j,k}$ where

$$\tilde{\kappa}_{i,j,k} = H_{1/2}(\frac{1}{2} - |\phi|)\kappa_{i,j,k}. \tag{27}$$

$\kappa_{i,j,k}$ is a result of computing the curvature near the interface and then extending the curvature outward. $\tilde{\kappa}_{i,j,k}$ will represent the curvature of the interface near the interface, but will gradually approach zero far from the interface. After non-dimensionalization, the dimensional length scale L (e.g. the radius of a bubble) corresponds to a length of 1 using dimensionless parameters. Thus for the example of a bubble or drop problem, (27) implies that $\tilde{\kappa}$ will be zero a length of half the bubble radius from the zero level set.

In the top row of Figure 2, we display the initial velocity field using our new method for handling the curvature as it appears in the surface tension term. In Figure 3, we display the contours of the new curvature term $\tilde{\kappa}$ for a rising gas bubble. The contours are smooth even across a coarse-fine grid interface. In Figure 3 we also show the curvature contours for the curvature $\kappa(\phi)$ as computed in the original single grid algorithm. Here, the contours are not smooth across coarse-fine grid boundaries.

5 Numerical Examples

We present gas/liquid computations on a two-dimensional axisymmetric (r-z) grid and on a fully three-dimensional grid. In all our examples below, the density ratio is at least 714:1. We validate our method via convergence checks, direct comparison with other numerical methods, and finally comparison with experiments. We show highly resolved computations of a gas bubble rising in liquid and also of a drop splashing against a pool of water. As seen by the results below, our method provides one with the capability of computing very complicated flows on a modest machine (e.g. Pentium-Pro 200Mhz).

5.1 Convergence Check

Our test problem will be a rising inviscid air bubble in water with surface tension. For this problem, we used an axisymmetric domain. The density ratio is 816 : 1 and the Weber number is 200. In Figure 4, we display the bubble at times $t = 0.0$, $t = 1.2$ and $t = 1.3$. The spatial mesh size on the finest level is $\Delta x^{\ell_{max}} = 6/512$ and the interfacial thickness parameter is $\alpha = 3$. The solid line represents results for the same problem except using the boundary integral method [31]. In order to compare with the boundary integral method, we use far-field boundary conditions on all sides of the domain except at $r = 0$. For our boundary condition, we assume that the pressure on the walls is $p = \rho_l z/Fr$. In Table I, we show that the solution converges at a rate of $O(h^{1.5})$ as progressively finer levels are added. The volume $V_{t=1.3}$ and error between successively refined computations $E_{t=1.3}$ were measured as:

$$V_t \equiv \int_\Omega (1 - H(\phi(t))d\Omega$$

$$E_t \equiv \int_\Omega |H(\phi_c(t)) - H(\phi_f(t))| d\Omega$$

ϕ_c is the result from a coarser computation, and ϕ_f is the result from the refined computation.

We have confidence that our adaptive solution is converging to the proper solution. First of all, we compare well with the boundary integral method (see [31] for single grid comparison with the boundary integral method). Secondly, we compared the adaptive results in which $\Delta x^{\ell_{max}} = 6/256$ to the single grid results in which $\Delta x = 6/256$ over the whole domain. The error between these two computations at $t = 1.3$ is 0.003 which is considerably less than the errors listed in table I. The speedup of the adaptive computation over the corresponding single grid computation was a factor of 2.4.

Table I: Convergence study for W=200 and $\alpha = 3$

$\Delta x^{\ell_{max}}$	$V_{1.3}$	$E_{1.3}$	order
6/64	4.111	N/A	N/A
6/128	4.154	0.170	N/A
6/256	4.177	0.059	1.5
6/512	4.187	0.019	1.6

5.2 Computations in which the interface has corners

In this section, we compute problems in which the finest levels of adaptivity are focused on regions of high curvature (such as corners) and high vorticity. For these problems, the zero level set does not necessarily have to be completely contained by the finest level. In section 4.6.1 care was taken in order to maintain a smooth level set function even across coarse-fine grid boundaries. It is possible to have the zero level set cross a coarse-fine grid boundary. In choosing our criteria for which regions to adapt, our choice on the first few levels would be to adapt in regions that contain the zero level set. In these regions there is a large jump in density. Our strategy for adapting on finer levels would be to focus attention to regions of high curvature or high vorticity. These regions typically cross the zero level set.

On the left of Figure 5, we compute the calculation of an axisymmetric rising air bubble in water in which the last two levels of refinement focus on regions of high vorticity and high curvature. Five levels of adaptivity are used for this problem. We compare these results to the case where only three levels are used (right side). The interfacial thickness for these two examples is the same. At time $t = 1.44$, we see that extra adaptivity allows us to resolve the thin structure near the breaking up of the bubble.

In Figure 6, we compute an axisymmetric gas bubble rising in a viscous liquid. As in the previous example, we use far-field boundary conditions at the wall. For this problem the density ratio is 714:1 and the viscosity ratio is 6667:1. The Reynolds number, Froude number and Weber number are 9.7, 0.78, and 7.6, respectively. These are the same parameters used in bubble experiments by Hnat and Buckmaster [18] and used in steady bubble computations by Ryskin and Leal [27]. For this problem, we have an extra level of adaptivity in the region of highest curvature. In Figure 7, we compare the volume of the bubble when computed with the extra adaptivity as opposed to without. In Figure 8, we display the position of the center of mass of the bubble versus time. The average dimensionless rise speed for this case was 1.006 which differs from the experiments by 0.6%. We believe that some of the error in the computed steady rise-speed is attributable to the fact that we compute in a limited domain and use far-field boundary conditions. When our computation was run in a domain one quarter the size (each dimension cut in half) the average dimensionless rise speed was 1.036, a difference of 4% from the experiments. The advantage of adaptivity here is that enlarging the domain adds cells (and therefore computational effort) only at the coarsest level; the fine grids covering the bubble remain the same size. The time to run from $t = 0$ to $t = 6.25$ in the large domain (5x20) was 6971 seconds (on a Pentium-Pro 200Mhz); the time to run on the small domain (2.5x10) was 5784 seconds (for $t > 6.25$, the bubble starts to exit the top end of the small domain).

5.3 Fully 3d simulations of rising gas bubble(s)

In this section we first show the computation of the rise of a fully three dimensional inviscid air bubble in water (see Figure 9). The density ratio is 816:1 and the Weber number is 200. The dimensions of the domain are 4x4x8 and the effective number of computational cells on the finest level of adaptivity (the third level) is 64x64x128. We use far-field boundary conditions on all sides of the domain. In Figure 10, we display a cross-section of the bubble at $t = 1.2$ and $t = 1.4$ and compare these results with the results of an axisymmetric bubble problem in which the effective fine grid resolution is 128x256. We point out here that we do not have to do any extra programming in transitioning from a spherical cap bubble into a toroidal bubble.

In Figure 11, we display the interaction of two inviscid gas bubbles in water. The density ratio is 816:1 and there is no surface tension active. The dimensions of the domain are 4x4x8 and the effective number of computational cells on the finest level of adaptivity (the second level) is 32x32x64. We use far-field boundary conditions on all sides of the domain.

