A Stream Function Solver for Liquid Simulations

Ryoichi Ando*
IST Austria

Nils Thuerey†
Technische Universität München

Chris Wojtan‡
IST Austria

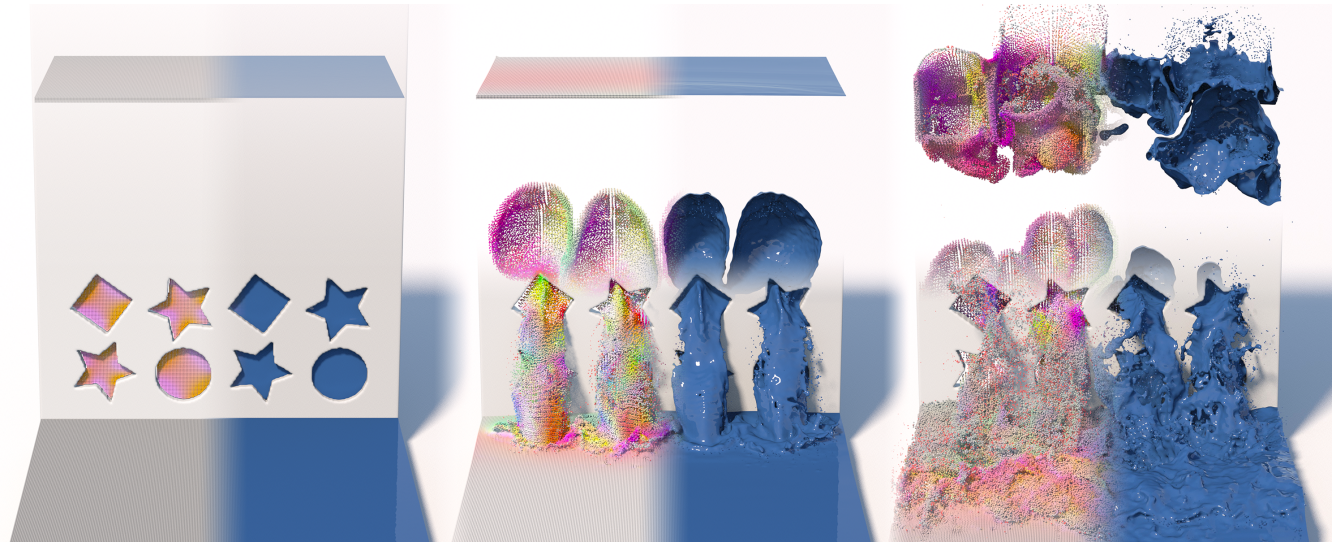

Figure 1: *Liquid streaming out of a closed container with holes. Our solver calculates the complex formation of bubbles without explicitly solving for the gas phase. Each of the images shows FLIP particles colored by the stream function on the left, and the surface on the right. The simulation ran with a* $256\times256\times128$ *resolution, 54 seconds per time step, and 3.9 minutes per video frame.*

Abstract

This paper presents a liquid simulation technique that enforces the incompressibility condition using a stream function solve instead of a pressure projection. Previous methods have used stream function techniques for the simulation of detailed single-phase flows, but a formulation for liquid simulation has proved elusive in part due to the free surface boundary conditions. In this paper, we introduce a stream function approach to liquid simulations with novel boundary conditions for free surfaces, solid obstacles, and solid-fluid coupling.

Although our approach increases the dimension of the linear system necessary to enforce incompressibility, it provides interesting and surprising benefits. First, the resulting flow is guaranteed to be divergence-free regardless of the accuracy of the solve. Second, our free-surface boundary conditions guarantee divergence-free motion even in the un-simulated air phase, which enables two-phase flow

simulation by only computing a single phase. We implemented this method using a variant of FLIP simulation which only samples particles within a narrow band of the liquid surface, and we illustrate the effectiveness of our method for detailed two-phase flow simulations with complex boundaries, detailed bubble interactions, and two-way solid-fluid coupling.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

*E-mail:and@verygood.aid.design.kyushu-u.ac.jp
†E-mail:nils.thuerey@tum.de
‡E-mail:wojtan@ist.ac.at

Keywords: fluid, two-phase flow, stream function, vector potential

1 Introduction

We wish to simulate the motion of a liquid surrounded by air. The inviscid, incompressible Navier-Stokes equations describe this motion:

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{1}{\rho}\nabla p + \boldsymbol{g} \tag{1}$$

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

where $\boldsymbol{u}$ is the fluid velocity field, t is time, ρ is fluid density, p is fluid pressure, and $\boldsymbol{g}$ is a body acceleration. The first equation describes the evolution of fluid momentum over time, and the second equation constrains the fluid motion such that it conserves mass. The typical approach to enforcing Eq. (2) is to find a pressure p whose gradient projects the velocity field into a divergence-free state. This projection can also be interpreted as a kinetic energy minimization problem [Batty et al. 2007]:

$$\underset{p}{\text{minimize}} \quad \int_\Omega \frac{1}{2}\rho||\boldsymbol{u}^* - \frac{\Delta t}{\rho}\nabla p||^2 dV \tag{3}$$

where Δt denotes the time step size, and $\boldsymbol{u}^*$ is the intermediate velocity after advection [Bridson 2008], Ω is the fluid domain, and V is the liquid volume. The solution yields a Poisson problem for the pressure:

$$\nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p \right) = \nabla \cdot \boldsymbol{u}^* \tag{4}$$

One can prevent flow through the boundary by enforcing the boundary condition $\boldsymbol{u} \cdot \boldsymbol{n} = \boldsymbol{v} \cdot \boldsymbol{n}$ (with surface normal $\boldsymbol{n}$ and boundary velocity $\boldsymbol{v}$). At the liquid's free surface, a common approach is to assume that the momentum of the liquid dominates that of the surrounding air to such an extent that the air pressure is negligible, resulting in the Dirichlet boundary condition $p = 0$. Alternatively, one can avoid this free-surface approximation by simulating the entire air phase as well as the liquid phase, resulting in a *two-phase* flow computation.

Unfortunately, actually solving Eq. (4) in this way can lead to a few frustrating numerical errors. The $p = 0$ condition at the free surface does not enforce incompressibility of the air phase, so the coupling between air and water is unrealistic and it results in artifacts like collapsing bubbles. The alternative of simulating the entire air phase in addition to the liquid phase requires a delicate treatment of the fluid interface [Boyd and Bridson 2012; Hong and Kim 2005] and it requires additional memory and computational effort to handle the relatively unimportant invisible air domain. In addition, the pressure projection is incapable of exactly enforcing the incompressibility condition. Eq. (4) relies on the solution to a linear system which necessarily accumulates numerical errors. In practice, researchers usually use an iterative solver which only satisfies Eq. (4) up to some tolerance. Consequently, Eq. (2) is only approximately satisfied at any given moment, and the fluid volume drifts over time.

