Efficient Liquid Animation: New Discretizations for Spatially Adaptive Liquid Viscosity and Reduced-Model Two-Phase Bubbles and Inviscid Liquids

by

Ryan Goldade

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Nils Thürey Associate Professor, Technische Universität München
Supervisor(s):	Christopher Batty Associate Professor, Dept. of Mathematics, University of Waterloo
Internal Member:	Justin Wan Professor, Dept. of Mathematics, University of Waterloo
Internal Member:	Stephen Mann Professor, Dept. of Mathematics, University of Waterloo
Internal-External Member:	Hans De Sterck Professor, Dept. of Applied Mathematics, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis consists of research material written for publication where Ryan Goldade was the lead author under the supervision of Dr. Christopher Batty and in collaboration with Yipeng Wang and Dr. Mridul Aanjaneya. Research presented in Chapters 4 and 5 largely consists of contribution and results sections adapted from Goldade et al. (2020) and Goldade et al. (2019), respectively.

Chapters 1, 2, 3, and 6 consist of a combination of sole-authored content by Ryan Goldade and a curation and paraphrasing of material from Goldade et al. (2019, 2020), adapted to the narrative structure of a PhD thesis.

Abstract

The work presented in this thesis focuses on improving the computational efficiency when simulating viscous liquids and air bubbles immersed in liquids by designing new discretizations to focus computational effort in regions that meaningfully contribute to creating realistic motion. For example, when simulating air bubbles rising through a liquid, the entire bubble volume is traditionally simulated despite the bubble's interior being visually unimportant. We propose our *constraint bubbles* model to avoid simulating the interior of the bubble volume by reformulating the usual incompressibility constraint throughout a bubble volume as a constraint over only the bubble's surface. Our constraint method achieves qualitatively similar results compared to a two-phase simulation ground-truth for bubbles with low densities (e.g., air bubbles in water). For bubbles with higher densities, we propose our novel *affine regions* to model the bubble's entire velocity field with a single affine vector field. We demonstrate that affine regions can correctly achieve hydrostatic equilibrium for bubble densities that match the surrounding liquid and correctly sink for higher densities. Finally, we introduce a tiled approach to subdivide large-scale affine regions into smaller subregions. Using this strategy, we are able to accelerate single-phase free surface flow simulations, offering a novel approach to adaptively enforce incompressibility in free surface liquids without complex data structures.

While pressure forces are often the bottleneck for inviscid fluid simulations, viscosity can impose orders of magnitude greater computational costs. We observed that viscous liquids require high simulation resolution at the surface to capture detailed viscous buckling and rotational motion but, because viscosity dampens relative motion, do not require the same resolution in the liquid's interior. We therefore propose a novel adaptive method to solve free surface viscosity equations by discretizing the variational finite difference approach of Batty and Bridson (2008) on an octree grid. Our key insight is that the variational method guarantees a symmetric positive definite linear system by construction, allowing the use of fast numerical solvers like the Conjugate Gradients method. By coarsening simulation grid cells inside the liquid volume, we rapidly reduce the degrees-of-freedom in the viscosity linear system up to a factor of 7.7× and achieve performance improvements for the linear solve between 3.8× and 9.4× compared to a regular grid equivalent. The results of our adaptive method closely match an equivalent regular grid for common scenarios such as: rotation and bending, buckling and folding, and solid-liquid interactions.

Acknowledgements

There are many people I would like to thank who helped me through the journey of this degree.

First and foremost, I would like to thank my supervisor, Christopher Batty. I owe a lot of my success to your encyclopedic knowledge of the research and endless source of brilliant ideas. You have been an incredible mentor and a friend and it has been an honour to work with you.

I have had the great fortune of working with many amazing people during my PhD. My adventure into computer graphics began during my Masters when Todd Keeler, Robert Bridson and Microsoft Canada took a chance on a sonar engineer. This opportunity changed my entire life and I am forever grateful. I cannot thank you enough. I would also like to thank Chris Wojtan for the opportunity to intern at IST Austria and for our collaboration that resulted in my first publication. Thank you to the wonderful people at SideFX. I learned so much during my time there and it helped guide my research. I would also like to thank the Simulation R&D team at Weta Digital. Although it was only a short internship, it was a great experience both professionally and personally. To Yipeng Wang, I really enjoyed working with you on our adaptive viscosity project and our carpool discussions about life and our careers. Finally, I would like to thank Mridul Aanjaneya. It has been an absolute joy collaborating with you over many research projects.

I would like to thank all the fellow graduate students in Sci-Com. Specifically, Parsiad Azimzadeh who I learned a lot of maths from and shared even more laughs with.

Thank you to my committee members for your time and valuable feedback.

I would like to thank my friends and family for their love and support.

To my wife, Monica, I couldn't have powered through those initial paper rejections without your endless support, and encouragement.

Dedication

To my grandmother, Vivian Davis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the major motivations in the field of computer graphics is the desire to recreate photorealistic environments, including their motion. Creating realistic simulations can often avoid expensive and dangerous practical effects. For example, filming a boat in a raging storm or a plane engine catching fire in mid-flight is far too dangerous given the alternatives of modern technology. Furthermore, since such simulations are too complex for an artist to generate by hand, they must rely on computational physical models to make materials in the scene move and behave realistically. Simulation research is therefore quickly adopted by the film, TV, and video game industries.

Fluid simulation has been a staple topic in computer graphics for several decades. Pioneering work achieved plausible motion of smoke and water and set the foundation for the computer graphics branch of computational fluid dynamics (CFD) (Foster and Metaxas, 1996; Stam, 1999). Due to obvious safety concerns, CFD applications in engineering primarily focus on the accuracy of both the physical model and the numerical methods used to solve them. However, computer graphics applications are, in general, principally concerned with qualitatively accurate behaviour and artistic control. Artists will often run multiple iterations of a simulation to achieve a desired look, driving the need for a fast turnaround of a given simulation. Given these different requirements in computer graphics, computational efficiency and stability for large simulation time steps are typically prioritized over accuracy, insofar as the simulation remains qualitatively believable. As an example, fluids are often assumed to be incompressible, eschewing the need for capturing shock waves and other complex phenomena that are computationally expensive and do not significantly improve visual quality in the vast majority of practical scenarios. While this assumption is not strictly true in the physical world, it is a reasonable assumption for many of the types of fluid behaviour that artists endeavour to create. On the other hand, purely

ad hoc or non-physical procedural methods break down for non-trivial three-dimensional scenarios, so we prefer to design simplified models that are still grounded in the classical Navier-Stokes model of fluid flow. Fundamentally, we follow the adage *everything should be made as simple as possible, but no simpler*, where we are motivated to develop the simplest model possible that still achieves qualitative believability. Despite invoking assumptions to simplify the model, this balance between realism, efficiency, stability and even artist controllability has made for a complex field of study. Indeed, even two decades since its inception the research community is still working to recreate phenomena found in the real world, in addition to designing new methods to improve simulation accuracy and efficiency.

The computational effort in a standard fluid simulation pipeline is largely focused on solving for two fundamental fluid forces: pressure and viscosity. Pressure is essential for enforcing incompressibility and interactions with solid boundaries; viscous forces are essential for capturing damped relative motion expected from liquids like honey. Solving these forces efficiently offers the artist faster turnaround time to iterate on the desired effect or, alternatively, to simulate at higher resolutions. Our research contributions focus on improving simulation efficiency of two phenomena: viscous liquids and bubbles immersed in liquids. Both phenomena are ubiquitous in everyday life and yet they are extremely costly to simulate using existing methods. Furthermore, one of our improvements to bubble simulation relies on a new reduced model, and we subsequently show that this model can also be applied as a general adaptive strategy to efficiently solve for pressure forces in the traditional *free surface* liquid simulation. Combined, our research contributions offer substantial performance improvements when solving pressure and viscous forces in an industrial fluid simulation pipeline.

1.1 Thesis Contributions

1.1.1 Reduced-Model Bubbles and Liquids

For a simulation to capture the *glugging* behaviour of liquid pouring out of a bottle or entrained air bubbles formed from splashy flows, the interaction between immiscible air bubbles and the surrounding liquid must be accounted for in the physical model of the fluids' pressures. However, modelling this two-phase flow phenomena is computationally expensive. Water and air differ in density by about three orders of magnitude, leading to ill-conditioned linear systems that strain numerical solvers (MacLachlan et al., 2008). It is therefore standard in computer graphics to ignore the air's physics altogether and assume a *free surface* boundary condition at the air-liquid interface (Bridson, 2015). Unfortunately,

this treats air as a massless void that leads to bubbles collapsing when entrained by the liquid, because there is no opposing force preserving its volume. Some previous methods simplify the air-volume's physical model by either enforcing incompressibility in the air implicitly through a costly liquid-only stream function model (Ando et al., 2015a), or explicitly with a weakly coupled compressible model that still requires advecting and solving for pressures (i.e., pressure projection) throughout the entire air volume (Aanjaneya et al., 2013; Patkar et al., 2013). We address this challenge with two new models: *constraint bubbles* and *affine regions*.