5.4 Water Splash Problem

In the work of Oguz & Prosperetti (1990) [23], the boundary integral method was used to study the impact of drops on liquid surfaces and the subsequent entrainment of an air bubble. Their work has important implications concerning the sound generated by rain. In results shown here, we use our adaptive level set method to compute the impact of a water drop on a pool of water along with the "splash" that comes afterward. With the level set method, we automatically handle the merge of the drop with the pool of water and also the break-up of the water splash. In our computations, we use dimensionless parameters based on the impact velocity U and the radius of the drop L. In Figure 12, we show results using $L = 1$ mm and $U = 4.0$ m/s. In Figure 13, we show results using $L = 1$ mm and $U = 7.6$ m/s. The dimensionless impact velocity is 1; we accelerate the drop with a fictitious gravitational force term $\frac{1}{Fr} = 1/2$ for a total dimensionless time 2. At dimensionless time $t = 2$, the drop will be traveling with dimensionless speed of 1 and will have traveled a dimensionless distance of 1 (which is the initial distance between the drop and the pool). For $U = 4.0m/s$ we have $Re = 3518$, $Fr = 1633$, and $We = 220$. For $U = 7.6m/s$ we have $Re = 6684$, $Fr = 5895$, and $We = 794$. As suggested by the difference in Weber number, the spray in the results for $U = 4.0m/s$ (Figure 12) coagulates at the tip whereas the spray in the results for $U = 7.6m/s$ (Figure 13) breaks up. The results for

$U = 7.6m/s$ do concern us. Is it a physical phenomenon that the spray at $t = 5.4$ should curl back towards $r = 0$? We ran the same problem using a uniform fine 128x256 grid (Figure 14) and obtained qualitatively the same results as when the problem is computed on an adaptive grid. The uniform grid calculation used 22441 seconds (on a Pentium-Pro 200Mhz), four times more than the adaptive computation.

6 Summary

We have presented an adaptive level set method for computing free-surface flows in which large jumps in density and viscosity occur at the free-surface. Surface tension forces are included in our numerical model. We show examples in two and three dimensions in which the arbitrary merging and break-up of fluid mass may take place. We use adaptive mesh methodology in order to focus computational effort on regions of high curvature or high vorticity. We have validated our method against the bubble experiments of Hnat and Buckmaster [18] and the boundary integral computations of Sussman and Smereka [31]. Finally, we have conducted a convergence study of our method in order to measure the order of accuracy for the problem of a rising inviscid air bubble in water.

References

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome. A conservative adaptive projection method for the incompressible Navier-Stokes equations in three dimensions. *LBNL Report 39075, UC-405*, 1996.

[3] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible Navier-Stokes equations based on an approximate projection. *SIAM J. Sci. Comput.*, 17(2), March 1996.

[4] J. B. Bell, M. J. Berger, J. S. Saltzman, and M. L. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 15(1):1277–138, January 1994.

[5] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, June 24–27, 1991.

[6] J. B. Bell and D. L. Marcus. A second-order projection method for variable-density flows. *J. Comput. Phys.*, 101:334–348, 1992.

[7] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[8] M. J. Berger and I. Rigoustsos. An algorithm for point clustering and grid generation. Technical Report NYU-501, New York University-CIMS, 1991.

[9] J.M. Boulton-Stone and Blake. Gas bubbles bursting at a free surface. *J. Fluid Mech.*, 254:437–466, 1993.

[10] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100:335–353, 1992.

[11] D. Chambers, D. Marcus, and M. Sussman. Relaxation spectra of surface waves. In *Proceedings of the 1995 International Mechanical Engineering Congress and Exposition*, November 1995.

[12] Y.C. Chang, T.Y. Hou, B. Merriman, and S. Osher. A level set formulation of eulerian interface capturing methods for incompressible fluid flows. *J. Comput. Phys.*, 1995. to appear.

[13] P. Colella. A direct Eulerian MUSCL scheme for gas dynamics. *SIAM Journal on Computing*, 6:104–117, January 1985.

[14] N.V. Deshpande. Fluid mechanics of bubble growth and collapse in a thermal ink-jet printer. In *SPSE/SPIES Electronic Imaging Devices and Systems Symposium*, January 1989.

[15] A. Esmaeeli and G. Tryggvason. An inverse energy cascade in two-dimensional low reynolds number bubbly flows. *J. Fluid Mech.*, 314, 1996.

[16] A. Harten. *J. Comput. Phys.*, 83:148–184, 1989.

[17] Munehiko Hinatsu and Haruya Takeshi. Breaking waves in front of a box-shaped ship. In *2nd Japan-Korea Workshop on Ship and Marine Hydrodynamics, Osaka, Japan*, 1993.

[18] J.G. Hnat and J.D. Buckmaster. Spherical cap bubbles and skirt formation. *Physics of Fluids*, 19 (2):182–194, 1976.

[19] M. S. Longuet-Higgins and E. D. Cokelet. Deformation of steep surface waves on water i: A numerical method of computation. *Proc. R. Soc. Lond. A.*, 350:1–26, 1975.

[20] T. S. Lundgren and N. N. Mansour. Oscillations of drops in zero gravity with weak viscous effects. *J. Fluid Mech.*, 194:479, 1988.

[21] T. S. Lundgren and N. N. Mansour. Vortex ring bubbles. *J. Fluid Mech.*, 224:177, 1991.

[22] M. L. Minion. On the stability of Godunov-projection methods for incompressible flow. *J. Comput. Phys.*, accepted for publication,1996.

[23] H.N. Oguz and A. Prosperetti. Bubble entrainment by the impact of drops on liquid surfaces. *J. Fluid Mech.*, 203:143–179, 1990.

[24] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[25] Elbridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William G. Rider. A second-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comput. Phys.*, 1996. accepted for publication.

[26] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal.*, 29(3):867–884, 1992.

[27] G. Ryskin and L.G. Leal. Numerical solution of free boundary problems in fluid mechanics. part 1 the finite-difference technique. *J. Fluid Mech.*, 148:1–17, 1984.

[28] C.W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes, ii. *J. Comput. Phys.*, 83:32–78, 1989.

[29] M. Sussman and E. Fatemi. An efficient, interface preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. submitted to the SIAM J. Sci. Stat. Comput.

[30] M. Sussman, E. Fatemi, P. Smereka, and S.J. Osher. An improved level set method for incompressible two-phase flows. *Journal of Computers and Fluids*, 1996. to appear.

[31] M. Sussman and P. Smereka. Axisymmetric free boundary problems. submitted to the Journal of Fluid Mechanics.

[32] M. Sussman, P. Smereka, and S.J. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[33] O. Tatebe. The multigrid preconditioned conjugate gradient method. In *6th Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, April 4–9 1993.

[34] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[35] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, New York, 1992.

[36] H. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127:179–195, 1996.

