MultiFLIP for Energetic Two-Phase Fluid Simulation

by

William Landon Boyd

B. Science, Dalhousie University, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

March 2011

Abstract

Physically-based liquid animations often ignore the influence of air, giving up interesting behaviour. We present a new method which treats both air and liquid as incompressible, more accurately reproducing the reality observed at scales relevant to computer animation.

The Fluid Implicit Particle (FLIP) method, already shown to effectively simulate incompressible fluids with low numerical dissipation, is extended to two-phase flow by associating a phase bit with each particle. The liquid surface is reproduced at each time step from the particle positions, which are adjusted to prevent mixing near the surface and to allow for accurate surface tension. The liquid surface is adjusted around small-scale features so they are represented in the grid-based pressure projection, while separate, loosely coupled velocity fields reduce unwanted influence between the phases.

The resulting scheme is easy to implement, requires little parameter tuning and is shown to reproduce lively two-phase fluid phenomena.

Preface

The entirety of this thesis has been submitted as a paper entitled “MultiFLIP for Energetic Two-Phase Fluid Simulation”. The authors listed on the paper are, in order, Landon Boyd and Robert Bridson.

The paper was written by Landon Boyd with minor revisions from Robert Bridson. The research was conducted by Landon Boyd, exploring and extending key ideas proposed by Robert Bridson: two velocity fields, particle bumping and level set adjustment for escaped particles. The MultiFLIP implementation was written by Landon Boyd as an extension to a single-phase fluid solver by Robert Bridson. The formula to estimate 3-D face fractions described in section 3.2.1 was contributed by Robert Bridson.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I was supported by a scholarship from Natural Sciences and Engineering Research Council of Canada. I am grateful to Christopher Batty, Tyson Brochu and Essex Edwards for many helpful discussions and suggestions. I would also like to thank my supervisor, Robert Bridson, and second reader Uri Ascher.

Chapter 1

Introduction

Computers have been used to simulate fluids for scientific and engineering applications since the advent of the electronic computer [9]. More recently, fluid simulation has become a valuable tool for movie production, producing realistic bodies of water [34], fire, smoke and more.

Traditionally, digital water effects have employed free-surface simulation in which the surrounding air is infinitely compressible and exerts no pressure on the liquid surface. Although plausible behaviour can be attained under this assumption, it forgoes the visually interesting phenomena arising from the interplay between water and air.

In reality, air trapped under water splashes form visible bubbles. This can be reproduced easily by simply pouring water quickly into a glass. Not only are these bubbles visually noticeable, they also contribute significantly to the sound of pouring water [26]. In a free surface simulation, these entrained bubbles collapse instantaneously, while in two-phase simulation, air can be treated as incompressible so the bubbles live on, contributing to visual (and potentially aural) realism.

Pouring water through a narrow spout is another everyday scenario where the influence of air is important. As we've all experienced when overturning a beverage container, water and air compete for space in the spout, resulting in the chaotic "glugging" shown in figure 1.1. In a free-surface simulation, the behaviour in this case would be as if the top of the container were open, allowing the water to flow in a smooth and boring fashion through the spout.

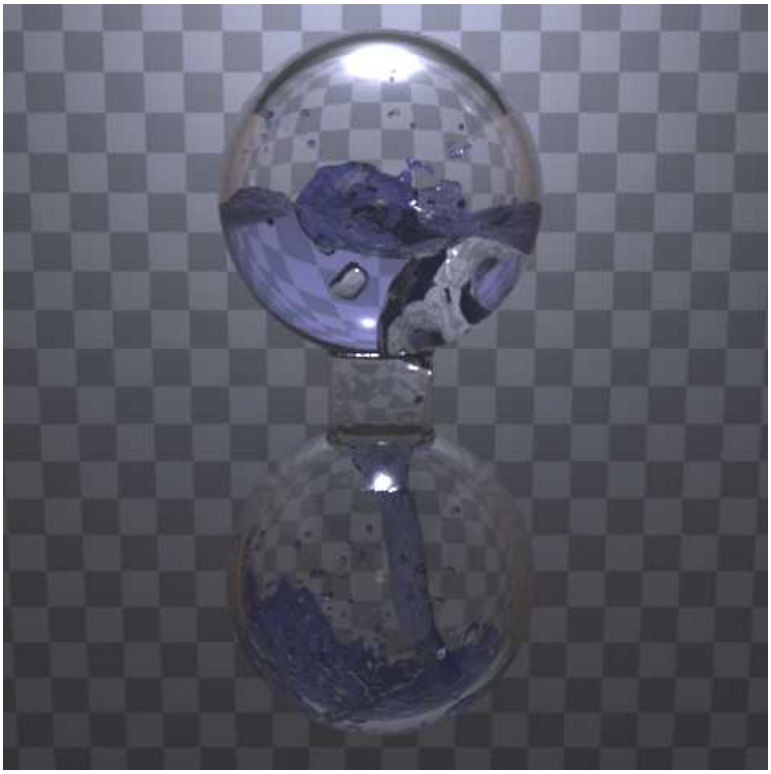


Figure 1.1: The "glugging" effect of water pouring through a spout cannot be reproduced with free-surface liquid simulation.

The goal of this work is to develop a two-phase fluid simulation method with the following properties:

Low numerical viscosity. For computer graphics, it is vital that energy be conserved as much as possible. For the gas phase, this allows for small vortices that can lend interesting behaviour to smoke, steam or mist. For the liquid phase, it avoids behaviour that viewers may characterize as too "goopy". At the interface between liquid and gas, it avoids an artificial boundary layer that would result in unwanted coupling between the two phases.

Plausible small-scale droplets and bubbles. In energetic fluid scenarios, it is inevitable that small droplets will pinch off splashes and small bubbles will be captured under the surface. Numerical simulations, particularly those

that partition the simulation domain into a grid, often lose these details or fail to reproduce their natural behaviour. We look for an elegant solution to this problem.

A practical implementation. The method should be easy to implement as an extension to an existing single phase fluid solver, and should not add unreasonable computational cost or require excessive parameter tuning.

Although our solution should be applicable to any two fluids with different densities, we imagine the most common use case to be some liquid interacting with air. Therefore, this paper will henceforth refer to the two fluids as "liquid" and "air".

The outcome of this work we call the MultiFLIP method. It is an extension of the Fluid Implicit Particle (FLIP) method [4] introduced to graphics and adapted to incompressible flow by Zhu and Bridson [41], which has gained popularity due to its excellent conservation of energy without the spurious oscillations associated with similarly undissipative central difference schemes. To our knowledge, MultiFLIP is the first application of FLIP to incompressible two-phase flow.

To evolve the liquid surface over time, we develop a new particle-based surface tracking scheme which identifies sub-scale bubbles and droplets. As opposed to previous methods which treat sub-scale particles with a separate Lagrangian model, MultiFLIP incorporates them into the grid-based fluid solve so they are subject to the same physical parameters as the rest of the simulation.

Finally, to avoid numerical viscosity at the liquid-air interface, MultiFLIP for the first time combines separate liquid and air velocity fields with a straightforward finite-volume treatment of divergence.

Chapter 2

Previous Work

Over the last decade, two-phase and multiphase fluid simulation has received attention from both the computational physics and computer graphics communities. The Ghost Fluid Method (GFM) [13, 18] paved the way for much of this work with a convenient discretization of discontinuities in quantities such as pressure at the fluid-fluid interface.

Hong and Kim [12] adapted GFM for computer graphics, employing semi-Lagrangian advection and Particle Level Set (PLS) for surface tracking. Losasso et al. [20] extended this scheme to multiphase flows by merging several overlapping level set functions. Kang et al. [14] added the ability to simulate both miscible and immiscible fluid interactions.

Song et al. [33] took a different approach, using a continuous model of the fluid properties at the interface but with higher-order constrained interpolation profile (CIP) advection. Zheng et al. [40] added Regional Level Set (RLS) surface tracking to simulate bubbles with thin films. Volume control for individual regions was added by Kim et al. [16]. Kim [15] improved the RLS formulation and added a Lagrangian model for small bubbles and droplets.

Sussman et al. [36] introduced another model for two-phase flow using Coupled Level Set and Volume of Fluid (CLSVOF) surface tracking. An interesting feature of this work is its use of separate velocity fields for the two fluid phases. Mihalef et al. [24] adapted this model to simulate boiling for computer graphics. Mihalef et al. [25] then replaced CLSVOF with Marker Level Set to simulate

two-phase flow with Lagrangian particles for small-scale droplets and bubbles.

Other works have focused specifically on reproducing lively behaviour of small scale bubbles [11] and liquid sprays [21].

All of the above methods advect a level set function representing the surface, perhaps correcting it using either particles or fluid volumes. MultiFLIP skips level set advection, instead reconstructing the surface from advected particles. Blinn [3] presented an influential early effort to constructing surfaces around particles and Zhu and Bridson [41] adapted the concept for single-phase fluid simulation.

Another difference of MultiFLIP from previous methods is its treatment of sub-grid bubbles and droplets. Rather than simulating these with a separate Lagrangian model, the liquid surface is slightly perturbed so that those features are visible on the grid for the pressure projection. This is similar in spirit to the approach of Kim et al. [17], wherein fine features were sampled onto a coarse grid by expanding the level set.

In parallel to the developments described above, some authors have attempted to implement multiphase fluids in a smoothed particle hydrodynamics (SPH) framework. We refer the reader to Solenthaler and Pajarola [32].

Chapter 3

The Method

The MultiFLIP method is based on the Euler equations for inviscid, incompressible fluid flow,

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \vec{f} \tag{3.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{3.2}$$

where $\vec{u}$ is velocity — we will often refer to its horizontal and vertical components, u and v, respectively; p is pressure; ρ is density and $\vec{f}$ encapsulates body forces such as gravity. For a complete introduction to these equations and their use in computer graphics, see Bridson [5].