We propose a constraint-based model for negligible-density *air* bubbles that considers only the net flux into (and out of) a given bubble. This model entirely eliminates both advection and pressure projection inside the bubble volume and requires just one Lagrange multiplier per distinct bubble. This surface-only approach introduces minimal additional overhead compared to a free surface method, easily integrates with existing free surface flow solvers, and realistically captures many familiar bubble behaviors. It can even allow for two or more distinct liquid bodies to correctly interact across completely unsimulated air, such as water volumes separated by air in a tube, which is not possible with prior bubble schemes (Aanjaneya et al., 2013; Ando et al., 2015a). Additionally, we augment our constraint model with a per-bubble volume-tracking and correction framework to minimize the effects of gradual volume drift.

Although our constraint-based model offers a practical solution to efficiently simulate negligible-density bubbles interacting with liquids, the model fails for non-negligible densities. For example, a stationary bubble with matching density to the surrounding liquid should experience neutral buoyancy and remain stationary; a bubble with a higher density should sink. However, our experiments demonstrate that the constraint method cannot capture this behaviour because the single *pseudo-pressure* constraint throughout the bubble is an insufficient model for larger bubble densities.

While nearly zero-density bubbles is a common situation, non-zero density coefficients also arise in many compelling two-phase scenarios. Therefore, we also propose a novel reduced model for an irregularly shaped fluid region over which we assume a single pointwise incompressible affine vector field. This approach requires only 11 interior velocity degrees-of-freedom per affine fluid region in 3D, and naturally handles the general case of non-negligible density coefficients. Applied to the interior of a secondary fluid phase immersed in water, our algorithm allows a sphere of neutrally buoyant (i.e., matching density) immiscible liquid to remain perfectly stationary, while a heavier immiscible liquid sphere correctly sinks. Thus, constraint bubbles and affine regions provide complementary treatments for two-phase scenarios: constraint bubbles offer a simple and efficient mechanism for zero-density bubbles, while affine regions accelerate the traditional two-phase approach (Kang et al., 2000; Hong

and Kim, 2005) for non-negligible density settings by significantly reducing the number of interior degrees-of-freedom.

Additionally, by enforcing an incompressible affine vector field over a coalesced set of grid cells, we have effectively constructed an *irregular coarse super-cell*. Similar in spirit to prior schemes exploiting tall-cell grids (Irving et al., 2006; Chentanez and Müller, 2011), stretched grids (Zhu et al., 2013), and octree grids (Losasso et al., 2004), this technique offers a convenient and flexible coarsening strategy that integrates readily with the usual staggered uniform regular grid; yet unlike its predecessors, it supports coarsened regions that are *arbitrary voxelized shapes* and provides an *analytically divergence-free interior*. Compared to octree, tetrahedra, or nested grid adaptivity schemes, our method requires simpler data structures and minimal overhead, and is more flexible than axially stretched or tall-cell approaches. We demonstrate its effectiveness with a new adaptive free-surface liquid solver whose interior affine regions are coarsened into a mix of tiles with regular and irregular shapes.

The pressure projection step in all of the above schemes can be reduced down to solving a symmetric and positive definite (SPD) linear system, which suggests the potential to develop fast numerical solvers. Therefore, to achieve efficiency in practice, we propose several Multigrid-based preconditioning schemes that are tailored to the particular characteristics of our new discretizations.

To summarize, our primary contributions for reduced-model two-phase and single-phase inviscid flow are:

- A constraint-based model for immersed zero-density bubbles, that entirely avoids advection and pressure projection inside the bubble,
- A reduced fluid model based on pointwise divergence-free affine vector fields that supports irregularly-shaped regions,
- A tile-based strategy for flexible and adaptive spatial coarsening that is built on top of the above affine model, with applications to single- and multi-phase flow simulations,
- Efficient Multigrid-based preconditioners for the Conjugate Gradients method that are tailored to the preceding models, and
- A region-tracking and volume-correction strategy to compensate for drift in multi-component, topology-changing flows.

1.1.2 Adaptive Viscosity

We observe that liquids with high viscosity coefficients can be slow to simulate with standard uniform resolution simulators, where solving viscous forces comprises up to 95% of the total simulation time. Compared to computing pressure forces on the same domain, viscous forces can often be slower by an order of magnitude or more. There are two main reasons why solving for the effects of viscosity is slow: large viscosity coefficients give rise to stiff linear systems that are generally slower to solve, and the boundary conditions necessary for plausible free surface behavior couple the different components of velocity together, yielding a system that is three times larger and contains twice as many non-zeros per row (Batty and Bridson, 2008). This computational bottleneck discourages artists from simulating viscous liquids at high resolutions, and in many cases, from simulating viscous liquids altogether.

Additionally, we observe that viscous simulations require a high simulation resolution near the liquid surface to capture fine-scale details of buckling and rotational motion and interactions with solid objects. However, because viscosity dampens relative motion, the velocity field deeper within a viscous liquid is generally quite smooth and requires less simulation resolution. Therefore, we propose to accelerate solving viscous forces by using *spatial adaptivity* to focus simulation resolution at the liquid surface and solid boundaries, and applying grid coarsening into the liquid's interior.

Although adaptivity has been extensively explored for solving liquid pressure forces (Losasso et al., 2004, 2006; Klingner et al., 2006; Chentanez et al., 2007; Batty et al., 2010; Brochu et al., 2010; Ando et al., 2013; Ferstl et al., 2014; Setaluri et al., 2014; Aanjaneya et al., 2017), there is surprisingly little previous work in computer animation considering adaptive Eulerian viscosity (Hong and Kim, 2005; Batty and Houston, 2011). Unfortunately, the viscosity model used by Hong and Kim (2005) does not support realistic rotational or bending motion achieved in the regular grid method of Batty and Bridson (2008), and the method of Batty and Houston (2011) uses tetrahedral meshes, which imposes significant additional overhead compared to modern octrees (Setaluri et al., 2014). Therefore, we propose and evaluate an efficient and practical adaptive viscosity solver for octree-based liquids that supports free surfaces and variable viscosity, while being geometrically compatible with the classic inviscid octree simulator of Losasso et al. (2004, 2006).

Our main contribution is the development and validation of a new *adaptive* variational finite difference methodology for fluids, with solving viscous forces considered as a specific case study. The regular grid variational finite difference framework was first proposed by Batty et al. (2007) for solving pressure forces and solid-fluid coupling; this approach has since been applied to a variety of uniform resolution fluid problems (Batty and Bridson, 2008;

Narain et al., 2010; Ando et al., 2013, 2015b; Larionov et al., 2017). Our novel generalization to the octree setting requires two core enhancements near T-junctions (i.e., borders between differing octree grid resolutions): first, the definition of modified sample points and control volumes for discretely approximating the necessary integrals; and second, the careful design of accurate adaptivity-aware finite difference stencils for derivative operators appearing in the problem's variational form (in our case, stencils for velocity gradients). Compared to direct finite difference/volume discretizations, our approach guarantees SPD linear systems by construction. Compared to a (hypothetical) finite element alternative, our method yields much sparser linear systems and reduces back to simple finite differences in uniform regions. Furthermore, we confirm the accuracy of the proposed method with comparisons against regular grids and with refinement studies showing velocity convergence at first order in L_∞ and second order in L_1. Although popular octree pressure projection schemes in graphics offer second order accuracy in pressure, their velocity accuracy is only first order (Losasso et al., 2006; Aanjaneya et al., 2017); our viscosity solver therefore offers comparable or better convergence rates.

We demonstrate the practical benefits of our octree viscosity by replacing the viscosity step of a standard regular grid simulator (Houdini) with our octree viscosity solver. Our solver offers speed-up factors up to 9.4× for the linear solve and 8.8× for a full viscosity step, and enables simulation at much higher resolutions than is currently feasible using regular grids (Batty and Bridson, 2008; SideFX, 2021).

To summarize, our primary contributions to viscous liquid animation are:

- The introduction of a novel adaptive variational finite difference methodology for octrees that guarantees symmetry,
- An efficient octree viscosity solver based on this methodology that handles free surfaces and variable coefficients,
- Numerical experiments confirming convergence of the discretization under spatial refinement, and
- The application of our method to dramatically increase the speed or resolution of viscous flow simulations produced with a commercial regular grid simulator.

