Multiple Interacting Fluids

Brendan Miller
Department of Computer Science
The University of Texas at Austin
brendanmiller.ca@gmail.com

August 19, 2019

Contents

1 Introduction

The following paper aims to describe the mathematical rules as well as the numerical solutions of multiphase flow. Additional methods, such as the Particle Level Set method, will be described. This paper assumes some background in calculus and tries to be accessible to those without background in fluid dynamics.

2 Governing Equations

The equations that govern the behavior of incompressible fluids are the Navier-Stokes equation for incompressible inviscid flow:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla \frac{p}{\rho} + g \tag{1}$$

and the incompressibility condition

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

To define our symbols: u represents a velocity, p represents pressure, ρ represents gravity, and g represents external force, in our case gravity. 1 describes how a velocity field changes over time, and 2 constrains the velocity field to have divergence.

First, we will describe the motivation for the incompressibility condition. An incompressible fluid does not compress or expand. If you were to fill a cup with water until its full, then add more water to it, the excess water simply spills out. In other words, if a container is always full (this notion will be explained more when we discretize incompressibility), the amount flowing in the the container must be equal to the amount flowing out.

Another more direct example is the classic Eulerian view of fluids. Say you are by the side of a river. If you look at a little cube of space within the river (so that it is always underwater), you will find that just as much water flows into the cube as flows out. For contradiction, if more fluid went in to our imaginary cube that flowed out, then the density in that cube would increase due to compression. We discretize divergence almost exactly as the person standing by the river - we look at discrete packets of space and make sure that just as much fluid is flowing in as is flowing out.

Now let us focus on 1. Going left to right, we see the different terms represent: change in u over time, convection (or self-advection) of u, the pressure gradient, and external forces. So for the purpose of updating the velocity field, we want to get the $\frac{\partial u}{\partial t}$ term on its own. This gives us

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u + g - \nabla \frac{p}{\rho} \tag{3}$$

This equation looks formidable, but can be split into clear, intuitive subroutines. First, we solve the convection term and get u^*. Then, we add external forces to get u^{**}. Then, we compute $\nabla \frac{p}{\rho}$ and add in the forces due to pressure. The first two subroutines are fairly clear to explain, and will be covered in the next section. Because of its complexity, I will give background for how we find the pressure gradient.

To obtain the formula that dictates our new pressure values, we take the divergence of 1

$$\nabla \frac{\partial u}{\partial t} + \nabla \cdot ((u \cdot \nabla)u - g) = \nabla \cdot (-\nabla \frac{p}{\rho}) \tag{4}$$

We make use of 2 here - since $\nabla \cdot u = 0$, then $\nabla \cdot \frac{\partial u}{\partial t} = 0$. So we substitute and get the following:

$$\nabla \cdot ((u \cdot \nabla)u - g) = \nabla \cdot (-\nabla \frac{p}{\rho}) \tag{5}$$

This can be simplified even further. Remember the meaning of the $(u \cdot \nabla)u$ term and the g term. They represent movement due to convection, and movement due to gravity. So the left hand side of this equation is equal to the divergence of u^{**} that we defined earlier.

$$-\nabla \cdot (\nabla \frac{p}{\rho}) = \nabla \cdot (u^{**}) \tag{6}$$

We now have a Poisson equation with $f(x) = -\nabla \cdot u^{**}(x)$ and $\phi(x) = \frac{p(x)}{\rho(x)}$. Note that x is a position, and not necessarily a scalar. Put in its canonical form, we have:

$$\triangle \phi = f \tag{7}$$

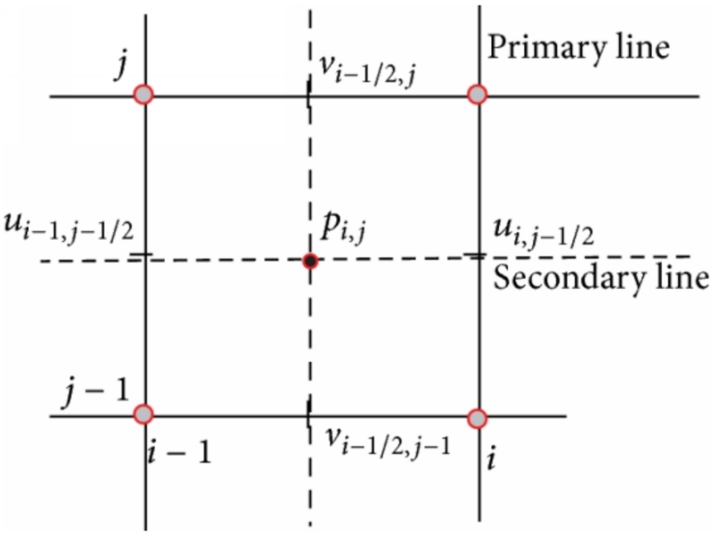
One more note on the Poisson equation. We are solving a varying coefficient Poisson equation which requires some extra numerical methods to prevent strange behavior near the boundaries. These methods will be clear in the next section.