Using the time splitting technique introduced to graphics by Stam [35], the Euler equations can be solved in a sequence of steps: an *advection* step to advance the velocity (and potentially other quantities such as smoke density) through the velocity field,

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = 0; \tag{3.3}$$

a step to apply body forces such as gravity,

$$\frac{\partial \vec{u}}{\partial t} = \frac{1}{\rho} \vec{f}; \tag{3.4}$$

and a *projection* step to enforce the incompressibility condition,

$$\frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho}\nabla p \tag{3.5}$$

$$\nabla \cdot \vec{u} = 0. \tag{3.6}$$

The MultiFLIP method, like FLIP [41], stores velocity samples on a staggered Marker and Cell grid [10]. While the body forces and pressure projection are applied to the grid velocities, randomly distributed particles carry velocity for the advection step.

In reality, the viscosity of air and water imposes a boundary layer between the two phases where their tangential velocities are coupled. However, this boundary layer is typically too small to be represented at the coarse grid resolutions used for computer graphics. Therefore, we treat the interface between water and air as a free-slip condition. To this end, MultiFLIP uses two velocity fields, $\vec{u}^L$ for liquid and $\vec{u}^A$ for air, so that there are two colocated velocity samples on each grid face. The two velocity fields contribute to a combined divergence that can be corrected using a single pressure projection.

Additionally, each particle is labeled as either liquid or air. During the advection step, liquid (air) particles only interpolate velocities to and from $\vec{u}^L$ ($\vec{u}^A$). The particle identities also allow the liquid-air surface to be reconstructed from the particle positions.

A MultiFLIP time step consists of the following sub-tasks, described in subsequent sections.

1. Advect velocities (§3.3)
2. Reconstruct the liquid surface (§3.1)
3. Bump and seed particles (§3.1.1)
4. Compute the liquid volume adjustment (§3.5)
5. Adjust the liquid surface around escaped particles (§3.1.2)
6. Apply body forces to both velocity fields
7. Project out the combined liquid/air divergence (§3.2)

3.1 Particle-Derived Level Set

An important aspect of simulating two-phase flow is tracking the interface between the two fluids, as the discontinuities in density and pressure must be accounted for along that interface. Many methods have been proposed to track fluid interfaces. Level Set Methods [27] define the interface as the zero level set of a scalar field, ϕ, stored on the grid. ϕ can be advected through the velocity field to track its evolution over time. A challenge of these methods is handling numerical dissipation which tends to smooth out features, shrinking high curvature regions over time.

The Particle Level Set (PLS) method [6] adds a band of particles on either side of the zero level set. By advancing both the particles and ϕ through the flow, more surface details are preserved.

Since FLIP requires particles to be maintained over the fluid domain, and since each MultiFLIP particle is identified as either liquid or air, it is natural to reconstruct the liquid surface from the particles.

Similarly to Foster and Fedkiw [8], we construct a distance function, ϕ as the union of spheres around particles. Whereas their method and its descendant PLS also advance ϕ over time to maintain a smooth interface, we avoid that added complexity and cost. Instead ϕ is reconstructed from particles at every time step, adjusting the positions and distribution of the particles near the interface to maintain smoothness.

Our task is then, given a distribution of liquid and air particles, to generate a scalar field ϕ which is less than zero in the region covered mostly by liquid particles and greater than zero in the region covered mostly by air particles (figure 3.1). It is important that this reconstruction be robust to intermingling of particles at the interface and gaps in the particle distribution, as these are inevitable side-effects of numerical integration over a time step.

The first step in our surface reconstruction algorithm is to construct two signed distance fields, ϕ_A and ϕ_L based on the union of spheres of radius r_s around air and liquid particles, respectively. A larger r_s results in a smoother surface around the particles but also smooths over finer features. [1] Figure 3.2 shows a 1-D example with $r_s = 0.4\Delta x$.

[1] In all tests, we used $r_s = 0.36\Delta x$.

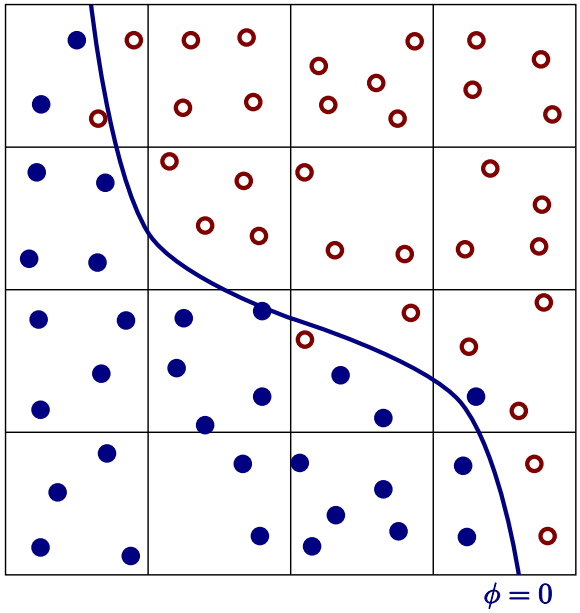


Figure 3.1: The liquid surface is reconstructed from the locations of air and water particles.

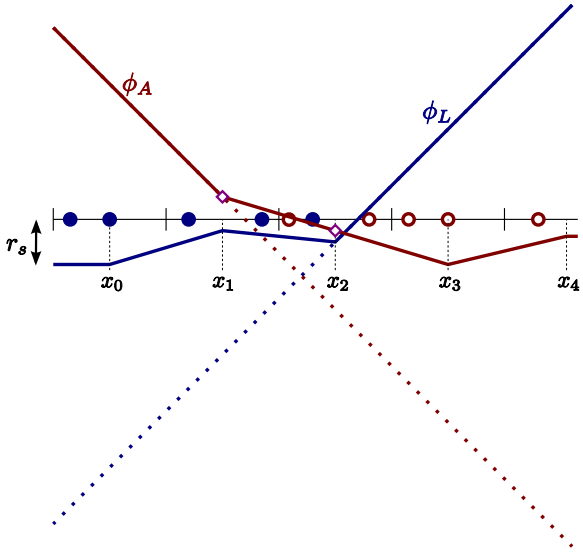


Figure 3.2: First, separate distance fields are constructed for liquid and air.

ϕ_A and ϕ_L are constructed as the distance to the nearest particle at every grid cell center minus r_s. As seen in figure 3.2, inside the air region ϕ_A hovers between 0 and $-r_s$, depending on the distribution of particles. For example, at x_2, the nearest air particle is $0.3\Delta x$ away, so $\phi_A = 0.3\Delta x - r_s = -0.1\Delta x$. Importantly, the value of ϕ_A here does not say anything about the extent of the air region.

For this reason, all negative values of ϕ_L and ϕ_A are discarded and instead extended from the positive ϕ values using the fast sweeping method of Zhao [39]. These reinitialized distance functions are shown as dashed lines in figure 3.2. Note that the reinitialized ϕ_A crosses zero in a new location which reflects a surface around the air particles.

Given ϕ_L and ϕ_A signed distance fields, we construct a merged ϕ (figure 3.3) via

$$\phi = \frac{\phi_L - \phi_A}{2}. \tag{3.7}$$

This is equivalent to the level set projection method proposed by Losasso et al. [20], which is robust to overlap and vacuum between surfaces.

A convenient side-effect of reconstructing the surface from both air and liquid particles is robustness to undersampling. A bubble can be distinguished from an undersampled liquid region by the presence of air particles.

3.1.1 Particle Bumping and Seeding

As previously mentioned, particles tend to mix at the liquid-air interface. Left unchecked, this mixing can quickly grow out of control, resulting in a homogeneous soup of liquid and air particles. To tackle this problem while maintaining a smooth surface, we strategically adjust the particle positions around the interface at every time step, correcting for the errors in numerical integration.

After constructing the signed distance function ϕ identifying the liquid-air interface, any particles that are less than r_s inside their respective regions are moved along the gradient of ϕ to the sr_s isocontour, where $s = +1$ for air or $s = -1$ for liquid particles.

Bumping to the zero isocontour is enough to prevent mixing between air and liquid particles but we found that too much noise developed on the surface to accurately simulate surface tension effects. By bumping to the $\pm r_s$ isocontour, the

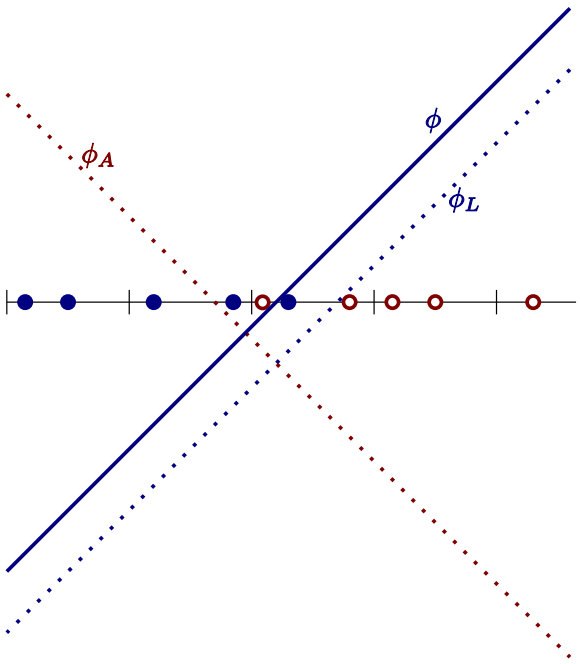


Figure 3.3: The liquid and air distance fields are merged. The new surface is halfway between the zero crossings of the individual fields.

union of spheres around the particles more closely approximate a smooth surface (figure 3.4). With additional particles seeded around around the interface, the approximation becomes even smoother.

The impact of bumping on accuracy should be minimal since the bump distance is generally below the order of one grid cell and because the particle velocity update in the advection step accounts for the change in position (§3.3).

In more detail, the bumping algorithm proceeds as follows.

1. Compute the curvature field, $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right)$, at grid cell centers using the second-order finite difference equation described by Osher and Fedkiw [27].

2. Identify candidate particles using the approximate signed distance to the interface, $d = \frac{\phi(x)}{|\nabla \phi(x)|}$, using fourth-order WENO interpolation [22] for $\phi(x)$ and $\nabla \phi(x)$. Candidates are those particles with $s\phi(x) \in (-r_s, +r_e)$. Particles with $s\phi(x) \geq r_e$ will be treated separately (§3.1.2).