Chapter 2

Related Work

Our work focuses entirely on grid-based fluid simulation methods for both viscous and inviscid fluids. Grid-based methods are popular because the regular grid structure offers algorithmic efficiencies over unstructured approaches like tetrahedral meshes or particle-based methods, in addition to simpler discretizations of the differential operators. Because the broader literature in CFD is vast, we focus our discussion primarily on computer animation techniques. Where appropriate, however, we also highlight methods from computational physics that are most directly relevant to our contributions. We introduce foundational concepts for grid-based fluid simulation in Chapter 3 and highlight selected related work below.

2.1 Free Surface Flow

Foster and Metaxas (1996) presented the first 3-D liquid (i.e., free surface flow) simulation method to the computer graphics community. Their proposed method employs the staggered marker-and-cell (MAC) grid (Harlow and Welch, 1965) that has become the industry standard. Stam's (1999) follow up work applied Chorin-style operator splitting (Chorin and Marsden, 1993) to solve the Navier-Stokes equations, added unconditional stability using a semi-Lagrangian backtracing method for advection, and a time-implicit integration scheme for solving viscous forces. Foster and Fedkiw (2001) and Enright et al. (2002) extended the method of Stam (1999) to liquid animation by combining liquid *marker* particles and level sets to track the liquid surface. Because the liquid-air boundary can move freely during a liquid animation, directly discretizing fluid pressures on the MAC grid will not correctly capture the position of the liquid boundary and results in visible artifacts on the

liquid surface. Gibou et al. (2002) introduced the ghost fluid method to properly capture the liquid boundary inside a grid cell, achieve second order accurate pressures and, more importantly, prevent grid artifacts from forming on the liquid surface. The semi-Lagrangian backtracing method for advection offers unconditional stability over explicit integration with finite differences, but the frequent interpolation required introduces significant numerical viscosity and results in visible damping of the fluid motion. Fedkiw et al. (2001) tackled this damping in smoke simulation by reinjecting lost vorticity into the simulation. Zhu and Bridson (2005) reduced damping in incompressible liquid motion by employing the fluid-implicit-particle (FLIP) method (Brackbill and Ruppel, 1986): an extension of the traditional particle-in-cell (PIC) method (Harlow and Welch, 1965) that stores velocities on fluid particles and offsets these particle-based velocities by the *change* of velocities during the grid-based pressure projection step. The FLIP method tracks liquid regions using particles and advects particles forward in time through the projected, divergence-free grid-based velocity field. Because FLIP only interpolates the *change* in the velocity field on the grid to the particles, only the change itself undergoes numerical damping. Indeed, the FLIP method was so successful at removing numerical damping that it is still the industry standard after a decade and half and we use it for all our results. For a thorough survey of the standard fluid simulation pipeline, please refer to Bridson's textbook (Bridson, 2015).

2.2 Multiphase Flow

2.2.1 Ghost Fluid Method

Popular immiscible two-phase flow methods in computer graphics are primarily derived from the boundary condition-capturing approach of Kang et al. (2000). This approach simulates both air and liquid regions, enforcing incompressibility through a pressure projection scheme that uses a ghost fluid method to model the density discontinuity at the liquid-air interface. Hong and Kim (2005) first made use of this scheme, with a variety of subsequent enhancements following later (Losasso et al., 2006; Mihalef et al., 2006; Kim et al., 2007; Boyd and Bridson, 2012). The work of Kim et al. (2007) is particularly relevant as it focuses on animating bubbles, but it differs from our work in that their air bubbles are all fully simulated. In contrast to these sharp interface approaches, authors such as Song et al. (2005) and Zheng et al. (2006) have used a continuous variable-density pressure solve to simulate multiphase flow, also referred to as a diffuse interface approach.

2.2.2 Sub-grid Bubbles

The use of secondary sub-grid scale particles is another natural way to add bubble details to free-surface flows; an early example is the work of Greenwood and House (2004). More recent instances of this strategy include the work of Hong et al. (2008), Kim et al. (2010) and Busaryev et al. (2012). An interesting recent hybrid is the approach of Patkar et al. (2013), which unifies the treatment of sub-grid and grid-scale compressible bubbles to allow tiny bubbles to oscillate and also coalesce into larger ones. Stomakhin et al. (2020) alternatively model incompressible, sub-grid bubbles using a drag model and momentum exchange with the surrounding bulk fluid. An approach along these lines is likely compatible with our method, but we restrict our attention to grid-resolvable bubbles.

2.2.3 Augmented Free Surface

Our work is closely related to bubble simulation methods that augment a free-surface flow solver with partially decoupled or fully unsimulated grid-scale bubbles (i.e., no computation of fluid equations in the interior of a bubble). Aanjaneya et al. (2013) proposed an equation-of-state approach to simulate two-way coupling of an incompressible liquid to a compressible, fully simulated air phase. They also suggested a simplified variant that assumes constant pressure in the air phase to approximate each bubble's influence with a single pressure degree of freedom (DOF) and thereby partially decouple the air phase. This approach produces a linear system for liquid incompressibility with a similar structure to our constraint bubbles. However, their method involves extra terms to support air compressibility and assumes that bubbles possess non-negligible air mass that must be tracked. This mass tracking necessitates a secondary pressure projection within each bubble and conservative advection for the air mass. As such, the method's computational cost scales with the full domain volume, whereas our constraint-bubbles method scales with the liquid volume. Furthermore, despite air density being present in the equations, we show later that Aanjaneya et al.'s (2013) constant pressure model fails unless the bubbles have densities close to zero. Lastly, while accurate bubble oscillations are critical to sound generation (e.g., (Zheng and James, 2009; Langlois et al., 2016)), they are irrelevant for a wide range of strictly visual applications, so we prefer a fully incompressible treatment.

Ando et al. (2015a) proposed a *stream function* approach for free-surface flows which reformulates the pressure projection step in terms of a vector potential. Standard vector calculus identities ensure that this representation provides incompressible velocities for the air by construction, even while assuming an air density of zero and without simulating air at all. This approach is quite elegant, yet potentially less attractive in practice for a

few reasons. First, the stream function approach entails a radically different and relatively complex discretization compared to standard solvers, requiring many standard fluid solver features be re-developed from the ground up. Second and more fundamentally, because the stream function is a three-component vector, the resulting linear systems are vector Poisson equations three times as large as the usual scalar Poisson equations for pressure projection, and are therefore significantly slower to solve. The method we propose instead requires only one extra DOF per bubble and a small additional computational cost over a standard pressure projection. Our constraint-bubble method also supports at-a-distance interactions with solids or between two liquid bodies, mediated only through the unsimulated air. This is not naturally supported by the stream function approach, since the air region contains no stream function DOFs that could enable the disjoint bodies to communicate.

2.3 Volume Correction

Small numerical errors in surface tracking and incompressibility inevitably lead to volume changes in long-term liquid simulation. Kim et al. (2007) first proposed to use divergence sources (Feldman et al., 2003) in a PID-style controller to recover lost volume. In particle-based models, direct particle position/density corrections have also been incorporated (Losasso et al., 2006; Takahashi and Lin, 2019; Kugelstadt et al., 2019). In the absence of particles, global corrections can lead to the wrong material component being adjusted (i.e., volume lost from one droplet spuriously being added to another). In a mesh-based surface tracking context, Thürey et al. (2010) used explicit topological change information to update per-component volume targets. In a volume-of-fluid (VOF) context, Langlois et al. (2016) tracked bubble identities and volumes using scalar fields in order to use their volumes for sound generation. These authors did not describe how to redistribute volume, especially when multiple kinds of topological changes occur simultaneously among nearby bubbles.

The VOF method provides unconditional mass conservation by storing volumes of fluid present in each simulation grid cell and advecting the volume through the fluid's incompressible velocity field (Sussman, 2003). However, maintaining a sharp boundary at the liquid-air interface is inherently challenging with this approach, leading to complex methods such as coupling a level set with the VOF method (Sussman, 2003). Furthermore, in contrast to the large time steps offered by semi-Lagrangian advection, the VOF method inherently requires small time steps to accurately account for volume exchanges between adjacent grid cells. In general, for computer graphics applications, the FLIP method offers acceptable volume conservation given the tradeoffs in performance and implementation

complexity associated with VOF methods.