We propose a *stream function* approach to solving the incompressibility condition. The Helmholtz-Hodge decomposition states that our intermediate velocity field is composed of three parts:

$$\boldsymbol{u}^* = \nabla \Theta + \nabla \times \boldsymbol{\Psi} + \boldsymbol{\gamma} \tag{5}$$

where the first term is the gradient of a scalar field, the second term is the curl of a vector field, and the third term is a harmonic vector field which is neither a curl nor a gradient. Removing the first component gives us a divergence-free vector field $\boldsymbol{u} = \nabla \times \boldsymbol{\Psi} + \boldsymbol{\gamma}$. Instead of solving for the gradient term and subtracting from $\boldsymbol{u}^*$, we propose to solve for Ψ, which directly gives us $\boldsymbol{u}$. This approach provides several contributions to the computer graphics community:

- We derive free-surface boundary conditions which enable, for the first time, a stream function approach to liquid simulation.
- We derive solid boundary conditions for two-way solid-fluid coupling.
- Our resulting flow is guaranteed to be divergence-free *regardless of the accuracy of the solve*.
- Our approach enforces incompressibility *even in the unsimulated air phase*, enabling realistic two-phase flow simulation by computing only a single phase.
- We provide an implementation of our method based on FLIP.

2 Related work

Solving the Navier-Stokes (NS) equations in computer graphics was popularized by Foster and Metaxas [1997], as well as the stable fluids approach [Stam 1999]. While the latter made the flow divergence free using an FFT, most follow up papers have resorted to a MAC discretziation [Harlow and Welch 1965]. For these and many other types of solvers, the pressure projection is crucial for making the flow incompressible as well as for introducing visually interesting motions. Thus, a large part of work has focused on improving it, e.g. with fast iterative solvers [Zhu et al. 2010; Ferstl et al. 2014] and by introducing flexible ways to couple multiple grids [English et al. 2013]. A variety of different discretizations have been proposed over the years, many of them using tetrahedral meshes [Klingner et al. 2006; Batty et al. 2010],

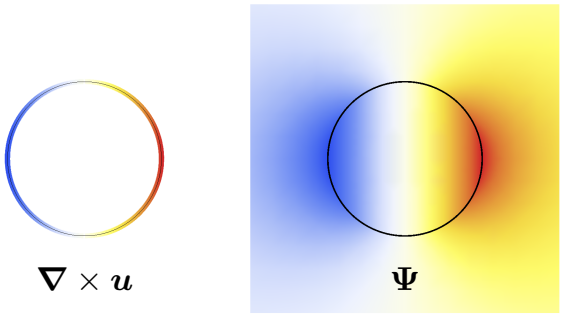

Figure 2: *A 2D illustration of the vorticity (left) and stream function (right) of a 2D water drop with constant downward velocity.*

Elcott and colleagues [2007] proposed a discretization based on generic simplices while focusing on preserving circulations in the flow. They also arrive at a stream function solve for the velocity update of their algorithm. We will outline similarities between this work and ours below, but one important difference is that their work focuses on smoke phenomena with static boundaries, while we target liquid simulations and support moving rigid bodies. The work by Elcott et al. is based around *vorticity* as a central quantity for fluids, and while vorticity and stream function are both vector-valued, they have different content: vorticity is defined as $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$, while the stream function $\boldsymbol{\Psi}$ relates to velocity by $\boldsymbol{u} = \nabla \times \boldsymbol{\Psi}$. Figure 2 illustrates this difference. Unlike vorticity, the stream function does not have an obvious physical meaning. However, both are related, and the stream function can be used to convert vorticity into a velocity (as is done in [Elcott et al. 2007]). Many variants of vorticity formulations have been proposed, e.g., based on Lagrangian vortex filaments [Angelidis and Neyret 2005; Weißmann and Pinkall 2010], vortex particles [Gamito et al. 1995; Selle et al. 2005], vortex sheets [Pfaff et al. 2012; Brochu et al. 2012], or with tightly coupled rigid bodies [Vines et al. 2014]. These methods focus on single-phase fluid simulation, but the algorithm of Golas et al. [2012] uses a grid vortex-particle hybrid and can simulate liquids by resorting to a pressure-based solve near the free surface and solid boundaries. Bridson et al. [2007] use an analytically computed stream function, making heavy use of the fact that the resulting velocity is guaranteed to be divergence-free.

While our algorithm is based on an Eulerian representation, a significant amount of work also covers Lagrangian representations [Ihmsen et al. 2014]. These are particularly popular for simulations of liquids, and most grid-based methods have also employed grid-particle hybrids to improve mass conservation, e.g. particle level-sets [Enright et al. 2003], or so-called FLIP methods [Bridson 2008]. We will make use of the latter for our simulations, but our approach is agnostic to the particular choice of surface tracking method. We believe FLIP is a good choice due to its wide spread use in industry [Budsberg et al. 2013] and academia [Boyd and Bridson 2012; Ando et al. 2013].

Although our approach makes use of a free surface model and simulates only the liquid phase, the special properties of stream

functions move it close to two-phase simulations. Simulations of multiple phases have, e.g., been investigated for level-set surfaces [Losasso et al. 2006], as well as volume fractions [Kang et al. 2010] and mesh-based surface tracking [Da et al. 2014]. There are also many papers from the field of particle-based simulations that target multiple phases [Müller et al. 2005; Ren et al. 2014]. Boyd and Bridson [2012] introduced a FLIP simulation model that employs two distinct velocity fields for the liquid and gas phases. Large density variations are notoriously difficult for both vortex methods and two-phase fluid formulations, especially when the Atwood ratio $(\rho_{\text{liquid}} - \rho_{\text{air}})/(\rho_{\text{liquid}} + \rho_{\text{air}})$ approaches 1. While our model also becomes stiffer with large density jumps, it becomes highly stable and efficient when the Atwood number actually equals one. No previous approach shares this property, and it allows us to create divergence-free motion for the gas phase without explicitly simulating it.

The idea of using stream functions has also been proposed in various works of the computational fluid dynamics community. Like vorticity formulations, they are not widely used for simulating liquids. Attempts have been made to push the density difference of multi-phase simulations using *vortex-in-cell* methods [Brecht and Ferrante 1989; Stock et al. 2008], where stream functions are also used for a conversion of vorticity to velocity. Many papers focused on two-dimensional stream function formulations, e.g., for lid-driven cavity studies [Barragy and Carey 1997]. While so called *velocity-vorticity* formulations are more widely used [G. Guj 1993], the advantages of stream function formulations have been demonstrated, e.g., for duct flows [Wong and Reizes 1984] and for accurate boundary handling [Wang and Zhang 2011]. Similar to works from computer animation, these publications focus purely on single-phase flows.

