A Boundary Condition Capturing Method for Multiphase Incompressible Flow [*]

Myungjoo Kang [†‡]
Ronald P. Fedkiw [§¶]
Xu-Dong Liu [‖**]

October 24, 2000

Abstract

In [6], the Ghost Fluid Method (GFM) was developed to capture the boundary conditions at a contact discontinuity in the inviscid compressible Euler equations. In [11], related techniques were used to develop a boundary condition capturing approach for the variable coefficient Poisson equation on domains with an embedded interface. In this paper, these new numerical techniques are extended to treat multiphase incompressible flow including the effects of viscosity, surface tension and gravity. While the most notable finite difference techniques for multiphase incompressible flow involve numerical smearing of the equations near the interface, see e.g. [19, 17, 1], this new approach treats the interface in a sharp fashion.

[*] We would like to thank Dr. David Wasson of Arete Entertainment (www.areteis.com) for developing the fast level set rendering software that was used in the visualization of the three dimensional calculations.

[†] Department of Mathematics, University of California Los Angeles, Los Angeles, California 90095

[‡] Research supported in part by NSF and DARPA grant NSF-DMS961854, for Virtual Integrated Prototyping (VIP)

[§] Computer Science Department, Stanford University, Stanford, California 94305

[¶] Research supported in part by ONR N00014-97-1-0027

[‖] Department of Mathematics, University of California Santa Barbara, Santa Barbara, California, 93106

[**] Research supported in part by NSF DMS-9805546

1 Introduction

The "immersed boundary" method [14] uses a δ-function formulation to compute solutions to the incompressible Navier-Stokes equations in the presence of a submersed elastic interface. This method allows one to incorporate the effects of the interface on a standard Cartesian mesh. For more details, see [15]. In [19], this approach was extended to treat multiphase incompressible flows in three spatial dimensions including complex topological changes. In [17] and [2], the authors replaced the front tracking formulations of [19] with level set formulations [12] that are generally easier to implement especially in the presence of three dimensional topological changes.

While the "immersed boundary" type methods of [14, 15, 19, 17, 2] are fairly attractive using finite differences on a Cartesian mesh, the inherent numerical smearing is known to have an adverse effect on the solution forcing continuity at the interface regardless of the appropriate interface boundary conditions. That is, the numerical solution is continuous at the interface even if the actual boundary conditions imply that the solution should be discontinuous. For example, surface tension forces induce a discontinuous pressure across a multiphase interface [10], while these methods smear the pressure profile into a numerically continuous function. While it is possible to formulate surface tension *models* based on continuous pressure profiles, see e.g. [1], one would hope that better results can be obtained if the jump conditions remain intact. However, this increases the possibility of introducing disturbances on the length scale of the mesh as discussed in [19], especially for calculations that are not well resolved.

In [6], the Ghost Fluid Method (GFM) was developed to capture the boundary conditions at a contact discontinuity in the inviscid Euler equations. In this paper, we extend those ideas to treat three dimensional multiphase incompressible flow including the effects of viscosity, surface tension and gravity eliminating the numerical smearing prevalent in the δ-function formulation of the "immersed boundary" method. Since a projection method [5] is used to solve for the pressure, a Poisson equation with both variable coefficients and a discontinuous solution needs to be solved at each time step. This is accomplished with the GFM related technique developed in [11] which yields a symmetric coefficient matrix for the associated linear system allowing for straightforward application of many "black box" solvers.

2 Equations

2.1 Navier-Stokes Equations

The basic equations for viscous incompressible flow are,

$$\rho_t + \vec{V} \cdot \nabla \rho = 0 \tag{1}$$

$$u_t + \vec{V} \cdot \nabla u + \frac{p_x}{\rho} = \frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho} \tag{2}$$

$$v_t + \vec{V} \cdot \nabla v + \frac{p_y}{\rho} = \frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y + (\mu(v_z + w_y))_z}{\rho} + g \tag{3}$$

$$w_t + \vec{V} \cdot \nabla w + \frac{p_z}{\rho} = \frac{(\mu(u_z + w_x))_x + (\mu(v_z + w_y))_y + (2\mu w_z)_z}{\rho} \tag{4}$$

where t is the time, (x, y, z) are the spatial coordinates, ρ is the density, $\vec{V} =< u, v, w >$ is the velocity field, p is the pressure, μ is the viscosity, g is gravity, and $\nabla = \left\langle \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right\rangle$. These equations are trivially derived from the Lagrangian form of the viscous compressible Navier-Stokes equations using the divergence free condition, $\nabla \cdot \vec{V} = 0$. The equations for the velocities can be written in condensed notation as a row vector

$$\vec{V} + \left(\vec{V} \cdot \nabla\right) \vec{V} + \frac{\nabla p}{\rho} = \frac{(\nabla \cdot \tau)^T}{\rho} + \vec{g} \tag{5}$$

where "T" represents the transpose operator, $\vec{g} =< 0, g, 0 >$, and τ is the viscous stress tensor for incompressible flow,

$$\tau = \mu \begin{pmatrix} 2u_x & u_y + v_x & u_z + w_x \\ u_y + v_x & 2v_y & v_z + w_y \\ u_z + w_x & v_z + w_y & 2w_z \end{pmatrix} = \mu \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} + \mu \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix}^T \tag{6}$$

2.2 Jump Conditions

Defining the unit normal vector

$$\vec{N} =< n_1, n_2, n_3 > \tag{7}$$

and applying conservation allows one to write the jump conditions for an interface moving with the local fluid velocity in the normal direction as

$$\left[\begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} (pI - \tau)\vec{N}^T\right] = \begin{pmatrix} \sigma\kappa \\ 0 \\ 0 \end{pmatrix} \tag{8}$$

where $\vec{T_1}$ and $\vec{T_2}$ are orthogonal unit tangent vectors, I is the identity matrix, σ is the coefficient of surface tension (a constant), κ is the local curvature of the interface and

$$[A] = A_{right} - A_{left} \tag{9}$$

defines "[$\cdot$]" as the jump across the interface. Equation 8 states that the net stress on the interface must be zero (since it has no mass). For more details, see [10, 7].

Using the definition of τ in equation 8 leads to

$$\left[\begin{pmatrix} p \\ 0 \\ 0 \end{pmatrix} - \mu \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} \nabla u \cdot \vec{N} \\ \nabla v \cdot \vec{N} \\ \nabla w \cdot \vec{N} \end{pmatrix} \right.$$
$$\left. -\mu \begin{pmatrix} \nabla u \cdot \vec{N} & \nabla v \cdot \vec{N} & \nabla w \cdot \vec{N} \\ \nabla u \cdot \vec{T_1} & \nabla v \cdot \vec{T_1} & \nabla w \cdot \vec{T_1} \\ \nabla u \cdot \vec{T_2} & \nabla v \cdot \vec{T_2} & \nabla w \cdot \vec{T_2} \end{pmatrix} \cdot \vec{N}\right] = \begin{pmatrix} \sigma\kappa \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

which can be written as three separate jump conditions

$$\left[p - 2\mu\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{N}\right] = \sigma\kappa \tag{11}$$

$$\Big[\mu\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_1} +$$
$$\mu\left(\nabla u \cdot \vec{T_1}, \nabla v \cdot \vec{T_1}, \nabla w \cdot \vec{T_1}\right) \cdot \vec{N}\Big] = 0 \tag{12}$$

$$\Big[\mu\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_2} +$$
$$\mu\left(\nabla u \cdot \vec{T_2}, \nabla v \cdot \vec{T_2}, \nabla w \cdot \vec{T_2}\right) \cdot \vec{N}\Big] = 0 \tag{13}$$

Since the flow is viscous, the velocities are continuous

$$[u] = [v] = [w] = 0 \tag{14}$$

as well as their tangential derivatives

$$[\nabla u \cdot \vec{T_1}] = [\nabla v \cdot \vec{T_1}] = [\nabla w \cdot \vec{T_1}] = 0 \tag{15}$$

$$[\nabla u \cdot \vec{T_2}] = [\nabla v \cdot \vec{T_2}] = [\nabla w \cdot \vec{T_2}] = 0 \tag{16}$$

so that the identity

$$\begin{multline}\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{N} + \left(\nabla u \cdot \vec{T_1}, \nabla v \cdot \vec{T_1}, \nabla w \cdot \vec{T_1}\right) \cdot \vec{T_1} + \\ \left(\nabla u \cdot \vec{T_2}, \nabla v \cdot \vec{T_2}, \nabla w \cdot \vec{T_2}\right) \cdot \vec{T_2} = \nabla \cdot \vec{V} = 0 \tag{17}\end{multline}$$

can be used to obtain

$$\left[\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{N}\right] = 0 \tag{18}$$

emphasizing that the normal derivative of the normal component of the velocity is continuous across the interface allowing equation 11 to be rewritten as

$$[p] - 2\,[\mu]\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{N} = \sigma\kappa \tag{19}$$

Next, the family of identities of the form

$$[AB] = \hat{B}[A] + \hat{A}[B] \tag{20}$$

$$\hat{A} = aA_{right} + bA_{left}, \quad \hat{B} = bB_{right} + aB_{left}, \quad a + b = 1 \tag{21}$$

is used along with equations 15 and 16 to rewrite equations 12 and 13 as

$$\left[\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_1}\right] = \frac{-[\mu]}{\hat{\mu}}\hat{\alpha} \tag{22}$$

and

$$\left[\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_2}\right] = \frac{-[\mu]}{\hat{\mu}}\hat{\beta} \tag{23}$$

where

$$\alpha = \left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_1} + \left(\nabla u \cdot \vec{T_1}, \nabla v \cdot \vec{T_1}, \nabla w \cdot \vec{T_1}\right) \cdot \vec{N} \tag{24}$$

and

$$\beta = \left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{T_2} + \left(\nabla u \cdot \vec{T_2}, \nabla v \cdot \vec{T_2}, \nabla w \cdot \vec{T_2}\right) \cdot \vec{N} \quad (25)$$

with the "*hat*" superscript defined as outlined above.

Finally, equations 15, 16, 18, 22, and 23 can be compiled to obtain

$$\begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} [\nabla u] \\ [\nabla v] \\ [\nabla w] \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T = \frac{-[\mu]}{\hat{\mu}} \begin{pmatrix} 0 & 0 & 0 \\ \hat{\alpha} & 0 & 0 \\ \hat{\beta} & 0 & 0 \end{pmatrix} \quad (26)$$

or more simply

$$\begin{pmatrix} [u_x] & [u_y] & [u_z] \\ [v_x] & [v_y] & [v_z] \\ [w_x] & [w_y] & [w_z] \end{pmatrix} = \frac{-[\mu]}{\hat{\mu}} \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T \begin{pmatrix} 0 & 0 & 0 \\ \hat{\alpha} & 0 & 0 \\ \hat{\beta} & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \quad (27)$$

Alternatively, equations 15, 16, and 18, can be compiled to obtain

$$\begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} [\mu\nabla u] \\ [\mu\nabla v] \\ [\mu\nabla w] \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T = [\mu] \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T +$$

$$[\mu] \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix}^T + \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} [\mu\nabla u] \\ [\mu\nabla v] \\ [\mu\nabla w] \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix}^T \quad (28)$$

or

$$\begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} [\mu\nabla u] \\ [\mu\nabla v] \\ [\mu\nabla w] \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T = [\mu] \begin{pmatrix} \vec{N} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T +$$

$$[\mu] \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix}^T - [\mu] \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix}^T \begin{pmatrix} \vec{N} \\ \vec{0} \\ \vec{0} \end{pmatrix}^T \quad (29)$$

using equations 12 and 13 as well. This can be rewritten as

$$\begin{pmatrix} [\mu u_x] & [\mu u_y] & [\mu u_z] \\ [\mu v_x] & [\mu v_y] & [\mu v_z] \\ [\mu w_x] & [\mu w_y] & [\mu w_z] \end{pmatrix} = [\mu] \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix}^T \begin{pmatrix} \vec{0} \\ \vec{T_1} \\ \vec{T_2} \end{pmatrix} +$$

$$[\mu]\,\vec{N}^T\vec{N}\begin{pmatrix}\nabla u\\ \nabla v\\ \nabla w\end{pmatrix}\vec{N}^T\vec{N}-[\mu]\begin{pmatrix}\vec{0}\\ \vec{T}_1\\ \vec{T}_2\end{pmatrix}^T\begin{pmatrix}\vec{0}\\ \vec{T}_1\\ \vec{T}_2\end{pmatrix}\begin{pmatrix}\nabla u\\ \nabla v\\ \nabla w\end{pmatrix}^T\vec{N}^T\vec{N} \tag{30}$$

noting that the right hand side of this equation only involves derivatives that are continuous across the interface as opposed to equation 27.