List of Figures

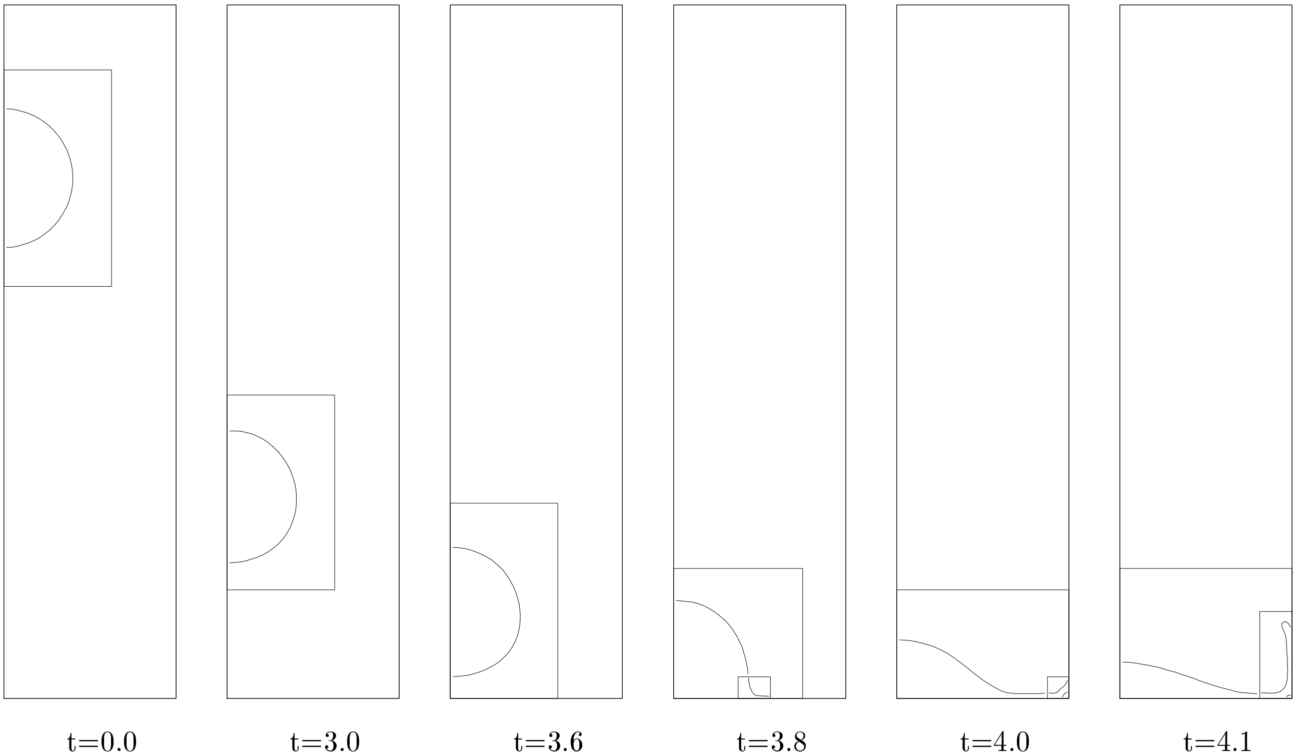


Figure 1: Falling 1cm spherical water drop in air; density ratio 816:1, viscosity ratio 71:1 64×256, $R = 2.75$, $W = 0.0135$. The grid lines shown are the grid boundaries of the level 1 grid, at later times, the level 2 grid as well.