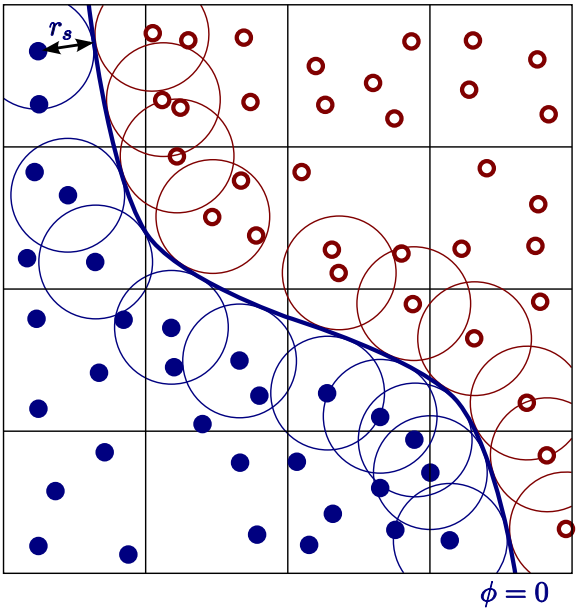


Figure 3.4: Particle bumping prevents mixing and results in a surface smooth enough for accurate surface tension.

3. Interpolate the curvature at each candidate particle location, $\kappa(x)$, using trilinear interpolation of κ. In regions where surface curvature approaches the maximum that can be represented on the grid, our bumping strategy changes to allow particles to move freely. When $-s\kappa(x) \in \left(\frac{1}{4\Delta x}, \frac{1}{2\Delta x}\right]$, the target distance inside the interface, *target*, is set to zero. When $-s\kappa(x) > \frac{1}{2\Delta x}$, the particle is not bumped at all. Figure 3.5 shows both of these scenarios. Otherwise, *target* is set to r_s. Without these extra cases, small splashes and tendrils are prevented from leaving the main fluid body.

4. Move the particle to the target isocontour. If $sd < 0$, the particle is on the wrong side of the interface. Find a vector $\vec{q}$ to the nearest point on the surface via Newton iterations with backtracking line search. Fourth-order WENO is again used for interpolation. Update the particle via

$$x = x - (target + |\vec{q}|)\frac{\vec{q}}{|\vec{q}|}. \tag{3.8}$$

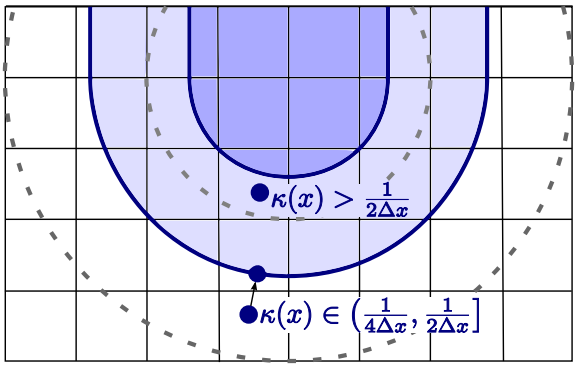


Figure 3.5: Bumping is modified in high curvature regions to enable splashing.

If $sd \in [0, target)$, the particle is between the interface and the target isocontour. Likewise, find $\vec{q}$ pointing to the nearest point on the surface. Update the particle position via

$$x = x + (target - |\vec{q}|)\frac{\vec{q}}{|\vec{q}|}. \tag{3.9}$$

After the bumping step, liquid and air particles are seeded and culled as necessary throughout the domain. In each cell, the number of liquid and air particles is counted. If the total number of particles in a cell is below the target of 8, particles are added to the cell at random locations, acquiring their identities and velocities from values of ϕ and $\vec{u}$ interpolated from the grid. If a location is randomly chosen within the r_s void around the liquid-air interface, no particle is created.

To improve evolution of the liquid-air interface, the target number of particles is increased for cells close to the surface. We increase the target number of particles by a factor of 4 in cell (i, j, k) when $\phi_{i,j,k} \leq \Delta x$.

If the number of air or liquid particles in a cell exceeds twice the target number, excess particles are deleted. Liquid and air particles are counted separately to prevent volume loss from removing all particles of a particular type.

3.1.2 Escaped Particle Handling

If a particle is too far away from the reconstructed liquid surface, it probably represents a feature too fine to be represented on the grid. Other papers [21, 25] have used this observation to identify sub-scale features in the context of the Marker Level Set and Particle Level Set methods.

Whereas other schemes have treated these "escaped" particles in a Lagrangian fashion, we propose to instead adjust the liquid surface around the particles so that they are represented in the grid-based pressure solve. This way, large fluid bodies and sub-grid features are subject to the same dynamics with no need to separately tweak parameters for both cases. Implementation is also relatively straightforward.

In our surface reconstruction scheme, an isolated particle may show up as a small "bump" in ϕ across the zero isosurface, depending on the density of surrounding particles and alignment with the grid. In order to treat all isolated particles equally, we first apply a simple filter to remove these bumps. Anywhere that the sign of ϕ differs from all its neighbors, i.e.

$$\text{sign}(\phi_{i,j,k}) \neq \text{sign}(\phi_{i',j',k'}) \; \forall \; (i',j',k') \neq (i,j,k) \\ i' \in [i-1,i+1], j' \in [j-1,j+1], k' \in [k-1,k+1],$$

the sign of $\phi_{i,j,k}$ is flipped. The resultant ϕ field is then redistanced.

Escaped particles are identified as those on the wrong side of the interface by more than r_e (or $\frac{r_e}{2}$ where $\kappa(x) > \frac{1}{2\Delta x}$) after the bumping step. ϕ is adjusted around these particles to create a spherical bubble or droplet of radius r_e (figure 3.6).

For all grid centers within $r_e + 2\Delta x$, ϕ is updated with

$$\phi_{i,j,k} = s \cdot \max(s\phi_{i,j,k}, r_e - d) \tag{3.10}$$

where d is the distance to the particle and s is the sign of the particle (-1 for liquid, $+1$ for air). Note that since the surface is reconstructed from particles at every time step, this adjustment of ϕ does not result in persistent volume change over time. When an escaped particle comes within r_e of the liquid surface, it is re-absorbed into the fluid body.

When ϕ is adjusted around an escaped particle, particles of the opposite type

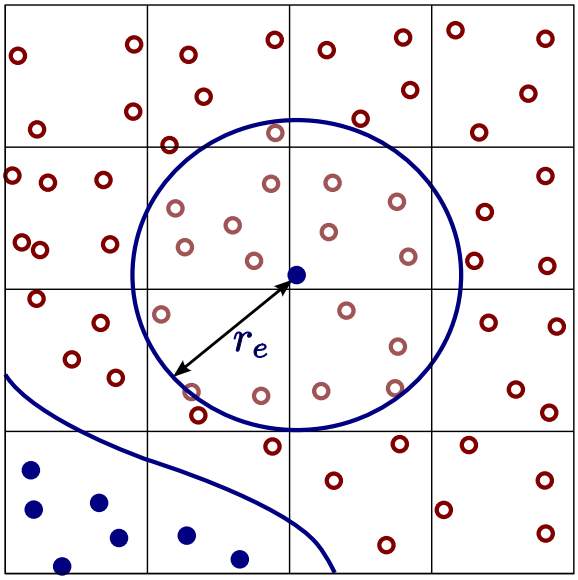


Figure 3.6: The liquid surface is adjusted around an escaped particle.

get covered by the expanded region. Consider an air particle covered by the expanded liquid region around an escaped liquid particle. We passively advect the covered air particle through the air velocity field. Since the pressure projection does not account for the existence of air at its location, we do not use it to exchange velocity information with the grid after the projection step. The velocity extrapolation strategy (§3.4) also helps reduce the influence of the surrounding velocity field on escaped particles.

r_e is chosen large enough for particles to be represented on the grid, but not so large as to introduce artifacts such as piles of particles that never merge. We used $r_e = 1.5\Delta x$ for our 2-D tests and $r_e = 1.1\Delta x$ for 3-D. Escaped particles can be rendered as spheres of any radius, but it is best for the rendered spheres not to differ too much in size from r_e or they will appear to float against boundaries.

3.2 Combined Divergence-Free Projection

3.2.1 Combined Divergence Measure

Given a divergent velocity field, $\vec{u}^*$, we obtain a new divergence-free field, $\vec{u}^n$ as follows. Assuming a pressure field p, advancing equation 3.5 over a time step gives

$$\vec{u}^n = \vec{u}^* - \Delta t \frac{1}{\rho} \nabla p. \tag{3.11}$$

Taking the divergence of both sides with $\nabla \cdot \vec{u}^n = 0$ gives

$$\Delta t \nabla \cdot \left(\frac{1}{\rho} \nabla p \right) = \nabla \cdot \vec{u}^*. \tag{3.12}$$

Solving equation 3.12 for p and substituting it into equation 3.11 obtains the divergence-free $\vec{u}^n$.

The right hand side of equation 3.12 corresponds to the net amount of fluid flowing into or out of each cell. The pressure projection then counteracts this flow so that no cell is a source or sink for fluid.

For the purpose of divergence, liquid and air are treated equally since both are incompressible. The total divergence for a cell is the sum of the fluxes of each fluid across each face. For example, the liquid flux across a face is the area of the face covered by liquid multiplied by the liquid velocity on that face.

Figure 3.7 shows the liquid and air velocity samples and face fractions on a cell containing air and liquid, where the tangential velocities of the two fluids are discontinuous.