2.4 Model Reduction

2.4.1 Solid-Fluid Coupling

The manner in which we couple grid-based fluids to our affine fluid model is related to monolithic (or strong) two-way solid-fluid coupling, especially in the rigid-body case. Specifically, as noted by Jiang et al. (2015), an affine vector field has a similar but more general structure compared to a rigid-body vector field. Thus, requiring matching normal fluxes at the boundary between the regular fluid and the reduced affine regions leads to linear systems that have parallels with rigid-body coupling. Klingner et al. (2006) first considered strong coupling of rigid bodies on tetrahedral meshes, though subsequent work on monolithic coupling has largely focused on regular grid approaches. Batty et al. (2007) considered rigid bodies, Robinson-Mosher et al. (2008, 2009) considered rigid bodies, volumetric elastic objects and thin shells, and Lu et al. (2016) considered reduced deformable solids. Aanjaneya (2018) recently proposed a fast solver for rigid-body coupling making use of Multigrid ideas. Another coupling approach is that of Golas et al. (2012), who used weak coupling between a vortex particle domain and a regular grid-based fluid simulator.

2.4.2 Fluid Model Reduction and Boundary-Based Approaches

Model reduction has been applied to smoke animation problems (Treuille et al., 2006; De Witt et al., 2012; Cui et al., 2018), by simulating in a reduced, divergence-free basis. The basis is typically global, specific to a particular domain, and precomputed for efficiency. To extend reduced fluid models to large domains, Wicke et al. (2009) considered tiling the space and applying coupling between tiles. Our incompressible affine fluid model could be viewed as a particularly convenient, reduced (low order polynomial) basis, constructed locally on the fly at each time step. Our reduced bubble models also share a philosophical connection with recent boundary-based models for fluids, in that the goal is to minimize the use of interior volumetric DOFs. Keeler and Bridson (2014) applied boundary integral techniques to ocean dynamics, and Da proposed both a vortex-based model for soap bubbles (Da et al., 2015), and a boundary element approach to surface-only liquids (Da et al., 2016).

2.5 Viscous Liquids

Viscosity was originally incorporated into the foundational work of Foster and Metaxas (1996) using explicit time integration. Because explicit integration of viscosity is subject to a particularly stringent stability restriction, Stam (1999) modeled viscosity using implicit diffusion (i.e., Laplacian smoothing) of velocity in the context of smoke simulation, and Carlson et al. (2002) applied this model to spatially varying viscosity and liquids with free surfaces. Fält and Roble (2003) used improved boundary conditions to support translation of free-flying liquid bodies, and Rasmussen et al. (2004) proposed an implicit/explicit integration method to properly treat spatially varying viscosity coefficients. Batty and Bridson (2008) developed an implicit variational finite difference viscosity model to allow for rotational motion of viscous free surfaces, thereby enabling realistic buckling and folding along with improved stability for variable viscosity. Larionov et al. (2017) further showed consistent viscous coiling can be achieved by treating viscosity and pressure terms simultaneously via the unsteady Stokes equations. Viscous forces have also been considered for multiphase flows, usually assuming constant viscosity per material (Hong and Kim, 2005; Losasso et al., 2006), and for specialized lower-dimensional mesh-based thread and sheet models (e.g., (Bergou et al., 2010; Batty et al., 2012; Zhu et al., 2015)). Nearly all of the discretizations above yield SPD linear systems, which allow for more efficient numerical solvers; ours does the same.

More generally, hierarchically adaptive node-based finite element-style elasticity methods (e.g., Capell et al. (2002); Grinspun et al. (2002)) could hypothetically be applied to our problem by replacing the elastic constitutive law with the fluid viscosity equations. However, our scheme adopts the staggered grids naturally preferred for incompressible fluids (Bridson, 2015), it does not require higher order basis function constructions, and finite difference schemes can offer significantly sparser stencils even compared to low order finite elements. For example, Zhu et al. (2010) used this last observation to motivate their staggered finite difference scheme for regular grid elasticity. Observe that for a regular 2D staggered grid, only 9 velocity DOFs are involved in a given matrix row (e.g., Batty and Bridson (2008), Figure 10); for linear nodal FVM/FEM, each row involves both velocity component DOFs of the 9 surrounding nodes ($9 \times 2 = 18$). This ratio is significantly worse in 3D: 15 staggered DOFs per row vs. 81 nodal DOFs per row.

2.6 Adaptive Methods

2.6.1 Affine Region Adaptivity

Spatial adaptivity has a long history in liquid simulation. Many different strategies for introducing adaptivity in the pressure projection have been proposed in the graphics literature, including octrees (Losasso et al., 2004; Ando and Batty, 2020), tetrahedra (Klingner et al., 2006; Batty et al., 2010; Ando et al., 2013), tall cells (Irving et al., 2006; Chentanez and Müller, 2011), warped grids (Ibayashi et al., 2018), Chimera grids (English et al., 2013), and Voronoi or Power diagrams (Brochu et al., 2010; de Goes et al., 2015; Aanjaneya et al., 2017), among others. Some advantages of our affine regions include relative simplicity of implementation, flexibility of element shapes, and exactly divergence-free interior velocities. The use of p-adaptivity (i.e., elements with varying degree polynomials) in computer graphics has been explored in a discontinuous Galerkin (DG) framework (Edwards and Bridson, 2014), allowing for highly detailed surfaces to be simulated; while we considered only affine fields, polynomial extensions of our method are also possible. The model of Edwards and Bridson (2014) assumes a standard polynomial basis, but a variety of pointwise divergence-free DG schemes have been developed outside of computer graphics (e.g., (Rhebergen and Wells, 2018)). Traditional finite element methods have considered divergence-free elements as well (Gustafson and Hartman, 1983). There has also been recent interest in finite element variants that support general polyhedral elements (Martin et al., 2008; Manzini et al., 2014; Edwards and Bridson, 2014). Our method has the practical advantage of being easy to incorporate into standard fluid animation tools based on regular grid finite volumes.

2.6.2 Octrees

The first successful *inviscid* octree-based free surface flow solver in computer graphics was proposed by Losasso et al. (2004), although Shi and Yu (2004) had earlier proposed a non-symmetric discretization for octree smoke. Losasso et al. (2004) developed a symmetric positive definite finite volume Laplacian discretization for the pressure projection and a semi-Lagrangian advection step relying on node-based trilinear velocity interpolation, but omitted viscosity. This formulation relied on an inaccurate sloped gradient approximation, leading to motion errors for hydrostatic scenarios. They later presented an enhanced version that constructs a single axis-aligned gradient shared by all child faces at a T-junction (Losasso et al., 2006). To improve its efficiency, Setaluri et al. (2014) proposed a memory-efficient sparse paged grid (SPGrid) data structure that constructs the octree as a hierarchy

of sparsely populated regular grids, rather than a standard pointer-based tree. They applied it to smoke simulation, and later Aanjaneya et al. (2017) applied it to liquid simulation using a finite volume power diagram discretization across T-junctions. We incorporate our viscosity discretization into the SPGrid framework to demonstrate a fully adaptive solver for viscous liquids.

There are relatively few octree-based fluid solvers that specifically address viscosity, particularly in the case of free surfaces and variable coefficients. Hong and Kim (2005) reused the octree Laplacian of Losasso et al. (2004) to add implicit viscosity via velocity diffusion, though this simple model precludes support for rotational effects and variable viscosity (Batty and Bridson, 2008). Ferstl et al. (2014) employed a cut-cell finite element method with a Multigrid solver for adaptive liquid animation, focusing on inviscid scenarios and pressure projection, but did not demonstrate nor elaborate on their treatment of viscosity. Nielsen and Bridson (2016) alluded to tree-based finite element viscosity in the BiFröst simulation toolkit for the 3D computer graphics application Maya (Autodesk, 2021), but omitted details.

Looking beyond computer graphics to computational physics, there exist many adaptive methods on nested regular grids that support viscosity, but most assume spatially constant viscosity without free surfaces (e.g., (Almgren et al., 1998; Min and Gibou, 2006; Guittet et al., 2015)). We discuss a few pertinent exceptions. In geophysics, Gerya et al. (2013) proposed an adaptive implicit finite difference discretization for free surface variable viscosity flows, though it is limited to 2D quadtrees. Their method also treats the free surface using an approximate "sticky air" layer rather than a sharp boundary condition, and yields non-symmetric systems. Nikitin et al. (2008; 2011) presented octree Navier-Stokes solvers that consider the free surface conditions and treat viscosity explicitly, though explicit viscosity methods can lead to stability issues. Olshanskii et al. (2013) proposed a related implicit discretization, but it does not handle the free surface and its reliance on least-squares fitting to construct adaptive stencils introduces asymmetry. Guittet et al. (2015) achieved an SPD system for the Laplacian form on non-graded trees using a Voronoi diagram of the staggered velocity face samples; however, this comes at the cost of frequent unstructured mesh generation and highly non-local stencils. Moreover, neither Voronoi nor power diagram discretizations of the Laplace operator can readily treat the cross-component derivative terms that arise for the more general viscosity PDE needed for free surfaces and variable coefficients. Lastly, setting aside viscosity, Horesh and Haber (2011) proposed a non-symmetric finite volume discretization of Maxwell's equations on octrees; their (adjoint of) curl stencils share some geometric similarities with our vector gradient stencils.