3 Discretizations and Numerical Methods

3.1 General Use Discretizations

This section will describe some of the essential concepts we make use of to solve the Navier-Stokes equations for incompressible inviscid flow.

We discretely store values of u, p, and others, on a staggered grid. We define a "cell" or "voxel" to be the atomic unit of our grid. We use the term "grid node", "cell center", or "voxel center" to refer to quantities that are stored in the center of our voxels. We use the term "grid face" to refer to the location of the sides of the voxels, specifically at the midpoint of a given side. We store scalar values like pressure at the grid nodes, and vector values like velocity at the grid faces. Note: the term "voxel" will be used (incorrectly) to describe packets of space in one, two, and three dimensions. This is simply because I am more concerned with conveying the notion of "a discrete packet of space" than with the precise, dimension-specific terminology.



To obtain the value of a field (for example pressure), we perform linear interpolation on the 2^n neighboring grid nodes (in this case), where n is the number of spatial dimensions.

Note: the motivation for using a staggered grid instead of storing all values in the grid nodes has to do with finding divergence and then the pressure gradient. Since we store divergence in the grid nodes, We can use second-order accurate central differencing to compute it. Here is an example in one dimension:

$$\nabla \cdot u_i = \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x} \tag{8}$$

where Δx is the length of a voxel. Note: in higher dimensions, we assume the grid is regular - so voxel length, height, and depth are all equal to the same Δx. Additionally, we make use of the staggered grid configuration when we apply the pressure gradient update, as velocity values are stored between pressure values.

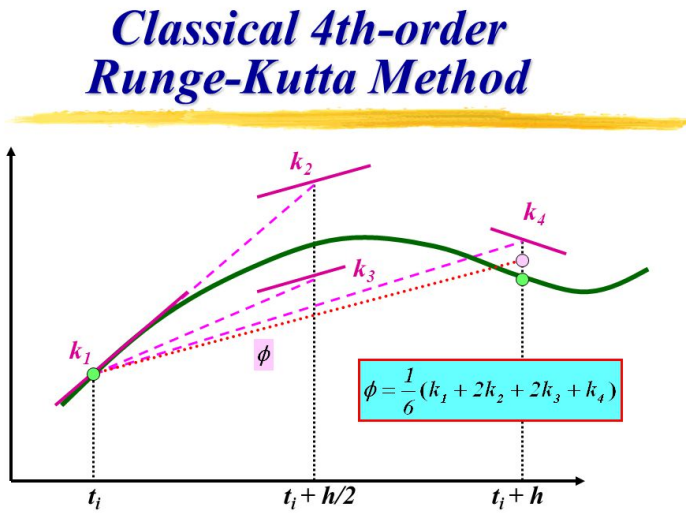
3.2 Integration

We integrate motion using the 4th order Runge-Kutta method. Forward Euler integration is only 1st order accurate with respect to Δx. When in higher dimensions, we can split by dimension and perform separate integration for the x, y, and z components. Here is the method in two dimensions, updating the x component

by a timestep Δt. In this case, u is defined by the x-component of velocity at a position.

$$\begin{aligned} L_1 &= \Delta t u(x_n, y_n) \\ L_2 &= \Delta t u(x_n + \frac{L_1}{2}, y_n + \frac{\Delta t}{2}) \\ L_3 &= \Delta t u(x_n + \frac{L_2}{2}, y_n + \frac{\Delta t}{2}) \\ L_4 &= \Delta t u(x_n + L_3, y_n + \Delta t) \\ x_{n+1} &= x_n + \frac{1}{6}(L_1 + 2L_2 + 2L_3 + L_4) \end{aligned} \tag{9}$$

y_{n+1} is computed similarly. The intuition behind these iterative methods is to sample the derivative at multiple points and interpolate them. The following diagram is an example of this method applied in one spatial dimension.



Note: all spatial integrations in this project make use of Fourth-Order Runge-Kutta unless specified.

3.3 Discretizing the Convection Term

Convection, or self advection, is the process of the velocity field moving within itself. For example, consider a draft of hot air moving upwards, and caryying along with it an upwards draft. So the velocity field in the air is moving itself. This is performed using the following equation:

$$(x_i)_{t+1} = RK4((x_i)_t, -\Delta t) \tag{10}$$

This gives us the position "where the velocity at that point came from" a certain duration of time ago. So we backtrack to see where things will end up in the future. Then we simply set the value of u_i to the interpolated value at the position $(x_i)_{t+1}$.

As a side note, we add in the effects of gravity by subtracting $\Delta t g$ from all vertical velocity values.

3.4 Discretizing the Poisson Equation

We use a modification of the standard second-order discretization of Poisson equation. In one dimension, we have:

$$\frac{\beta_{i+\frac{1}{2}}(\frac{p_{i+1}-p_i}{\Delta x}) - \beta_{i-\frac{1}{2}}(\frac{p_i-p_{i-1}}{\Delta x})}{\Delta x} = \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x} \tag{11}$$

$\beta(i+\frac{1}{2})$ is the density at the voxel face between i and $i+1$. If the interface between two fluids is between i and $i+1$, $\beta(i+\frac{1}{2})$ is handled specially, and called $\hat{\beta}$. Also note that p is the unknown in this equation.