In MultiFLIP, the net flux across each face is computed as a weighted sum of the liquid and air velocities (figure 3.8), similarly to the scheme used by Roble et al. [30] for solid boundaries. In 2-D, this can be expressed as

$$\begin{aligned} \nabla \cdot \vec{u}_{i,j} = & - u^L_{i,j} ufrac_{i,j} - u^A_{i,j} (1 - ufrac_{i,j}) \\ & + u^L_{i+1,j} ufrac_{i+1,j} + u^A_{i+1,j} (1 - ufrac_{i+1,j}) \\ & - v^L_{i,j} vfrac_{i,j} - v^A_{i,j} (1 - vfrac_{i,j}) \\ & + v^L_{i,j+1} vfrac_{i,j+1} + v^A_{i,j+1} (1 - vfrac_{i,j+1}) \end{aligned} \tag{3.13}$$

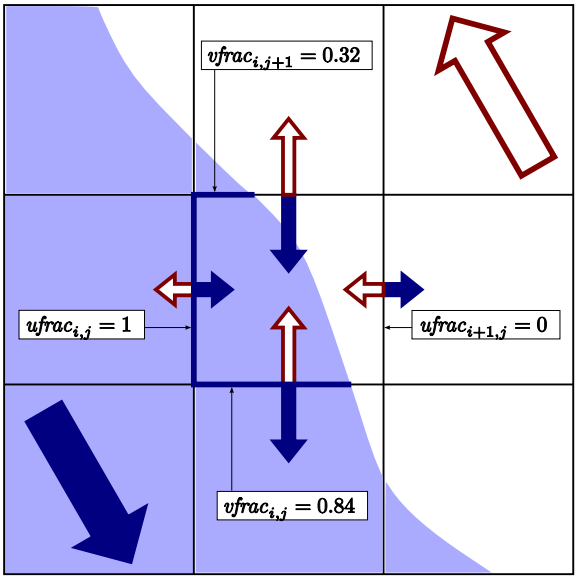


Figure 3.7: Separate liquid (blue) and air (red) velocities are colocated on the faces of cell (i,j).

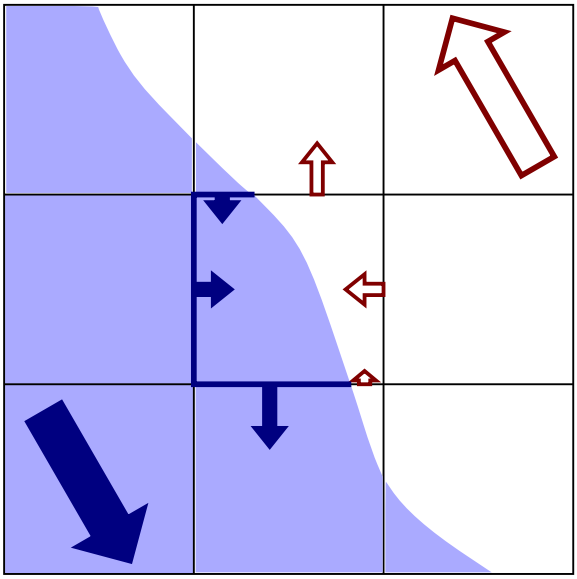


Figure 3.8: The combined divergence is a weighted combination of liquid and air velocities.

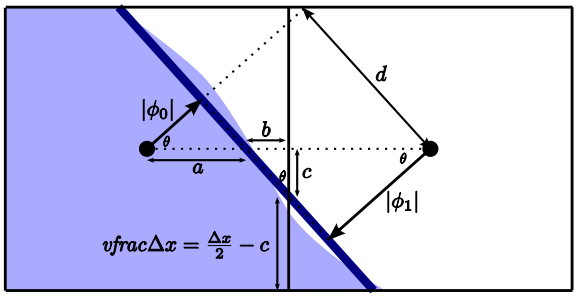


Figure 3.9: The face fraction is approximated using adjacent ϕ values. Here, ϕ_0 and ϕ_1 have different signs.

where *ufrac* and *vfrac* are the fractions of the vertical and horizontal cell faces covered in liquid.

There are several possible ways to approximate *ufrac* and *vfrac*. For example, in 2-D one can interpolate ϕ onto the corners of each cell. Then, each face has two ϕ values at its endpoints. If these values are of opposite sign, the fraction is simply $\frac{\phi^-}{\phi^- - \phi^+}$. Otherwise, the fraction is 1 or 0, depending on the sign.

Unfortunately, this method does not extend trivially to 3-D, where each face is a square rather than a line segment. Instead, we use a geometric approach that considers only the two ϕ values adjacent to the face. By assuming ϕ describes the distance to a plane parallel to two of the face's edges, the third dimension can be ignored and we only need to compute the fraction of a 1-D face under a 1-D surface, as shown in figures 3.9 and 3.10.

When two adjacent ϕ values have different signs, as for ϕ_0 and ϕ_1 in figure 3.9, the identities

$$\begin{aligned} a + b &= \Delta x/2 \\ \frac{c}{b} &= \frac{\phi_1 - \phi_0}{d} \\ \frac{a}{\phi_0} &= \frac{\Delta x}{\phi_1 - \phi_0} \end{aligned}$$

give

$$c = \Delta x \left(\frac{\phi_0 + \phi_1}{2d} \right).$$

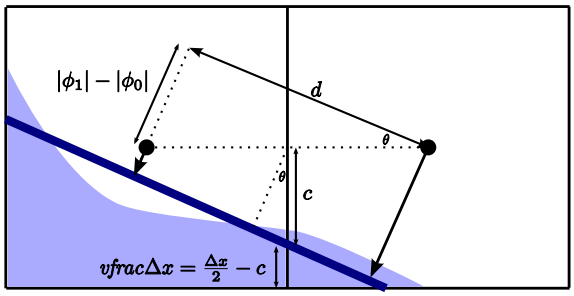


Figure 3.10: Here, a face fraction is approximated from ϕ_0 and ϕ_1 with the same sign.

Then

$$vfrac = \frac{1}{2} - \frac{\phi_0 + \phi_1}{2d}, \tag{3.14}$$

where

$$d = \sqrt{\Delta x^2 - (\phi_1 - \phi_0)^2}.$$

We clamp *vfrac* within $[0, 1]$.

When ϕ_0 and ϕ_1 have the same sign and the interface crosses above or below ϕ_0 and ϕ_1, as in figure 3.10, the face fraction can again be derived from similar triangles. Conveniently, this also gives equation 3.14.

3.2.2 Pressure Projection

At the liquid-air interface, there is a sharp jump in density, pressure gradient and, in the presence of surface tension, pressure. In order to capture these discontinuities in the pressure projection, we use the Ghost Fluid Method [7].

Our goal is to solve a variable coefficient Poisson problem, equation 3.12. Liu et al. [18] show how the Ghost Fluid Method can accommodate jumps in p, ∇p and ρ while still resulting in the usual Laplacian stencil for the matrix on the left hand side. It is illustrative to repeat some of that discussion here in 1-D.

Figure 3.11 shows the pressure across the liquid-air interface with surface tension. The pressure sample p_k at position x_k is in a liquid region while the pressure sample p_{k+1} is in air. The exact interface between the two fluids is at position $x_k + \theta\Delta x$. We say that the pressure at that position is p_I while moving an infinitesi-

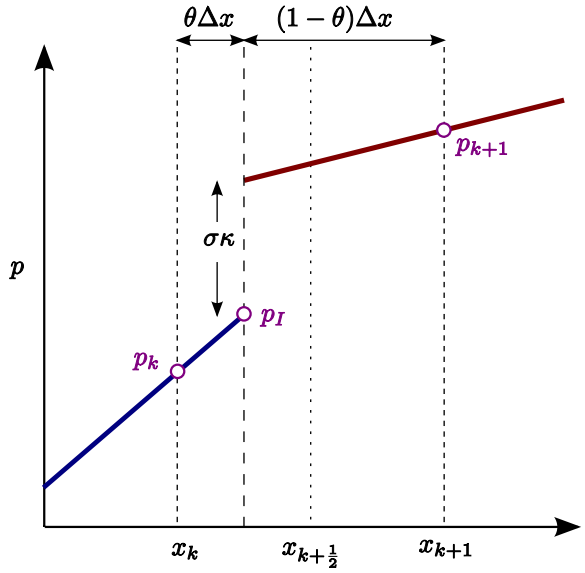


Figure 3.11: Surface tension causes a jump in pressure at the liquid-air interface.

mal amount to the right into the air region, the pressure jumps to $p_I + \sigma\kappa$. σ is the surface tension coefficient for the interface and κ is the curvature of the interface at this position.[2]

To obtain second-order finite differences for $\nabla \cdot \left(\frac{1}{\rho}\nabla p\right)$ at cell center x_k, we will need the gradients across the cell's two faces, $x_{k+\frac{1}{2}}$ and $x_{k-\frac{1}{2}}$.

The gradient at $x_{k-\frac{1}{2}}$ can be determined in the usual fashion since there is no interface between x_{k-1} and x_k.

$$\left(\frac{1}{\rho}\nabla p\right)_{k-\frac{1}{2}} = \frac{1}{\rho_L}\frac{p_k - p_{k-1}}{\Delta x}. \tag{3.15}$$

At the interface between x_k and x_{k+1}, the pressure is discontinuous but the variable density gradient, $\frac{1}{\rho}\nabla p$ is continuous. Assuming that $\frac{1}{\rho}\nabla p$ is the same for water and air within this cell allows $\frac{p}{\rho}$ to be extended to ghost values, $(\frac{p}{\rho})^G$, as shown in figure 3.12.

[2] κ is only meaningful in higher dimensions. The actual magnitude of the jump is unimportant for this discussion.

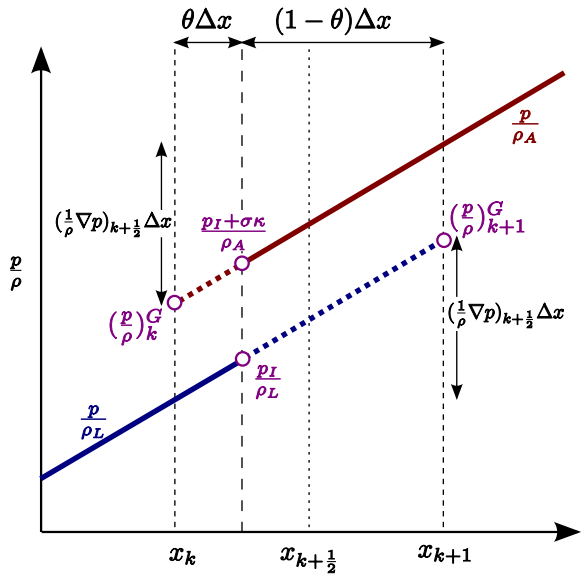


Figure 3.12: GFM uses a linear extension of p/ρ across interface.