In viscous flows, the velocity is continuous across the interface implying that the material derivative or Lagrangian acceleration is continuous as well. That is,

$$\left[\frac{Du}{Dt}\right]=\left[\frac{Dv}{Dt}\right]=\left[\frac{Dw}{Dt}\right]=0 \tag{31}$$

are valid jump conditions allowing one to write

$$\left[\frac{p_x}{\rho}\right]=\left[\frac{(2\mu u_x)_x+(\mu(u_y+v_x))_y+(\mu(u_z+w_x))_z}{\rho}\right] \tag{32}$$

$$\left[\frac{p_y}{\rho}\right]=\left[\frac{(\mu(u_y+v_x))_x+(2\mu v_y)_y+(\mu(v_z+w_y))_z}{\rho}\right] \tag{33}$$

$$\left[\frac{p_z}{\rho}\right]=\left[\frac{(\mu(u_z+w_x))_x+(\mu(v_z+w_y))_y+(2\mu w_z)_z}{\rho}\right] \tag{34}$$

based on equations 2, 3, and 4.

2.3 Level Set Equation

The level set equation

$$\phi_t+\vec{V}\cdot\nabla\phi=0 \tag{35}$$

is used to keep track of the interface location as the set of points where $\phi=0$. To keep the values of ϕ close to those of a signed distance function, i.e. $|\nabla\phi|=1$, the reinitialization equation

$$\phi_t+S(\phi_o)\left(|\nabla\phi|-1\right)=0 \tag{36}$$

is iterated for a few steps in ficticious time. The level set function is used to compute the normal

$$\vec{N}=\frac{\nabla\phi}{|\nabla\phi|} \tag{37}$$

and the curvature

$$\kappa = -\nabla \cdot \vec{N} \tag{38}$$

in a standard fashion. For more details on the level set function see [6, 12, 17].

3 Numerical Method

A standard MAC grid is used for discretization where $p_{i,j,k}$, $\rho_{i,j,k}$, $\mu_{i,j,k}$, and $\phi_{i,j,k}$ exist at the cell centers (grid points) and $u_{i\pm\frac{1}{2},j,k}$, $v_{i,j\pm\frac{1}{2},k}$, and $w_{i,j,k\pm\frac{1}{2}}$ exist at the appropriate cell walls. See [9] and [13] for more details.

3.1 Level Set Equation

The level set function is evolved in time from ϕ^n to ϕ^{n+1} using nodal velocities defined by $u_{i,j,k} = \frac{u_{i-\frac{1}{2},j,k}+u_{i+\frac{1}{2},j,k}}{2}$, $v_{i,j,k} = \frac{v_{i,j-\frac{1}{2},k}+v_{i,j+\frac{1}{2},k}}{2}$, and $w_{i,j,k} = \frac{w_{i,j,k-\frac{1}{2}}+w_{i,j,k+\frac{1}{2}}}{2}$. Detailed discretizations for equations 35 and 36 are given in [6]. Note that the 5th order WENO discretization from [6] is used to discretize the spatial terms in equations 35 and 36 for the numerical examples in this paper.

The normal vector can be computed as

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|} \tag{39}$$

using standard central differencing everywhere the denominator is non-zero. In the rare case that the denominator is identically zero, one sided differences are used to calculate ϕ_x, ϕ_y, and ϕ_z instead of central differencing allowing at least one nonzero value to be calculated as long as ϕ has been properly reinitialized to a signed distance function. Once the normal is computed as $\vec{N} =< n_1, n_2, n_3 >$, tangent vectors can be found in the following fashion. First find the $min\,\{|n_1|, |n_2|, |n_3|\}$. For the sake of exposition, suppose that n_1 has the smallest magnitude (the other cases are treated similarly). Then choose the unit vector in the direction of the component with the smallest magnitude, which in this case is $< 1, 0, 0 >$, and define

$$\vec{T_1} = \frac{\vec{N}\times\langle 1,0,0\rangle}{|\vec{N}\times\langle 1,0,0\rangle\,|} = \left\langle 0, \frac{n_3}{\sqrt{n_2^2+n_3^2}}, \frac{-n_2}{\sqrt{n_2^2+n_3^2}} \right\rangle \tag{40}$$

as one tangent vector. The other tangent vector is defined as $\vec{T_2} = \vec{N}\times\vec{T_1}$. Note that the unit tangent vectors T_1 and T_2 are not necessarily continuously defined and care should be exercised when using these vectors.

The curvature at each grid point is defined as

$$\begin{aligned} \kappa \;=\; & -\Big(\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - 2\phi_x\phi_z\phi_{xz} + \phi_z^2\phi_{xx} \\ & +\phi_y^2\phi_{zz} - 2\phi_y\phi_z\phi_{yz} + \phi_z^2\phi_{yy}\Big) \,/\, \Big(\phi_x^2+\phi_y^2+\phi_z^2\Big)^{1.5} \end{aligned} \tag{41}$$

and discretized using standard central differences. In the rare case that the denominator in equation 41 is identically zero, one sided differences are used to calculate ϕ_x, ϕ_y, and ϕ_z instead of central differencing allowing at least one nonzero value to be calculated as long as ϕ has been properly reinitialized to a signed distance function. Note that the curvature is limited with

$$|\kappa| \leq \frac{1}{\min\{\triangle x, \triangle y, \triangle z\}} \tag{42}$$

so that under-resolved regions do not erroneously contribute large surface tension forces.

3.2 Projection Method

First, $\vec{V}^\star = \langle u^\star, v^\star, w^\star \rangle$ is defined by

$$\frac{\vec{V}^\star - \vec{V}^n}{\triangle t} + \left(\vec{V} \cdot \nabla\right)\vec{V} = \frac{(\nabla \cdot \tau)^T}{\rho} + \vec{g} \tag{43}$$

and then the velocity field at the new time step, $\vec{V}^{n+1} = \langle u^{n+1}, v^{n+1}, w^{n+1} \rangle$, is defined by

$$\frac{\vec{V}^{n+1} - \vec{V}^\star}{\triangle t} + \frac{\nabla p}{\rho} = 0 \tag{44}$$

so that combining equations 43 and 44 to eliminate $\vec{V}^\star$ results in equation 5. Taking the divergence of equation 44 results in

$$\nabla \cdot \left(\frac{\nabla p}{\rho}\right) = \frac{\nabla \cdot \vec{V}^\star}{\triangle t} \tag{45}$$

after setting $\nabla \cdot \vec{V}^{n+1}$ to zero. Note that equation 45 defines the pressure in terms of the value of $\triangle t$ used in equation 43. Defining a scaled pressure by $p^\star = p\triangle t$ leads to

$$\vec{V}^{n+1} - \vec{V}^\star + \frac{\nabla p^\star}{\rho} = 0 \tag{46}$$

and

$$\nabla \cdot \left(\frac{\nabla p^\star}{\rho}\right) = \nabla \cdot \vec{V}^\star \tag{47}$$

in place of equations 44 and 45 where $p^\star$ no longer depends on $\triangle t$. In the case of a spatially constant density one can proceed even further defining $\hat{p} = \frac{p\triangle t}{\rho}$ leading to

$$\vec{V}^{n+1} - \vec{V}^\star + \nabla \hat{p} = 0 \tag{48}$$

and

$$\triangle \hat{p} = \nabla \cdot \vec{V}^\star \tag{49}$$

where $\hat{p}$ no longer depends on $\triangle t$ or ρ.

Boundary conditions are applied to the velocity and are not needed for the pressure. In order to apply boundary conditions to $\vec{V}^{n+1}$, one simply applies them to $\vec{V}^\star$ after computing $\vec{V}^\star$ in equation 43 and before solving equation 45. Then in equation 45, one sets $\nabla p \cdot \vec{N} = 0$ on the boundary where $\vec{N}$ is the unit normal to the boundary. Since the flow is incompressible, the compatibility condition

$$\int_\Gamma \vec{V}^\star \cdot \vec{N} = 0 \tag{50}$$

needs to be satisfied when specifying the boundary condition on $\vec{V}^\star$ in order to guarantee the existence of a solution. Here, Γ represents the boundary of the compuational domain and $\vec{N}$ is the unit normal to that boundary. [13]

3.3 Runge Kutta

The projection method is a special splitting method that allows one to advance a solution forward one time step, $\triangle t$, with Euler's method. To simplify notation, let E define an Euler update so that

$$\vec{V}^{n+1} = E\left(\vec{V}^n\right) \tag{51}$$

can be used to describe a temporal update using Euler's method. Moreover,

$$\vec{V}^{n+1} = \frac{1}{2}\vec{V}^n + \frac{1}{2}E\left(E\left(\vec{V}^n\right)\right) \tag{52}$$

is 2nd order TVD Runge Kutta (also known as 2nd order Runge Kutta, the modified Euler method, the midpoint rule, or Heun's predictor corrector method) [16]. In a similar fashion, one can write

$$\vec{V}^{n+1} = \frac{1}{3}\vec{V}^n + \frac{2}{3}E\left(\frac{3}{4}\vec{V}^n + \frac{1}{4}E\left(E\left(\vec{V}^n\right)\right)\right) \tag{53}$$

for 3rd order TVD Runge Kutta [16].

It is obvious that 2nd and 3rd order TVD Runge Kutta can be written as a convex combination of Euler updates. This is exactly what makes them TVD as described in [16]. Another interesting fact is that these TVD Runge Kutta methods can be numerically implemented with a minimal amount of memory. More specifically, only two copies of the independent variables are needed for 2nd order TVD Runge Kutta

$$\begin{aligned} \vec{V} &\leftarrow \vec{V}^n \\ \vec{V} &\leftarrow E\left(\vec{V}\right) \\ \vec{V} &\leftarrow E\left(\vec{V}\right) \\ \vec{V} &\leftarrow \frac{1}{2}\vec{V}^n + \frac{1}{2}\vec{V} \end{aligned}$$

and only two copies of the independent variables are needed for 3nd order TVD Runge Kutta

$$\begin{aligned} \vec{V} &\leftarrow \vec{V}^n \\ \vec{V} &\leftarrow E\left(\vec{V}\right) \\ \vec{V} &\leftarrow E\left(\vec{V}\right) \\ \vec{V} &\leftarrow \frac{3}{4}\vec{V}^n + \frac{1}{4}\vec{V} \\ \vec{V} &\leftarrow E\left(\vec{V}\right) \\ \vec{V} &\leftarrow \frac{1}{3}\vec{V}^n + \frac{2}{3}\vec{V} \end{aligned}$$

which is not true for non-TVD Runge Kutta methods. Note that 3rd order TVD Runge Kutta is used in the examples section.

3.4 Convection Terms

The MAC grid stores u values at $\vec{x}_{i\pm\frac{1}{2},j,k}$. Updating $u^\star_{i\pm\frac{1}{2},j,k}$ in equation 43 requires the discretization of $\vec{V}\cdot\nabla u$ at $\vec{x}_{i\pm\frac{1}{2},j,k}$. First, simple averaging can be used to define $\vec{V}$ at $\vec{x}_{i\pm\frac{1}{2},j,k}$. For example

$$v_{i+\frac{1}{2},j,k} = \frac{v_{i,j-\frac{1}{2},k} + v_{i,j+\frac{1}{2},k} + v_{i+1,j-\frac{1}{2},k} + v_{i+1,j+\frac{1}{2},k}}{4} \tag{54}$$

and

$$w_{i+\frac{1}{2},j,k} = \frac{w_{i,j,k-\frac{1}{2}} + w_{i,j,k+\frac{1}{2}} + w_{i+1,j,k-\frac{1}{2}} + w_{i+1,j,k+\frac{1}{2}}}{4} \tag{55}$$

define v and w at $\vec{x}_{i+\frac{1}{2},j,k}$ while u is already defined there. Then the $\vec{V} \cdot \nabla u$ term on the offset $\vec{x}_{i\pm\frac{1}{2},j,k}$ grid can be discretized in the same fashion as the $\vec{V} \cdot \nabla \phi$ term on the regular $\vec{x}_{i,j,k}$ grid using the method outlined in [6] for equation 35. Note that the 3rd order ENO discretization from [6] is used in the examples section. Along the same lines, updating $v^\star_{i,j\pm\frac{1}{2},k}$ and $w^\star_{i,j,k\pm\frac{1}{2}}$ in equation 43 requires the discretization of $\vec{V} \cdot \nabla v$ at $\vec{x}_{i,j\pm\frac{1}{2},k}$ and $\vec{V} \cdot \nabla w$ at $\vec{x}_{i,j,k\pm\frac{1}{2}}$ respectively. Once again, with the aid of simple averaging to define $\vec{V}$, these terms can be discretized on offset grids in a fashion similar to the $\vec{V} \cdot \nabla \phi$ term on the regular grid.

Ghost cells are used to aid in the discretization near the boundaries. For example, if the computational boundary is a solid wall, a reflection condition is used to populate the ghost cells with the appropriate velocities.