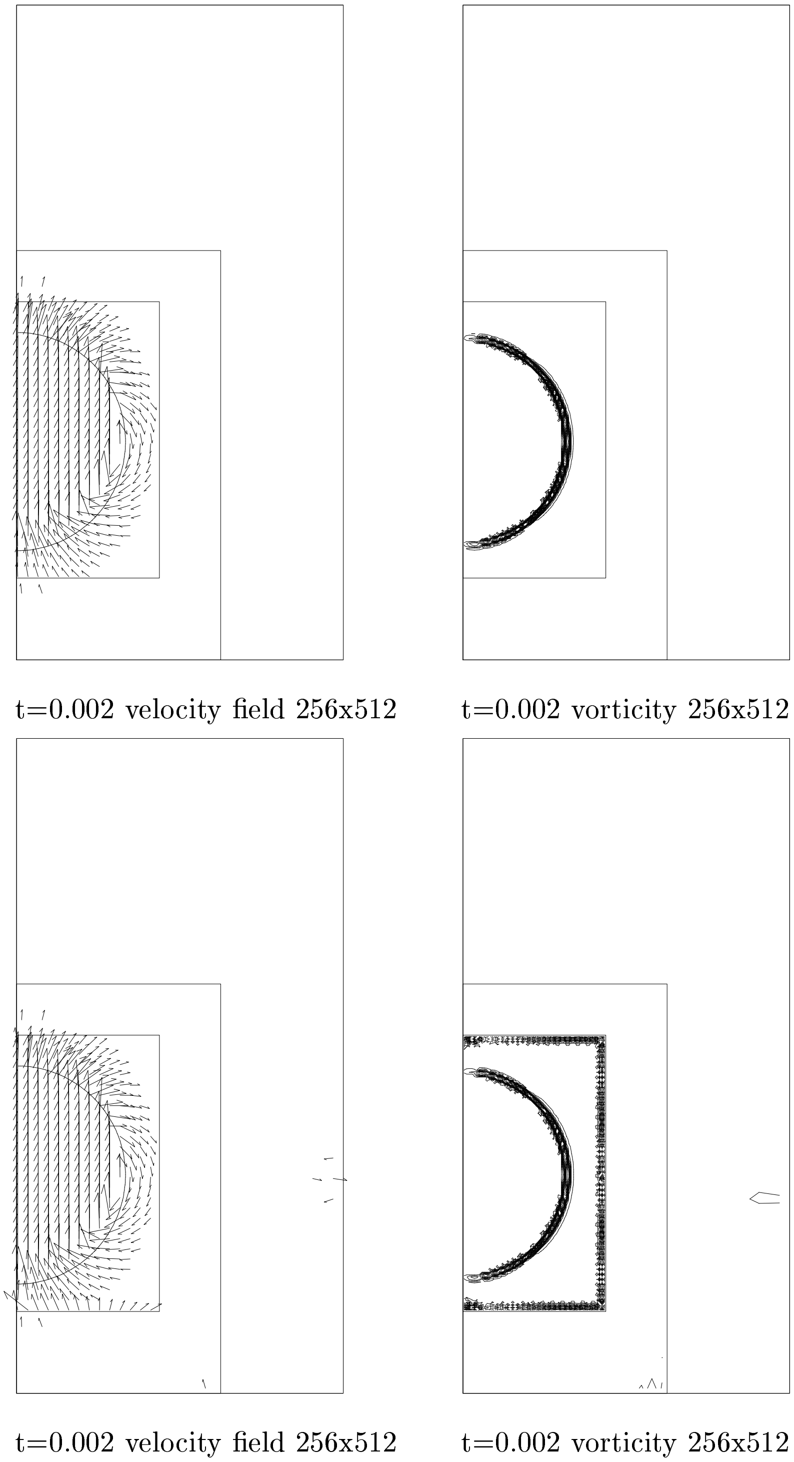
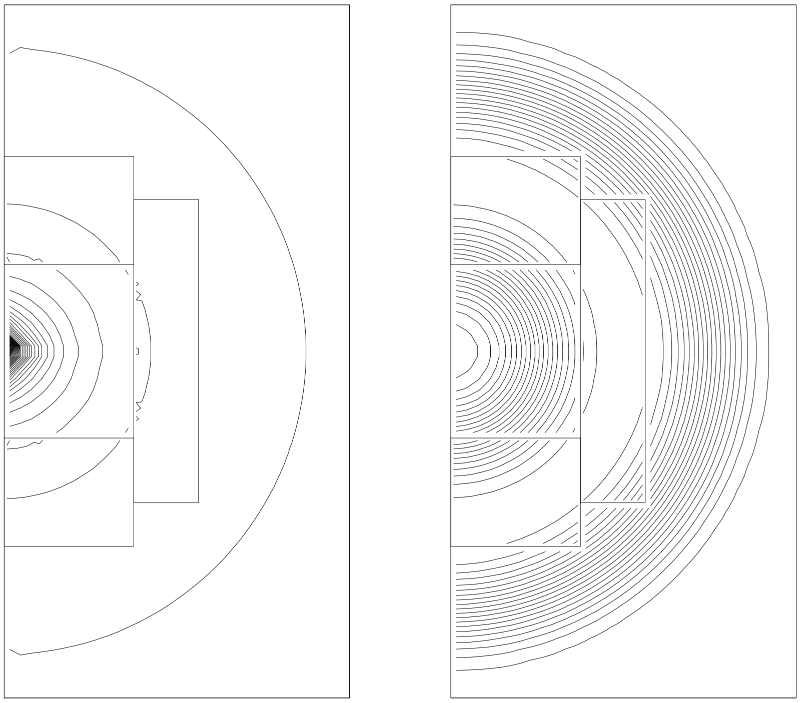
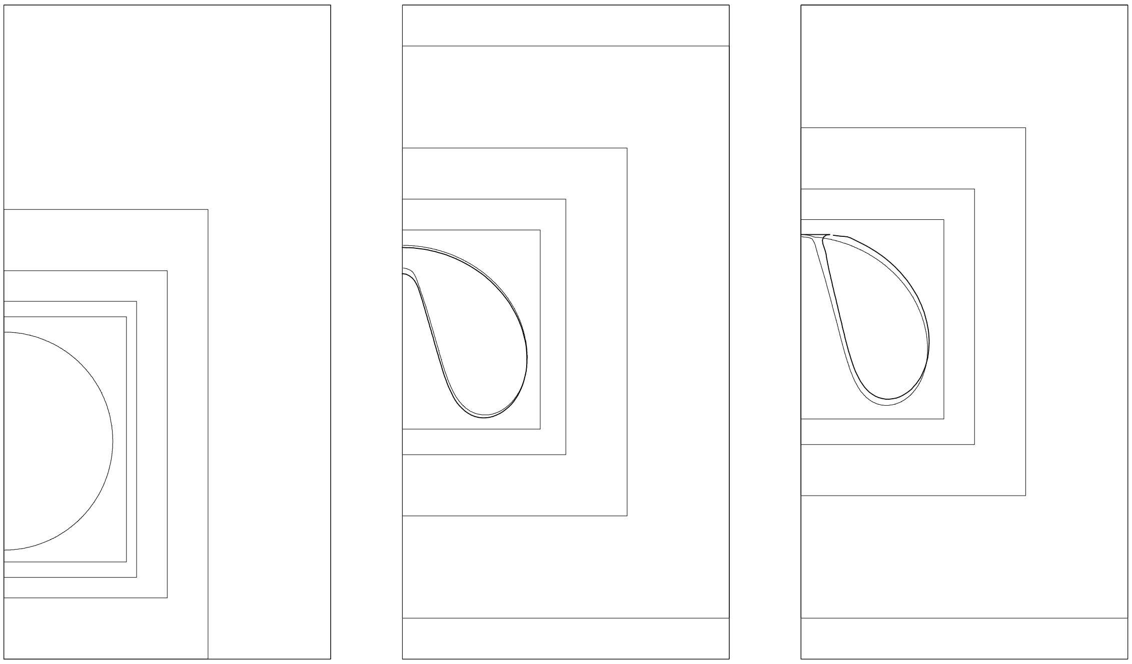


Figure 2: Spherical gas bubble in liquid; density ratio 816:1; We=10. Bottom: initial velocity field and vorticity after first time step due to irregularity in curvature across coarse-fine interface. Top: initial velocity field and vorticity after appropriate measures are taken in order to keep the curvature smooth across coarse-fine interface.



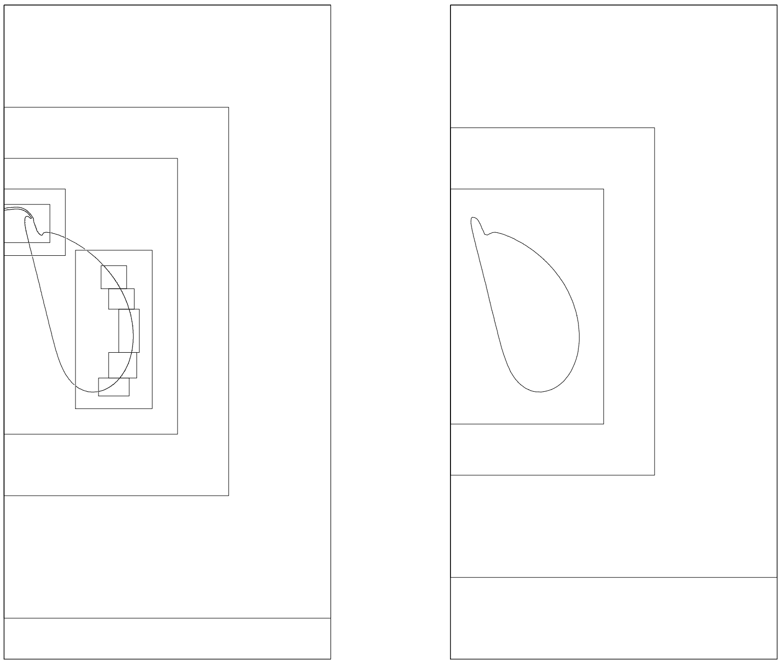
t=0.002 curvature κ t=0.002 smoothed curvature $\tilde{\kappa}$

Figure 3: Spherical gas bubble in liquid; initial contours of curvature variable κ before (left) and modified curvature variable $\tilde{\kappa}$ after (right) fix. Notice that $\tilde{\kappa}$ is smooth across coarse-fine grid boundaries.