3 Stream function solver

For simplicity of exposition, we will assume that our domain is a subset of $\mathbb{R}^3$ with a simple boundary, so that the harmonic component $\boldsymbol{\gamma}$ in Eq. (5) is zero, and thus the divergence-free velocity field is $\boldsymbol{u} = \nabla \times \boldsymbol{\Psi}$. We will use this idea to replace the pressure projection in a standard Navier-Stokes simulation. Following the time-splitting approach in [Bridson 2008], we wish to solve:

$$\boldsymbol{u} = \boldsymbol{u}^* - \frac{\Delta t}{\rho} \nabla p \tag{6}$$

We multiply by ρ, substitute $\boldsymbol{u} = \nabla \times \boldsymbol{\Psi}$, and take the curl of both sides to remove p:

$$\nabla \times (\rho \nabla \times \boldsymbol{\Psi}) = \nabla \times (\rho \boldsymbol{u}^*) \tag{7}$$

The solution to Eq. (7) is the solution to the following quadratic kinetic energy minimization problem:

$$\underset{\boldsymbol{\Psi}}{\text{minimize}} \quad \int_\Omega \frac{1}{2} \rho ||\boldsymbol{u}^* - \nabla \times \boldsymbol{\Psi}||^2 dV \tag{8}$$

This equation is a kinetic energy minimization, but it can also be interpreted as the search for the unique incompressible velocity field $\nabla \times \boldsymbol{\Psi}$ which is as close as possible to $\boldsymbol{u}^*$ in the weighted least squares sense (with weights equal to $\rho\ dV/2$). The least squares problem is solved exactly by taking the gradient with respect to $\boldsymbol{\Psi}$ and setting it to zero, yielding Eq. (7).

The curl operator $\nabla\times$ has a non-trivial nullspace, as illustrated by the operation $\nabla\times(\boldsymbol{\Psi} + \nabla\phi) = \nabla\times\boldsymbol{\Psi} + \nabla\times\nabla\phi = \nabla\times\boldsymbol{\Psi}$, where ϕ denotes an arbitrary scalar field. Therefore, an infinite number of vector fields have the same curl, and thus there are an infinite number of solutions to Equations (7) and (8). Enforcing a specific divergence on $\boldsymbol{\Psi}$ removes this null space. Specifically, we can pin down ϕ by enforcing an extra requirement $\nabla \cdot \boldsymbol{\Psi} = 0$, which is equivalent to adding a specific regularizer to Eq. (8):

$$\underset{\boldsymbol{\Psi}}{\text{minimize}} \quad \int_\Omega \frac{1}{2} \rho ||\boldsymbol{u}^* - \nabla \times \boldsymbol{\Psi}||^2 dV + \frac{1}{2} \rho (\nabla \cdot \boldsymbol{\Psi})^2 dV \tag{9}$$

This regularizer acts exactly on the nullspace of the original problem, so that a solution to Eq. (9) will also optimally solve Eq. (8). Discretizing the curl in Eq. (9) with the matrix $[\nabla\times]$ and the divergence with $[\nabla\cdot]$ (described in more detail in Section 5.1) gives:

$$\begin{aligned}\underset{[\boldsymbol{\Psi}]}{\text{minimize}} \Big(& \frac{1}{2} ([\boldsymbol{u}^*] - [\nabla\times][\boldsymbol{\Psi}])^T [\rho][V]([\boldsymbol{u}^*] - [\nabla\times][\boldsymbol{\Psi}]) \\ & + \frac{1}{2} ([\nabla\cdot][\boldsymbol{\Psi}])^T [\rho][V]([\nabla\cdot][\boldsymbol{\Psi}]) \Big)\end{aligned} \tag{10}$$

where $[\rho]$ and $[V]$ are diagonal matrices denoting density and cell volume. Taking the derivative of Eq. (10) with respect to $[\boldsymbol{\Psi}]$ and setting it to zero yields a symmetric positive-definite linear system:

$$\big([\nabla\times]^T [\rho][V][\nabla\times] + [\nabla\cdot]^T [\rho][V][\nabla\cdot]\big)[\boldsymbol{\Psi}] = [\nabla\times]^T [\rho][V][\boldsymbol{u}^*] \tag{11}$$

We re-scale this equation by dividing both sides by the volume of a grid cell and the density of liquid. This effectively transforms $[V]$ into an identity matrix $[I]$ (because we discretize it on a regular grid) and leads to values between 0 and 1 for ρ.

Next, we introduce a change of variables $[\rho] = [I] + [\Delta\rho]$; the previously-mentioned re-scaling sets liquid density to 1, so $[\Delta\rho]$ encodes the deviation from liquid density, and it must be negative. Substituting $[\Delta\rho]$ into Eq. (11) and applying the vector calculus identity $[\nabla\times]^T[\nabla\times] + [\nabla\cdot]^T[\nabla\cdot] = [\nabla^2]$ (where $[\nabla^2]$ is the vector Laplacian operator[1]), the linear system transforms into a variant of a vector-valued Poisson equation:

$$\big([\nabla^2] + [\nabla\times]^T [\Delta\rho][\nabla\times] + [\nabla\cdot]^T [\Delta\rho][\nabla\cdot]\big)[\boldsymbol{\Psi}] = [\nabla\times]^T [\rho][\boldsymbol{u}^*] \tag{12}$$

Note that on the right hand side of this equation, we simplified $[\Delta\rho] + [I]$ back to $[\rho]$ (both for brevity, and because it is intuitive to implement this way).

If we make the assumption that ρ=0 (and thus $[\Delta\rho]$=-1) in the air phase, then the entries of the linear system completely disappear outside of the liquid. The intuition here is that the air will automatically be divergence-free for any Ψ, and further varying Ψ within the air phase has no effect on the kinetic energy minimization because the air is massless. Within the liquid domain, the matrix entries are identical to those of a standard vector Laplacian matrix, and the $\Delta\rho$ terms are only non-zero in the narrow band near the liquid surface. Furthermore, the incompressibility constraint is implicitly satisfied within the air domain for free.

In the above exposition, we assumed that the harmonic component $\boldsymbol{\gamma}$ in Eq. (5) was zero, and we leave it set to zero for the simulations shown in Section 6. However, domains with non-trivial topology could be handled as described in [Elcott et al. 2007]: by first explicitly generating a harmonic vector field basis for the domain, and then projecting out and preserving the harmonic component of $\boldsymbol{u}^*$ before the stream function solve. Alternatively, we can simply treat complex boundary geometry as a dense two-way coupled rigid body, as we explain in Section 4.3.

[1] Note that we do not use Δ for the Laplacian; Δ will be reserved for changed quantities.

4 Boundary conditions

Suitable boundary conditions are essential in any fluid simulation. This section describes how to set free-surface and solid boundaries for our stream function solver.