3.5 Viscous Terms - δ-function formulation

Updating $u^\star_{i\pm\frac{1}{2},j,k}$ in equation 43 requires discretization of

$$\frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho} \tag{56}$$

at $\vec{x}_{i\pm\frac{1}{2},j,k}$. Likewise, updating $v^\star_{i,j\pm\frac{1}{2},k}$ and $w^\star_{i,j,k\pm\frac{1}{2}}$ requires discretization of

$$\frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y + (\mu(v_z + w_y))_z}{\rho} \tag{57}$$

at $\vec{x}_{i,j\pm\frac{1}{2},k}$ and

$$\frac{(\mu(u_z + w_x))_x + (\mu(v_z + w_y))_y + (2\mu w_z)_z}{\rho} \tag{58}$$

at $\vec{x}_{i,j,k\pm\frac{1}{2}}$ respectively. Since the velocities are continuous (see equation 14), the first derivatives are computed directly using central differencing. For example,

$$(u_x)_{i,j,k} = \frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\triangle x} \tag{59}$$

$$(u_y)_{i+\frac{1}{2},j+\frac{1}{2},k} = \frac{u_{i+\frac{1}{2},j+1,k} - u_{i+\frac{1}{2},j,k}}{\triangle y} \tag{60}$$

and

$$(u_z)_{i+\frac{1}{2},j,k+\frac{1}{2}} = \frac{u_{i+\frac{1}{2},j,k+1} - u_{i+\frac{1}{2},j,k}}{\triangle z} \tag{61}$$

are used to compute the first derivatives of u.

Suppose that μ^- and μ^+ are the viscosities for the fluid where $\phi \leq 0$ and $\phi > 0$ respectively. Then a continuous viscosity can be defined as

$$\mu(\phi) = \mu^- + \left(\mu^+ - \mu^-\right) H(\phi) \tag{62}$$

where

$$H(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2} + \frac{\phi}{2\epsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right) & -\epsilon \leq \phi \leq \epsilon \\ 1 & \epsilon < \phi \end{cases} \tag{63}$$

defines the Heaviside function based on ϕ. Since ϕ is only defined at the grid points, simple averaging is used to define ϕ elsewhere. For example $\phi_{i+\frac{1}{2},j,k} = \frac{\phi_{i,j,k}+\phi_{i+1,j,k}}{2}$ and $\phi_{i+\frac{1}{2},j+\frac{1}{2},k} = \frac{\phi_{i,j,k}+\phi_{i+1,j,k}+\phi_{i,j+1,k}+\phi_{i+1,j+1,k}}{4}$. Then viscosities can be defined where needed using equation 62.

Since the viscosity has been forced to be continuous, equation 27 implies that all the first derivatives of the velocity are continuous as well. Therefore, the second derivative terms in equations 56, 57 and 58 can be calculated directly using central differencing. For example,

$$\frac{\mu\left(\phi_{i+\frac{1}{2},j+\frac{1}{2},k}\right)(u_y + v_x)_{i+\frac{1}{2},j+\frac{1}{2},k} - \mu\left(\phi_{i+\frac{1}{2},j-\frac{1}{2},k}\right)(u_y + v_x)_{i+\frac{1}{2},j-\frac{1}{2},k}}{\triangle y} \tag{64}$$

is used to approximate $(\mu(u_y + v_x))_y$ at $\vec{x}_{i+\frac{1}{2},j,k}$.

Finally, a smeared out density profile is computed

$$\rho(\phi) = \rho^- + \left(\rho^+ - \rho^-\right) H(\phi) \tag{65}$$

using the Heaviside function.

For more details on the δ-function formulation, see [17].

3.6 Viscous Terms - without δ-functions

Instead of using equations 62 and 65 to define smeared out versions of μ and ρ, one can simply use the sign of the level set function to determine μ as μ^- or μ^+ and to determine ρ as ρ^- and ρ^+ in a sharp fashion. Then one can consider updating each fluid independently and pay particular attention to the jump conditions at the interface. In particular, both μ and ρ are spatially constant on either side of the interface allowing one to write

$$\frac{\mu\left(u_{xx}+u_{yy}+u_{zz}\right)}{\rho} \tag{66}$$

$$\frac{\mu\left(v_{xx}+v_{yy}+v_{zz}\right)}{\rho} \tag{67}$$

and

$$\frac{\mu\left(w_{xx}+w_{yy}+w_{zz}\right)}{\rho} \tag{68}$$

in place of equations 56, 57, and 58 with the help of $\nabla \cdot \vec{V} = 0$. Once again, since the velocities are continuous (see equation 14), the first derivatives can be computed directly using central differencing along the lines of equations 59, 60, and 61. However, one must be particularly careful when computing the second derivatives of the velocity, since the first derivatives are discontinuous according to equation 27. In particular, equation 30 is used to aid in the computation of the second derivatives. (Note that one *could* directly compute the second derivatives ignoring the jump conditions. This produces a numerical method with some degree of smearing.)

To compute the right hand side of equation 30, the continuous velocity field is transfered from the MAC grid to the grid points with simple averaging, i.e. $u_{i,j,k} = \frac{u_{i-\frac{1}{2},j,k}+u_{i+\frac{1}{2},j,k}}{2}$, $v_{i,j,k} = \frac{v_{i,j-\frac{1}{2},k}+v_{i,j+\frac{1}{2},k}}{2}$, and $w_{i,j,k} = \frac{w_{i,j,k-\frac{1}{2}}+w_{i,j,k+\frac{1}{2}}}{2}$. Next, central differencing is used to compute the first derivatives at each grid point, e.g. $(u_x)_{i,j,k} = \frac{u_{i+1,j,k}-u_{i-1,j,k}}{2\triangle x}$. Then these first derivatives can be multiplied by the appropriate components of the normal and tangent vectors at each grid node to complete the calculation of the right hand side of equation 30 which can be denoted by J rewriting equation 30 as

$$\begin{pmatrix} [\mu u_x] & [\mu u_y] & [\mu u_z] \\ [\mu v_x] & [\mu v_y] & [\mu v_z] \\ [\mu w_x] & [\mu w_y] & [\mu w_z] \end{pmatrix} = \begin{pmatrix} J^{11} & J^{12} & J^{13} \\ J^{21} & J^{22} & J^{23} \\ J^{31} & J^{32} & J^{33} \end{pmatrix} \tag{69}$$

at each grid point.

Note that the unit tangent vectors are used to choose a coordinate system where discontinuous derivatives are replaced with continuous derivatives using equations 12 and 13. In fact, this is how equation 29 follows from equation 28. Once the discontinuous derivatives have been replaced, the results are transformed back to the Cartesian coordinate system. Since this entire computation is done at a grid node, one need not worry about averaging values of J since they live in Cartesian space and are not dependent on the particular choice of T_1 and T_2 at each node. Since J is a spatially smooth function, spatial averages of J are well defined and can be used to define J elsewhere. For example, $J_{i+\frac{1}{2},j,k} = \frac{J_{i,j,k}+J_{i+1,j,k}}{2}$ and $J_{i+\frac{1}{2},j+\frac{1}{2},k} = \frac{J_{i,j,k}+J_{i+1,j,k}+J_{i,j+1,k}+J_{i+1,j+1,k}}{4}$.

Once J has been computed, the second derivatives can be computed using the techniques developed in [11] for computing second derivatives of the variable coefficient Poisson equation. As a specific example, consider the discretization of μu_{xx} at $x_{i+\frac{1}{2},j,k}$ using $u_M = u_{i+\frac{1}{2},j,k}$ and its neighbors $u_L = u_{i-\frac{1}{2},j,k}$ and $u_R = u_{i+\frac{3}{2},j,k}$. In addition, we will need the averaged values of ϕ and J^{11} at $\phi_L = \phi_{i-\frac{1}{2},j,k}$, $\phi_M = \phi_{i+\frac{1}{2},j,k}$, $\phi_R = \phi_{i+\frac{3}{2},j,k}$, $J_L = J^{11}_{i-\frac{1}{2},j,k}$, $J_M = J^{11}_{i+\frac{1}{2},j,k}$, and $J_R = J^{11}_{i+\frac{3}{2},j,k}$ so that they are defined in the same spatial locations as the corresponding u terms. If ϕ_L, ϕ_M, and ϕ_R are all greater than zero, we define

$$(\mu u_x)_L = \mu^+ \left(\frac{u_M - u_L}{\triangle x} \right) \tag{70}$$

and

$$(\mu u_x)_R = \mu^+ \left(\frac{u_R - u_M}{\triangle x} \right) \tag{71}$$

arriving at

$$(\mu u_{xx})_{i+\frac{1}{2},j,k} = \frac{(\mu u_x)_R - (\mu u_x)_L}{\triangle x} \tag{72}$$

in the standard fashion. A similar discretization holds when ϕ_L, ϕ_M, and ϕ_R are all less than or equal to zero.

Next, suppose that $\phi_L \leq 0$ and $\phi_M > 0$ so that the interface lies in between the associated grid points. Then

$$\theta = \frac{|\phi_L|}{|\phi_L| + |\phi_M|} \tag{73}$$

can be used to estimate the interface location. That is, the interface splits this cell into two pieces of size $\theta\triangle x$ on the left and size $(1-\theta)\triangle x$ on the right. At the interface, we denote the continuous velocity by u_I and calculate the jump as

$$J_I = \theta J_M + (1-\theta) J_L \tag{74}$$

noting that it is continuous across the interface as well. As discussed in [11], we discretize the jump condition $[\mu u_x] = J_I$ as

$$\mu^+ \left(\frac{u_M - u_I}{(1-\theta)\triangle x} \right) - \mu^- \left(\frac{u_I - u_L}{\theta \triangle x} \right) = J_I \tag{75}$$

and then solve for u_I to get

$$u_I = \frac{\mu^+ u_M \theta + \mu^- u_L (1-\theta) - J_I \theta (1-\theta) \triangle x}{\mu^+ \theta + \mu^- (1-\theta)} \tag{76}$$

so that we can write

$$(\mu u_x)_L = \mu^+ \left(\frac{u_M - u_I}{(1-\theta)\triangle x} \right) = \hat{\mu} \left(\frac{u_M - u_L}{\triangle x} \right) + \frac{\hat{\mu} J_I \theta}{\mu^-} \tag{77}$$

where

$$\hat{\mu} = \frac{\mu^+ \mu^-}{\mu^+ \theta + \mu^- (1-\theta)} \tag{78}$$

defines an effective μ. Similarly, if $\phi_L > 0$ and $\phi_M \leq 0$,

$$(\mu u_x)_L = \mu^- \left(\frac{u_M - u_I}{(1-\theta)\triangle x} \right) = \hat{\mu} \left(\frac{u_M - u_L}{\triangle x} \right) - \frac{\hat{\mu} J_I \theta}{\mu^+} \tag{79}$$

where

$$\hat{\mu} = \frac{\mu^- \mu^+}{\mu^- \theta + \mu^+ (1-\theta)} \tag{80}$$

defines an effective μ.

In similar fashion, if $\phi_R > 0$ and $\phi_M \leq 0$, then

$$\theta = \frac{|\phi_R|}{|\phi_R| + |\phi_M|} \tag{81}$$

is used to estimate the interface location with $(1-\theta)\triangle x$ on the left and $\theta\triangle x$ on the right. Then

$$J_I = \theta J_M + (1-\theta) J_R \tag{82}$$

is used to discretize the jump condition as

$$\mu^+ \left(\frac{u_R - u_I}{\theta \triangle x} \right) - \mu^- \left(\frac{u_I - u_M}{(1-\theta)\triangle x} \right) = J_I \tag{83}$$

resulting in

$$u_I = \frac{\mu^- u_M \theta + \mu^+ u_R (1-\theta) - J_I \theta (1-\theta) \triangle x}{\mu^- \theta + \mu^+ (1-\theta)} \tag{84}$$

and

$$(\mu u_x)_R = \mu^- \left(\frac{u_I - u_M}{(1-\theta)\triangle x} \right) = \hat{\mu} \left(\frac{u_R - u_M}{\triangle x} \right) - \frac{\hat{\mu} J_I \theta}{\mu^+} \tag{85}$$

where

$$\hat{\mu} = \frac{\mu^- \mu^+}{\mu^- \theta + \mu^+ (1-\theta)} \tag{86}$$

defines an effective μ. If $\phi_R \le 0$ and $\phi_M > 0$,

$$(\mu u_x)_R = \mu^+ \left(\frac{u_I - u_M}{(1-\theta)\triangle x} \right) = \hat{\mu} \left(\frac{u_R - u_M}{\triangle x} \right) + \frac{\hat{\mu} J_I \theta}{\mu^-} \tag{87}$$

where

$$\hat{\mu} = \frac{\mu^+ \mu^-}{\mu^+ \theta + \mu^- (1-\theta)} \tag{88}$$

defines an effective μ.

For more on the details and motivation behind this method, see [11].

3.7 Poisson Equation

Once $\vec{V}^\star$ has been updated with equation 43, the right hand side of equation 47 is discretized using standard central differencing, see e.g. equation 59. Then the techniques presented in [11] for the variable coefficient Poisson equation are used to solve equation 47 for the pressure at the grid nodes.

Finally, the resulting pressure is used to find $\vec{V}^{n+1}$ in equation 46. One should take care to compute the derivatives of the pressure in equation 46 in *exactly* the same way as they were computed in equation 47 using the techniques in [11].