Assume i and $i+1$ contain the same fluid. Then we set $\beta_{i+\frac{1}{2}}$ is set to $\frac{1}{\rho}$, where ρ is the density of that fluid.

Now assume i contains fluid A and $i+1$ contains fluid B. Then the interface between fluids A and B is between i and $i+1$. Due to jump conditions between fluid regions, p_i may be different for the different fluids. Essentially, jump conditions may make $(p)_x$ discontinuous across the interface. Consider a pressurized shock wave expanding throughout an ambient atmosphere - there is a sudden jump in pressure across the interface between the pressurized gas and the rest of the ambient atmosphere.

To explain this more concretely, consider the discretization in 11. In the term with $p_{i+1} - p_i$. Since we might have jump conditions, we need to replace p_{i+1} with an adjusted fictitious pressure - the pressure at the interface p_I. Taking the pressure at the interface allows us to avoid dealing with jump conditions since we are not crossing the interface.

Before we proceed, lets designate the fluid region on the $i+1$ side as the $+$ region, and the other region as the $-$ region. We know that flux across the interface is continuous, so

$$\beta^- p^- = \beta^+ p^+ \tag{12}$$

First, define theta, which gives us the relative position of the interface to the two nodes it lies between.

$$\theta = \frac{|\phi^+_{i+1}|}{|\phi^+_{i+1}| + |\phi^-_i|} \tag{13}$$

We then discretize 12 similarly to 11, but using a fictitious interface pressure p_I:

$$\frac{\beta^-(p_I - p_i)}{\theta \Delta x} = \frac{\beta^+(p_{i+1} - p_I)}{(1-\theta)\Delta x} \tag{14}$$

We then solve for p_I, and substitute it back in to 14 to get a modified $\hat{\beta}_{i+\frac{1}{2}}$:

$$p_I = \frac{(\theta\beta^+ p_{i+1} + (1-\theta)\beta^- p_i}{\theta\beta^+ + (1-\theta)\beta^-} \tag{15}$$

which results in

$$\hat{\beta}_{i+\frac{1}{2}} = \frac{\beta^-\beta^+}{\theta\beta^+ + (1-\theta)\beta^-} \tag{16}$$

This method handles the jump conditions without explicitly defining them, which makes them straightforward to implement.

Now that we have defined our discretization method, we can outline our solution method. We put the linear system that we have obtained into matrix-vector form. For a given row (which designates a fluid cell), the columns which correspond to neighboring fluid cells are given the coefficient $\frac{\beta}{\Delta x}$ where β is sampled between the central cell and the neighboring cell. The coefficient on the diagonal is equal to the negative sum of this value. If we multiply each entry by -1, we get a symmetric, weakly diagonal matrix that is easily solved with a Preconditioned Conjugate Gradient solver. The right hand side of the equation is simply the discrete divergence at each voxel.

After computing the new pressure, we apply the force of pressure to the current velocity field u^{**}. Again, when sampling density in this case we use our method to find $\hat{\beta}$ if two neighboring cells contain two different fluids.

Note: I am only lightly covering the method to solve our system of equations, as the solution method is identical to a standard constant coefficient Poisson equation solution.

4 Level Set Methods

Now that we have described the laws governing our computational domain, as well as the numerical methods we use to evolve the domain forward in time, we can describe the method used for interface tracking. Each fluid in the domain is equipped with a level set function that describes its interface.

Much of the following has been described in earlier papers - here is a summary of the level set methods used in this paper (Note: The subsections are listed loosely in order (with the only exception of correcting the

level set once more after reinitialization) so that the reader can get a feel for the subroutines within the simulation loop):

A level set function $\phi(x, y)$ takes in a position and returns a signed distance from the surface. So if we have a ϕ function that represents a drop of water, ϕ will be negative for points within the water drop, positive if outside the water, and zero on the surface.

We begin the simulation by initializing our fluids with predetermined level set values. Each fluid has its own ϕ, and each ϕ is evolved and corrected by marker particles independently and then all ϕ's are merged in the projection step.

4.1 Advection of ϕ

We advect ϕ through the velocity field. This is with the equation

$$\phi_t = -V \cdot \nabla\phi \tag{17}$$

where V is the velocity field.