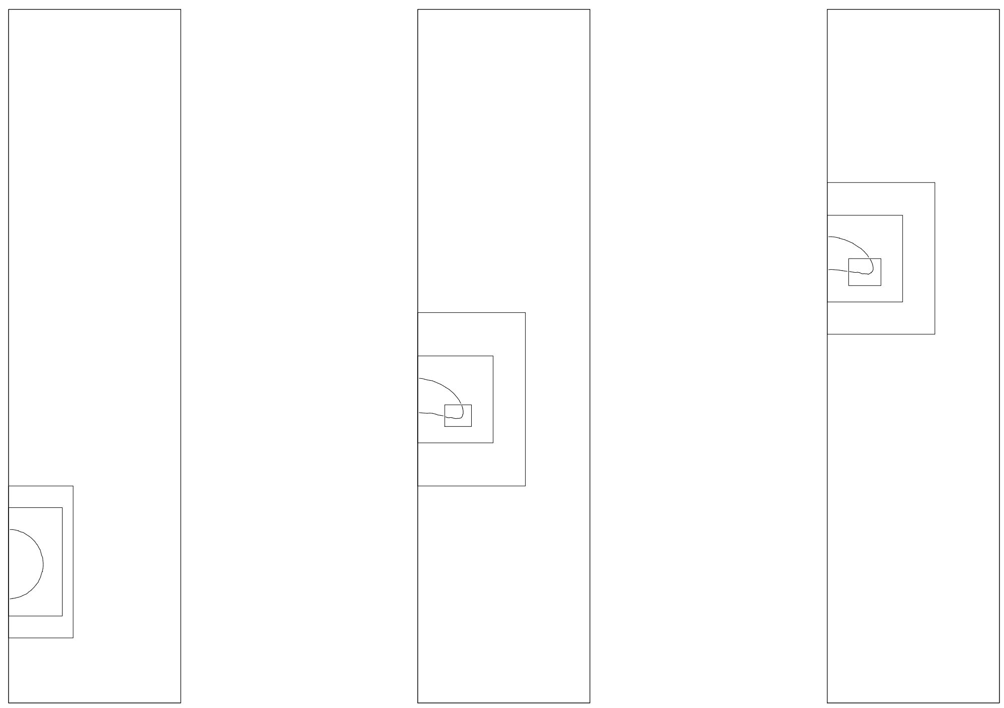
t=0 256x512 t=1.2 256x512 t=1.3 256x512

Figure 4: Spherical gas bubble in liquid: density ratio 816:1, We=200. Results computed using the adaptive levelset method (thin contour) are compared to results computed using the boundary integral method (thick contour)



t=1.44 512x1024 t=1.44 128x256

Figure 5: Spherical, inviscid gas bubble in liquid; Density ratio 816:1, We=200. For results on the left, two extra levels of adaptivity focus on regions of high vorticity and high curvature.



t=0.0 t=4.8 t=8.9

Figure 6: Rise of an initially spherical gas bubble in viscous liquid. An extra level of adaptivity is automatically added when corner forms in the ensuing cap bubble. density ratio 714:1, viscosity ratio 6667:1, $Re = 9.7$, $We = 7.6$, $Fr = 0.78$

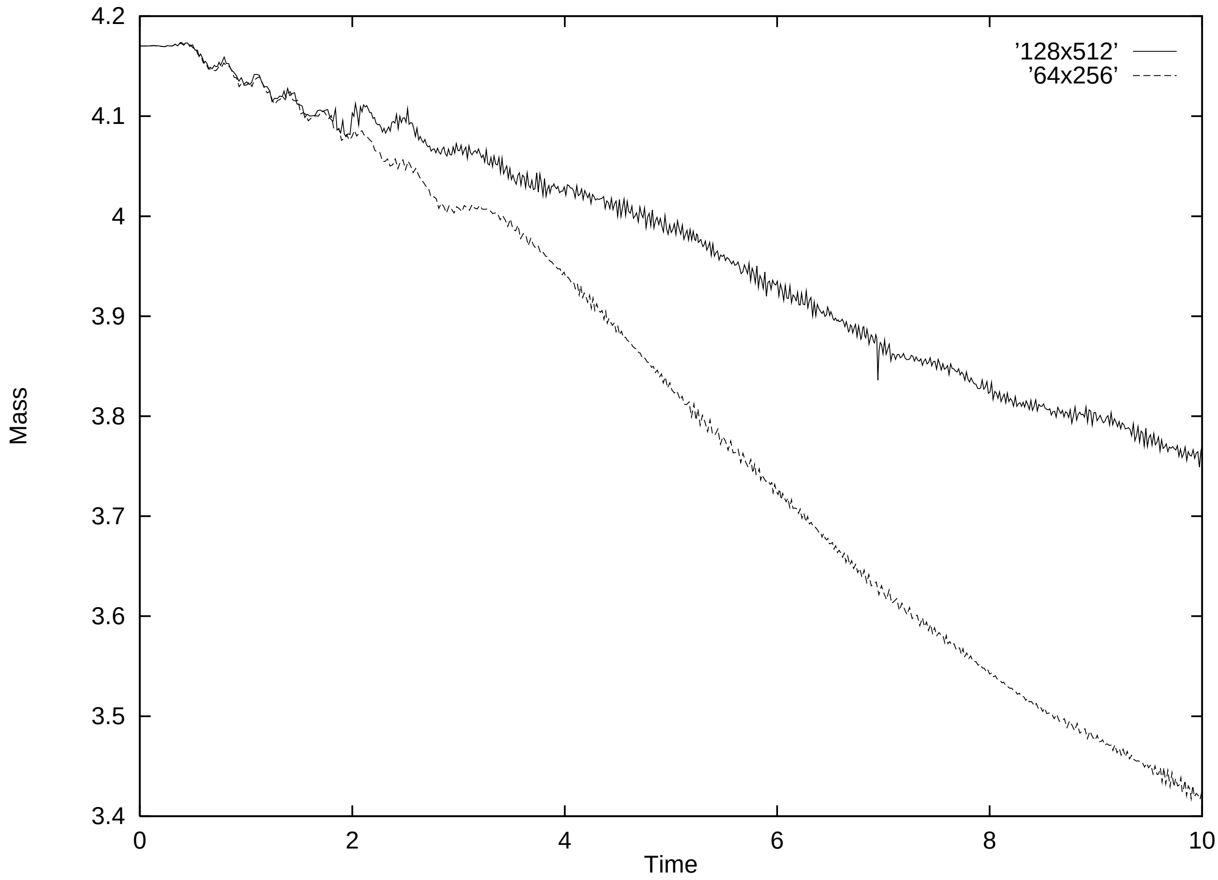


Figure 7: Plot of mass of a rising cap bubble vs. time. Data corresponding to "128x512" was computed in which an extra level of adaptivity was added when the corner formed on the cap (about $t = 2$). For data corresponding to "64x256," an extra level of adaptivity was not added.