4.1 Liquid-air interface

Unlike other methods that must treat the free surface with special boundary conditions, our method handles the liquid-air interface automatically. By simply dropping the density ρ to zero in the air domain, the system matrix already incorporates the necessary boundary condition at the liquid-air interface. Surface tension effects can be integrated by adding a force $\sigma \boldsymbol{H}$ to $\boldsymbol{u}^*$ at the liquid-air interface, where σ is the surface tension strength and $\boldsymbol{H}$ is the mean curvature normal of the interface. Surface tension can be an important visual cue for flows on small scales, but as we will aim for large-scale, splashing flows in Section 6, none of our examples includes surface tension forces.

The assumption in Section 3 that $\rho_{\text{air}} = 0$ corresponds to setting the Atwood ratio to 1. In contrast to most methods, which become unstable as the Atwood ratio approaches unity, our method actually becomes *more* stable and computationally simpler.

4.2 Static solid boundaries

The simplest way to handle solid boundaries in our framework is to set the density ρ to a large value within the solid domain. However, the resulting large density variations cause slower convergence and potential numerical instabilities. Instead, we can ensure $\boldsymbol{u} = 0$ at a solid boundary by setting the stream function equal to the gradient of a scalar potential, $\boldsymbol{\Psi} = \nabla\hat{\phi}$, where $\hat{\phi}$ contains the degrees of freedom along the surface of the solid. This way, $\nabla \times \boldsymbol{\Psi} = \nabla \times (\nabla\hat{\phi}) = 0$, so the $\boldsymbol{u} = 0$ condition is enforced by construction. To handle more accurate free-slip boundary conditions, we use fractional weighting of the fluid velocity, as explained in Section 5.1.

4.3 Rigid body coupling

Let $\boldsymbol{x}_c$, $\boldsymbol{u}_c$ and $\boldsymbol{\omega}$ be the center of mass, translational velocity, and angular velocity of a rigid body. The rigid body's velocity $\breve{\boldsymbol{u}}$ evaluated at location $\boldsymbol{x}$ is given by:

$$\breve{\boldsymbol{u}} = \boldsymbol{\omega} \times \boldsymbol{x}_{\text{rel}} + \boldsymbol{u}_c \tag{13}$$

where $\boldsymbol{x}_{\text{rel}} = \boldsymbol{x} - \boldsymbol{x}_c$, the distance relative to the body's center of mass. Within the solid domain, we find a stream function $\breve{\boldsymbol{\Psi}}$ that satisfies $\nabla \times \breve{\boldsymbol{\Psi}} = \breve{\boldsymbol{u}}$, which is given by:

$$\breve{\boldsymbol{\Psi}} = -\frac{1}{2}\,\text{diag}\begin{bmatrix} y^2+z^2 \\ z^2+x^2 \\ x^2+y^2 \end{bmatrix} \boldsymbol{\omega} + \begin{bmatrix} 0 & 0 & y \\ z & 0 & 0 \\ 0 & x & 0 \end{bmatrix} \boldsymbol{u}_c + \nabla\hat{\phi} \tag{14}$$

where $\nabla\hat{\phi}$ denotes the gradient of a scalar field $\hat{\phi}$, and x, y, z are the components of $\boldsymbol{x}_{\text{rel}}$. Instead of solving for the unconstrained stream function within the solid, the degrees of freedom are replaced by the scalar field $\hat{\phi}$ and the spatial constants $\boldsymbol{\omega}$ and $\boldsymbol{u}_c$. The corresponding density and the velocity for the resulting linear system are given by those of the rigid body. The resulting $\boldsymbol{\omega}$ and $\boldsymbol{u}_c$ are then used to integrate the rigid body dynamics in between fluid time steps. While this two-way rigid body coupling approach is similar to that of Carlson et al. [2004], ours is monolithically coupled, while theirs is weakly coupled using a pressure projection followed by a rigidity solve.

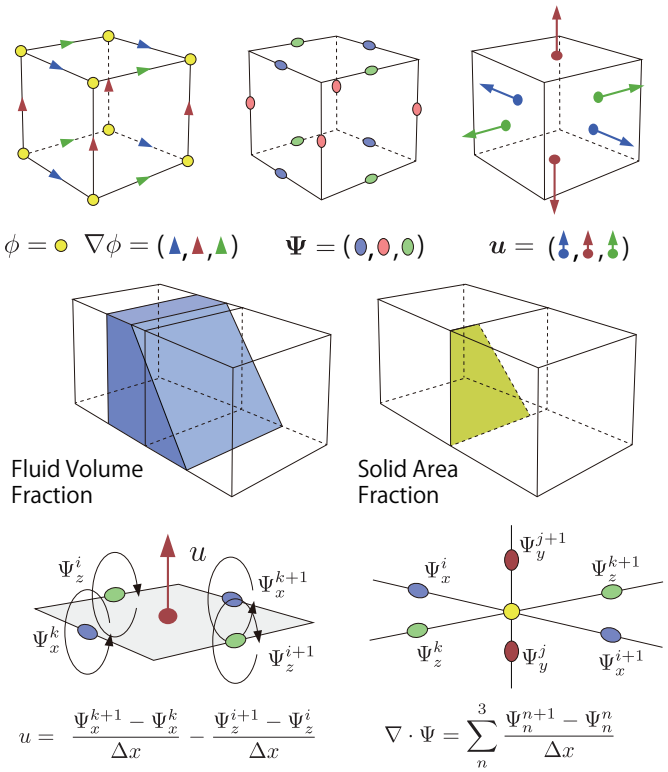

Figure 3: *Discretization overview*

To use a rigid body as a scripted kinematic solid boundary, we simply fix $\boldsymbol{\omega}$ and $\boldsymbol{u}_c$ to the prescribed values and set $\rho = 1$ within the rigid body. This reduces the degrees of freedom within the solid to $\hat{\phi}$, which can be solved just as in Section 4.2. The details of how we discretize these solid boundary conditions are discussed in Section 5.1.

4.4 Arbitrary solid boundary motions

Sections 4.2 and 4.3 describe how to constrain $\boldsymbol{\Psi}$ to achieve specific boundary motions. In principle, such an approach can be extended to any motion that can be represented as a stream function. Boundary motion that does not conserve volume (and thus cannot be represented by a stream function) is not straightforward in our framework and requires future research.

5 Computation

This section will explain several practical aspects, such as how to discretize the equations and how to realize a working implementation of our approach.

5.1 Discretization

The computational domain is represented by a regular voxel grid, storing components of $\boldsymbol{\Psi}$ on cell edges, components of $\boldsymbol{u}$ on cell faces, and the scalar field $\hat{\phi}$ on vertices (Figure 3, top). The location of these variables is consistent with discrete exterior calculus [de Goes et al. 2013], which locates 0-forms on vertices, 1-forms on edges, and 2-forms (fluxes) on faces.