The ghost values themselves are not as interesting as the gradient across the cell. To obtain this gradient, first observe that

$$\frac{1}{\rho_L}\frac{p_I - p_k}{\theta\Delta x} = \frac{1}{\rho_A}\frac{p_{k+1} - (p_I + \sigma\kappa)}{(1-\theta)\Delta x}. \tag{3.16}$$

This can be rearranged to obtain

$$p_I = \frac{\rho_L(p_{k+1} - \sigma\kappa)\theta + \rho_A p_k(1-\theta)}{\rho_L\theta + \rho_A(1-\theta)}. \tag{3.17}$$

Substituting p_I back into either side of equation 3.16 gives

$$\left(\frac{1}{\rho}\nabla p\right)_{k+\frac{1}{2}} = \frac{1}{\hat{\rho}}\frac{p_{k+1} - p_k - \sigma\kappa}{\Delta x} \tag{3.18}$$

where

$$\hat{\rho} = \rho_L\theta + \rho_A(1-\theta). \tag{3.19}$$

Returning to the Poisson problem, the left side of equation 3.12 can be dis-

cretized at x_k using equations 3.15 and 3.18.

$$\Delta t \frac{\frac{1}{\hat{\rho}} \frac{p_{k+1}-p_k-\sigma\kappa}{\Delta x} - \frac{1}{\rho_L} \frac{p_k-p_{k-1}}{\Delta x}}{\Delta x} = \nabla \cdot \vec{u}^* \tag{3.20}$$

Moving the $\sigma\kappa$ term to the right hand side reveals the usual Poisson sparsity structure, making it convenient to solve using a standard Preconditioned Conjugate Gradient solver.

$$\frac{\Delta t}{\hat{\rho}} \frac{p_{k+1}-p_k}{\Delta x^2} - \frac{\Delta t}{\rho_L} \frac{p_k-p_{k-1}}{\Delta x^2} = \nabla \cdot \vec{u}^* + \frac{\Delta t}{\hat{\rho}} \frac{\sigma\kappa}{\Delta x^2} \tag{3.21}$$

This discussion extends naturally to higher dimensions [18]. Forming the variable coefficient system is simply a matter of substituting $\hat{\rho}$ on the left hand side and adding a surface tension term to the right hand side, wherever the liquid-air interface passes between two pressure samples.

Once pressure values have been obtained, both velocity fields are updated according to equation 3.11 so that $\nabla \cdot \vec{u}^n = 0$. Where a liquid-air interface exists between pressure samples, it's also necessary to use the averaged density and include the pressure jump term

$$\vec{u}^n = \vec{u}^* - \frac{\Delta t}{\hat{\rho}} \nabla p + \frac{\Delta t}{\hat{\rho}} \frac{\sigma\kappa}{\Delta x}. \tag{3.22}$$

The liquid (air) velocity samples are only updated from pressures where the liquid (air) velocity has influenced the divergence term, i.e. on faces whose liquid (air) fraction is nonzero. Elsewhere, the velocity is extrapolated as described in section 3.4.

3.3 Two-Phase Velocity Advection

Velocity advection in MultiFLIP is similar to FLIP [41] with minor adjustments to deal with two velocity fields and particle bumping. Particles only interact with the velocity field corresponding to their identity. Liquid (air) particles only interpolate to and from $\vec{u}^L$ ($\vec{u}^A$) and are advected through $\vec{u}^L$ ($\vec{u}^A$). This is important near the liquid-air interface as in figure 3.13 to reduce smearing of the velocities across the interface.

At the beginning of a time step, each FLIP particle is advected through the

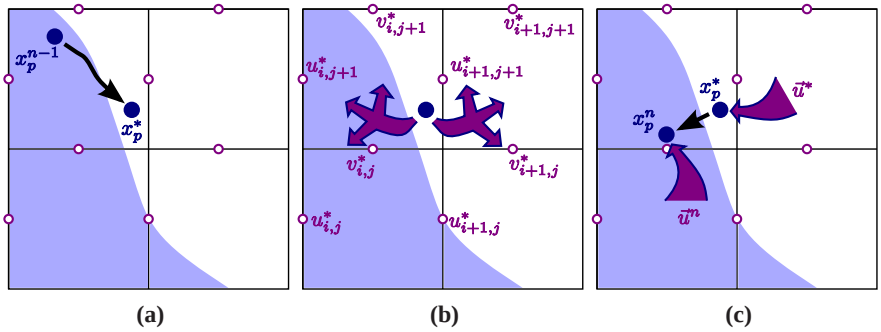


Figure 3.13: The MultiFLIP update: advection, interpolation onto the grid, interpolation onto particles.

divergence-free velocity field, u^{n-1} from the previous pressure projection (figure 3.13 (a)). We use a third-order Runge-Kutta scheme [28] for this step.

The particle velocities are then interpolated onto the grid at the new location to get $\vec{u}^*$ (figure 3.13 (b)), which may have nonzero divergence. If more than one particle updates a particular velocity sample, the particle velocities are weighted by the distances of the particles to the sample.

If a velocity sample is not updated by any particles, it is marked for extrapolation (§3.4).

The MultiFLIP time step proceeds, applying body forces and performing the pressure projection, resulting in a new, divergence-free velocity field, $\vec{u}^n$. Also during the time step, particles close to the liquid-air interface are bumped. In the case of figure 3.13, the particle at x_p^* is bumped to x_p^n.

The central idea of FLIP is to *increment* the particle velocity by the *difference* in the grid velocity interpolated at the particle position over the time step. The FLIP update to the particle is therefore

$$\vec{u}_{FLIP}^n = \vec{u}_p^{n-1} + (\vec{u}_{Interp}^n - \vec{u}_{Interp}^*). \tag{3.23}$$

To alleviate noise inherent to FLIP, the FLIP update is combined with a PIC update which considers only the current velocity field. The strength of this regularization

term is controlled by the parameter α. [3]

$$\vec{u}_{PIC}^{n} = \vec{u}_{Interp}^{n} \tag{3.24}$$

$$\vec{u}_{p}^{n} = \alpha\vec{u}_{PIC}^{n} + (1-\alpha)\vec{u}_{FLIP}^{n} \tag{3.25}$$

$\vec{u}_{Interp}^{*}$ is interpolated at position x_p^*, while $\vec{u}_{Interp}^{n}$ uses position x_p^n so that $\vec{u}_{Interp}^{n} - \vec{u}_{Interp}^{*}$ accounts for changes to the velocity field around the particle as well as the bumping of the particle over the time step (figure 3.13 (c)).

3.4 Velocity Extrapolation

When updating particles near the liquid-air interface from the grid, the stencil may include velocity samples outside the particle's region. In free-surface simulation, the velocities for these samples are typically populated via extrapolation to avoid dissipation near the interface.

Because MultiFLIP has separate velocity fields for liquid and air, the same procedure can be followed. Any velocity sample that was not influenced by the pressure projection can obtain an extrapolated value. Likewise, when updating the grid from particle values, samples not influenced by any particles can be extrapolated.

Like the redistancing procedure for ϕ, velocity extrapolation is based on fast sweeping [39]. The distance to the nearest known velocity sample is propagated in eight sweeps over the grid (one for each combination of $\Delta i = \pm 1$, $\Delta j = \pm 1$ and $\Delta k = \pm 1$). Each of the unknown velocities is then set to the value of the nearest known sample.

Even with extrapolated values near the interface, the first-order accuracy of time splitting and the Ghost Fluid Method allows some unwanted influence between liquid and air. This is most visible as unwanted drag on escaped water particles. This is a form of numerical viscosity, since in truly inviscid flow there would be no drag.

To test the coupling of escaped particles with the surrounding flow, we used the following 2-D setup. An air velocity field was initialized throughout the domain to

[3] In all tests, we used $\alpha = 0.03$.

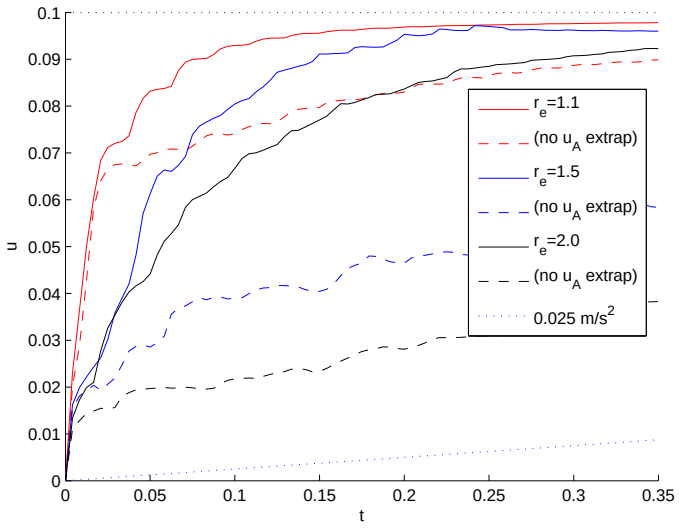


Figure 3.14: Extrapolating only the air velocity reduces coupling of small droplets to the surrounding air velocity.

$0.1m/s$ in the positive x direction. As the flow carried air particles rightward, new particles were seeded on the left edge of the domain with velocities interpolated from the grid.

A single liquid particle was seeded in the middle of domain and treated as escaped (§3.1.2). The velocity of this liquid particle was tracked over time. For comparison, the actual drag force [25] on a water drop with radius $R = 3mm$ traveling at $\Delta u = 0.1m/s$ relative to air with kinematic viscosity $\nu_g = 15.7 \times 10^{-6} m^2/s$ and density $\rho_g = 1.2kg/m^3$ is $F_{drag} = 22.8\rho_g \nu_g^{0.72} (R\Delta u)^{1.28} = 2.85 \times 10^{-6} N$. This corresponds to a maximum acceleration of $0.025m/s^2$.

Figure 3.14 shows the results of this experiment with r_e values of $1.1\Delta x$, $1.5\Delta x$ and $2.0\Delta x$. In all cases, the initial acceleration of the droplet was much greater than $0.025m/s$. With larger r_e, this unwanted acceleration decreased. This is still sometimes visible as water droplets getting carried around by air currents instead of following mostly ballistic trajectories.