2.6.3 Tetrahedral Meshes

Tetrahedral meshes are another alternative for adaptive liquid simulation with either Eulerian or Lagrangian approaches. In the Eulerian setting, Klingner et al. (2006) used adaptive staggered tetrahedral meshes for smoke; Chentanez et al. (2007) extended this approach to liquids. To the best of our knowledge, the subsequent node-based finite volume viscosity scheme of Batty and Houston (2011) is the only prior Eulerian scheme to animate high viscosity liquids on adaptive tetrahedral meshes that supports rotational surface motion and spatially varying viscosity. However, earlier work supported Laplacian-style viscosity on closed domains or used (nearly) uniform-resolution tetrahedral meshes (Wendt et al., 2007; Elcott et al., 2007; Bonito et al., 2006).

The possibility of purely *Lagrangian* tetrahedral liquid animation was first hinted at by the finite element viscoplastic solid approaches of Bargteil et al. (2007) and Wojtan and Turk (2008). Subsequently, incompressible Lagrangian liquid simulation with surface-conforming tetrahedral meshes was achieved by Misztal et al. (2010; 2014) and Clausen et al. (2013), including natural handling of viscous free surfaces. However, while tetrahedral methods flexibly support adaptive viscous flows, the computational expense, complexity, and memory overhead of accessing, manipulating, and remeshing such structures makes them less attractive compared to modern optimized octrees (Setaluri et al., 2014; Aanjaneya et al., 2017).

2.7 Particle-based Models

For our overall framework, we prefer the hybrid grid-particle FLIP (see §2.1) method because of the efficiencies offered by the grid structure. However, both viscous and multiphase liquids have been modelled with the fluid state fully defined on the particles. We detail the relevant previous work below.

2.7.1 Smoothed Particle Hydrodynamics

The smoothed particle hydrodynamics method (SPH) was first introduced to computer graphics by Desbrun and Gascuel (1996) and popularized for liquid simulation and real-time applications by Müller et al. (2003). This purely particle-based formulation uses smooth kernels with compact support to represent continuum-based properties like fluid density and velocity. To enforce incompressibility, pressure forces opposing compression and expansion

are derived from implied fluid densities based on the SPH particle distribution and are integrated explicitly in time. This equation of state method leads to liquids appearing *squishy* and requires small time steps to prevent instability. The predictor corrector SPH method (Solenthaler and Pajarola, 2009) and, subsequently, the implicit imcompressible SPH method (Ihmsen et al., 2014) solve pressure forces that remove compression and ease the time step restrictions at the cost of more computation per time step. Adaptively sized particles allow the simulator to focus computational effort at geometrically important regions of the simulation (Adams et al., 2007; Solenthaler and Gross, 2011), with the added complexity of coalescing many small particles to one big particle or splitting one big particle into many small particles. Because pressure forces are derived from particle density, these splitting and merging operations must be handled carefully to avoid inducing spurious forces in the liquid. To simulate the effects of viscosity in SPH-based fluid, Takahashi et al. (2015) solved for viscous forces implicitly in time using the full viscous tensor, applied to the smooth kernel representation of the fluid's velocity field. Although variations to this approach have been proposed (Peer et al., 2015; Weiler et al., 2018; Peer and Teschner, 2017), the approach of Takahashi et al. (2015) is most similar to the variational method of Batty and Bridson (2008) that we extend to adaptive grids. Simulating immiscible air bubbles entrained in liquid was made possible by using particle densities for their corresponding materials (Solenthaler and Pajarola, 2008) and modelling the interaction of the two materials using drag forces (Ihmsen et al., 2011).

2.7.2 Material Point Method

The material point method (MPM) was originally proposed by Sulsky et al. (1995) as an extension of the PIC method for simulating granular material and was first introduced to computer graphics by Stomakhin et al. (2013) for simulating snow. An MPM material is represented by a collection of particles that store material information such as mass, momentum and the deformation gradient tensor. At each simulation time step, the material parameters are transferred from the particles to the grid using finite element basis functions. Forces acting on the material (e.g., elastic stress, contacts, etc.) are resolved at the grid level and the updated material properties are transferred back to the particles. Finally, the particles are integrated in time using their updated velocities. The benefit of this approach for granular material is that topology changes are handled implicitly instead of requiring tricky mesh operations common with tetrahedral-based finite element approaches (Bargteil et al., 2007). MPM has since been applied to non-Newtonian fluids (Stomakhin et al., 2014; Yue et al., 2015), viscoelastic fluids (Ram et al., 2015), granular materials such as sand (Klár et al., 2016; Daviet and Bertails-Descoubes, 2016), and many more interesting phenomena

(Jiang et al., 2016). Gao et al. (2017) recently developed an adaptive MPM scheme for octrees using carefully designed basis functions with C^1 continuity. Importantly, even regular grid MPM schemes require higher order basis functions to avoid discontinuities in forces and, as a result, suffer from relatively dense matrices that are inefficient to solve (Yue et al., 2018). This limitation motivates MPM practitioners to use explicit time integration methods, in contrast to our preference for stable implicit methods.

Chapter 3

Background

In this chapter, we give a brief overview of a standard fluid animation pipeline that our contributions build on, focusing primarily on solving for pressure and viscous forces. Solving for pressure is essential for imposing the incompressibility constraint, capturing momentum transfer between immiscible liquids (i.e., two-phase simulations), and enforcing solid boundaries. Solving for viscous forces is essential for capturing damped relative motion of viscous liquids, but more importantly, for capturing rotational buckling and folding motion expected of highly viscous liquids such as honey. We assume the reader is familiar with concepts from vector calculus, numerical linear algebra, and numerical differential equations.

3.1 Navier-Stokes

Movement of fluids is described by the Navier-Stokes momentum equation (Chorin and Marsden, 1993),

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \boldsymbol{f}, \tag{3.1}$$

where $\boldsymbol{u}$ is the fluid's velocity, ρ is the density, p is the pressure, $\boldsymbol{\tau}$ is the deviatoric viscous stress tensor, $\boldsymbol{\tau} = \mu\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathsf{T}}\right)$ where μ is the dynamic viscosity coefficient, and $\boldsymbol{f}$ are external forces (e.g., gravity). We make the simplification that our fluid is incompressible by imposing a divergence-free constraint on the fluid's velocity:

$$\nabla\cdot\boldsymbol{u} = 0. \tag{3.2}$$

Given the non-linear term, $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$, a numerical approach for solving the entirety of (3.1) as a single operation is computationally expensive (Mullen et al., 2009). Instead, we adopt the standard operator splitting technique to update the velocity field through a daisy chain of operations (Chorin and Marsden, 1993; Stam, 1999):

- Advect fluid: $\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = 0$,
- Integrate forces: $\rho \frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{f}$,
- Apply viscosity: $\rho \frac{\partial \boldsymbol{u}}{\partial t} = \nabla \cdot \boldsymbol{\tau}$,
- Apply pressure: $\rho \frac{\partial \boldsymbol{u}}{\partial t} = -\nabla p$, s.t. $\nabla \cdot \boldsymbol{u} = 0$.

This splitting technique allows us to solve for pressure and viscous forces as solutions to linear PDEs. Although this simplification does decrease the accuracy of the method to first order with respect to time, the simplification of the numerical methods and the accompanied performance improvement is a worthwhile (and widely adopted) tradeoff (Chorin and Marsden, 1993; Bridson, 2015). Previous work has achieved higher order accuracy within the operator splitting framework, albeit with additional overhead cost (Narain et al., 2019). The advection step advances the simulation forward in time, moving fluid attributes (e.g., density, level set surface, etc.) and its velocity field to the next time step (Bridson, 2015). We adopt the particle-based method, FLIP, which stores fluid velocities on each particle and moves these particles forward through the velocity field using numerical integration (Zhu and Bridson, 2005). External forces are integrated into the fluid's velocity field with forward Euler, $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{f}$.