In fact, our $[\nabla\times]$ and $[\nabla\cdot]$ can be computed in the same manner as the discrete exterior calculus operators defined by Elcott et al. [2007] for tetrahedral meshes. Our curl operator $[\nabla\times]$ is a rectangular matrix mapping edges to faces. The y component of the

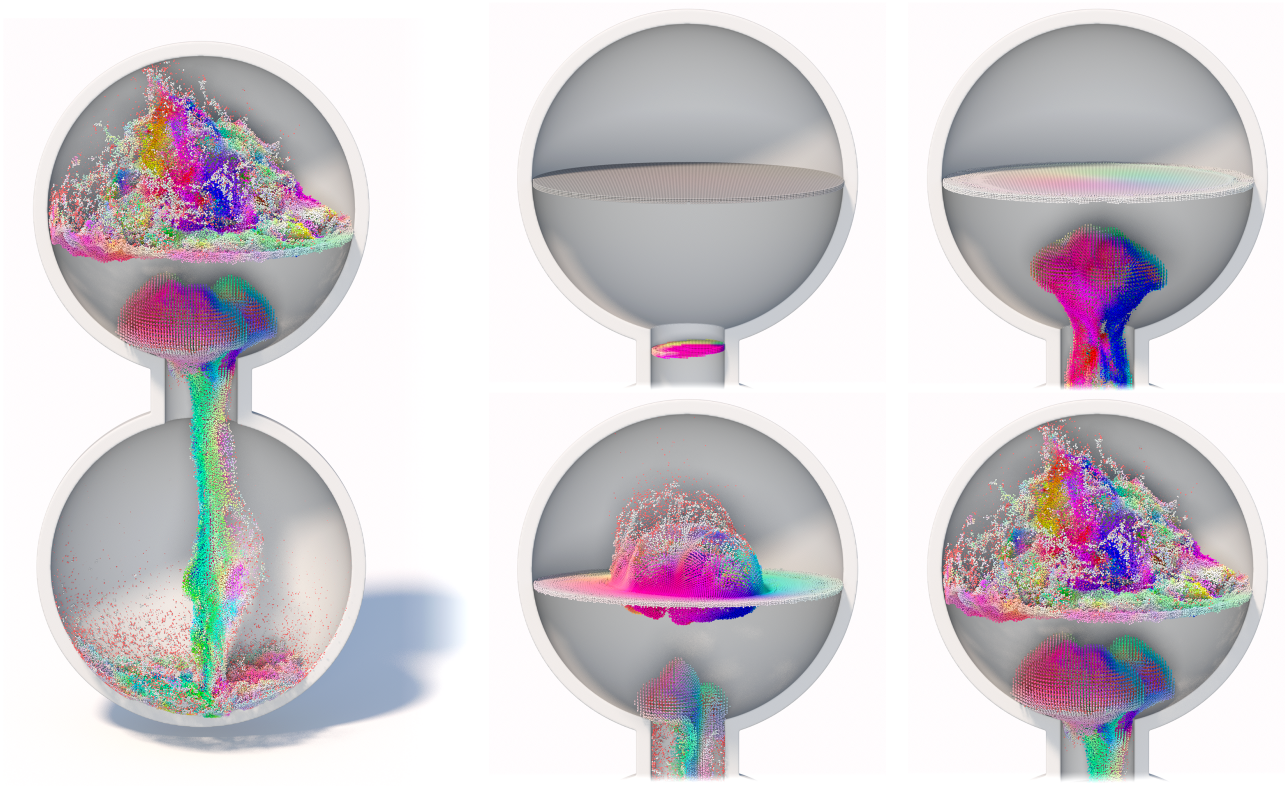

Figure 4: *Liquid glugging down a closed container:* $128\times256\times128$ *resolution, 18 seconds per time step, and 1.5 minutes per video frame.*

curl is defined as $\frac{\partial \Psi_x}{\partial z} - \frac{\partial \Psi_z}{\partial x}$ (Figure 3 bottom left), and the x and z components are defined similarly. The divergence operator $[\nabla\cdot]$ is a rectangular matrix mapping edges to vertices by summing up the vector components on the oriented edges adjacent to each vertex (Figure 3, bottom right).

We also evaluate the fractions of a cell occupied by liquid (for the liquid-air boundary conditions) and solid obstacles (for solid boundary conditions). Our implementation stores different signed distance functions to implicitly represent the liquid and solid surfaces. We store the liquid distance function (the fluid level set) at cell centers. We directly use $[\rho]$ for liquid volume fractions, which is valid if we normalize liquid density to 1 and air density to 0 as explained in Section 3.

While there are many ways to approximate the solid volume fraction [Batty et al. 2007], we use the ghost-fluid-style technique of computing the extent of the fluid on the line segment between cell centers [Gibou et al. 2002]. We store the solid distance function on cell vertices, and we compute the solid area fraction of a face using the area given by a "marching squares"-style contour extraction [Lorensen and Cline 1987] (See Figure 3 middle). We truncate each fraction to zero if it is smaller than 0.01, and we switch to the solid boundary condition from Section 4.2 if the solid fraction is larger than 0.99.

5.2 Tolerating convergence errors

Given the solid boundary conditions described in Section 4.2, we define our stream function as:

$$[\mathbf{\Psi}] = [Z] \begin{bmatrix} \hat{\mathbf{\Psi}} \\ \hat{\phi} \end{bmatrix} \tag{15}$$

where $\hat{\mathbf{\Psi}}$ is the stream function within the liquid, $\hat{\phi}$ is the scalar within the solid boundary, and $[Z]$ is a matrix mapping liquid regions to $\hat{\mathbf{\Psi}}$ and solid regions to the gradient of $\hat{\phi}$. The boundary conditions in Section 4.3 are handled analogously by adding rigid body degrees of freedom.

We can then solve for $[\mathbf{\Psi}]$ using the ideas in Section 3, which works perfectly well when solving the system accurately. However, when allowing for errors, perhaps by terminating the iterative system solve early, the inaccuracies manifest as spurious damping. The iterative solve begins with an initial guess whose curl is zero and converges to the correct solution whose curl is the desired velocity field. Terminating this process early yields a blend between the correct solution and zero motion. To allow for errors without any spurious damping, we reformulate the solution vector as offsets $\Delta\hat{\mathbf{\Psi}}$ and $\Delta\hat{\phi}$ from the vector potential at the previous time step, $\mathbf{\Psi}_{t-\Delta t}$:

$$[\mathbf{\Psi}] = [Z] \begin{bmatrix} \Delta\hat{\mathbf{\Psi}} \\ \Delta\hat{\phi} \end{bmatrix} + [\mathbf{\Psi}_{t-\Delta t}] \tag{16}$$

In addition to avoiding spurious damping, the resulting velocity field $\boldsymbol{u} = \nabla \times \mathbf{\Psi}$ will also be divergence-free by construction, even in the presence of numerical errors.