To combat this problem, instead of extrapolating the air velocity $\vec{u}^A$ into the liquid region, $\vec{u}^A$ is set to $\vec{u}^L$ there. This way, the liquid velocity is smeared into the

air velocity near the interface, thereby reducing the impact of numerical smearing on the liquid velocity field. As shown in the dashed lines of figure 3.14, this results in less coupling between the droplet and surrounding air.

For improved droplet behaviour, it would be sufficient to alter the extrapolation strategy only in the expanded liquid regions around escaped liquid particles. However, we did not find the additional smearing of the air velocity near the interface to have a visually noticeable effect elsewhere. This is consistent with the intuition that the liquid should impart more influence on the much lower density air. Similarly, Sussman et al. [37] extrapolated liquid velocity to obtain solutions for high density ratios that converged to solutions of a single-phase liquid simulation.

Velocity extrapolation is oblivious to solid boundaries. By extrapolating this way, a free-slip condition at solid boundaries is maintained. Although the pressure solve enforces zero normal velocity at boundaries, noise from the particles can result in non-zero normal velocities on the grid there. If a particle does stray into a solid region, it is bumped to the surface in similar fashion to the bumping employed at the liquid-air interface.

Our velocity extrapolation is not inherently divergence-free. Rasmussen et al. [29] eliminated divergence in extrapolated velocities using an additional projection. It is possible that this would make a difference for some scenarios, however we did not find it to be worth the extra cost in our implementation.

3.5 Volume Control

When simulating splashy liquids, i.e. flows with high Weber number, MultiFLIP reproduces the visual excitement of the flow through the creation of escaped particles. Unfortunately, since escaped particles carry no volume, this tends to erode the overall volume of liquid.

To combat this, we use a simple global volume control mechanism. The current liquid volume is estimated as the sum of the face fractions, in 2-D

$$V^n = \frac{\Delta x^2}{2} \sum_{i,j} ufrac_{i,j} + \frac{\Delta x^2}{2} \sum_{i,j} vfrac_{i,j}. \tag{3.26}$$

This is compared to the target volume calculated on the first time step, V^0, using

the proportional controller of Kim et al. [16] to obtain a target divergence, ΔV^n. We found it best to use an aggressive controller gain, $k_P = 2.3/2\Delta t$ and a conservative PI gain, $k_I = (k_P/512)^2$ to avoid oscillations.

To avoid artifacts in the liquid-air interface, the divergence is applied only to cells whose faces are either completely covered by air or completely covered by liquid. ΔV^n is divided by the number of all-liquid cells to obtain a per-liquid-cell divergence, which is subtracted from equation 3.13 for those all-liquid cells. Likewise, ΔV^n is divided evenly among all-air cells and added to the corresponding entries in equation 3.13.

Chapter 4

Results

Figures 4.1–4.3 show selected frames from a 2-D scenario to test liquid behaviour in the presence of strong air currents. Using physical parameters for surface tension and density, two spherical water drops of diameter $2cm$ and three $1cm$ drops are dropped into a liquid pool, within a $10cm \times 10cm$ box. At this scale, surface tension is strong enough to keep the drops intact until they reach the liquid surface.

For comparison, the scenario was run with a single velocity field (left column) and separate air and liquid velocity fields (right column). At $t = 92ms$, the unwanted influence of air can be seen in the deformed splash in the single velocity field version. At $t = 208ms$, a splash on the right side of the frame gets stretched out and pulled along by the air. This stretching action results in additional escaped particles which are more influenced by the surrounding air. Even after $833ms$, they are still floating around in disturbingly non-physical fashion.

In the two velocity field version, a few escaped particles are generated but they follow mostly ballistic trajectories. Throughout the simulation, small entrained bubbles are generated and float to the surface. A larger bubble can be seen in the lower-left at $t = 417ms$.

Figure 4.4 shows a 3-D ellipsoidal blob of liquid in zero gravity, used to test the accuracy of surface tension as per Torres and Brackbill [38]. The initial shape of the ellipsoid is given by

$$\frac{x^2}{3^2} + \frac{y^2}{2^2} + \frac{z^2}{2^2} = 1,$$

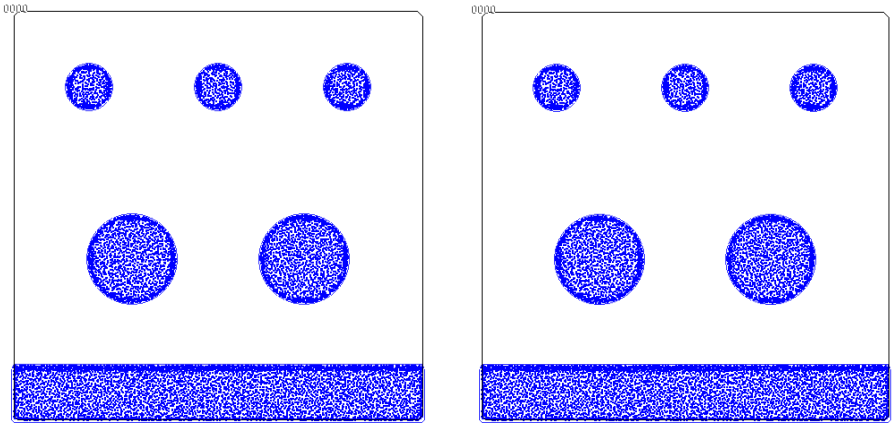


Figure 4.1: Comparison of a 2-D splashing drop simulation using a single velocity field (left) and two velocity fields (right). The initial state, shown here, is the same.

which has the volume of a sphere with radius $r_0 = \sqrt[3]{12}$. The surface tension coefficient was set to $\sigma = 1$, and the densities for the ellipse and surrounding fluid are $\rho_L = 1$ and $\rho_A = 0.01$. The expected oscillation frequency can be determined analytically as

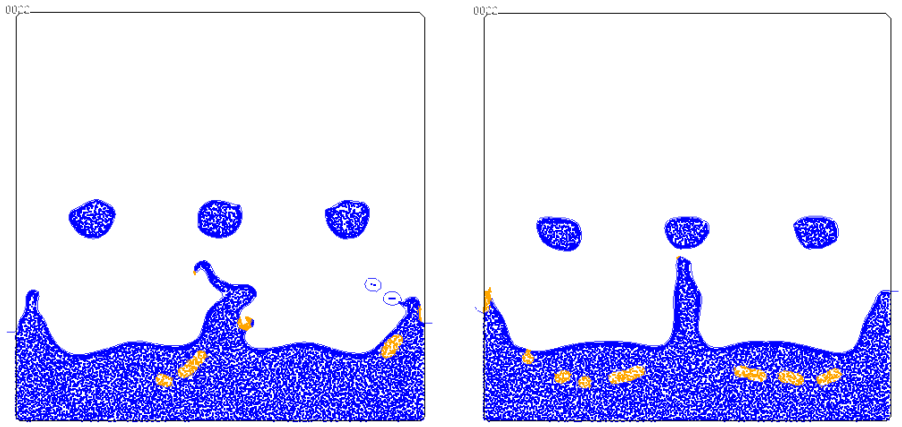
$$\omega_2^2 = \frac{6\sigma}{(\rho_L + \rho_A) r_0^3} \tag{4.1}$$

$$\approx 0.495, \tag{4.2}$$

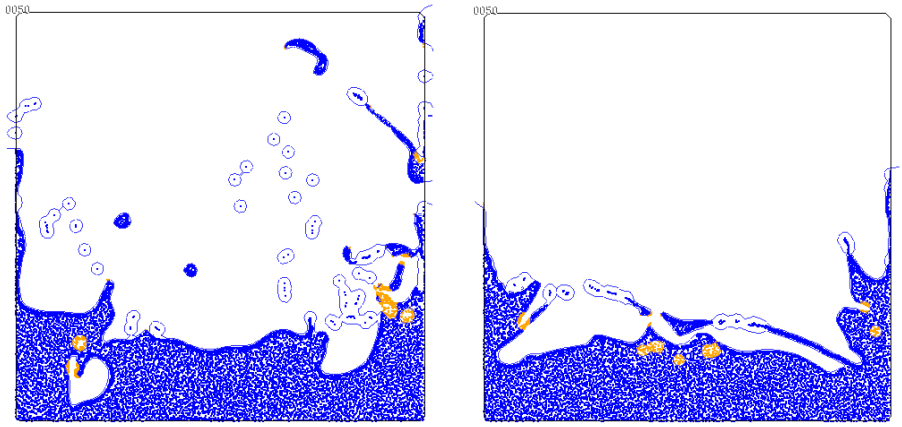
corresponding to a period of 8.93 seconds.

For our experiment, we used a coarse 20^3 grid and time steps dictated by a CFL number of 5. One-sided differences were employed at the boundaries of the grid. Figure 4.5 shows kinetic energy over time, with and without the volume control code described in §3.5. Every second dip in the energy corresponds to a return to the original configuration.