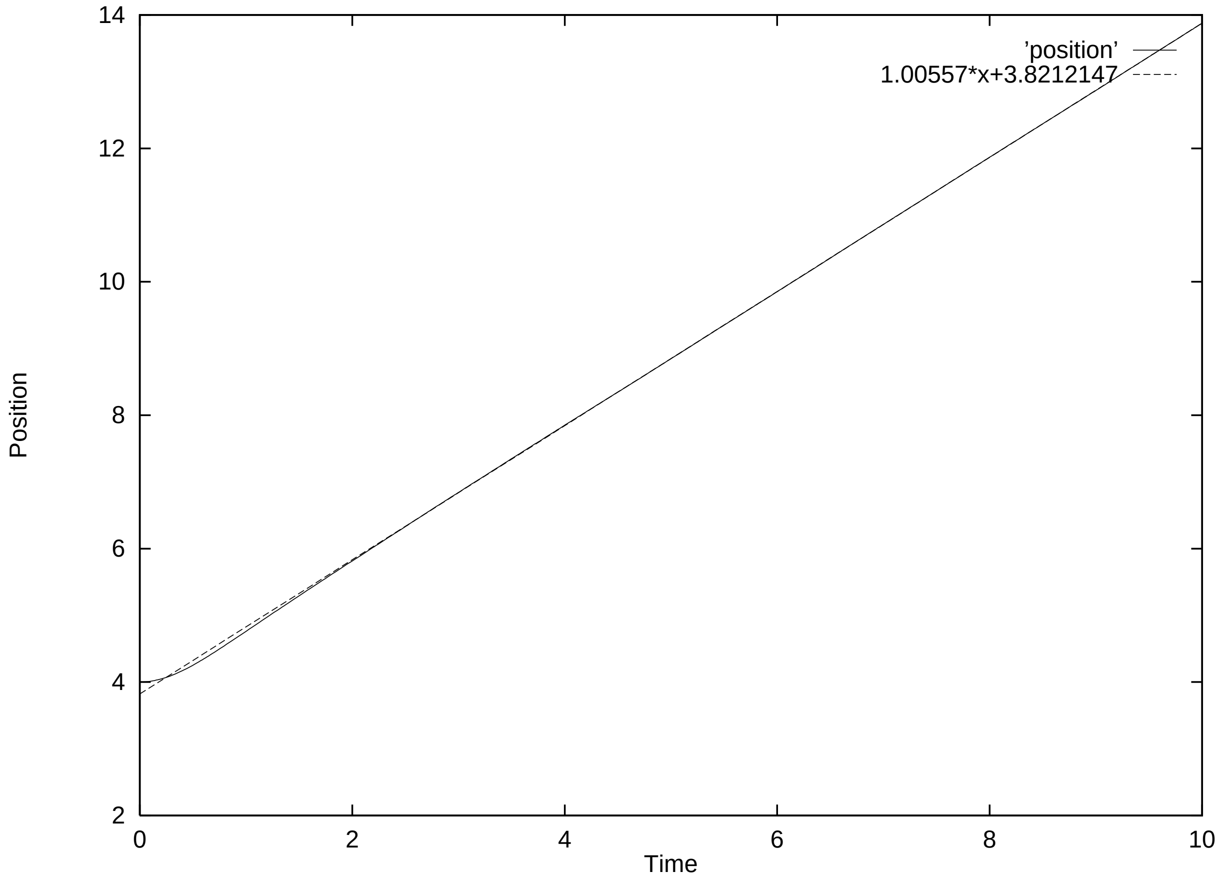
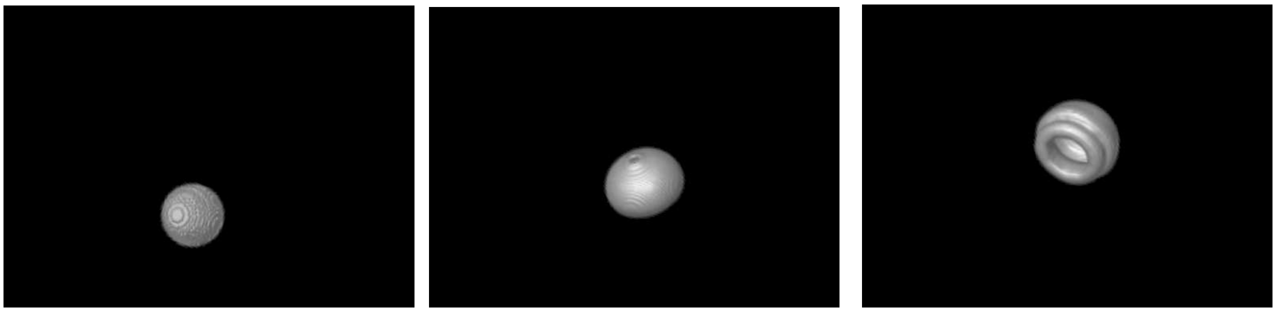
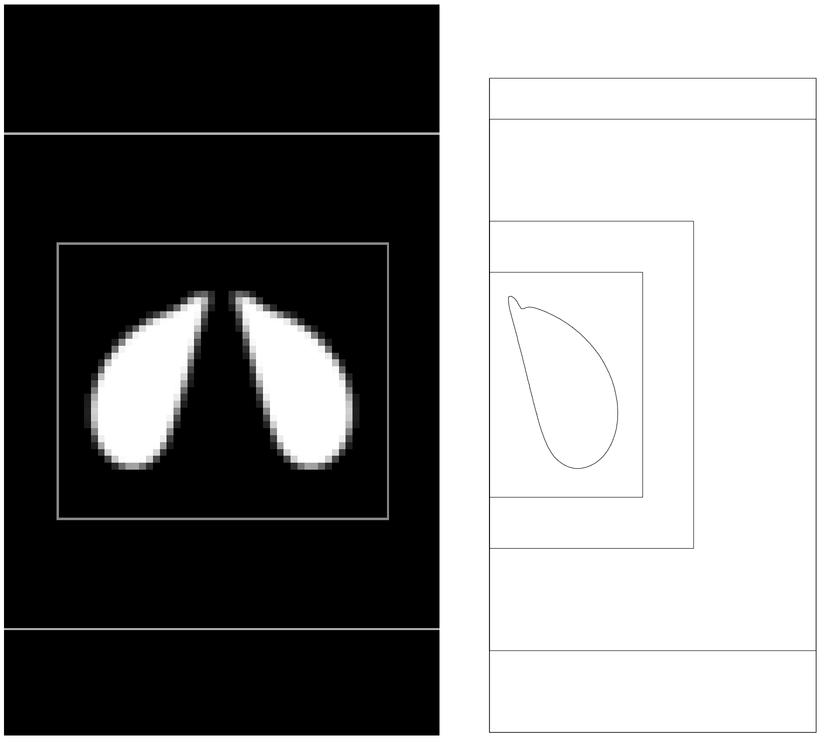


Figure 8: Plot of the center of mass of a rising cap bubble vs. time. We compare this data with the linear best fit for $2 < time < 10$. Expected slope is 1.