The techniques in [11] require a level set function to describe the interface location. We use ϕ^{n+1} as opposed to ϕ^n, since we wish to find the pressure that will make $\vec{V}^{n+1}$ divergence free in equation 46. This implies that both equation 46 and equation 47 should use $\rho^{n+1} = \rho(\phi^{n+1})$. In contrast, some conventional discretizations of equation 43 use $\mu^n = \mu(\phi^n)$ and $\rho^n = \rho(\phi^n)$ to discretize the viscous terms. At this point it is instructive to consider the jump conditions in equations 32, 33, and 34 that keep the interface from "tearing apart" due to a jump in acceleration. These equations illustrate that the density used for the Poisson equation and the density used for the viscous terms should be identical. Therefore, we use ρ^{n+1} when discretizing the viscous terms as opposed to ρ^n.

Note that one can set $\left[\frac{p_x}{\rho}\right] = \left[\frac{p_y}{\rho}\right] = \left[\frac{p_z}{\rho}\right] = 0$ when solving the Poisson equation using the method in [11] in spite of the non-zero jumps in equations 32, 33, and 34. Since the full equations 2, 3 and 4 are continuous across the interface, one can take the divergence of the full equations without considering jump conditions, and this is exactly how equation 45 is derived. That is, jump conditions only need to be considered when discretizing individual parts of the full equations, and can be ignored when taking the divergence of the full equations themselves, since the full equations are continuous across the interface. Moreover, the jumps in the derivatives of the pressure in equation 45 are already balanced on the right hand side by the appropriate jumps in the viscous terms which have been included in $V^\star$. This gives further justification to the use of ρ^{n+1} when discretizing the viscous terms.

On the other hand, the jump in pressure defined in equation 19 needs to be accounted for when solving the Poisson equation with the method in [11]. Equation 19 is rewritten as

$$[p^\star] - 2\triangle t\,[\mu]\left(\nabla u \cdot \vec{N}, \nabla v \cdot \vec{N}, \nabla w \cdot \vec{N}\right) \cdot \vec{N} = \triangle t \sigma \kappa \tag{89}$$

for use equation 47. The $[p^\star]$ is computed at each grid node. The derivatives of the velocities are computed with standard central differencing of the averaged nodal velocities analogous to the way that J was computed when discretizing the viscous terms. The normals are computed using ϕ^n to be consistent with the velocities and the computation of the viscous terms. In general, the computation of this viscosity related term is not that sensitive since it is continuous across the interface. The curvature is discretized using ϕ^{n+1}.

In the case of a continuous or smeared out viscosity, the jump in pressure reduces to $[p^\star] = \triangle t \sigma \kappa$. This can be further reduced to $[p^\star] = 0$ by using a continuous surface force (CSF) model for the surface tension [1]. In [17], the CSF model is implemented by adding a term of the form

$$\frac{\delta \sigma \kappa \vec{N}}{\rho} \tag{90}$$

to the right hand side of the momentum equations. Here ρ is calculated along the lines of equation 65, and the smeared out delta function

$$\delta(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon}\cos\left(\frac{\pi\phi}{\epsilon}\right) & -\epsilon \le \phi \le \epsilon \\ 0 & \epsilon < \phi \end{cases} \tag{91}$$

is calculated by taking the derivative of the smeared out Heaviside function in equation 63. Note that ϕ^{n+1} is used when calculating all the relevant terms in equation 90.

After discretizing the Poisson equation for the pressure, the resulting system of linear equations is solved with a preconditioned conjugate gradient (PCG) method using an Incomplete Choleski preconditioner [8]. The PCG algorithm is applied once for every Euler time step, or a total of three times for a third order Runge Kutta cycle.

3.8 Time Step Restriction

Adaptive time stepping is used in the examples section choosing the overall time step based on convection, viscosity, surface tension and gravity. The convective time step restriction is given by

$$\triangle t \left(\frac{|u|_{max}}{\triangle x} + \frac{|v|_{max}}{\triangle y} + \frac{|w|_{max}}{\triangle z} \right) \le 1 \tag{92}$$

where $|u|_{max}$, $|v|_{max}$, and $|w|_{max}$ are the maximum magnitudes of the velocities. The viscous time step restriction is given by

$$\triangle t \left(\max\left\{ \frac{\mu^-}{\rho^-}, \frac{\mu^+}{\rho^+} \right\} \left(\frac{2}{(\triangle x)^2} + \frac{2}{(\triangle y)^2} + \frac{2}{(\triangle z)^2} \right) \right) \le 1 \tag{93}$$

where the “max” function is defined in the obvious way.

Gravity can be included in the convection estimate noting that $|v|_{max} + |g|\triangle t$ is a linear approximation to a bound on the vertical component of the velocity at the end of a time step due to the effects of gravity. Then $\triangle t\left(\frac{|v|_{max}+|g|\triangle t}{\triangle y}\right) \leq 1$ leads to $\triangle t \leq \left(\frac{-|v|_{max}+\sqrt{|v|^2_{max}+4|g|\triangle y}}{2|g|}\right)$ or

$$\frac{\triangle t}{2}\left(\frac{|v|_{max}}{\triangle y} + \sqrt{\left(\frac{|v|_{max}}{\triangle y}\right)^2 + \frac{4|g|}{\triangle y}}\right) \leq 1 \tag{94}$$

as a time step restriction for the velocity in the vertical direction. Rewriting equation 92 as $\triangle t C_{cfl} \leq 1$ and equation 93 as $\triangle t V_{cfl} \leq 1$ allows one to write

$$\frac{\triangle t}{2}\left((C_{cfl}+V_{cfl}) + \sqrt{(C_{cfl}+V_{cfl})^2 + \frac{4|F_x|}{\triangle x} + \frac{4|F_y|}{\triangle y} + \frac{4|F_z|}{\triangle z}}\right) \leq 1 \tag{95}$$

where $\vec{F} = (F_x, F_y, F_z)$ is the net acceleration due to forces such as gravity and surface tension. Note that equation 95 was derived along the lines of the gravity estimates above.

The acceleration due to curvature can be written as $\frac{\delta\sigma\kappa}{\rho}$ where the δ-function has been included since the force only appears on the interface. In the δ-function formulation this term is added to the right hand side of the equations for velocity. In the GFM formulation, curvature enters the equations through $[p] = \sigma\kappa$ which contributes to the $\frac{\nabla p}{\rho}$ term in the equations for velocity. Noting that numerical δ-functions take the form $\frac{1}{\triangle x}$ leads one to $\frac{\sigma\kappa}{\rho\triangle x}$ for *both* formulations. Equation 95 can then be written as

$$\triangle t\left(\frac{(C_{cfl}+V_{cfl}) + \sqrt{(C_{cfl}+V_{cfl})^2 + 4\left(G_{cfl}\right)^2 + 4\left(S_{cfl}\right)^2}}{2}\right) \leq 1 \tag{96}$$

where

$$G_{cfl} = \sqrt{\frac{|g|}{\triangle y}} \tag{97}$$

and

$$S_{cfl} = \sqrt{\frac{\sigma|\kappa|}{\min\left\{\rho^+, \rho^-\right\}\left(\min\left\{\triangle x, \triangle y, \triangle z\right\}\right)^2}} \tag{98}$$

represent the restrictions due to gravity and surface tension respectively. If κ is replaced by $\frac{1}{\triangle x}$ in equation 98, one can see a resemblance between this equation and the time step restriction given in [1].

In the numerical simulations, a CFL restriction of $\frac{1}{2}$ is used. That is,

$$\triangle t \left(\frac{(C_{cfl} + V_{cfl}) + \sqrt{(C_{cfl} + V_{cfl})^2 + 4\,(G_{cfl})^2 + 4\,(S_{cfl})^2}}{2} \right) \leq \frac{1}{2} \qquad (99)$$

is used.

4 Examples

In the numerical examples, we use the following constants unless otherwise specified: $g = -9.8\frac{m}{s^2}$, $\sigma = .0728\frac{kg}{s^2}$, $\rho_{water} = 1000\frac{kg}{m^3}$, $\mu_{water} = 1.137 \times 10^{-3}\frac{kg}{ms}$, $\rho_{air} = 1.226\frac{kg}{m^3}$ and $\mu_{air} = 1.78 \times 10^{-5}\frac{kg}{ms}$. When evaluating the smeared out Heaviside and delta functions in equations 63 and 91 respectively, we use $\epsilon = 1.5\triangle x$. In addition, all calculations were performed in a box with standard no-slip solid wall boundary conditions applied at the edges of the computational domain unless otherwise specified.

4.1 Example 1

Consider a $[-.01m, .01m] \times [-.01m, .02m]$ computational domain which is initially filled with water except for a circular air bubble of radius $\frac{1}{300}m$ centered at the origin. In order to show the effects of grid refinement, the calculation was carried out on meshes of size 40×60, 80×120, 160×240 and 320×480. Figure 1 shows the four calculations at times of $t = 0$, $t = .02$, $t = .035$, and $t = .05$ seconds for the smeared out delta function method. The numerical results are color coded black, red, green and blue from the coarsest to the finest mesh. The numerical results indicate first order accurate convergence for the interface location. Note that the top of the air bubble is in practically the same location for every mesh indicating that the bubble rise velocity is a rather easy quantity to predict (in this case) even on a fairly coarse mesh. The area loss for these four calculations was 24.72%, 8.10%, .55% and $-.0068\%$ (area gain) respectively. All area loss results were computed at the final time of $t = .05$ seconds. Figure 2 shows the corresponding results for the sharper GFM. Similar to figure 1, the results indicate first order accurate convergence for the interface location. Here, the area loss was 17.23%, 5.76%, 1.54% and .0036% respectively.

In order to illustrate the behavior of the method on larger unstable bubbles, the same calculations were carried out on a $[-1m, 1m] \times [-1m, 2m]$ computational domain with a circular air bubble of radius $\frac{1}{3}m$ centered at the origin. Here, the surface tension forces are too small to dominate the inherent Kelvin-Helmholtz and Rayleigh-Taylor instabilities due to density and velocity differences respectively. Figure 3 shows the four calculations at times of $t = 0$, $t = .2$, $t = .35$, and $t = .5$ seconds for the delta function method with area loss results of -1.42%, .30%, $-.82\%$ and $-.51\%$ respectively. Figure 4 shows the corresponding results for the GFM where the area loss was 4.07%, 1.78%, .18% and $-.94\%$ respectively. While the numerical results demonstrate a certain "consistency" under grid refinement, they do not (and can not) converge, since the underlying solution is unstable. Note that higher order accurate finite difference schemes such as ENO and WENO have nonsymmetric stencils and do not necessarily produce symmetric numerical results.

4.2 Example 2

Consider a $[-.01m, .01m] \times [-.02m, .01m]$ computational domain which is initially filled with air except for a circular water droplet of radius $\frac{1}{300}m$ centered at the origin. In order to show the effects of grid refinement, the calculation was carried out on meshes of size 40×60, 80×120, 160×240 and 320×480. Figure 5 shows the four calculations at times of $t = 0$, $t = .02$, $t = .035$, and $t = .05$ seconds for the delta function method with area loss results of -6.38%, -5.78%, -4.12% and -2.39% respectively. Figure 6 shows the corresponding results for the GFM where the area loss was 2.23%, $.64\%$, $.0089\%$ and $-.0091\%$ respectively. The numerical results are color coded black, red, green and blue from the coarsest to the finest mesh. The numerical results indicate first order accurate convergence for the interface location for both methods. Note that the calculations agree surprisingly well on the sequence of meshes until the drop approaches the bottom wall of the container indicating that drop velocity is a rather easy quantity to predict (in this case) even on a fairly coarse mesh.

In order to illustrate the behavior of the method on larger unstable droplets, the same calculations were carried out on a $[-1m, 1m] \times [-1m, 2m]$ computational domain with a circular water droplet of radius $\frac{1}{3}m$ centered at the origin. Here, the surface tension forces are too small to dominate the inherent Kelvin-Helmholtz and Rayleigh-Taylor instabilities due to density and velocity differences respectively. Figure 7 shows the four calculations at times of $t = 0$, $t = .2$, $t = .35$, and $t = .5$ seconds for the delta function method with area loss results of 5.37%, 3.93%, 2.36% and $.38\%$ respectively. Figure 8 shows the corresponding results for the GFM where the area loss was 21.06%, 3.38%, 6.24% and 6.03% respectively. While the numerical results demonstrate a certain "consistency" under grid refinement, they do not (and can not) converge, since the underlying solution is unstable. Note that higher order accurate finite difference schemes such as ENO and WENO have nonsymmetric stencils and do not necessarily produce symmetric numerical results.

4.3 Example 3

Figures 3, 4, 7 and 8 might require some further explanation for the novice reader unfamiliar with the behavior of numerical methods on unstable problems. Note that figure 4 contains a higher degree of instability than figure 3 and that figure 8 contains a higher degree of instability than figure 7 indicating that the GFM contains a higher degree of instability than the delta function method. The standard explanation of this behavior can be traced to the artificial numerical dissipation inherent in the numerical method. Since the delta function smears out the interface, it incorrectly damps out physical flow features producing a more stable result on the unstable problems.