The observed oscillation period is within 10% of the expected result. Some noise can be seen as the kinetic energy does not return to zero, but this additional energy does not show signs of getting out of hand. Although volume control does cause the elongation of the ellipsoid to decay more quickly, the overall behaviour

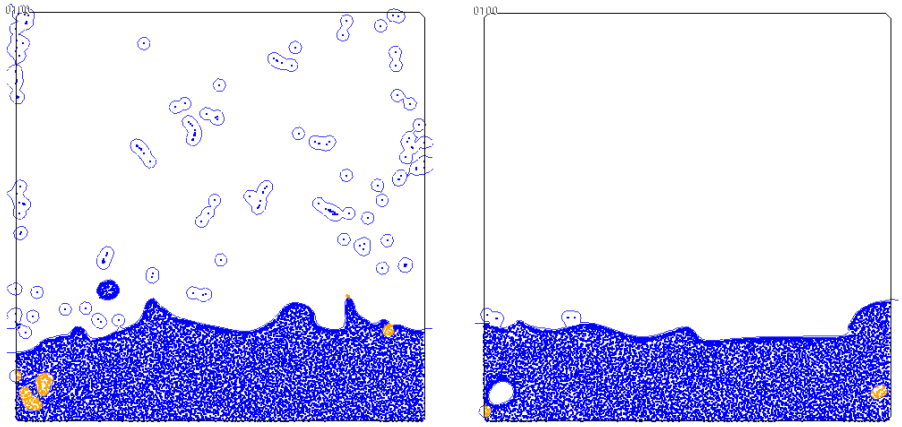


$t = 92ms$

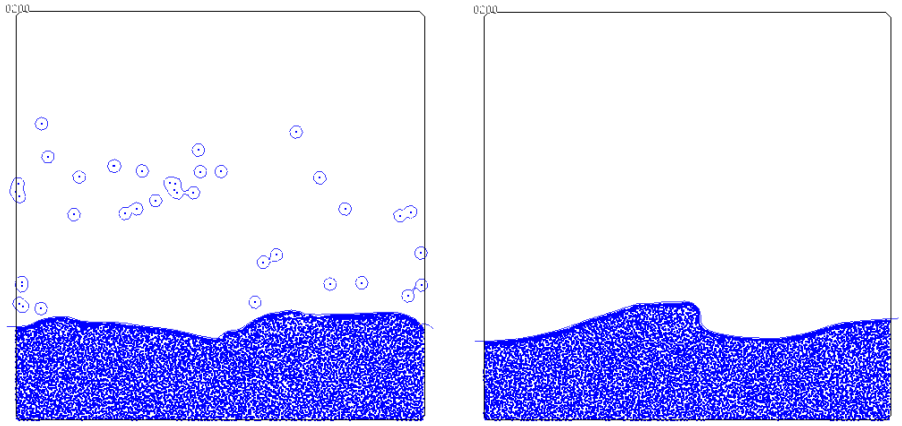


$t = 208ms$

Figure 4.2: Comparison of a 2-D splashing drop simulation using a single velocity field (left) and two velocity fields (right). Without separate velocity fields, the liquid gets pulled along and stretched by air currents.



$t = 417ms$



$t = 833ms$

Figure 4.3: Comparison of a 2-D splashing drop simulation using a single velocity field (left) and two velocity fields (right). Droplets float around unconvincingly in the single velocity field version.

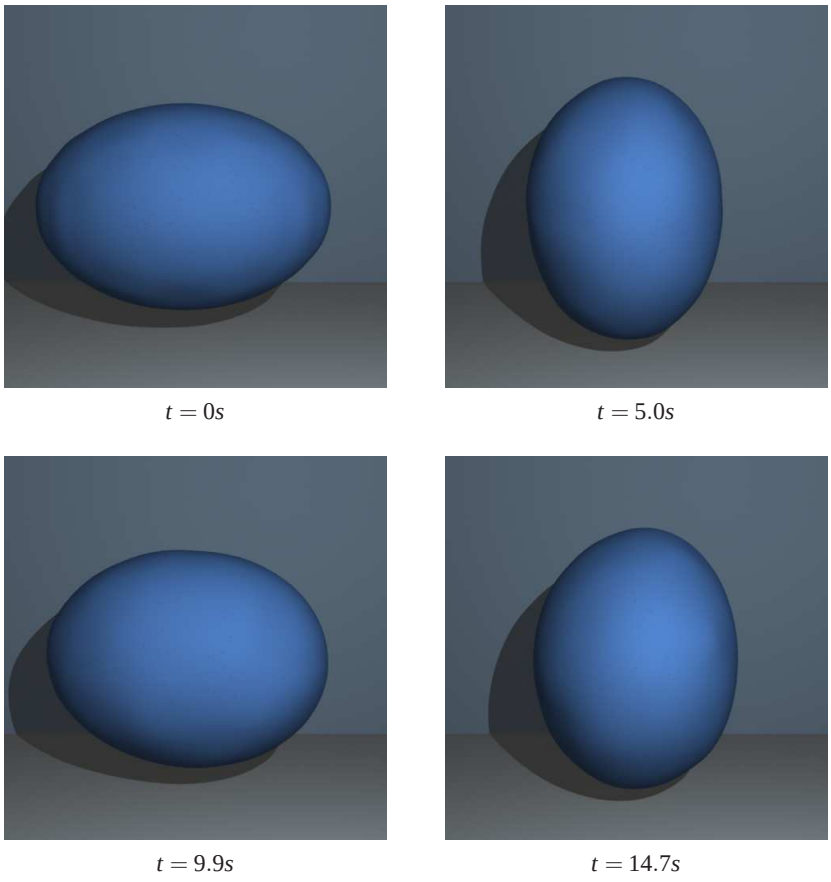


Figure 4.4: A 3-D ellipsoid in zero gravity oscillates due to surface tension.

is still good, showing that volume control does not significantly compromise the curvature calculation around the liquid surface.

Figure 4.6 shows the challenging scenario of a volume of liquid exiting an enclosed space through a narrow spout. The water exiting the container must be replaced by air, producing the visually interesting glugging effect. In a single-phase simulation, glugging does not occur because the pressure difference between the air inside and outside of the container is not considered. In our result, air can be seen rushing up through the spout, producing bubbles that rise to replace the lost liquid in the top container.

For our test, we simulated two spherical containers of diameter $3cm$ connected

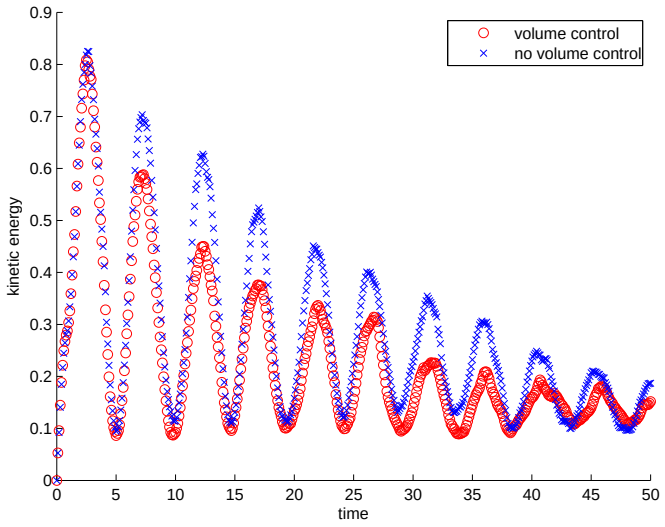


Figure 4.5: A 3-D oscillating ellipsoid shows the expected evolution of kinetic energy.

by a cylinder with diameter $0.8cm$, the top container half-filled with water. Physical values were used for surface tension, $\sigma = 0.073kg/s^2$, and density, $\rho_L = 1000kg/m^3$, $\rho_A = 1.2041kg/m^3$. The small scale of the simulation and resulting high surface tension forces prevented the water from breaking up too much and losing volume. However, there is still perhaps more break-up than one would expect at this scale, indicating that MultiFLIP may benefit from a more sophisticated particle escape condition.

The test used a $40 \times 40 \times 80$ simulation grid on a 2.4 GHz Intel Core2 Duo with 3 GB of RAM. The strong surface tension required small time steps, $0.23ms$ on average, or about 18 steps per frame at 240 frames per second. On average, each time step took $6.34s$ to execute, with contributions from sub-tasks shown in table 4.1. Most sub-tasks were parallelized through the use of OpenMP but the Poisson solve used only a single core. The next step towards improving performance would be to employ a parallel Poisson solver.

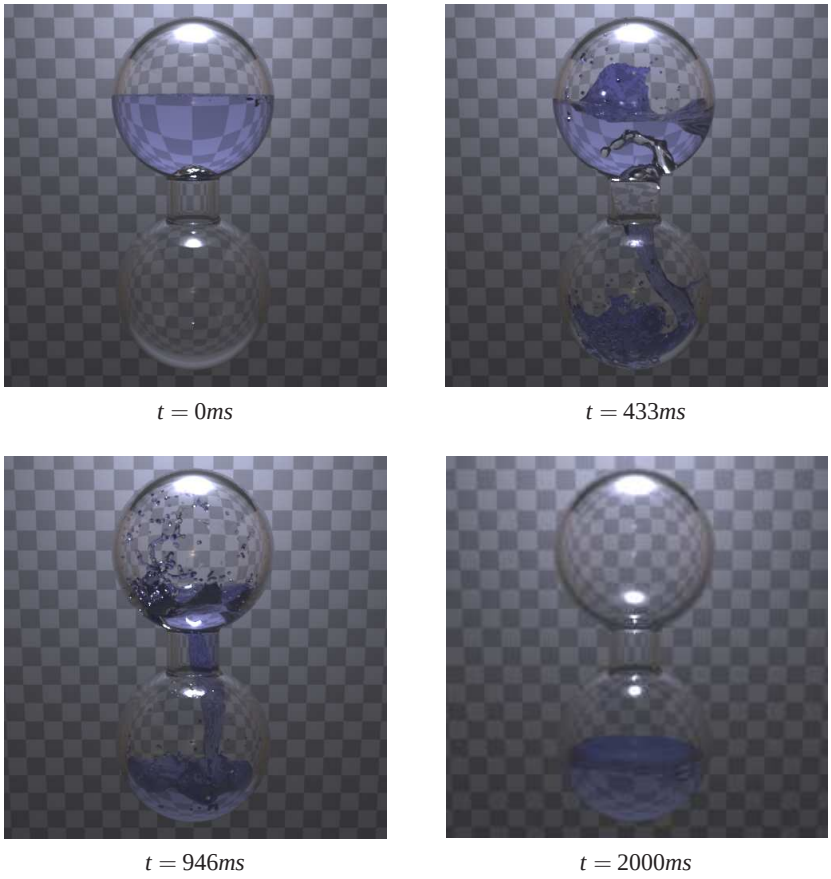


Figure 4.6: Simulated water flows through a spout. By treating air as incompressible, visually exciting glugging is reproduced.

Sub-task	% execution time
Poisson solve	36
Particle velocities to grid	21
Pressure update to grid	15
Particle bumping	13
ϕ computation	7
Grid to particle velocities	2
Particle advection	2
Particle seeding	1

Table 4.1: Sub-task execution time

Chapter 5

Conclusion

We have presented a practical method for two-phase fluid simulation. Key aspects of our solution are:

- FLIP to integrate the Euler equations of fluid motion with minimal numerical dissipation
- the Ghost Fluid Method to discretize the jump in density and pressure at the liquid-air interface
- separate velocity fields with a combined divergence formulation to enforce overall incompressibility while maintaining a free-slip condition at the interface
- a new particle-based surface tracking method
- level set adjustment around escaped particles for sub-scale bubble and droplet behaviour

Our results show that the method effectively reproduces splashy two-phase flows with plausible behaviour of small-scale features and accurate surface tension effects.