We use upwinding with respect to velocity to compute $\nabla\phi$. The following is an example of our upwinding scheme in one dimension:

$$\frac{d\phi}{dx} = \begin{cases} \frac{\phi_{i+1}-\phi_i}{h} & V < 0 \\ \frac{\phi_i-\phi_{i-1}}{h} & V >= 0 \end{cases} \tag{18}$$

An explanation for this is that when we advect, we are "backtracking" from a position. So if the flow is going mostly to the right, we want to look at values to the left so as to best "go upstream"

4.2 Particle Level Set Method - Reseed Particles

Reseeding is performed periodically (in this case once every 10 simulation steps). First, excessively distant (particles with a local ϕ value of greater than $3\Delta x$) particles are removed. Then, each particle is counted, and if there are more than 64 particles in any grid cell, they are deleted until exactly 64 remain. Then, for any grid cells with less than 64 marker particles that are within the narrow band (of $\phi < 3\Delta x$), we add particles to make sure there are 64 in each cell near the interface. We then attract the particles towards the boundary using the formula, where x_p and ϕ_p are the position and ϕ value of the particle, the ϕ_{goal} is a number we would like to narrow the band to. This implementation attracts particles to a clamped ϕ_{goal} between $\pm 0.1\Delta x$ and $\pm 2.0\Delta x$.

$$x_p^{\text{new}} = x_p + (\phi_{\text{goal}} - \phi_p)\frac{\phi}{\|\phi\|}, \tag{19}$$

4.3 Particle Level Set Method - Marker Particle Advection

Marker particles used in the Particle Level Set method are advected through the velocity field with the Fourth-Order Runge-Kutta method. Their positions are then clamped in bounds, and any particles in solid voxels are pushed out of the solid by moving them along the gradient of the solid's ϕ function.

Each particle stores its current position, its initial ϕ value, and its radius, all of which will be explained in the next subsection.

4.4 Particle Level Set Method - Levelset Correction

This step is the underpinning of the level set method. We begin by creating 2 copies of the ϕ array that represents a given fluid, lets call then ϕ^+ and ϕ^- We iterate through every particle to see if it has "escaped". A particle is considered escaped if it satisfies 2 conditions, the initial ϕ value of the particle is the opposite sign of the local ϕ value at the particles current position, and that the difference between those two values is at least the radius of the particle. In other words, we consider a particle as escaped if it has "overstepped the boundary by more than its radius". For each escaped particle, we check its neighboring 8 grid nodes. For each of these nodes, we compute a value ϕ_p, with the particle's initial phi value ϕ_{p0}

$$\phi_p = \text{sign}(\phi_{p0})(\text{radius}_p - \left\|\text{position}_{\text{node}}, \text{position}_p)\right\|; \tag{20}$$

which represents an extension of the escaped particles ϕ value. We now take ϕ^+ at each node to be the maximum of the initial ϕ at that node, and all ϕ_p's corresponding to that node. We do a similar sequence to

construct ϕ^-, with the difference that we take the minimum of all such values. The final step is to "merge" the two: at each grid node, we set our corrected ϕ to be equal to ϕ^+ if $|\phi^+| < |\phi^-|$, and equal to ϕ^- otherwise. This can be thought of as a reconstruction biased towards the fluid interface, as small *phi* values are given priority.

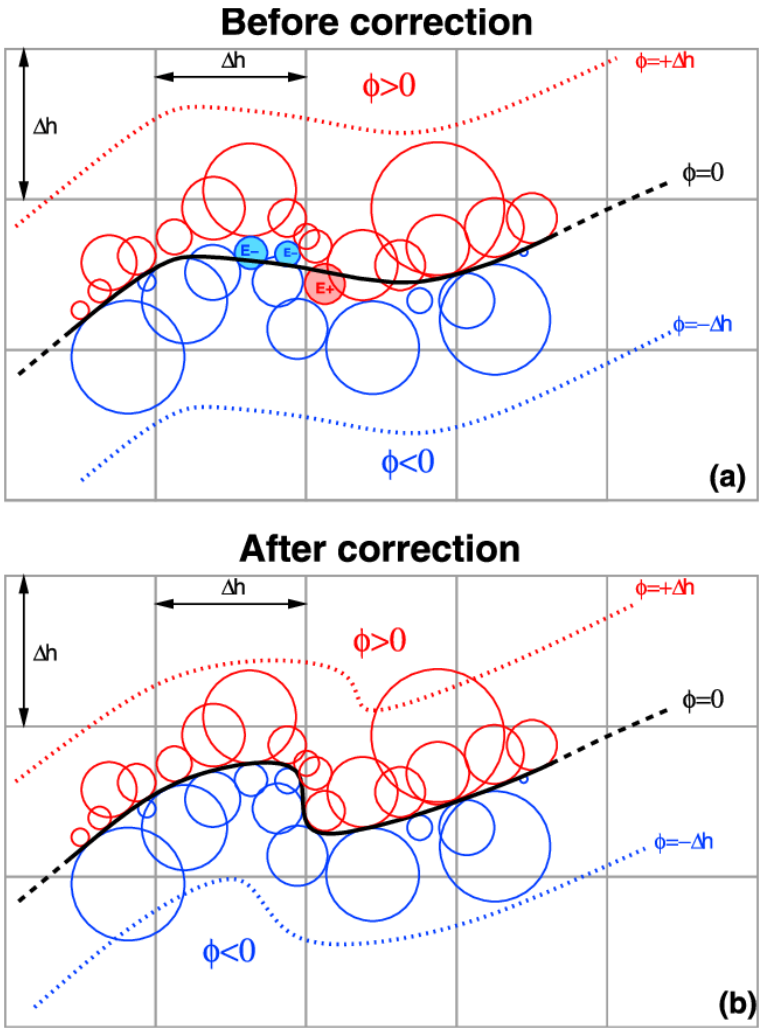


Figure 1: A diagram of a level set before being corrected by particles that have escaped "far enough", and after being corrected (Samuel and Evonuk 2010).