We emphasize that the higher degree of instability demonstrated by the GFM is due to the more accurate interface representation and not due to nonphysical parasitic flows. Consider a $[0m, .04m] \times [0m, .04m]$ computational domain initially filled with water except for a circular air bubble of radius $.01m$ centered in the middle of the domain. Here we set gravity to zero in order to demonstrate the ability of our scheme to accurately compute sharp pressure jumps across the interface using the numerical algorithm in [11]. Figure 9 shows the initial data as compared to the solution at $t = .05$ seconds on a 40×40 mesh. These two solutions lie directly on top of each other. The interface is not visibly distorted by parasitic flows in part because our scheme is able to sharply resolve the pressure jump as shown in figure 10. Figure 11 shows the initial data plotted on top of the exact solution for the delta function method. Note that the results are comparable with the GFM even though the pressure is smeared out as shown in figure 12. However, the largest velocity produced by the GFM is around $1 \times 10^{-4} \frac{m}{s}$ while the largest velocity produced by the delta function method is 1000 times larger at $.1 \frac{m}{s}$ indicating that it is the delta function method that is more likely to suffer from nonphysical parasitic flows due to the nonphysical smearing of the interface. Note that the exact solution has an identically zero velocity field.

4.4 Example 4

Consider a $[-1m, 1m] \times [-1m, 1m] \times [-1m, 2m]$ computational domain which is initially filled with water except for a spherical air bubble of radius $\frac{1}{3}m$ centered at the origin. The calculation was carried out on a $60 \times 60 \times 90$ Cartesian mesh using the GFM. Figure 13 shows 12 evenly spaced snapshots ($\triangle t = .05$) of the solution from $t = 0$ to $t = .55$ seconds.

4.5 Example 5

Consider a $[-1m, 1m] \times [-1m, 1m] \times [-2m, 1m]$ computational domain consisting of water below $z = -1m$ and air above $z = -1m$. In addition, a spherical water droplet of radius $\frac{1}{3}m$ is centered at the origin. The calculation was carried out on a $60 \times 60 \times 90$ Cartesian mesh using the GFM. Figure 14 shows the solution at $t = 0$, .35, .45, .50, .55, .80, .95, 1.00, 1.05, 1.15, 1.25, and 1.35 seconds.

4.6 Example 6

In order to illustrate the potential of this new method, figure 15 shows water waves generated by the impact of a solid object. Note that the solid object is not rendered so that the surrounding flow field can be more easily visualized. Figure 16 shows the results obtained when a rather large box is filled with water.