Solving for viscosity accounts for stress in the fluid caused by internal friction and the effect of solid boundaries acting on the fluid. Advecting the fluid, adding forces and incorporating viscosity typically leaves the velocity field in a divergent state; solving for pressure returns the velocity field to the divergence-free state required by the constraint in (3.2). Since our contributions build on the existing pressure and viscosity methods, we detail their standard discretizations below.

3.2 Solving For Pressure

We transform a fluid velocity field, $\boldsymbol{u}^n$, to a divergence-free state, $\boldsymbol{u}^{n+1}$, through the internal pressure of the fluid, where the gradient of pressure is a force acting on the velocity field,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \frac{\Delta t}{\rho} \nabla p, \tag{3.3}$$

and Δt is the size of our discrete simulation time step.[1] However, since both pressure and the final velocity are unknown, we cannot simply solve (3.3) directly. Using the constraint that the final velocity must be divergence-free, we apply a divergence operator to both sides of (3.3) to remove dependence on the unknown final velocity field:

$$-\Delta t \nabla \cdot \frac{1}{\rho} \nabla p = -\nabla \cdot \boldsymbol{u}^n. \tag{3.4}$$

Solving this Poisson equation provides us with the fluid pressure needed to ensure $\boldsymbol{u}^{n+1}$ is divergence-free when recovered from (3.3). This operation is essentially a projection of the fluid's velocity field onto the divergence-free subspace and commonly referred to as the *pressure projection* operation.

3.2.1 Discretization

We solve the pressure Poisson equation (3.4) discretely using finite differences to approximate the gradient and divergence operators. We use the standard staggered grid approach to place pressure scalars at grid cell centers and discrete velocity vector components at grid cell edges in 2D or grid cell faces in 3D. This placement avoids instabilities that can occur when pressure and velocity are co-located (Bridson, 2015). Figure 3.1a illustrates a 2D staggered grid with the associated indices for each component (where u and v are the x- and y-axis vector components of the velocity field, oriented along the face normals). This grid format extends naturally to 3D with the three velocity vector components (u,v,w) placed at centers of cell faces. Given our sampling scheme, we construct a linear system of discrete approximations to (3.4) using centered differences. For example, axis-aligned components of pressure gradients are co-located with velocity samples at cell edges,

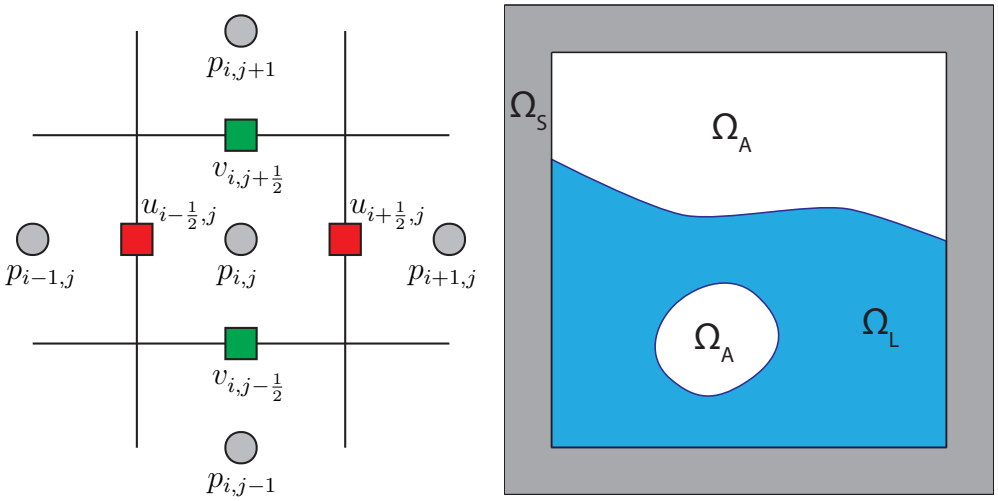
$$(\nabla \mathrm{p})_{i-\frac{1}{2},j} = \frac{p_{i,j} - p_{i-1,j}}{\Delta x}, \tag{3.5}$$

and the divergence of velocities are co-located with pressure samples at cell centers,

$$(\nabla \cdot \mathbf{u})_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y}.$$

The discrete divergence of pressure gradients in the Poisson equation follows directly from divergence of velocities. These discretizations are applied at every grid cell in the liquid

[1] We use $\boldsymbol{u}^{n+1}$ to represent the final velocity field after applying an individual operator. Likewise, we will use $\boldsymbol{u}^n$ to represent the velocity field before applying the operator.



(a) (b)

Figure 3.1: The fluid domain is discretized on a staggered grid (a), with pressure samples at grid centers and velocity components at grid faces. For liquids with free surfaces (b) the staggered grid fills the liquid volume, Ω_L. The liquid boundary, $\partial\Omega_L$, is composed of Dirichlet boundary conditions at the air-liquid boundary, $\partial\Omega_L \cap \partial\Omega_A$, and Neumann boundary conditions at the solid-liquid boundary, $\partial\Omega_L \cap \partial\Omega_S$.

volume and together form a linear system of equations that is SPD, relatively sparse, and simple to implement. In Figure 3.1b, the bulk of the liquid volume Ω_L can be discretized using the above methods. However, we must modify the discretizations at the boundaries of the liquid volume, $\partial\Omega_L$, to account for interactions with solid objects ($\partial\Omega_L \cap \partial\Omega_S$) and other fluids ($\partial\Omega_L \cap \partial\Omega_A$).

3.2.2 Boundary Conditions

Accounting for boundaries inherently changes the discretization of (3.3) and (3.4) because there are no valid liquid pressure or velocity samples outside of Ω_L. We aim to modify our linear system of discrete pressure equations to account for boundaries while still remaining SPD, sparse, and simple to implement.

Free Surface

In the *free surface* model for the air-liquid boundary, we assume that air is substantially less dense than liquid and provides vanishingly-small resistance to the liquid's motion. Given this assumption, we simplify our simulation by assigning air volumes, Ω_A, a uniform pressure of zero, i.e., $p_A = 0$, which effectively removes air cells from the linear system for pressures.

It is reasonable to wonder about the consequences of such a simplification. Would this assumption cause air bubbles to simply collapse? Indeed it will, leaving artists with a difficult trade-off. Simulating the entire atmosphere in a large ocean simulation would be prohibitively expensive and may not add realism to the scene. On the other hand, as demonstrated in Figure 4.14, water pouring through the neck of a container requires simulated air resistance on the liquid to properly capture the expected *glugging* motion. For sake of exposition, we begin by focusing only on the *air-as-a-vacuum*, free surface, approach to liquid simulations and introduce the proper interacting *two-phase* flow method in a later section.

At first glance, applying the zero-pressure Dirichlet boundary condition at the free-surface seems trivial. Since we are assuming that air pressure is uniformly zero, any discrete pressure samples that fall inside the air volume are no longer unknowns and can be removed from the linear system. However, simply setting the air pressure to be zero at these discrete samples fails to realize zero pressure precisely at the air-liquid boundary. Consider the example presented in Figure 3.2 where the boundary does not pass directly through a discrete pressure sample. In this case, pressure will vary linearly from the liquid sample,

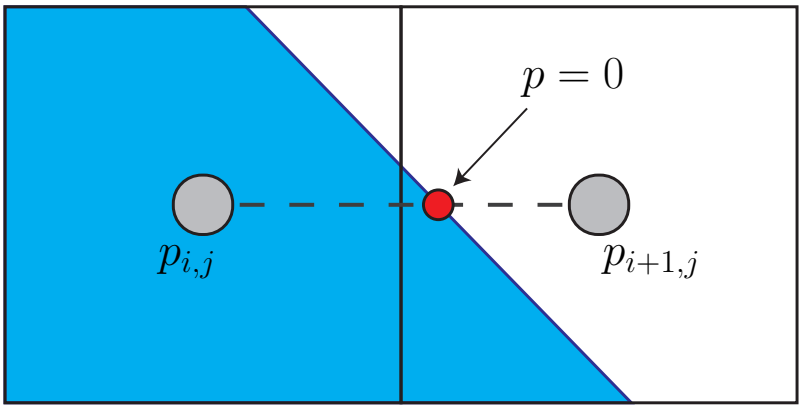


Figure 3.2: A liquid-air boundary passes between liquid pressure sample $p_{i,j}$ and air sample $p_{i+1,j}$. The ghost fluid method extrapolates the liquid pressure to precisely place the zero air pressure at the interface (*red* circle).

$p_{i,j}$, to finally reach zero at the exterior air sample, $p_{i+1,j} = 0$, implying a non-zero pressure value at the actual air-liquid boundary (*red* circle). Failing to enforce zero-pressure directly on the surface can generate stair-step artifacts on the surface, impacting the visual quality of the simulation.