4.5 Levelset Reinitialization

Advecting ϕ moves our fluid interface as expected, but the newly advected ϕ is not necessarily a signed distance function with desired properties. We want our ϕ to have a normalized gradient everywhere:

$$\|\nabla\phi\| = 1 \tag{21}$$

So we fix this by solving the following differential equation

$$\phi_t + S(\phi_0)(\|\nabla\phi\| - 1) = 0 \tag{22}$$

where $S(\phi_0)$ is the sigmoid smearing function:

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + \Delta x^2}} \tag{23}$$

We use Godunov's scheme to obtain the gradients for use in this calculation, as we want information to propagate from the fluid's boundaries outward (Osher and Fedkiw 2003). We will now explain Godunov's scheme, as well as why it causes information to correctly propagate.

Godunov's Scheme:
In this context, we are using Godunov's scheme to propagate information from the zero isocontour of our level set to other points on the grid. This is used specifically to reinitialize our ϕ to be a signed distance function - that is, at all points in our space, we want

$$\|\phi\| = 1 \tag{24}$$

so that we can make use of having a well behaved normal and curvature of the fluid surface. This shares similarities to upwinding (in that it determines which method of approximation for the gradient based on data from the surrounding areas), but takes into account the position relative to the level set function. A compact form of this method is seen for the direction of derivative of ϕ with respect to x:

$$\phi_x = \begin{cases} \phi_x^2 = \max(\max(\phi_x^-, 0)^2, \min(\phi_x^+, 0)^2) & \phi \leq 0 \\ \phi_x^2 = \max(\min(\phi_x^-, 0)^2, \max(\phi_x^+, 0)^2) & \phi > 0 \end{cases} \tag{25}$$

Consider a point inside of the fluid, where $\phi < 0$. If both ϕ_x^- and phi_x^+ are both positive, we the value of ϕ_x according to Godunov's scheme is ϕ_x^-. In other words, we know that the direction to the fluid's surface is towards the positive x direction. So similarly to upwinding, we take the derivative in the direction *opposite* to the flow of information (since we want information about ϕ to flow from the surface of the fluid away from it) - in this case ϕ_x^-.

Similarly, now consider a point inside the fluid with both ϕ_x^- and ϕ_x^+ less than 0. In this case, the value of ϕ_x according to Godunov's scheme is ϕ_x^+ since we know that the direction to the fluid's surface is towards the negative x direction.

Now again consider a point inside the fluid in which $\phi_x^- > 0$ and $\phi_x^+ < 0$. We know on the side towards the negative x direction that $\phi_x > 0$, so the boundary is towards the positive x direction. And on the side towards the positive x direction, we have $\phi_x < 0$, so the boundary is towards the negative x direction. This implies that the boundary is somewhere in between the point to the left of our center point, and the point to the right of our center point. Knowing that, we let ϕ_x be set to either ϕ_x^- or ϕ_x^+, whichever has the largest magnitude. Since this magnitude measures rate of increase *towards* the surface, since we want to sample our derivative *opposite* to the flow of information (again, away from the surface).

Similar results can be seen with $\phi > 0$.

So Godunov's scheme produces the best directional derivatives (which is to say derivatives which facilitate flow from the surface away from it), even when the surface is somewhere in between some discrete points neighbors (or more simply put, when a point is very close to the surface).

Once we have our $\nabla\phi$, we compute the temporal derivative of ϕ and repeatedly apply it to our existing ϕ with forward Euler steps until the average $\|\nabla\phi\|$ is within a certain tolerance of 1. We compute ϕ_t with

$$\phi_t = -\Delta t S(\phi_0)(\|\nabla\phi - 1\|) \tag{26}$$

To break down equation 26, we see that change in ϕ at each iteration is directly proportional to the timestep (in this case we set the timestep to $0.1 \cdot h$, a value which produces stable results that I arrived at by experimentation).

The other components of the equation: $-S(\phi_0)(\|\nabla\phi\| - 1)$ are dependent on both the sign of ϕ_0 and whether the gradient norm is greater or less than 1.

First consider the case where $\phi_0 > 0$ and $\|\nabla\phi\| > 1$. Then ϕ_t will be negative. Then consider the case where $\phi_0 > 0$ and $\|\nabla\phi\| < 1$, then ϕ_t will be positive. In other words, if our ϕ_0 is positive, then we will want to decrease ϕ if our gradient norm is too large, and increase ϕ if our gradient norm is too small. Since ϕ_0 is positive, a negative ϕ_t reduces its magnitude in order to correct for a gradient norm that is too large. And a positive ϕ_t will correct for a gradient norm which is too small.