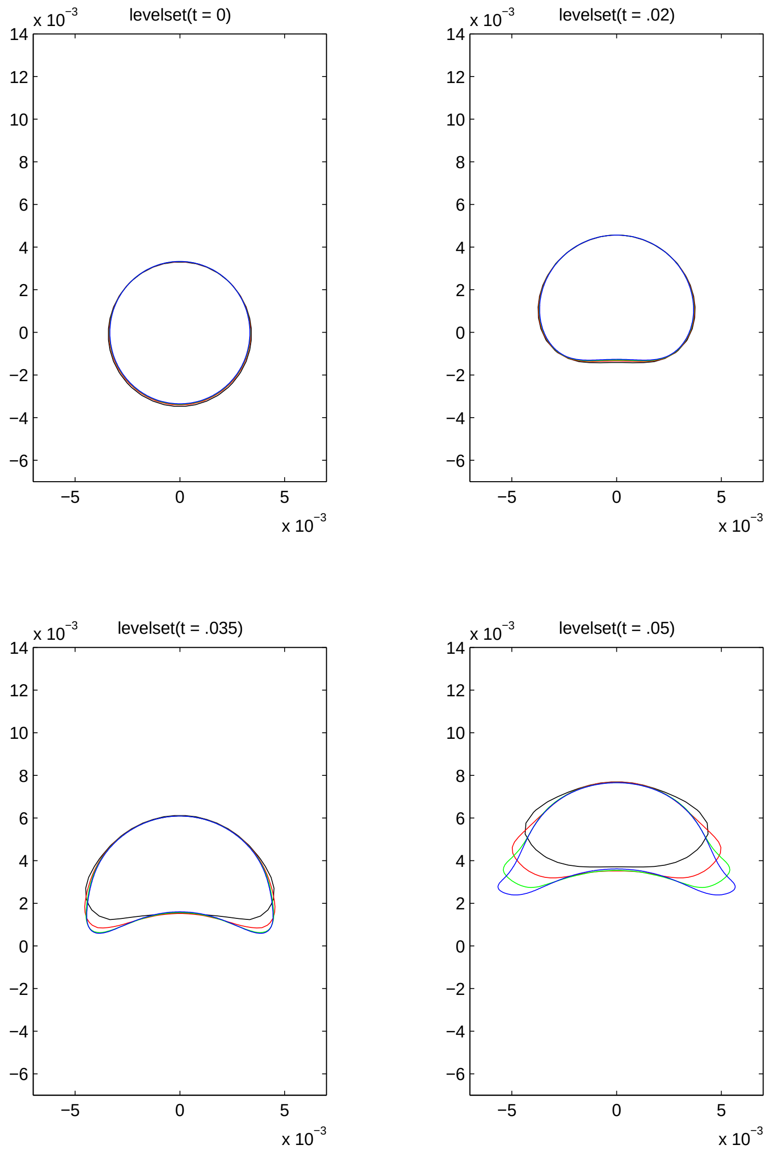

Figure 1: Small Air Bubble - Delta Function Method

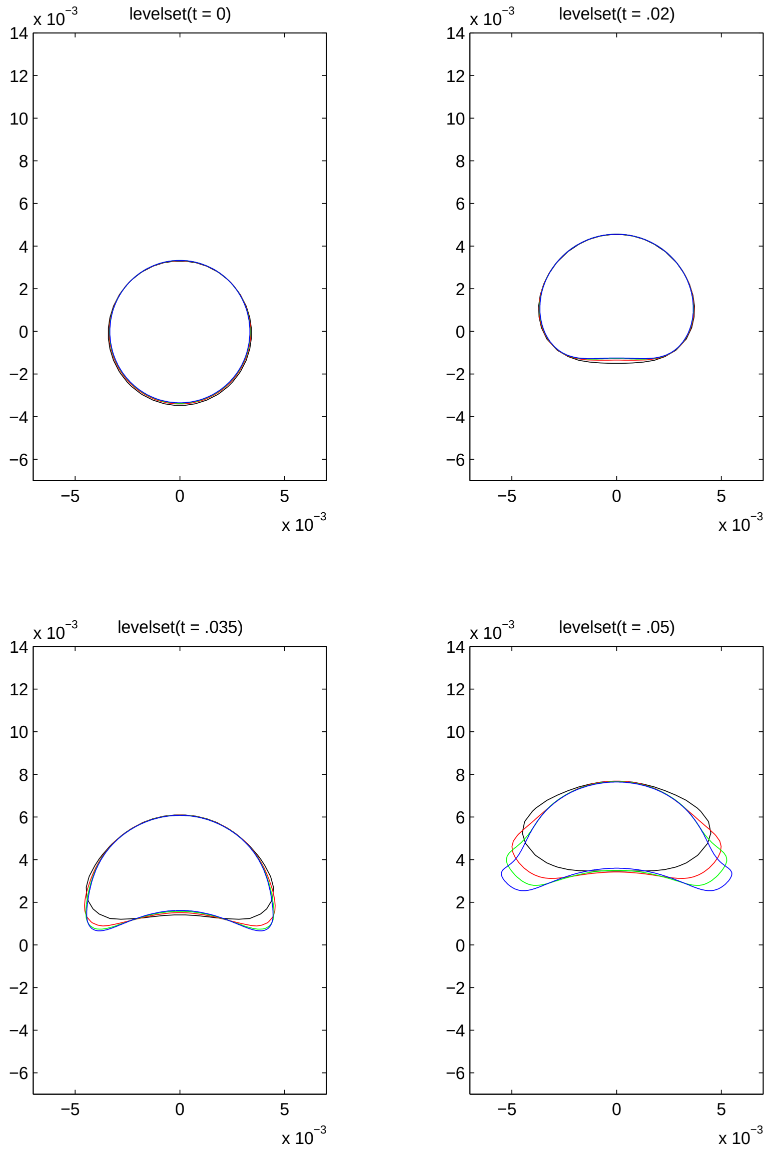

Figure 2: Small Air Bubble - Ghost Fluid Method

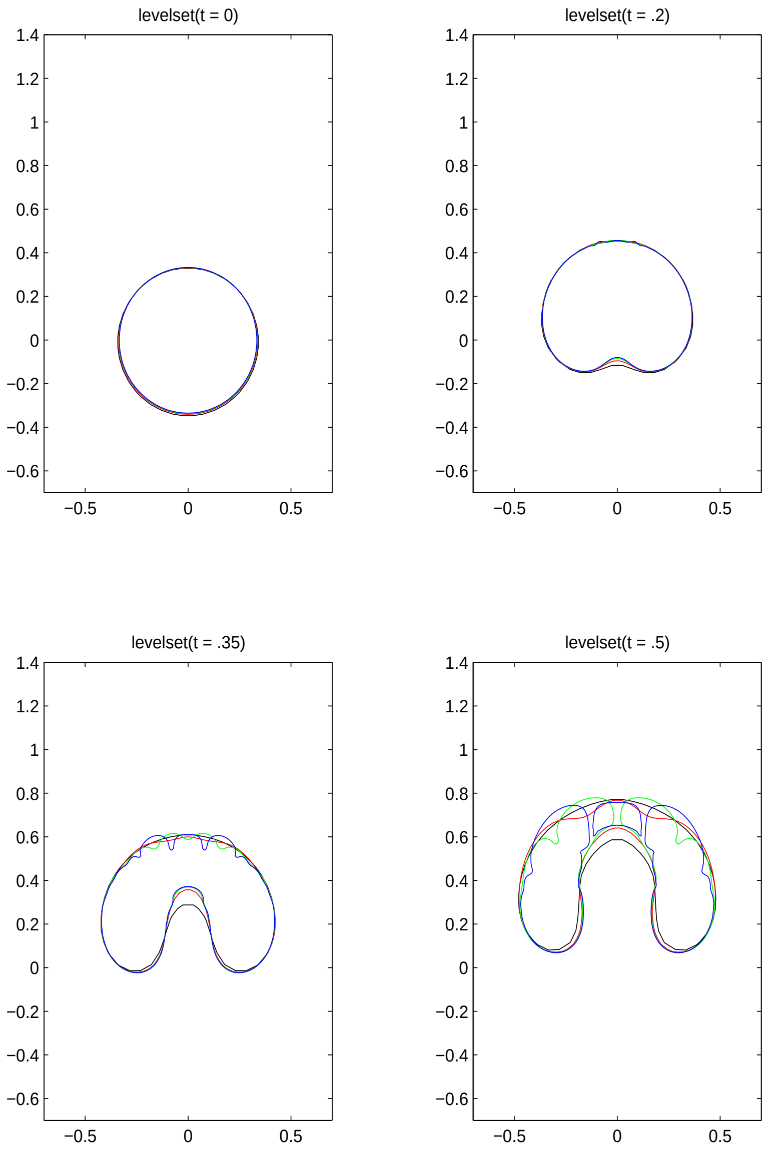

Figure 3: Large Air Bubble - Delta Function Method

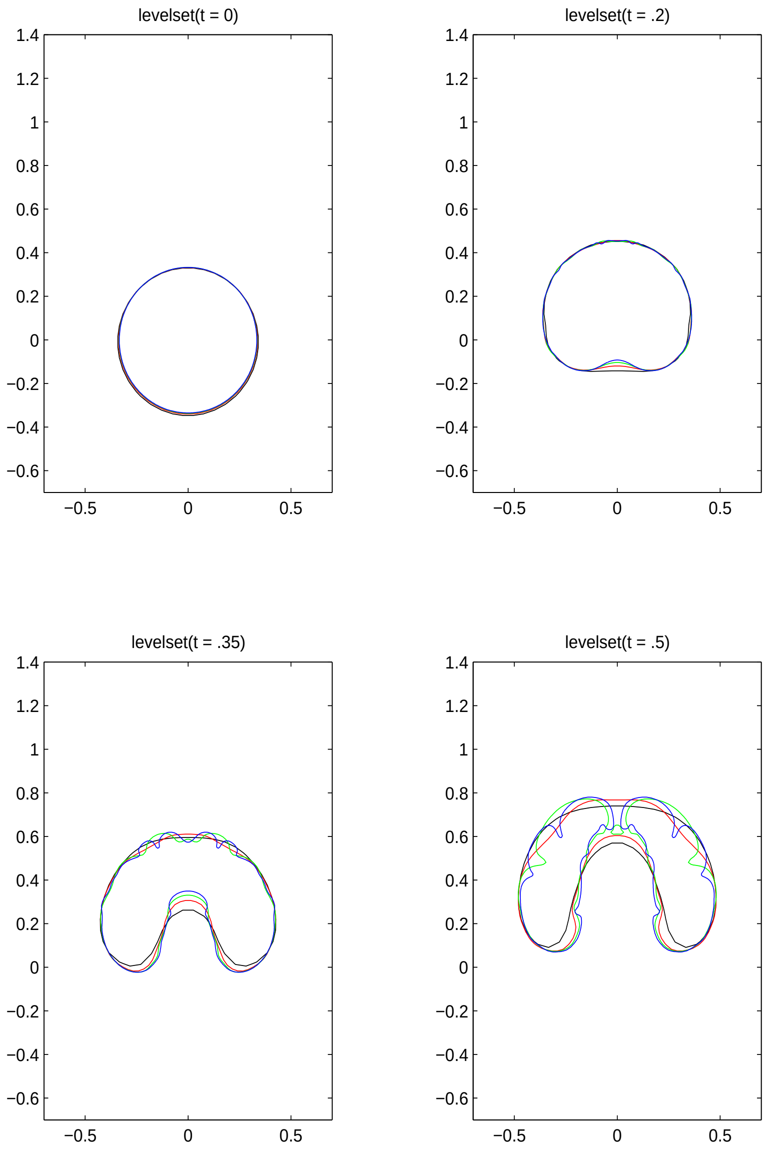

Figure 4: Large Air Bubble - Ghost Fluid Method

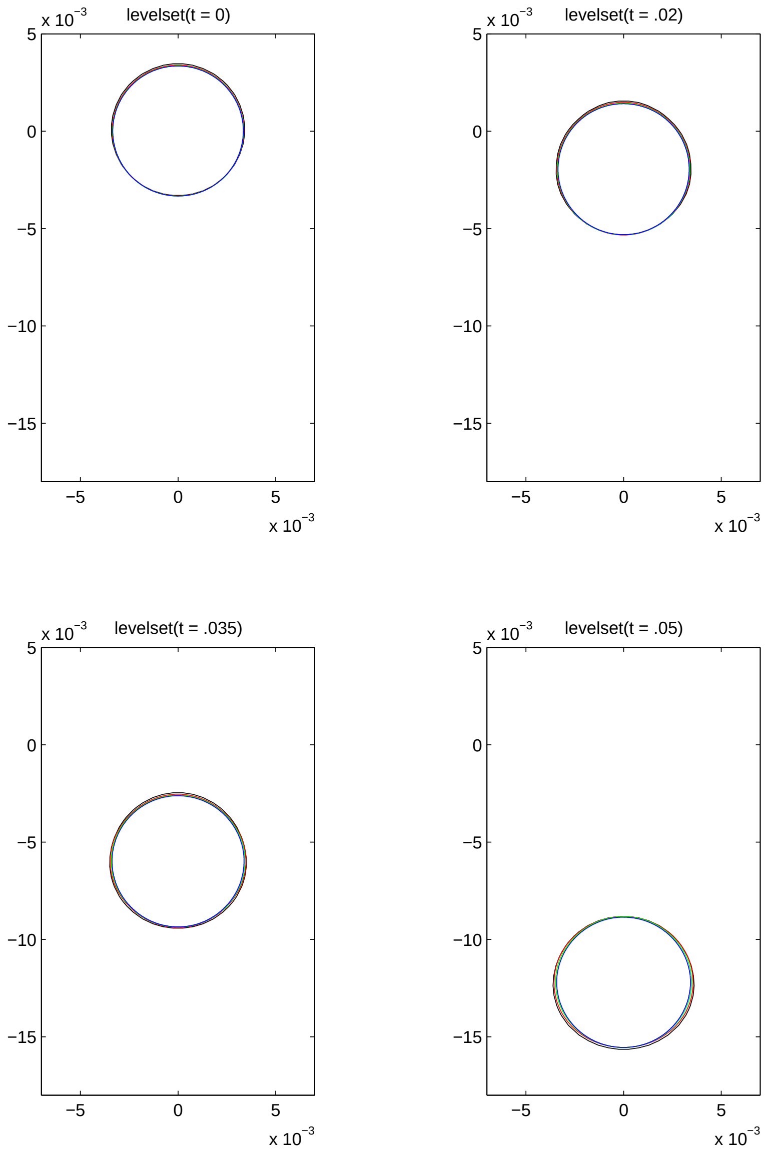

Figure 5: Small Water Droplet - Delta Function Method

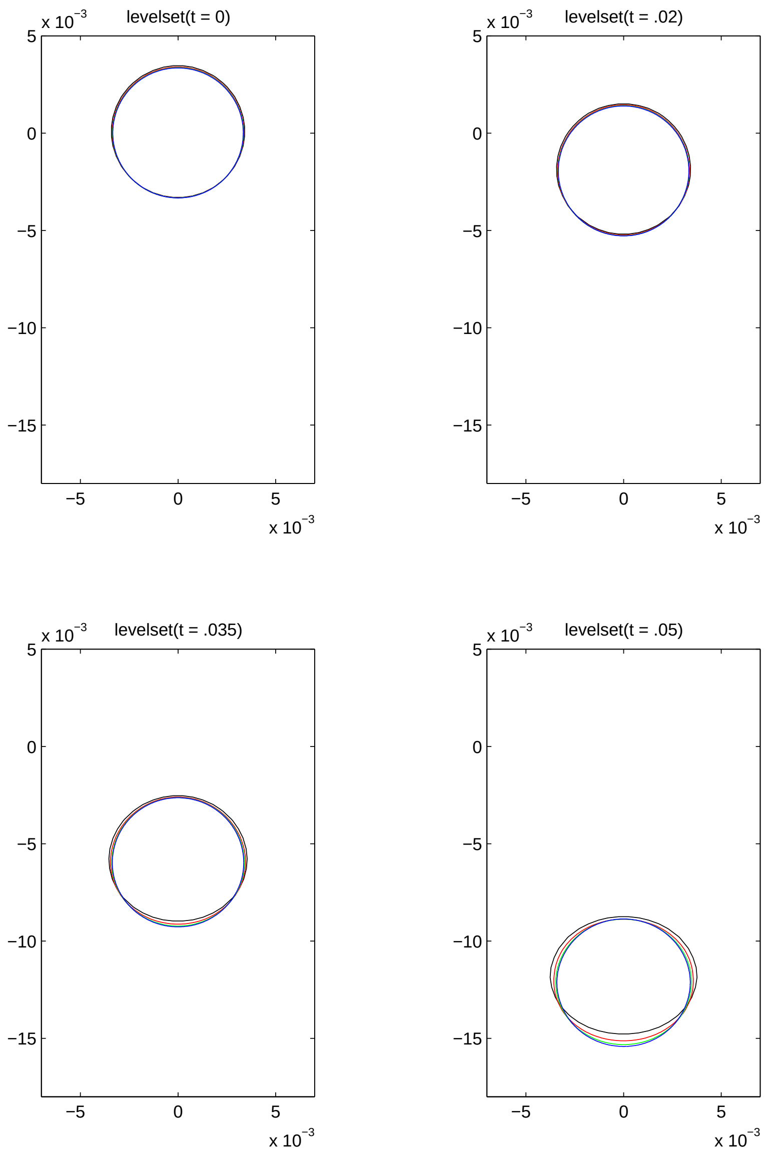

Figure 6: Small Water Droplet - Ghost Fluid Method

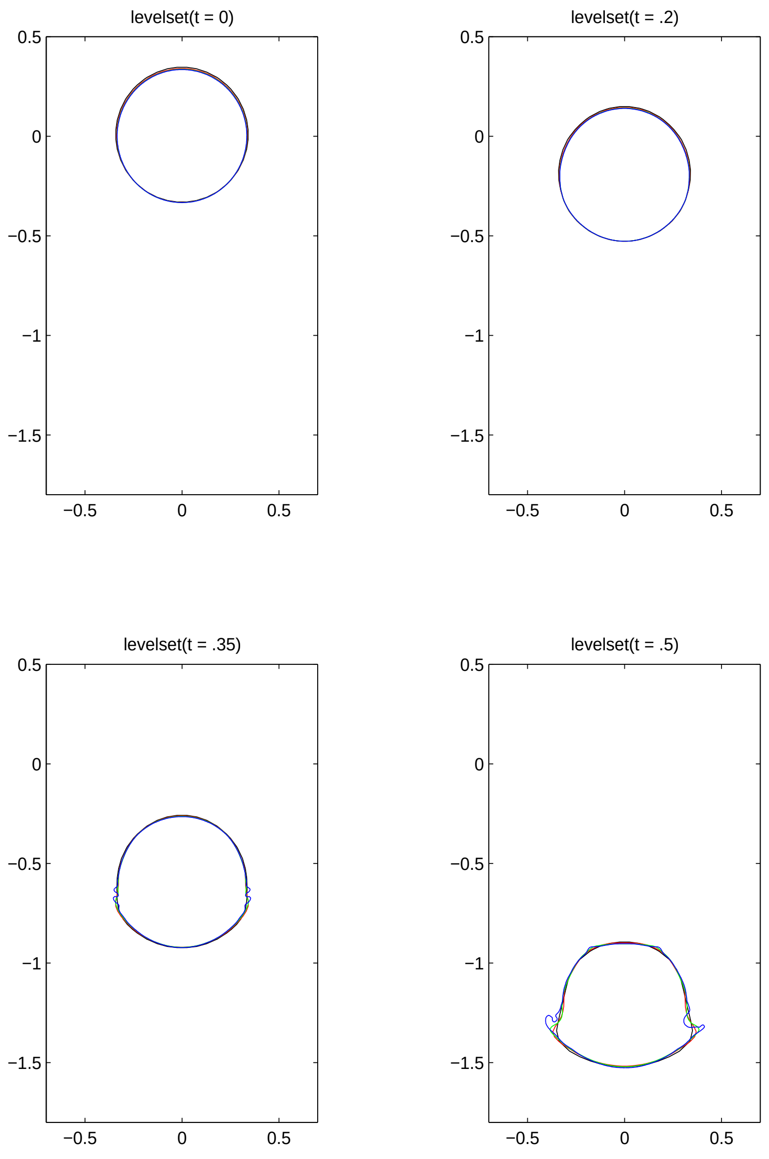

Figure 7: Large Water Droplet - Delta Function Method

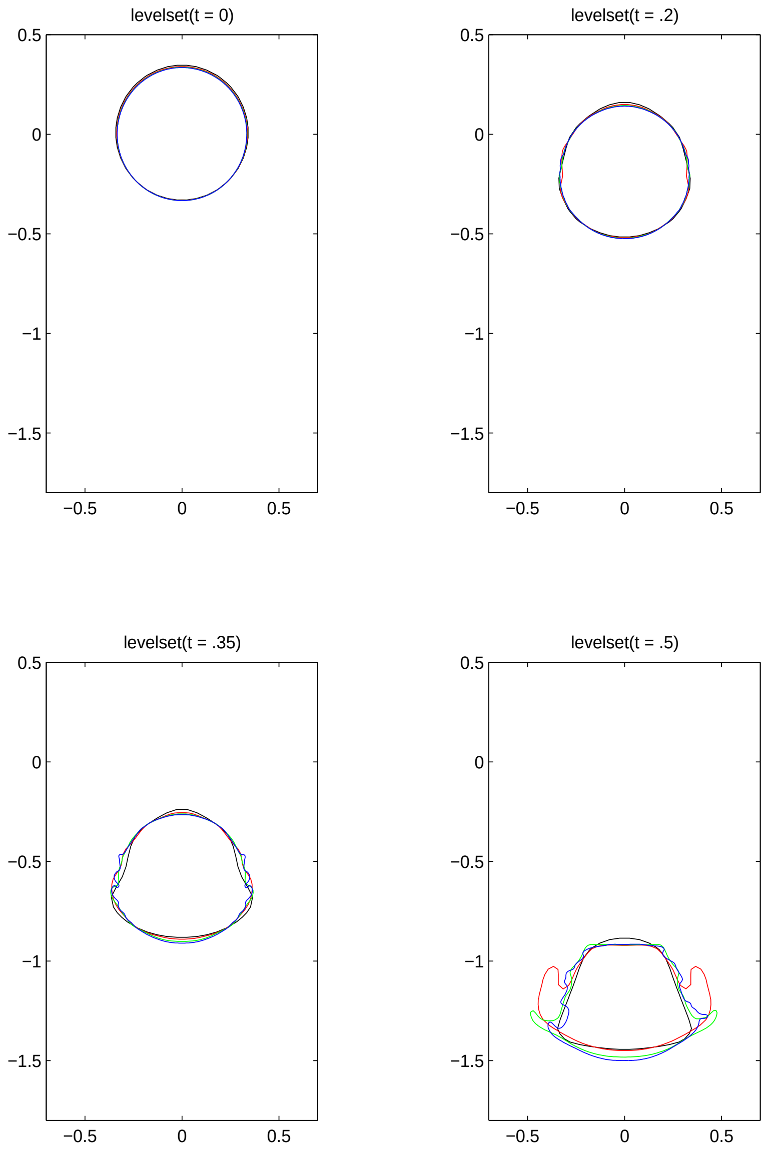

Figure 8: Large Water Droplet - Ghost Fluid Method

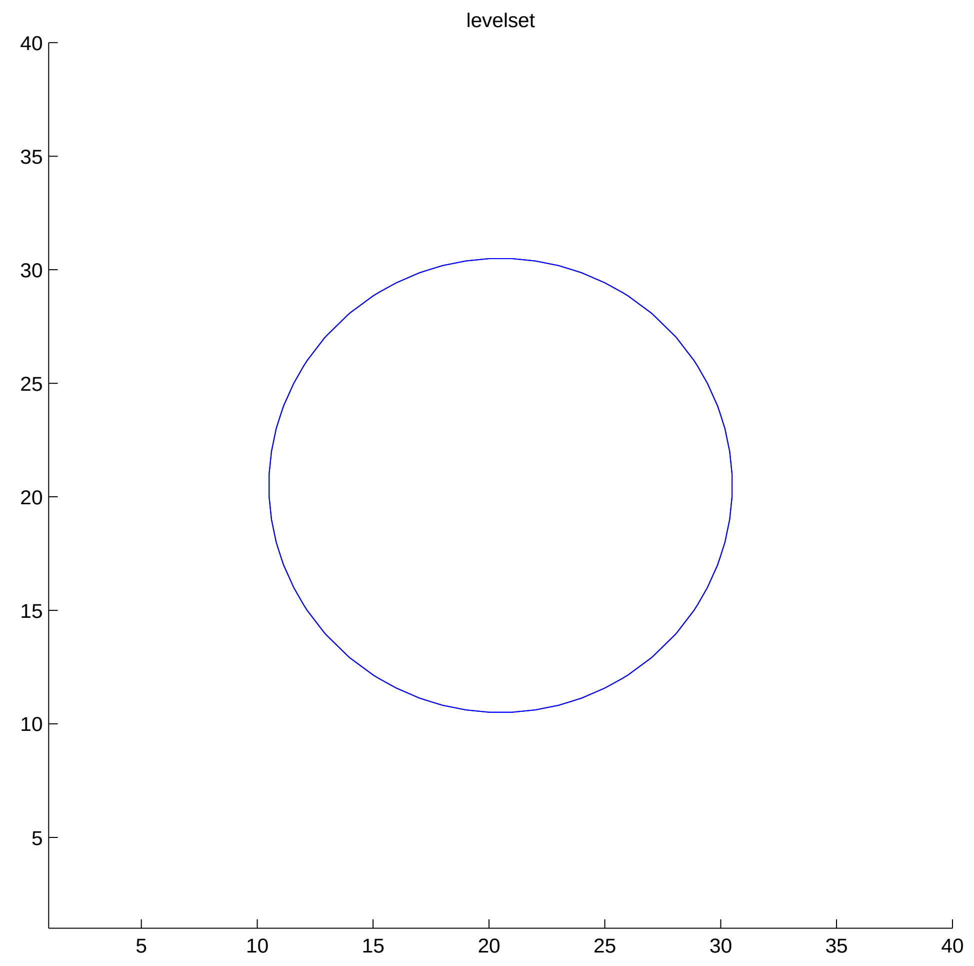

Figure 9: Steady State Air Bubble - Ghost Fluid Method

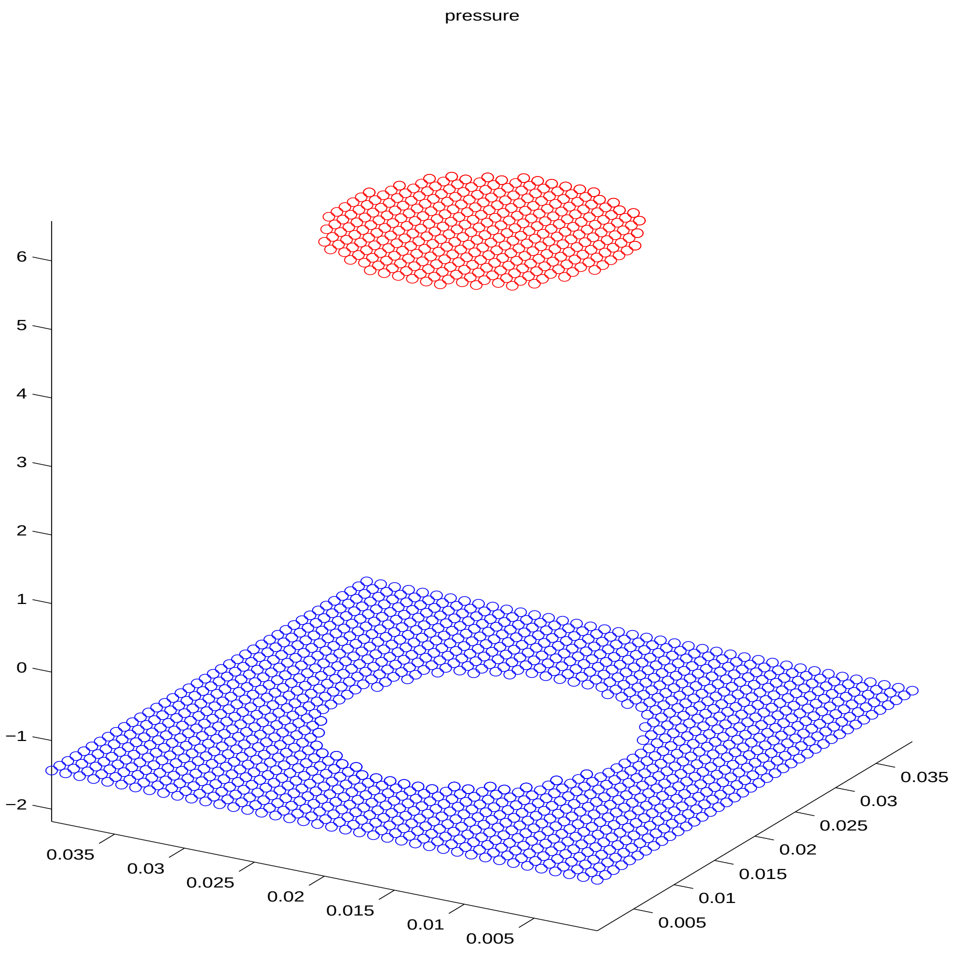

Figure 10: Steady State Air Bubble - Ghost Fluid Method

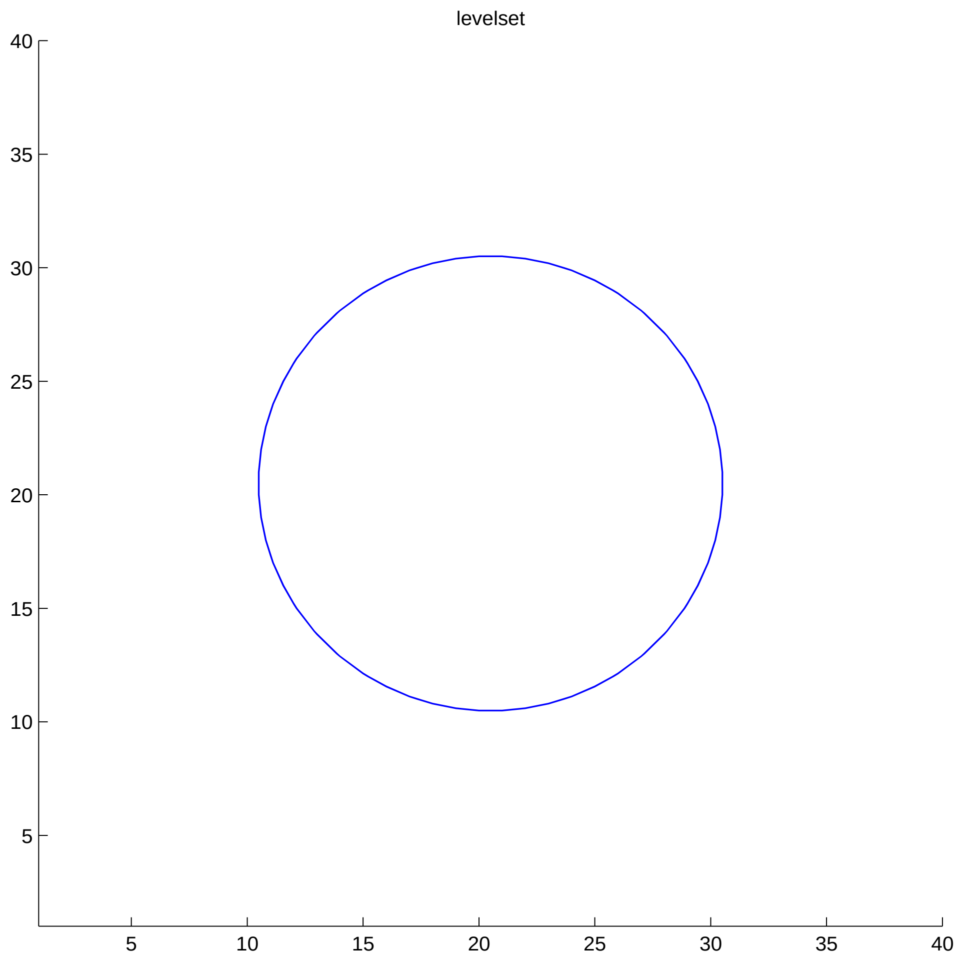

Figure 11: Steady State Air Bubble - Delta Function Method

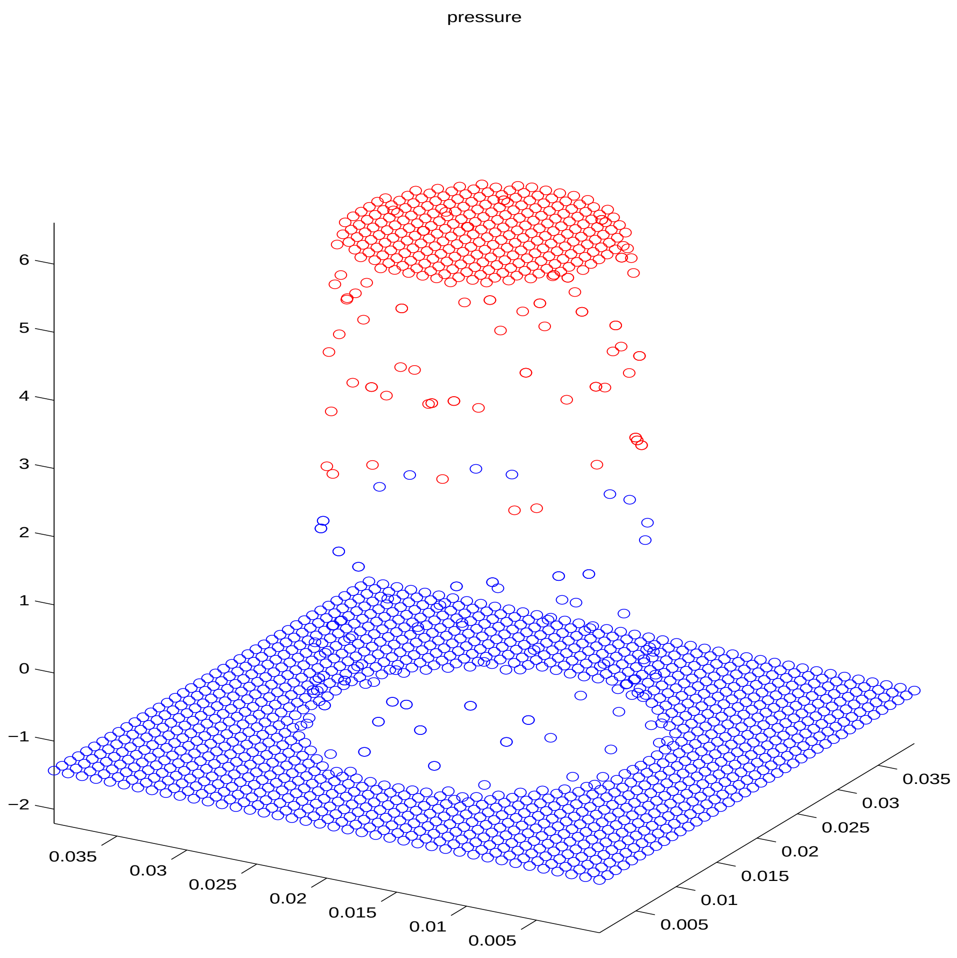

Figure 12: Steady State Air Bubble - Delta Function Method

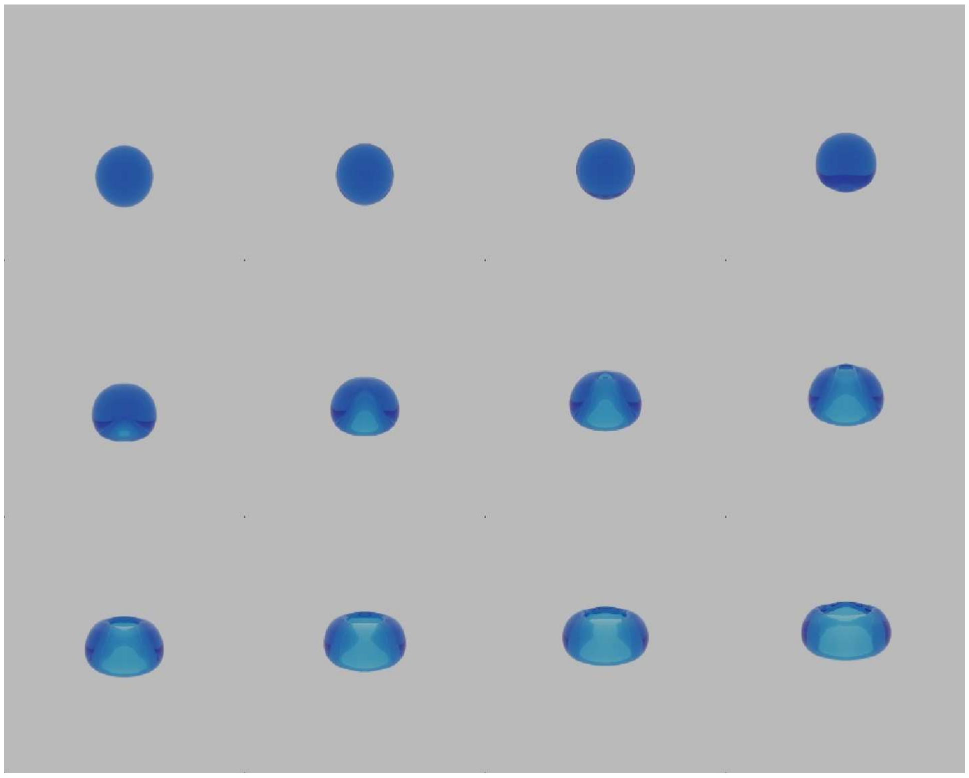

Figure 13: Large Air Bubble - Ghost Fluid Method

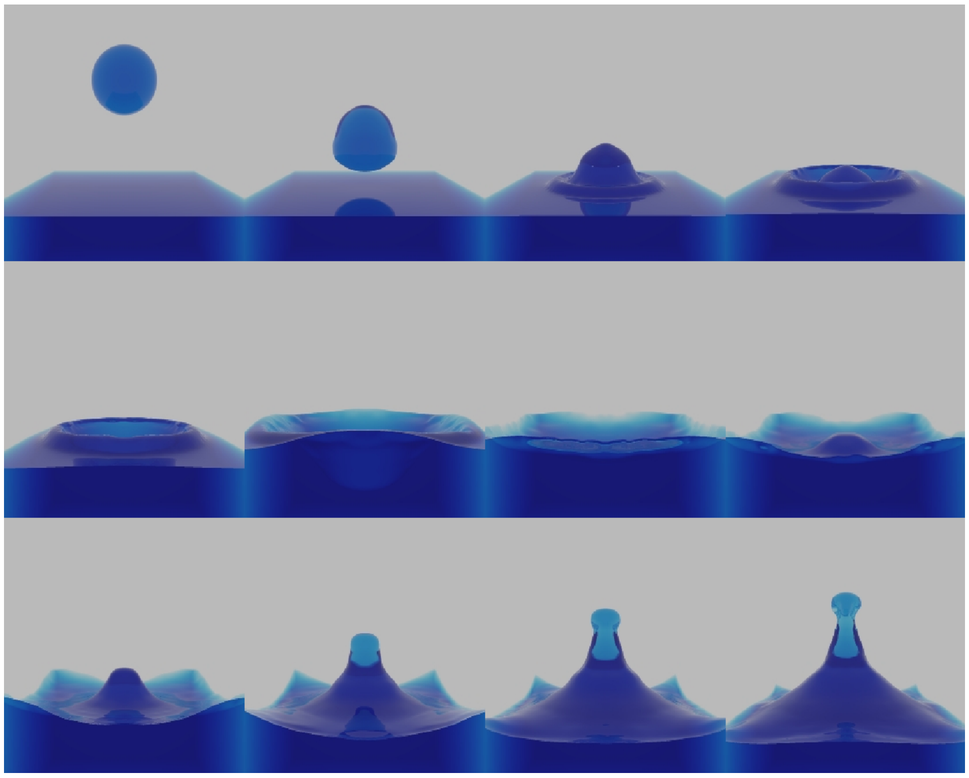

Figure 14: Large Water Droplet - Ghost Fluid Method

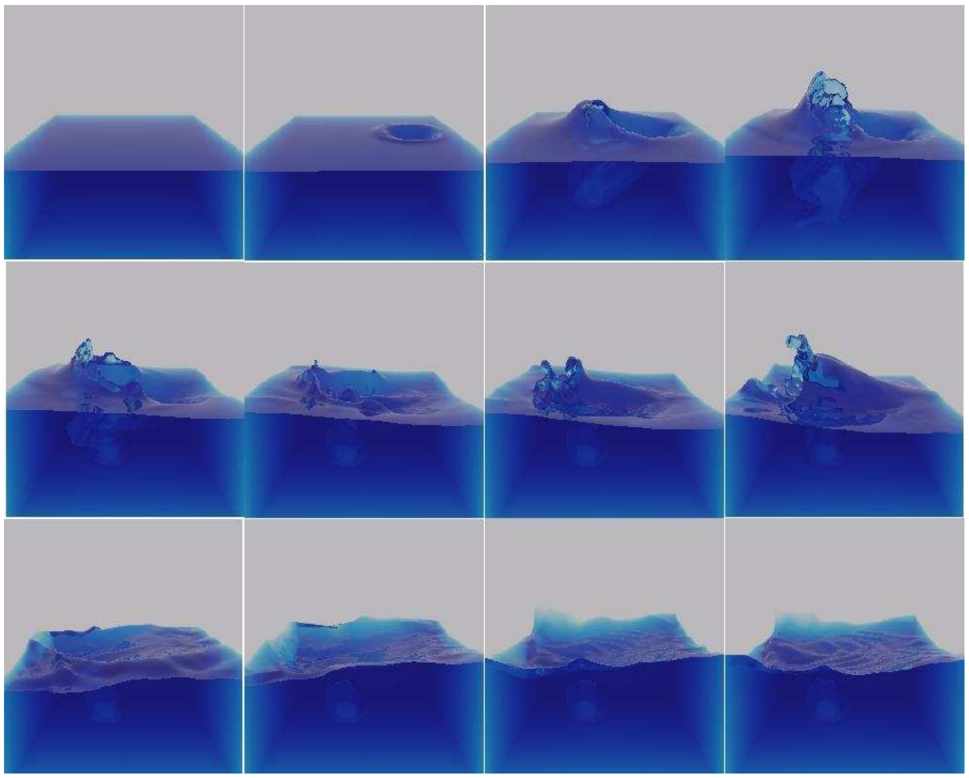

Figure 15: Water Waves Generated by the Impact of an (invisible) Solid Object - Ghost Fluid Method

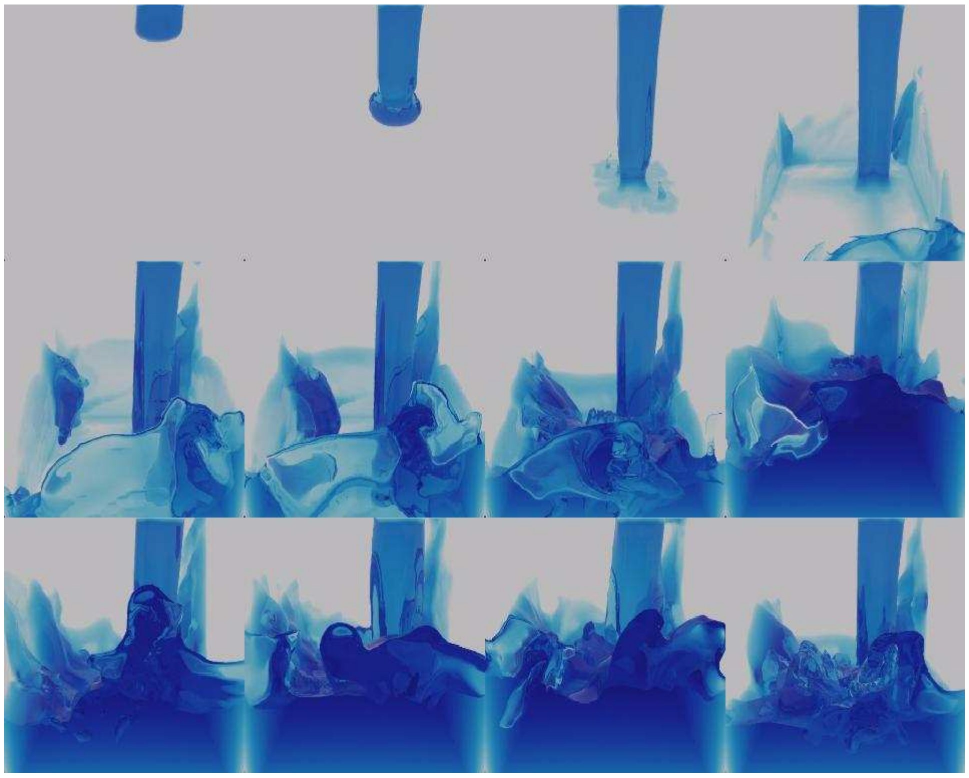
Figure 16: Filling a Box With Water - Ghost Fluid Method

5 Conclusions and Future Work