5.3 Implementation details

The only difference between our method and previous time-splitting algorithms for liquid simulation [Bridson 2008] is the use of a stream function solve instead of a pressure solve; we re-use existing algorithms for advection, extrapolation, etc. Here, we explain how we implement the stream function solver.

Algorithm 1: Stream function projection

input : velocity field after advection $\boldsymbol{u}^*$,
narrow-banded levelset function of fluid θ_F and solid θ_S
output: divergence-free velocity field $\boldsymbol{u}$

if *first time called* **then**
  Pre-compute $[Z]$ from θ_S
  Pre-compute $[P] \leftarrow$ Eq. (17)
  Pre-compute $[B] \leftarrow [Z]^T[\nabla\times]^T$
  Set $\begin{bmatrix}\hat{\boldsymbol{\Psi}}_{t-\Delta t}\\ \hat{\phi}_{t-\Delta t}\end{bmatrix} \leftarrow 0$
Compute diagonal density matrix $[\rho]$ and $[\Delta\rho]$ from θ_F
Compute matrix $[Q] \leftarrow$ Eq. (18)
Compute matrix $[K] \leftarrow [P] + [Q]$
Assemble the linear system in Eq. (19)
Solve the linear system Eq. (19) by PCG
Compute $\begin{bmatrix}\hat{\boldsymbol{\Psi}}_t\\ \hat{\phi}_t\end{bmatrix} \leftarrow \begin{bmatrix}\Delta\hat{\boldsymbol{\Psi}}\\ \Delta\hat{\phi}\end{bmatrix} + \begin{bmatrix}\hat{\boldsymbol{\Psi}}_{t-\Delta t}\\ \hat{\phi}_{t-\Delta t}\end{bmatrix}$
Compute $[\boldsymbol{u}] \leftarrow [\nabla\times][Z]\begin{bmatrix}\hat{\boldsymbol{\Psi}}_t\\ \hat{\phi}_t\end{bmatrix}$

We first compute a matrix $[P]$ which encodes the solid boundaries:

$$[P] = [Z]^T[\nabla\times]^T[\frac{1}{A}][\nabla\times][Z] + [Z]^T[\nabla\cdot]^T[\frac{1}{A}][\nabla\cdot][Z] \quad (17)$$

where $[\frac{1}{A}]$ denotes a diagonal matrix where entries are the reciprocal of the area occupied by liquid (the compliment of the solid area calculated in Section 5.1). Thus, A=1 for regions completely inside the liquid and A=0 inside a solid. As mentioned earlier, we switch to the solid boundary condition in Section 4.2 when A<0.01.

The $[Z]^T[\nabla\cdot]^T[\frac{1}{A}][\nabla\cdot][Z]$ term serves as a regularizer in the original problem. We do not compute it near solid boundaries in order to avoid a denser matrix; this optimization does not change the solution. Away from solid boundaries, $[P]$ is a standard Laplacian matrix $[\nabla^2]$, which can be efficiently encoded. $[P]$ can be pre-computed, but we must reassemble it whenever solid boundaries change. We also pre-compute $[B] = [Z]^T[\nabla\times]^T$ for assembling the right hand side of the final system. We then compute a matrix $[Q]$ to encode density jumps:

$$[Q] = [Z]^T[\nabla\times]^T[\frac{\Delta\rho}{A}][\nabla\times][Z] + [Z]^T[\nabla\cdot]^T[\frac{\Delta\rho}{A}][\nabla\cdot][Z]. \quad (18)$$

which must be re-computed each time step. Note that $[Q]$ is non-zero only in a narrow band around the liquid-air interface. We sum these matrices to get $[K] = [P] + [Q]$. Here, $[K]$ is non-zero only inside the bulk of the liquid, and zero for both air and the static solid boundaries not touching the liquid. The linear system for our stream function is then

$$[K]\begin{bmatrix}\Delta\hat{\boldsymbol{\Psi}}\\ \Delta\hat{\phi}\end{bmatrix} = [B][\rho][\boldsymbol{u}^*] - [K]\begin{bmatrix}\hat{\boldsymbol{\Psi}}_{t-\Delta t}\\ \hat{\phi}_{t-\Delta t}\end{bmatrix}. \quad (19)$$

which we solve for $\Delta\hat{\boldsymbol{\Psi}}$ and $\Delta\hat{\phi}$ each time step. We perform this solve using a preconditioned conjugate gradient (PCG) method[2] [Bridson 2008]. The steps of this stream function projection are reviewed in Algorithm 1.

6 Results

To reduce the overhead for particle tracing, we seed FLIP particles only near the free surfaces, as proposed by Chentanez et al. [2014].

[2]With MIC(0) parameters $\tau = 0.97$ and $\sigma = 1.0$

The PCG solver uses a residual tolerance of $\|r\|_2/\|r^0\|_2 < 10^{-4}$ (with current residual r and initial residual r^0) for all examples. We ran our examples on several different Linux workstations (2.70GHz Intel Xeon E5-2697 or 3.3GHz Intel Core i7-3960X).

The setup of Figure 4 illustrates the effect of an incompressible gas phase around the liquid. Without explicitly simulating the second phase, our solver captures the violent splashes caused by the air moving in the opposite direction of the liquid. In the figures throughout our paper, the particles are color-coded to visualize the three components of the stream function (normalized per frame).

The wall with holes that is shown in Figure 1, exhibits a similar effect: the container behind the wall is closed, so that air has to enter through the same holes the liquid tries to escape from. For this example, the matrix assembly per time step took 11.5 seconds on average, while the PCG solver took 16 seconds. A regular pressure solve results in a comparatively boring pouring motion for this example (shown in the accompanying video). Note that this is an example of a high-genus geometry, and the velocity field can potentially have a non-zero harmonic component γ. Our result illustrates that flows without visual artifacts are possible even when setting γ to zero. To illustrate that our method results in flows that settle down over time, the energy of this simulation is shown in Figure 8.

The simulation of Figure 6 highlights the capabilities of our solver to preserve the gas phase explicitly: a single large bubble rises in a long tube of liquid. The complex deformations and breakup effects are captured by the divergence-free motion our stream function solver computes for the region around the liquid.

Figure 5 highlights the ability of our stream function solver to handle non-grid-aligned solid boundaries, a two-way coupled rigid body, and bubbles in a strongly coupled manner. As the rigid body coupling leads to residual terms for the fluid and rigid bodies being combined in one vector, we modified our PCG settings for this example: we use an accuracy threshold of $\|r\|_\infty < 10^{-1}$, and we perform at most 10^3 iterations.

We compared our algorithm with a naive two-phase flow algorithm (a FLIP-based variable-density pressure solver without any special treatment of the liquid-air density jump) in Figure 7. A density disparity of 1000× in the naive solver contributes to a stiffer and less stable linear system, causing very high CFL numbers, slower solver convergence, and errors in volume conservation. The FLIP advection also experiences unphysical shearing and a visibly disturbing "mist" of particles around the liquid-air interface. In contrast, our method remains stable and treats the interface appropriately without any special treatment of FLIP particles.