The ghost fluid method (Enright et al., 2003) treats the pressure sample in the air volume as an extrapolation from the true, zero pressure at the interface. To enforce a zero pressure at the liquid surface, consider a linear interpolation of pressure samples at the free surface, $(1-\theta)p_{i,j} + \theta p_{i+1,j} = 0$, where θ is the normalized length fraction of liquid between both pressures samples. The standard approximation for this fraction is the normalized distance to the zero isosurface, given by $\theta = \frac{\phi_{i,j}}{\phi_{i,j}-\phi_{i+1,j}}$ where $\phi_{i,j}$ and $\phi_{i+1,j}$ are the associated signed distances to the liquid surface, located at the corresponding pressure samples. We simply solve for the *ghost* pressure in terms of the active liquid pressure, $p_{i+1,j} = \frac{\theta-1}{\theta}p_{i,j}$. The pressure gradient used in (3.3) and (3.4) at the cell-face between the air and liquid pressure samples reduces to $(\nabla \mathrm{p})_{i+\frac{1}{2},j} = \frac{p_{i+1,j}-p_{i,j}}{\Delta x} = \frac{p_{i,j}}{\theta \Delta x}$.

Solid Boundaries

In contrast to the free surface model, where air imposes no resistance against the motion of the surrounding liquid, scripted-motion solids provide an infinitely strong resistance to the liquid's motion, forcing the liquid to move in accordance with them. Although methods for coupling liquid and dynamic rigid-body or deformable solids provide a more realistic

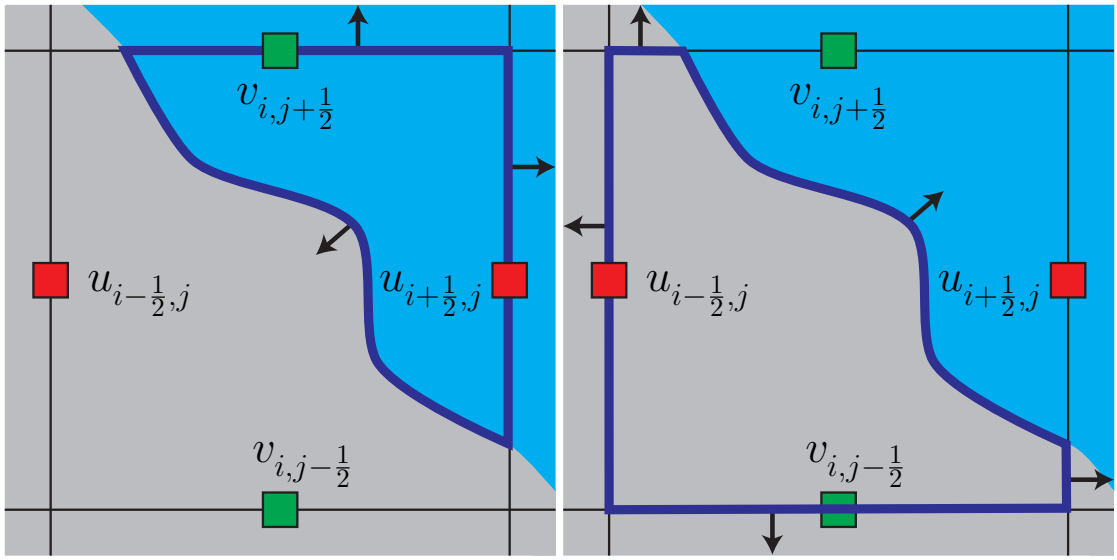


Figure 3.3: Surface integrals (*dark blue*) for the fluid subregion (a) and solid subregion (b) within a grid cell.

interaction between the liquid and solids (Carlson et al., 2004; Batty et al., 2007; Gibou and Min, 2012; Zarifi and Batty, 2017), they are beyond the scope of our work. We assume that the liquid cannot penetrate into the moving solid nor can it separate and create a vacuum between the two regions, but liquids can freely flow tangentially along the solid boundary (i.e., free-slip condition). We model this interaction by constraining the liquid velocity to match the solid's velocity in the normal direction of the solid boundary, $\boldsymbol{u}^{n+1} \cdot \boldsymbol{n}_S = \boldsymbol{u}_S \cdot \boldsymbol{n}_S$.

Enforcing this solid boundary condition in the pressure Poisson equations (3.4) is not as straightforward as free surfaces. Consider the free-slip condition in the continuous setting for the pressure update (3.3), $(-\frac{\Delta t}{\rho}\nabla p) \cdot \boldsymbol{n}_S = (\boldsymbol{u}_S - \boldsymbol{u}^n) \cdot \boldsymbol{n}_S$. Because pressure gradients in the discrete setting are staggered and aligned to grid face normals, directly applying this free-slip condition to the Poisson equations will fail to accurately capture solid boundary normals in all but trivially grid-aligned solid boundaries. This mismatch of normals leads to stair-step artifacts in the resulting velocity field and creates unrealistic motion when fluid slides along a curved boundary. More advanced methods that account for solid geometry passing through a grid cell successfully remove these artifacts (Batty et al., 2007; Ng et al., 2009) with only a small modification to the discrete Poisson equation for pressure.

The finite volume approach of Ng et al. (2009) accurately enforces solid geometry that

cuts through a grid cell using the aptly named *cut-cell* method. We briefly detail the main concepts of the cut-cell method. First, consider the integral form of (3.4) over the intersection of a liquid volume, Ω_L, and a discrete grid cell, C_i:

$$-\Delta t \int_{C_i \cap \Omega_L} \nabla \cdot \frac{1}{\rho} \nabla p \, dV = -\int_{C_i \cap \Omega_L} \nabla \cdot \boldsymbol{u}^n \, dV. \tag{3.6}$$

Since both velocities and pressure gradients are defined at cell boundaries, we apply the divergence theorem to transform (3.6) into a boundary integral. The grid cell illustrated in Figure 3.3 is only partially occupied by the liquid volume, which breaks the boundary integral into components along the cell boundary ($\partial C \cap \Omega_L$) and the interior solid-liquid boundary ($C \cap \partial\Omega_L$):

$$-\Delta t \int_{\partial C_i \cap \Omega_L} \frac{\nabla p}{\rho} \cdot \boldsymbol{n}_C \, dA - \Delta t \int_{C_i \cap \partial\Omega_L} \frac{\nabla p}{\rho} \cdot \boldsymbol{n} \, dA = -\int_{\partial C_i \cap \Omega_L} \boldsymbol{u}^n \cdot \boldsymbol{n}_C \, dA - \int_{C_i \cap \partial\Omega_L} \boldsymbol{u}^n \cdot \boldsymbol{n} \, dA. \tag{3.7}$$

The integrals along the cell boundary can straightforwardly be discretized using our staggered grid and midpoint quadratures. However, we want to avoid integration along solid boundaries interior to a cell. To remove the cell-interior integral, we first substitute the pressure gradient along the solid boundary with the free-slip boundary condition:

$$-\Delta t \int_{C_i \cap \partial\Omega_L} \frac{\nabla p}{\rho} \cdot \boldsymbol{n}_S \, dA = \int_{C_i \cap \partial\Omega_L} (\boldsymbol{u}_S - \boldsymbol{u}^n) \cdot \boldsymbol{n}_S \, dA. \tag{3.8}$$

We then substitute the left-hand side integral along the liquid-solid boundary in (3.7) with (3.8) and replace the solid boundary normal with the equivalent liquid boundary normal, $\boldsymbol{n}_S = -\boldsymbol{n}$:

$$-\Delta t \int_{\partial C_i \cap \Omega_L} \frac{\nabla p}{\rho} \cdot \boldsymbol{n}_C \, dA = -\int_{\partial C_i \cap \Omega_L} \boldsymbol{u}^n \cdot \boldsymbol{n}_C \, dA - \int_{C_i \cap \partial\Omega_L} \boldsymbol{u}_S \cdot \boldsymbol{n} \, dA. \tag{3.9}$$