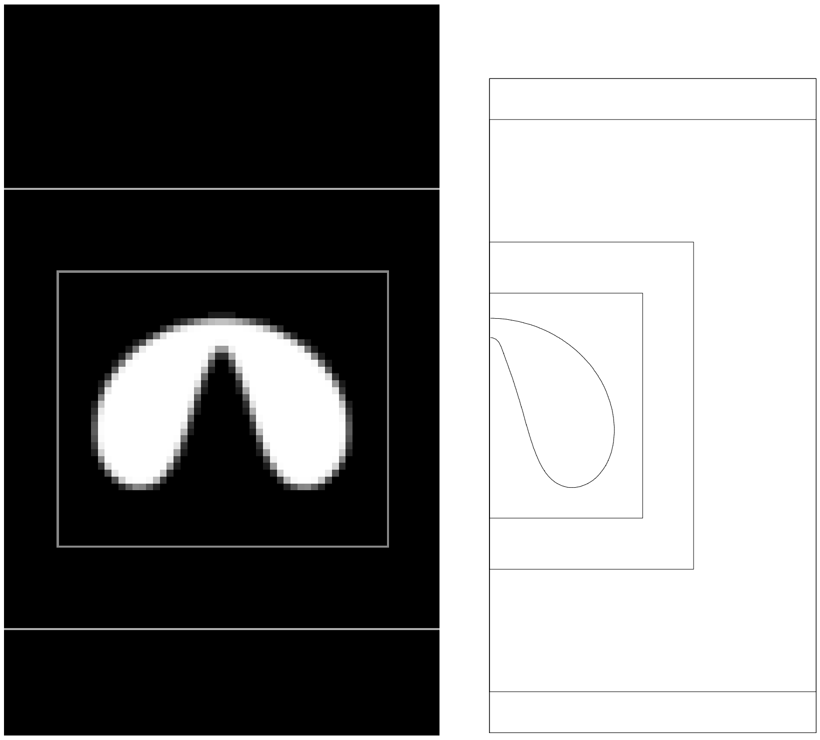


t=0.0 t=1.4 t=3.0

Figure 9: Rise of inviscid air bubble in water. $We = 200$, effective fine grid 64x64x128.



t=1.4 t=1.4



t=1.2 t=1.2

Figure 10: Spherical gas bubble in liquid; density ratio 816:1; We=200. Left: Cross section of three-dimensional results (y=2, x-z plane), effective fine grid 64x64x128, dimensions of domain: 4x4x8. Right: Axisymmetric results, effective fine grid 128x256, dimensions of domain 3x6.

t=0.0 t=1.27 t=1.43

Figure 11: Merge of two inviscid gas bubbles, effective fine grid 32x32x64.

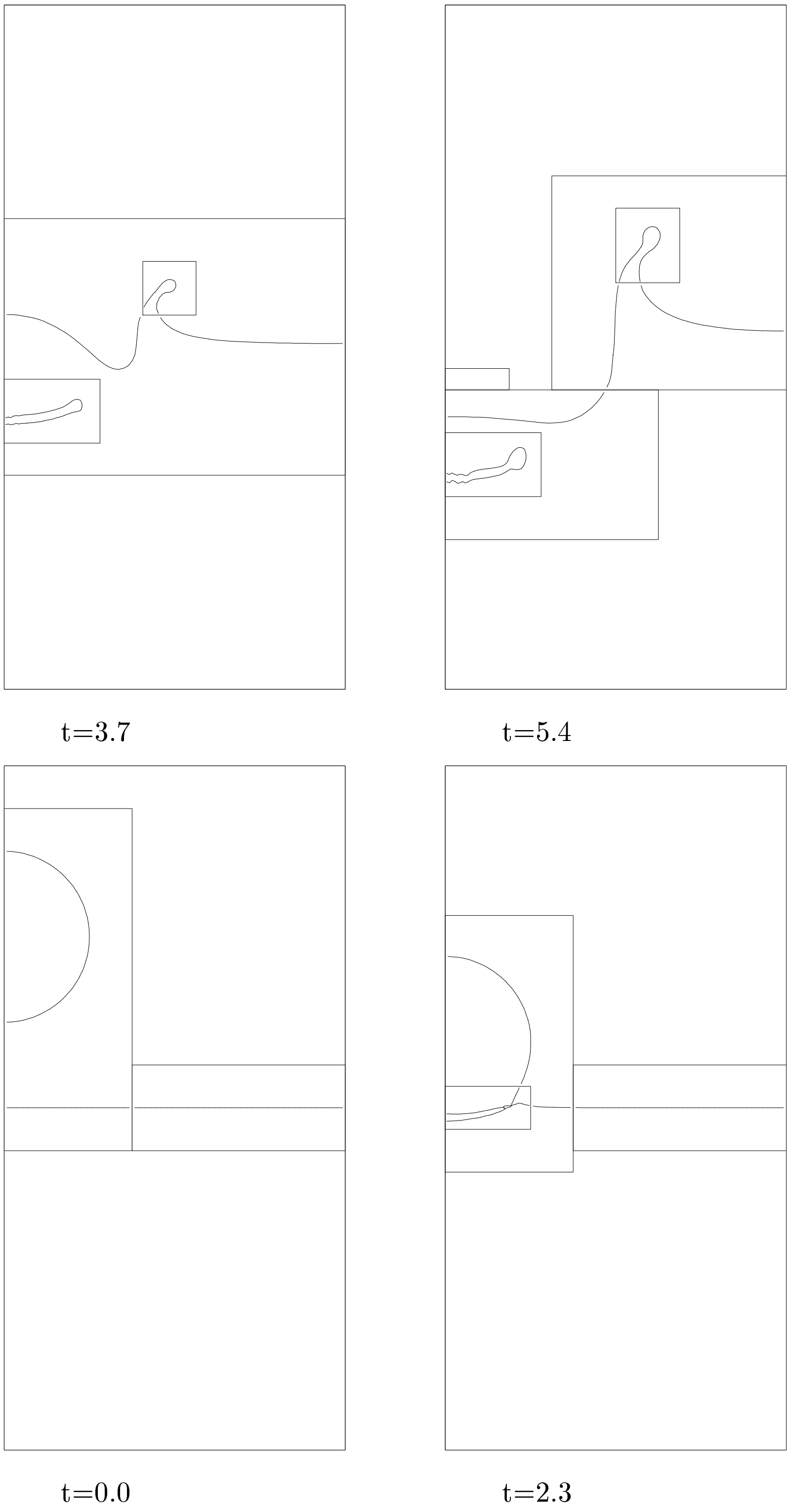
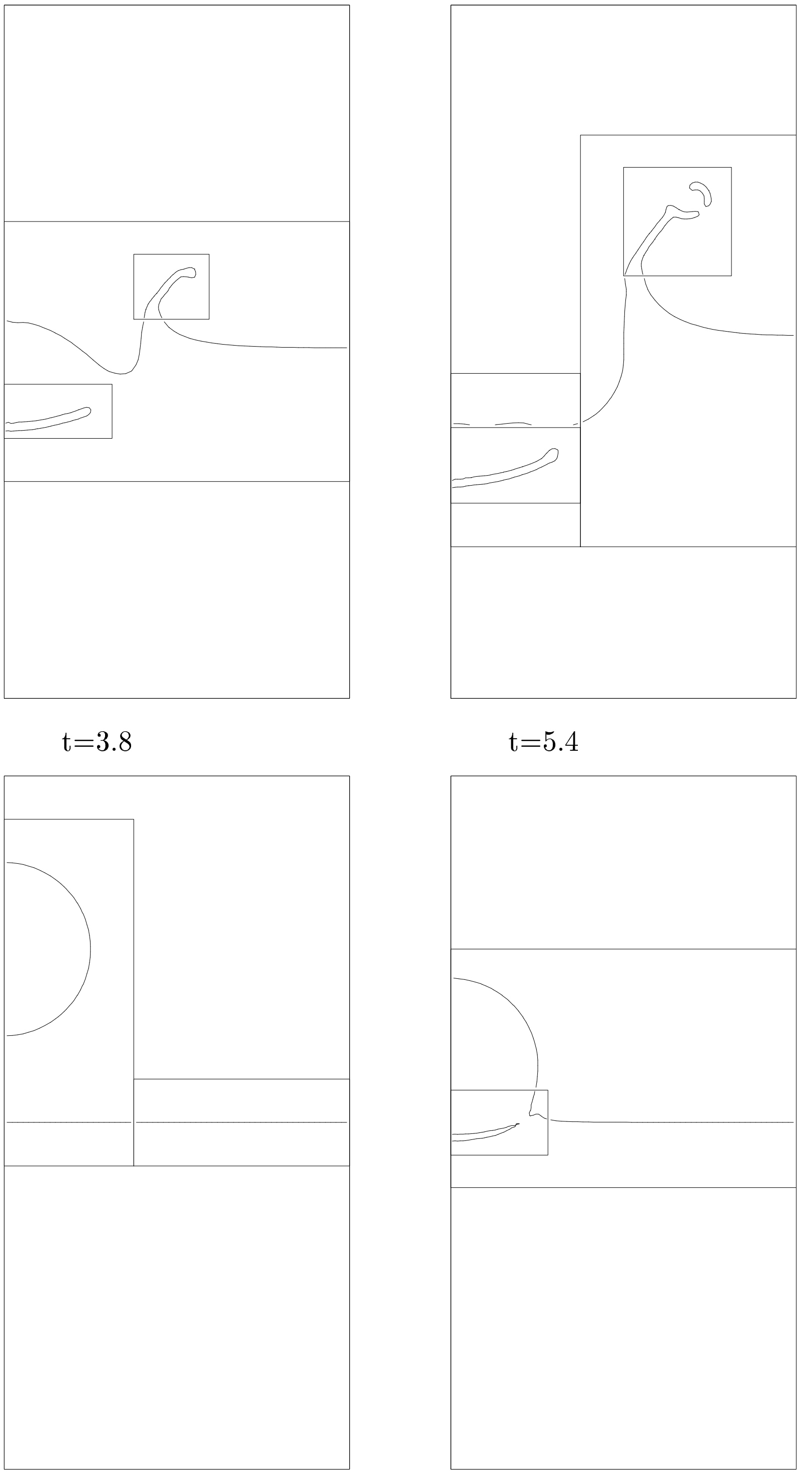