As discussed in Section 5.2, our method produces divergence-free velocity fields regardless of the accuracy of the stream function solve. Indeed, our supplementary comparison video shows how our method gets less accurate as the solve accuracy decreases, while still generating divergence-free velocities.

7 Discussion and Limitations

The linear system Eq. (19) at the heart of our method solves a vector-valued Poisson equation, so it has about three times more unknowns than a standard pressure solver. Unlike some previous work for smoke simulation, our free surface and solid boundary conditions strongly couple the three stream function components together, and we were not able to split the system into three independent scalar Poisson equations. Consequently, our method is more expensive than a standard pressure solver. However, we noticed that the change of variables in Section 5.2 dramatically increased efficiency by only computing the deviation from a previous

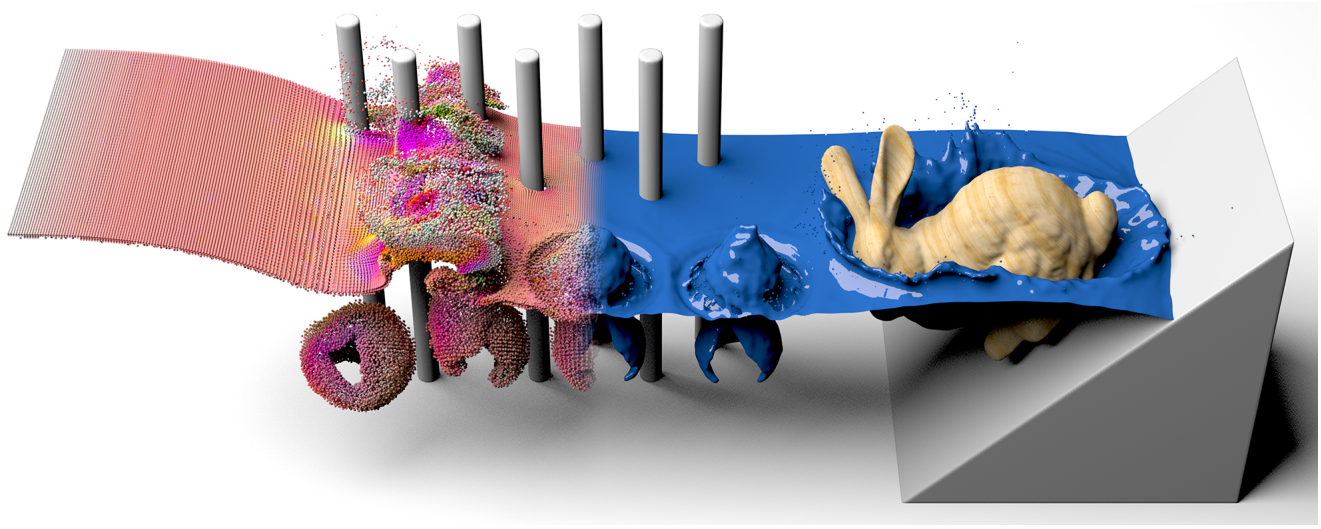

Figure 5: *Interaction with a two-way coupled rigid body:* $256\times128\times64$ *resolution, 49 seconds per time step, and 1.5 minutes per frame.*

solution. The entire stream function projection (including matrix assembly and PCG) was 5.6 times slower than a regular single-phase pressure projection. Ultimately, one whole time-step of our method was about 3.0 times slower (on average) than a time-step of a solver using a regular pressure projection.

It is difficult to compare the timings of our method directly with a two-phase flow solver, because setting the air density to zero (as we do) makes such models unstable. When using a finite air density in a two-phase flow model, large density jumps induce large CFL numbers near the interface, which must be treated with a careful choice of smaller time steps. In addition, the size of the computational domain can become arbitrarily different from ours, as we only simulate the liquid phase.

The introduction of our divergence regularization term in Eq. (9) is likewise motivated by the runtime of our method. Omitting this term would make the linear solver converge poorly, or not converge at all. Another practical consequence of this specific regularizer is that makes the matrix significantly sparser within the liquid.

As discussed in Section 4.4, our strategy of encoding the obstacle motion with a stream function limits us to divergence-free velocity fields. We will leave it for future work to extend these boundary conditions to more general cases, e.g., arbitrarily deforming elastic bodies. Compressible flows cannot be handled directly by our stream function solver and require the treatment of an additional gradient term in the Helmholtz-Hodge decomposition (Eq. (5)). We also have not yet addressed inflow/outflow boundary conditions, but we believe this should be possible in our method provided that the boundary conditions integrate to zero total flux and thus conserve volume. It may also be possible to sidestep these restrictions by making the flow locally compressible.

We believe one of the most interesting aspects of this work is the insight that stream-function solves are a viable alternative to regular pressure solves. Furthermore we have demonstrated that, contrary to popular opinion, free-surface boundary conditions can be seamlessly integrated into a stream-function solver. Having the stream function available during fluid simulations has interesting implications. For example, the velocity computed from the stream function is by construction always divergence-free up to the numerical precision of the curl operator. Thus, it would be possible to construct interpolation schemes that are guaranteed to uphold this property, in contrast to interpolations of velocity data, which currently introduce divergence errors.

8 Conclusion and Outlook

We have presented a stream function approach to enforcing incompressibility in a liquid simulation. Our approach has many theoretical benefits, including guaranteed divergence-free velocity fields regardless of the accuracy of the solve, and a straightforward extension to rigid-body coupling. Our novel variable-density formulation yields a dramatic drop in computation in the limit that air density tends to zero, while still satisfying the incompressibility constraint everywhere.

In the future we would like to generalize the circulation-preserving advection algorithm of [Elcott et al. 2007] to work with density jumps. We may also be able to only track fluid variables at the liquid-air interface. Combining such a strategy with an analytical stream function could allow large-scale two-phase liquid simulations without any volumetric computations.

9 Acknowledgements

The first author was supported by a JSPS *Postdoctoral Fellowship for Research Abroad*. This work was also supported by the ERC projects ERC-2014-StG-637014 *realFlow* and ERC-2014-StG-638176 *BigSplash*. Additionally, we want to thank Reiji Tsuruno for providing us with computational resources, Keenan Crane for discussions about the Hodge decomposition, Florian Ferstl for discussions about FLIP, and the anonymous reviewers for their helpful comments.

References

ANDO, R., THUEREY, N., AND WOJTAN, C. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. 32*, 4, 103.

ANGELIDIS, A., AND NEYRET, F. 2005. Simulation of smoke based on vortex filament primitives. In *Proceedings of the Symposium on Computer Animation*, ACM, 87–96.