The numerical experiments performed in this paper indicate that the our new numerical method preforms quite well in both two and three spatial dimensions. Numerical comparisons with the delta function approach proposed in [17] confirm this as well. In this paper, fully two phase water and air mixtures were considered. In future work, we will consider extending our approach to treat free surface flows where the air is replaced with a vacuum. For more information on free surface flows, see [9], [3] and [4] where marker particles were used to track the free surface. We note that free surface flows admit a Poisson equation for the pressure which is significantly different than the one discussed in this paper for fully two phase flow. Therefore, in the case of a free surface, one may want to consider methods similar to that proposed in [18] when solving for the pressure.

References

[1] Brackbill, J.U., Kothe, D.B. and Zemach, C., *A Continuum Method for Modeling Surface Tension*, J. Comput. Phys. 100, 335-354 (1992).

[2] Chang, Y.C., Hou, T.Y., Merriman, B. and Osher, S., *A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows*, J. Comput. Phys. 124, 449-464 (1996).

[3] Chen, S., Johnson, D. and Raad, P., *Velocity Boundary Conditions for the Simulation of Free Surface Fluid Flow*, J. Comput. Phys. 116, 262-276 (1995).

[4] Chen, S., Johnson, D., Raad, P. and Fadda, D., *The Surface Marker and Micro Cell Method*, Int. J. for Num. Methods in Fluids 25, 749-778 (1997).

[5] Chorin, A.J. *Numerical Solution of the Navier-Stokes Equations*, Math. Comp. 22, 745-762 (1968).

[6] Fedkiw, R., Aslam, T., Merriman, B., and Osher, S., *A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method)*, J. Comput. Phys. 152, 457-492 (1999).