Similarly, with a negative ϕ_0, we have the opposite. When our gradient norm is too large, we will have a positive ϕ_t, which reduces the magnitude of ϕ. And when our gradient norm is too small, we will have a negative ϕ_t, which increases the magnitude of ϕ, thereby increasing our gradient norm.

Additionally, the change at each step is directly proportion to $-(\|\nabla\phi\| - 1)$. In other words, the closer the gradient norm that is to 1 (our desired value) the less we change it. If $\|\nabla\phi\|$ is greater than 1, then we will This serves to not needlessly change "correct" values, as well as to more strongly change "incorrect" values.

After we reinitialize our level set to be a signed distance function, our interface may have moved due to numerical errors. So we correct our levelset with the marker particles once again.

4.6 Particle Level Set Method - Adjust Particle Radii

This step simply sets the new particle radius to the local ϕ value of its current position, clamped between $0.1\Delta x$ and $0.5\Delta x$, as per (Enright et al. 2002).

4.7 Projection of ϕ

We project ϕ after advecting and correcting, and reinitializing our level sets, but before any dynamics calculations. So essentially, we perform the Particle Level Set Method on each fluid to move them each individually, fix points where fluids overlap or create vacuums (in the case where air is treated as a low density fluid), and then continue with the simulation with a set of correctly placed fluid surfaces. This step serves to correct the numerical errors that occur when we independently evolve the level sets of each fluid, as they might contradict after we evolve them (Losasso et al. 2006).

Consider a number of fluids, each with a corresponding level set. If only one fluid has a negative ϕ value at a given point, we know that that ϕ value serves as a valid signed distance for all other fluids, as it measures the correct distance from its interface. Moreover, the second smallest ϕ value at that point is equal to the negative of the smallest value - since they two fluids share an interface. We want to project all of our ϕ values in order to obtain these properties

To correct the values of ϕ, we take the average of the two smallest ϕ values at a given point, and subtract it from all fluid ϕs. This has the property that the two smallest ϕ values are now negatives of each other, and all ϕ values but one are negative. This works both for overlaps (double or more negative ϕ values) and gaps (no negative ϕ values) with the exact same method.

5 Examples

These methods allow for the robust simulation of multiple interacting liquids of varying density. Some applications are pictured below.