BARRAGY, E., AND CAREY, G. 1997. Stream function-vorticity driven cavity solution using p finite elements. *Computers and Fluids 26*, 5, 453–468.

Figure 6: *An initially cubical bubble rises up (cutaway view):* $128\times256\times128$ *resolution, 52 seconds per time step, and 1.9 minutes per video frame.*

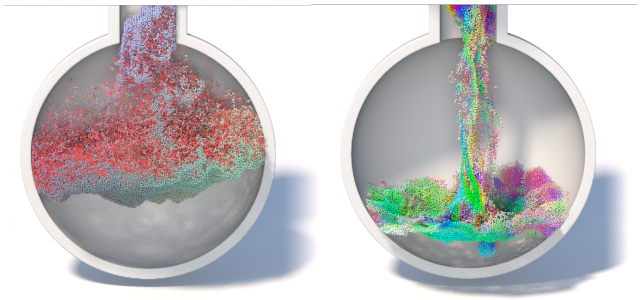

Figure 7: *A two-phase FLIP solver with variable density (with densities of* $\rho_{liquid} = 1$ *and* $\rho_{air} = 1/1000$*) quickly leads to very high velocities and a diffuse interface. In comparison, our solver keeps the air phase divergence free and clearly separated (shown on the right).*

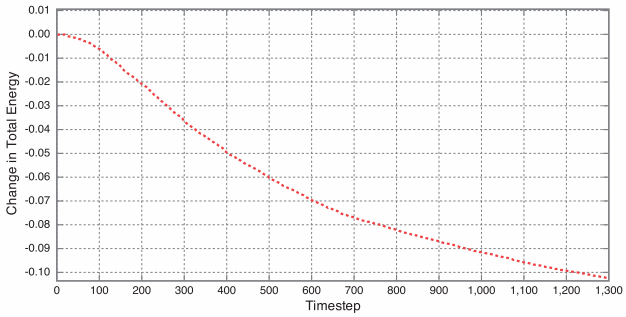

Figure 8: *This graph shows the change in total energy (kinetic plus potential energy) of the simulation from Figure 1 over time. The flow settles, similar to simulations with pressure projection solvers.*

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (July).

BATTY, C., XENOS, S., AND HOUSTON, B. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Proceedings of Eurographics*.

BOYD, L., AND BRIDSON, R. 2012. Multiflip for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2, 16:1–16:12.

BRECHT, S. H., AND FERRANTE, J. R. 1989. Vortex-in-cell simulations of buoyant bubbles in three dimensions. *Physics of Fluids A: Fluid Dynamics (1989-1993) 1*, 7, 1166–1191.

BRIDSON, R., HOURIHAM, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (TOG) 26*, 3, 46.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press.

BROCHU, T., KEELER, T., AND BRIDSON, R. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, Eurographics Association, 87–95.

BUDSBERG, J., LOSURE, M., MUSETH, K., AND BAER, M. 2013. Liquids in "The Croods". In *ACM SIGGRAPH Digital Production Symposium (DigiPro)*.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 377–384.

CHENTANEZ, N., MLLER, M., AND KIM, T.-Y. 2014. Coupling 3D Eulerian, Heightfield and Particle Methods for Interactive Simulation of Large Scale Liquid Phenomena. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, The Eurographics Association, V. Koltun and E. Sifakis, Eds.

DA, F., BATTY, C., AND GRINSPUN, E. 2014. Multimaterial mesh-based surface tracking. *ACM Trans. on Graphics (SIGGRAPH 2014)*.

DE GOES, F., CRANE, K., DESBRUN, M., SCHRÖDER, P., ET AL. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*, ACM, 7.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG) 26*, 1, 4.

ENGLISH, R. E., QIU, L., YU, Y., AND FEDKIW, R. 2013. Chimera grids for water simulation. In *Proceedings of the Symposium on Computer Animation*, 85–94.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free-Surface Flows. *Proc. of the 4th ASME-JSME Joint Fluids Engineering Conference*.

FERSTL, F., WESTERMANN, R., AND DICK, C. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE Trans. Vis. Comput. Graph. 20*, 10, 1405–1417.

FOSTER, N., AND METAXAS, D. N. 1997. Controlling fluid animation. In *Computer Graphics International*, 178–188.

G. GUJ, F. S. 1993. A vorticity-velocity method for the numerical solution of 3d incompressible flows. *Journal of Computational Physics 106*, 2, 286–298.

GAMITO, M. N., LOPES, P. F., AND GOMES, M. R. 1995. Two-dimensional simulation of gaseous phenomena using vortex particles. In *In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*, Citeseer.

GIBOU, F., FEDKIW, R. P., CHENG, L.-T., AND KANG, M. 2002. A second-order-accurate symmetric discretization of the poisson equation on irregular domains. *Journal of Computational Physics 176*, 1, 205–227.

GOLAS, A., NARAIN, R., SEWALL, J., KRAJCEVSKI, P., DUBEY, P., AND LIN, M. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. 31*, 6 (Nov.), 148:1–148:9.

HARLOW, F., AND WELCH, E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids 8 (12)*, 2182–2189.

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, SIGGRAPH '05, 915–920.

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH fluids in computer graphics. In *Eurographics - State of the Art Reports*, Eurographics Association, 21–42.

KANG, N., PARK, J., YONG NOH, J., AND SHIN, S. Y. 2010. A Hybrid Approach to Multiple Fluid Simulation using Volume Fractions. *Computer Graphics Forum 29*, 685–694.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3, 820–825.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '87, 163–169.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graph. 25*, 3, 812–819.

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. H. 2005. Particle-based fluid-fluid interaction. In *Proceedings of the Symposium on Computer Animation*, ACM, 237–244.

PFAFF, T., THUEREY, N., AND GROSS, M. H. 2012. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph. 31*, 4, 112.

REN, B., LI, C., YAN, X., LIN, M. C., BONET, J., AND HU, S. 2014. Multiple-fluid SPH simulation using a mixture model. *ACM Trans. Graph. 33*, 5, 171.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. 24*, 3 (July), 910–914.

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH '99*, ACM, 121–128.

STOCK, M. J., DAHM, W. J., AND TRYGGVASON, G. 2008. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. *Journal of Computational Physics 227*, 21, 9021–9043.

VINES, M., HOUSTON, B., LANG, J., AND LEE, W.-S. 2014. Vortical inviscid flows with two-way solid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 20*, 2 (Feb.), 303–315.

WANG, S., AND ZHANG, X. 2011. An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows. *Journal of Computational Physics 230*, 9, 3479–3499.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics (TOG) 29*, 4, 115.

WONG, A. K., AND REIZES, J. A. 1984. An effective vorticity-vector potential formulation for the numerical simulation of three-dimensional duct flow problems. *Journal of Computational Physics 55*, 98–114.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2.