[7] Fedkiw, R. and Liu, X.-D., *The Ghost Fluid Method for Viscous Flows*, Progress in Numerical Solutions of Partial Differential Equations, Arachon, France, edited by M. Hafez, July 1998.

[8] Golub, G. and Van Loan, C., *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1989.

[9] Harlow, F.H. and Welch, J.E. *Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with a Free Surface*, The Physics of Fluids 8, 2182-2189 (1965).

[10] Landau, L.D. and Lifshitz, E.M., *Fluid Mechanics*, Pergamon Press, NY, 1978.

[11] Liu, X.-D., Fedkiw, R. and Kang, M., *A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains*, J. Comput. Phys. 160, 151-178 (2000).

[12] Osher, S. and Sethian, J.A., *Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, J. Comput. Phys. 79, 12-49, (1988).

[13] Peyret, R. and Taylor, T.D., *Computational Methods for Fluid Flow*, Springer-Verlag, NY, 1983.

[14] Peskin, C., *Numerical Analysis of Blood Flow in the Heart*, J. Comput. Phys. 25, 220-252 (1977).

[15] Peskin, C. and Printz, B., *Improved Volume Conservation in the Computation of Flows with Immersed Elastic Boundaries*, J. Comput. Phys. 105, 33-46 (1993).

[16] Shu, C.W. and Osher, S., *Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes*, J. Comput. Phys. 77, 439-471 (1988).

[17] Sussman, M., Smereka, P. and Osher, S., *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow*, J. Comput. Phys. 114, 146-154 (1994).

[18] Tau, E.Y. *A Second Order Projection Method for the Incompressible Navier-Stokes Equations in Arbitrary Domains*, J. Comput. Phys. 115, 147-152 (1994).

[19] Unverdi, S.O. and Tryggvason, G., *A Front-Tracking Method for Viscous, Incompressible, Multi-Fluid Flows*, J. Comput. Phys. 100, 25-37 (1992).