5.1 Large air bubble immersed in water

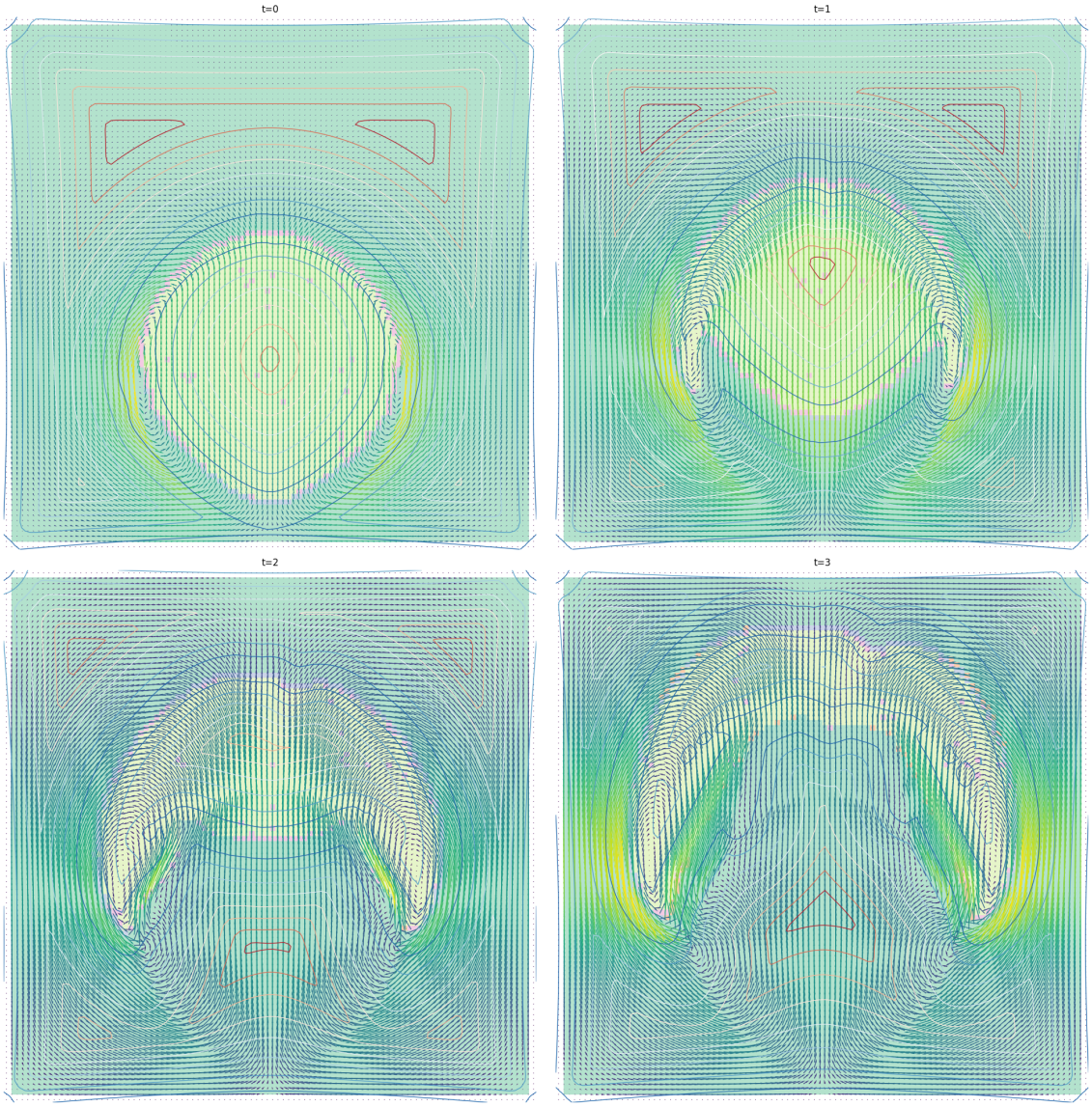


Figure 2: An air bubble flowing up through a container of water

5.2 Water above, oil below, dense fluid in center

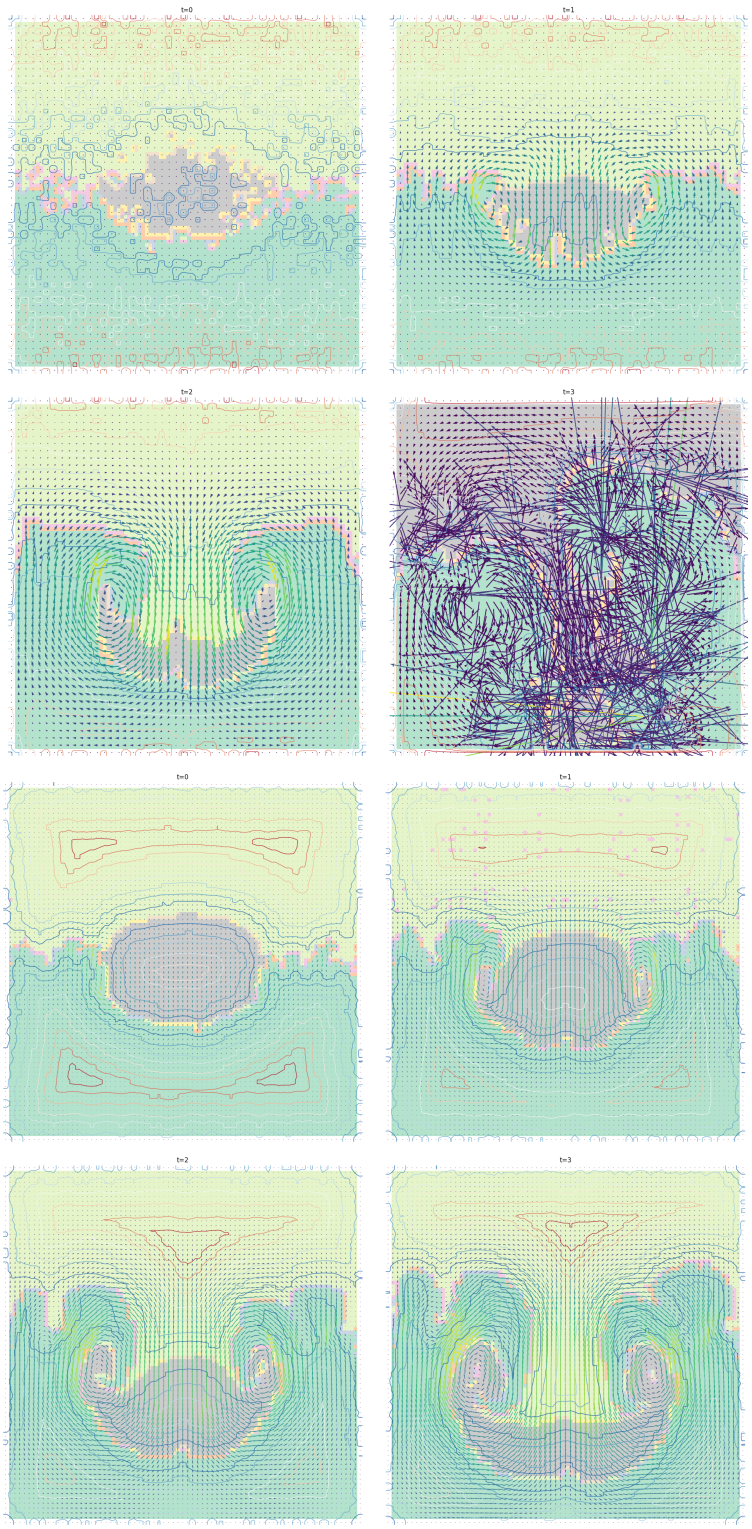


Figure 3: Top: without Particle Level Set Method correction
Bottom: with Particle Level Set Method correction
Note: the instability in the top image is due to excessive volume loss, a common weakness in level set methods