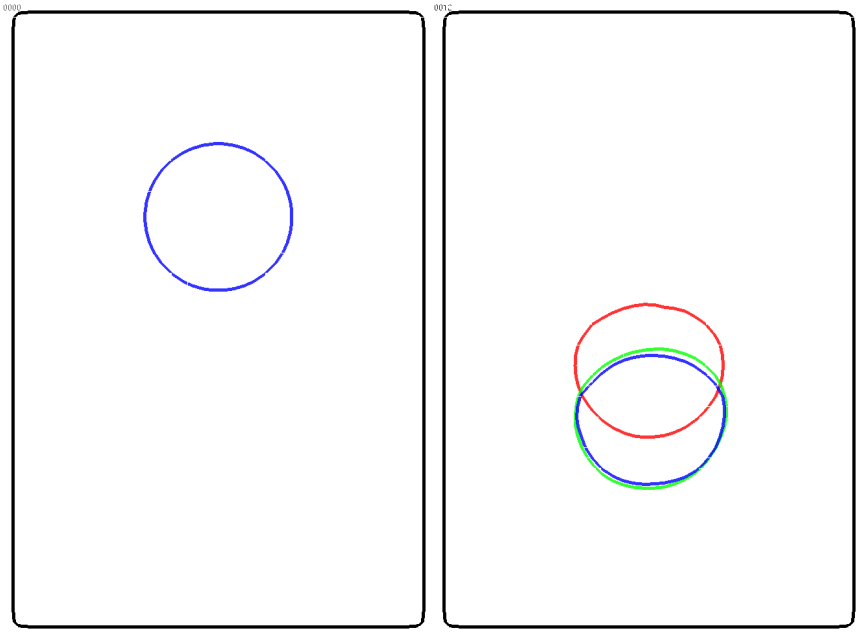


Figure 5.1: A small 2-D water droplet falls using two velocity fields (blue), one velocity field (red) and two fields with volume control (green).

5.1 Future Work

Figure 5.1 shows a water drop of diameter $6\frac{2}{3}mm$ falling in air as per Example 2 in Kang et al. [13]. With a grid resolution of 40×60 and CFL restriction of 0.5, MultiFLIP does a good job predicting the acceleration of the droplet. Its speed after $50ms$ is $0.48m/s$, giving $9.6m/s^2$.

However, without explicit volume control, the volume of the droplet decreases by 13.4%, showing that there is room for improvement in tracking the liquid-air interface. For comparison, the test was also performed with a single velocity field, giving slightly less volume loss (10.9%), but at significantly slower acceleration ($7.46m/s^2$). As expected, with volume control enabled volume loss was much less (1.8%).

An interesting challenge for future research would be to reproduce the water drop shapes seen in nature at terminal velocity [2, 23]. At present, decimation of

the droplet volume makes it difficult to approach the terminal velocity of approximately $8m/s$.

Thin features are another source of volume change inherent to grid-based fluid simulation. While our global volume correction scheme helps compensate for this, it would be better to address the problem at its source. In MultiFLIP, thin features usually manifest themselves as escaped particles, so associating some volume with those particles could be a more localized solution to volume change. Regional level sets [15] have also been shown to be useful for controlling volume change.

Other papers (e.g. [1, 19]) have used adaptive grids to concentrate computing power where it is needed. This could also be employed with MultiFLIP. For example, higher resolution in the area of escaped particles would allow the adjusted ϕ to cover more grid cells, reducing unwanted coupling with the surrounding fluid.

Finally, the current MultiFLIP implementation extrapolates liquid velocities throughout the entire domain twice: once when updating the grid velocities from particles and again when updating the grid velocities from the pressure projection. The extrapolated velocities are only really needed in a band around the liquid-air interface, so some performance might be gained from using a fast marching method [31] to populate only that band.

Bibliography

[1] C. Batty, S. Xenos, and B. Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Computer Graphics Forum*, volume 29, pages 695–704. Wiley Online Library, 2010. → pages 37

[2] K. Beard and C. Chuang. A new model for the equilibrium shape of raindrops. *Journal of the Atmospheric sciences*, 44(11):1509–1524, 1987. ISSN 0022-4928. → pages 36

[3] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982. ISSN 0730-0301. → pages 5

[4] J. Brackbill and H. Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986. ISSN 0021-9991. → pages 3

[5] R. Bridson. *Fluid simulation for computer graphics*. AK Peters Ltd, 2008. → pages 6

[6] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002. → pages 8

[7] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 152(2):457–492, 1999. → pages 19

[8] N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: http://doi.acm.org/10.1145/383259.383261. → pages 8

[9] F. Harlow. Fluid dynamics in group T-3 Los Alamos national laboratory. *Journal of Computational Physics*, 195(2):414–433, 2004. ISSN 0021-9991. → pages 1

[10] F. Harlow, J. Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965. → pages 7

[11] J. Hong, H. Lee, J. Yoon, and C. Kim. Bubbles alive. In *ACM SIGGRAPH 2008 papers*, pages 1–4. ACM, 2008. → pages 5

[12] J.-M. Hong and C.-H. Kim. Discontinuous fluids. *ACM Trans. Graph.*, 24 (3):915–920, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073283. → pages 4

[13] M. Kang, R. Fedkiw, and X. Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 15(3): 323–360, 2000. → pages 4, 36

[14] N. Kang, J. Park, J. Noh, and S. Shin. A Hybrid Approach to Multiple Fluid Simulation using Volume Fractions. In *Computer Graphics Forum*, volume 29, pages 685–694. Wiley Online Library, 2010. → pages 4

[15] B. Kim. Multi-phase fluid simulations using regional level sets. In *ACM Transactions on Graphics (TOG)*, volume 29, page 175. ACM, 2010. → pages 4, 37

[16] B. Kim, Y. Liu, I. Llamas, X. Jiao, and J. Rossignac. Simulation of bubbles in foam with the volume control method. *ACM Transactions on Graphics (TOG)*, 26(3):98, 2007. ISSN 0730-0301. → pages 4, 27

[17] D. Kim, O. Song, and H. Ko. Stretching and wiggling liquids. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–7. ACM, 2009. → pages 5

[18] X. Liu, R. Fedkiw, and M. Kang. A boundary condition capturing method for Poisson's equation on irregular domains. *Journal of Computational Physics*, 160(1):151–178, 2000. → pages 4, 19, 22

[19] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (TOG)*, 23(3): 457–462, 2004. ISSN 0730-0301. → pages 37

[20] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. Multiple interacting liquids. *ACM Trans. Graph.*, 25(3):812–819, 2006. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1141911.1141960. → pages 4, 10

[21] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled SPH and particle level set fluid simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(4):797–804, 2008. → pages 5, 14

[22] C. Macdonald and S. Ruuth. Level set equations on surfaces via the Closest Point Method. *Journal of Scientific Computing*, 35(2):219–240, 2008. → pages 11

[23] J. McDonald. The shape of raindrops. *Sci. Am*, 190(2):64–68, 1954. → pages 36

[24] V. Mihalef, B. Unlusu, D. Metaxas, M. Sussman, and M. Hussaini. Physics based boiling simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 317–324. Eurographics Association, 2006. ISBN 3905673347. → pages 4

[25] V. Mihalef, D. Metaxas, and M. Sussman. Simulation of two-phase flow with sub-scale droplet and bubble effects. In *Computer Graphics Forum*, volume 28, pages 229–238. John Wiley & Sons, 2009. → pages 4, 14, 25

[26] W. Moss, H. Yeh, J. Hong, M. Lin, and D. Manocha. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Transactions on Graphics (TOG)*, 29(3):1–13, 2010. ISSN 0730-0301. → pages 1

[27] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer Verlag, 2003. → pages 8, 11

[28] A. Ralston. Runge-Kutta methods with minimum error bounds. *Mathematics of computation*, 16(80):431–437, 1962. → pages 23

[29] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–202, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi: http://doi.acm.org/10.1145/1028523.1028549. → pages 26

[30] D. Roble, N. Zafar, and H. Falt. Cartesian grid fluid simulation with irregular boundary voxels. In *ACM SIGGRAPH 2005 Sketches*, page 138. ACM, 2005. → pages 16

[31] J. Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999. ISSN 0036-1445. → pages 37

[32] B. Solenthaler and R. Pajarola. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 211–218. Eurographics Association, 2008. → pages 5

[33] O. Song, H. Shin, and H. Ko. Stable but nondissipative water. *ACM Transactions on Graphics (TOG)*, 24(1):81–97, 2005. ISSN 0730-0301. → pages 4

[34] C. Sprenger, D. Trazzi, A. Hemberger, and S. Marino. Digital Water for Avatar. In *ACM SIGGRAPH 2010 Talks*, page 1. ACM, 2010. → pages 1

[35] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999. → pages 6

[36] M. Sussman, M. Hussaini, K. Smith, R. Zhi-Wei, and V. Mihalef. A Second-Order Adaptive Sharp-Interface Method for Incompressible Multiphase Flow. *Computational Fluid Dynamics 2004*, pages 643–648, 2006. → pages 4

[37] M. Sussman, K. Smith, M. Hussaini, M. Ohta, and R. Zhi-Wei. A sharp interface method for incompressible two-phase flows. *Journal of Computational Physics*, 221(2):469–505, 2007. → pages 26

[38] D. Torres and J. Brackbill. The point-set method: front-tracking without connectivity. *Journal of Computational Physics*, 165(2):620–644, 2000. ISSN 0021-9991. → pages 28

[39] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–628, 2005. → pages 10, 24

[40] W. Zheng, J. Yong, and J. Paul. Simulation of bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 325–333. Eurographics Association, 2006. ISBN 3905673347. → pages 4

[41] Y. Zhu and R. Bridson. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, pages 965–972. ACM, 2005. → pages 3, 5, 7, 22