Figure 12: Falling 1mm spherical water drop onto pool of water; density ratio 1000:1, 128×256, impact speed $U = 4m/s$



t=3.8

t=5.4

t=0.0

t=2.4

Figure 13: Falling 1mm spherical water drop onto pool of water; density ratio 1000:1, 128×256, impact speed $U = 7.6m/s$

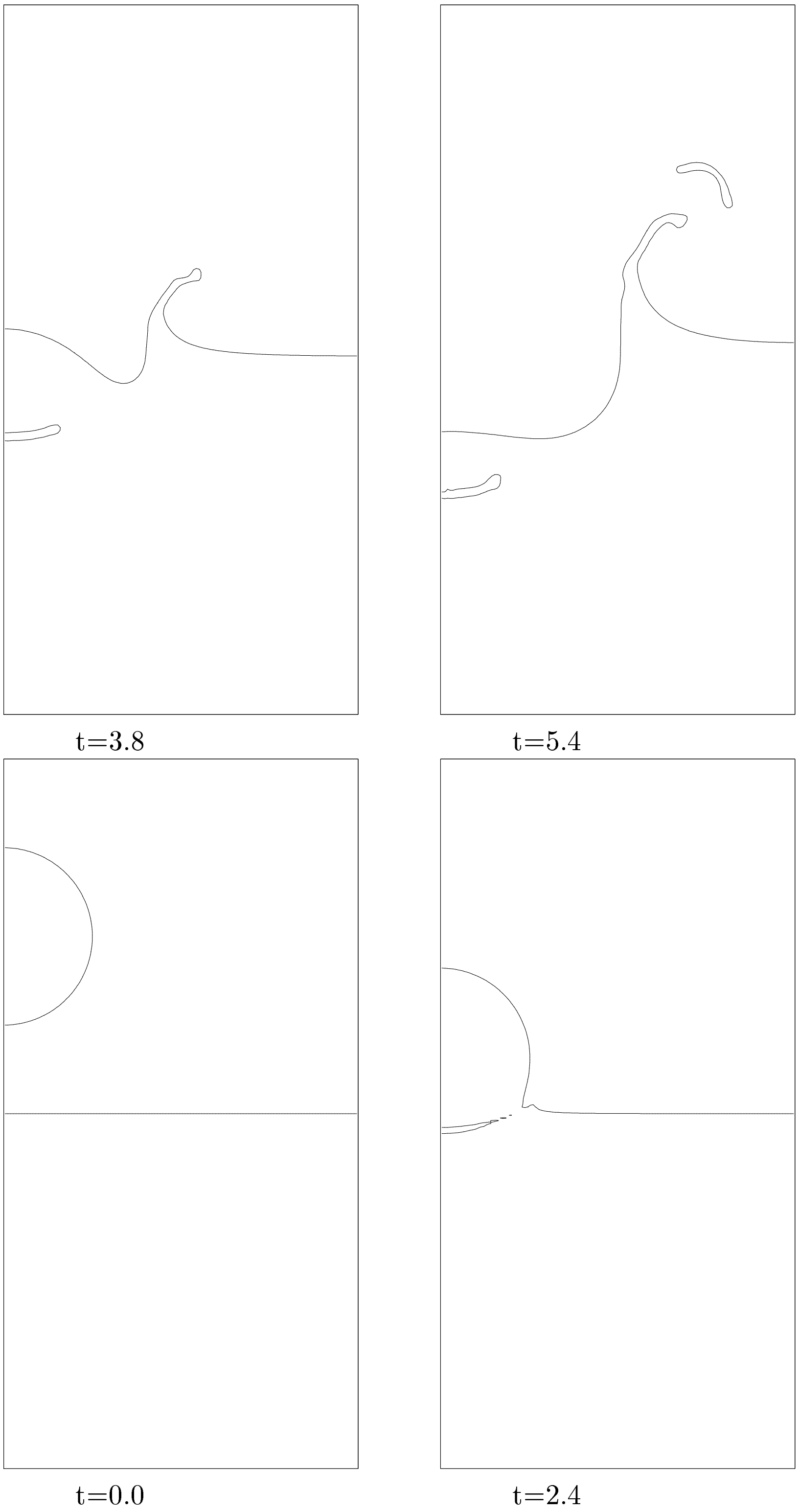


Figure 14: Falling 1mm spherical water drop onto pool of water; density ratio 1000:1, 128×256, impact speed $U = 7.6m/s$. Adaptive mesh refinement turned off.