5.3 Rayleigh-Taylor instability

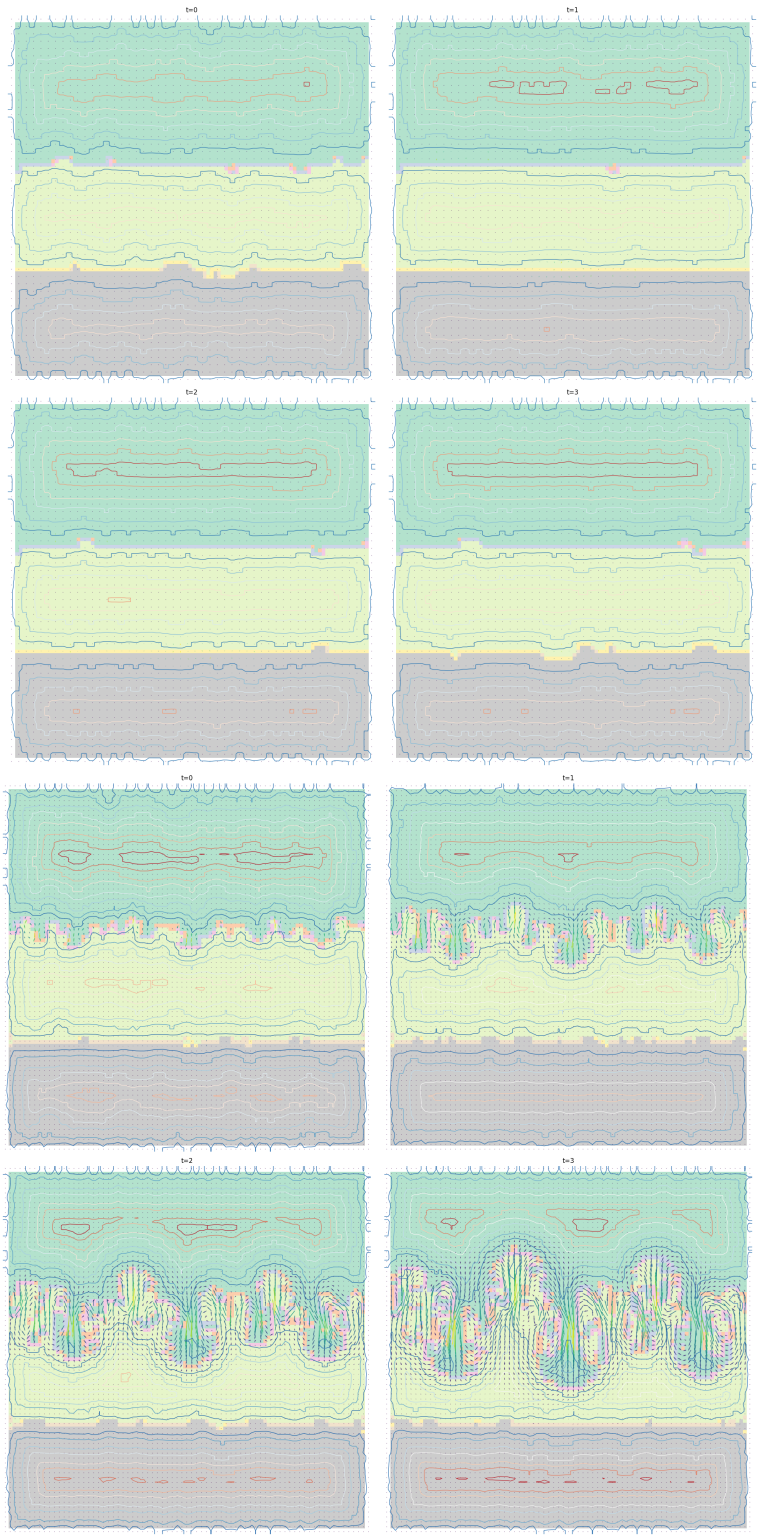


Figure 4: Top: stable interaction of fluids of increasing density from top to bottom
Bottom: Chaotic interactions of (top to bottom) a high density fluid, a low density fluid, and a medium density fluid

References

Enright, Douglas et al. (2002). "A Hybrid Particle Level Set Method for Improved Interface Capturing". In: *Journal of Computational Physics* 183.1, pp. 83–116. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2002.7166. URL: http://www.sciencedirect.com/science/article/pii/S0021999102971664.

Losasso, Frank et al. (2006). "Multiple Interacting Liquids". In: *ACM Trans. Graph.* 25.3, pp. 812–819. ISSN: 0730-0301. DOI: 10.1145/1141911.1141960. URL: http://doi.acm.org/10.1145/1141911.1141960.

Osher, S. and R. Fedkiw (2003). *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag.

Samuel, H and M Evonuk (2010). "Modeling advection in geophysical flows with particle level sets". In: *Geochemistry Geophysics Geosystems - GEOCHEM GEOPHYS GEOSYST* 11. DOI: 10.1029/2010GC003081.