Finally, we need to remove the right-hand side integral along the liquid-solid boundary. We assume that the solid object's velocity is divergence-free (which holds for any rigid body motions) and apply the divergence theorem once more for the integral along the solid domain inside the cell (see Figure 3.3b),

$$\int_{C_i \cap \partial\Omega_S} \boldsymbol{u}_S \cdot \boldsymbol{n}_S \, dA = -\int_{\partial C_i \cap \Omega_S} \boldsymbol{u}_S \cdot \boldsymbol{n}_C \, dA, \tag{3.10}$$

and then substitute the solid-liquid boundary integral for the cell-boundary integral to construct the cut-cell finite volume method in terms of velocities along the cell boundary:

$$-\Delta t \int_{\partial C_i \cap \Omega_L} \frac{\nabla p}{\rho} \cdot \boldsymbol{n}_C \, dA = -\int_{\partial C_i \cap \Omega_L} \boldsymbol{u}^n \cdot \boldsymbol{n}_C \, dA - \int_{\partial C_i \cap \Omega_S} \boldsymbol{u}_S \cdot \boldsymbol{n}_C \, dA. \tag{3.11}$$

The discrete form of (3.11) applied to the staggered grid is simply the discrete Poisson equation (3.4) using discrete pressure gradients (3.5), scaled by the area fraction (or length fraction in 2D) of the grid face (edge) inside the fluid or solid volume,

$$-\Delta t \sum_{\partial C_i} \frac{1}{\rho} \nabla \mathrm{p} \cdot \mathbf{n}_C \, \mathsf{dA} = -\sum_{\partial C_i} \mathbf{u}^n \cdot \mathbf{n}_C \, \mathsf{dA} - \sum_{\partial C_i} \mathbf{u}_S \cdot \mathbf{n}_C \, \mathsf{dA}_S, \tag{3.12}$$

where dA and dA_S are the (finite) areas of the cell face inside the liquid and solid volumes, respectively. The power of this approach arises from the liquid-solid boundary integral cancellation when combining the two sub-regions: a solid boundary with complicated geometry is implicitly handled with the cut-cell method.

Because the ghost fluid method modifies the gradient operator and the cut-cell method modifies the divergence operator, cells with both air-liquid and liquid-solid boundaries can straightforwardly enforce both boundary types. Additionally, because both methods modify the weighted flux at boundaries between pressure samples, the linear system remains SPD.

Two-Phase Flow

For performance sake, industrial fluid simulators typically avoid simulating the ambient air volume in addition to the liquid of interest. Modelling the air volume as a vacuum offers significant performance benefits but also limits the range of phenomena that can be captured. Below we detail how one might incorporate the air volume into a fluid simulator to achieve the *glugging* effect of liquid pouring out of a bottle, air bubbles entrained in liquids and many other desired effects resulting from two fluids interacting. We will focus primarily on two-phase flow scenarios but the concepts directly generalize to multi-phase flows with three or more fluids. We will also only consider incompressible fluids. Although air is easier to compress than water, this effect is largely imperceptible for most applications in computer graphics.

In order to capture the interaction between fluids and enforce incompressibility for each fluid domain, pressure inside the air region can no longer be assumed zero. Within the air volume, the resulting discrete Poisson equation for pressure is effectively the same as in the liquid region, with a smaller density coefficient. Therefore, we only need to modify the pressure system at the boundaries between the two materials. We assume the fluids are immiscible, implying there is a sharp interface between the two fluids and a corresponding discontinuity in fluid densities. Kang et al. (2000) modelled this discontinuity using the ghost fluid method to construct a variable coefficient Laplacian as the discrete Poisson equation across the two-phase fluid boundary. Similar to the free surface approach, the

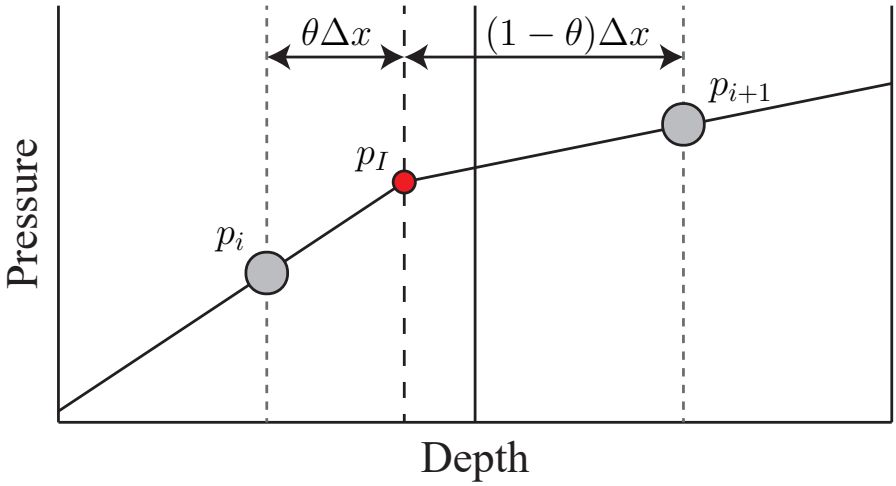


Figure 3.4: A two-phase fluid in 1D has a C^0 continuous pressure at the interface, p_I. The length fraction θ to the interface from the pressure sample p_i is used to construct the pressure gradients on both sides of the interface: $\frac{p_I - p_i}{\theta\Delta x}$ and $\frac{p_{i+1} - p_I}{(1-\theta)\Delta x}$. (Adapted from Figure 12 in Boyd and Bridson (2012).)

resulting linear system remains SPD. Immiscibility of the fluids implies that the fluids cannot separate from, or penetrate into, one another. A consequence of this restriction is that the fluids must move in lockstep, meaning that the component of the fluid velocities normal to the interface between them must be equal. Therefore, forces induced by pressure at the interface must agree for both liquids, and pressure must be continuous across the boundary.[2] Figure 3.4 illustrates a pressure profile of a 1D two-phase simulation with a material interface pressure, p_I, between the discrete liquid pressure p_i and the discrete air pressure p_{i+1}. In the discrete setting, the velocity at the face between materials defines the motion of the interface. Therefore, the forces induced by pressure gradients must also be continuous across the interface:

$$\frac{1}{\rho_L}\frac{p_I - p_i}{\theta\Delta x} = \frac{1}{\rho_A}\frac{p_{i+1} - p_I}{(1-\theta)\Delta x}, \tag{3.13}$$

where ρ_L and ρ_A are the liquid and air densities, respectively, and θ is the length fraction of liquid between the two pressure samples. By solving for p_I and substituting it into either

[2]Pressure is discontinuous when surface tension is included in the model. We will ignore surface tension effects in our models but they can easily be incorporated with simple level set operations and interpolation at the interface if desired (Kang et al., 2000; Hong and Kim, 2005; Boyd and Bridson, 2012; Popinet, 2018).

side of (3.13), we solve for the pressure gradient at the grid face:

$$\left(\frac{1}{\rho}\nabla \mathrm{p}\right)_{i+\frac{1}{2}} = \frac{1}{\theta\rho_L + (1-\theta)\rho_A}\frac{p_{i+1} - p_i}{\Delta x}. \tag{3.14}$$

Because the normal of the interface for 2D and 3D simulations is often not aligned to a grid face normal, this approach is only first-order accurate (Kang et al., 2000). However, standard finite difference/volume second order accurate methods for solving elliptic PDEs with sharp coefficient discontinuities inherently result in a non-symmetric linear system (Crockett et al., 2011; Coco and Russo, 2018). For this reason, the first order, SPD, ghost fluid method is typically preferred for multi-phase simulations in computer graphics (Boyd and Bridson, 2012).

Incorporating the air volume into the fluid simulation can result in a significant increase in computation time compared to an equivalent free surface method. This performance impact is caused by the additional DOFs in the linear system associated with the air region and the large density ratios between the two fluids that worsen the conditioning of the linear system. As a result, iterative solvers like the diagonally preconditioned Conjugate Gradients method are slow to converge.

3.2.3 Multigrid Preconditioners

Achieving modern cinematic-quality resolutions for fluid simulations often require hundreds of millions of DOFs in the linear system for pressure, making direct solvers infeasible due to both memory consumption and time cost. Alternatively, iterative solvers like the Conjugate Gradients method scale much more efficiently. Although numerical solvers like the Multigrid method generally offer fast convergence for elliptic PDEs, the irregular boundaries that arise in free surface liquids simulators are known to impact efficiency and require special handling. For this reason, Multigrid methods in free surface simulations are typically employed as a preconditioner inside of Conjugate Gradients (McAdams et al., 2010; Setaluri et al., 2014). Below we briefly review the main concepts of Multigrid schemes and how they are applied to free surface liquids.

We aim to solve a discrete elliptic PDE, $\mathbf{A}_h\mathbf{x}_h = \mathbf{b}_h$, defined over a uniform grid of size h, where $\mathbf{A}_h$ is the discrete differential operator and $\mathbf{x}_h$ is the unknown solution vector. Beginning with an approximate solution, $\mathbf{x}_h^n$, an iterative solver aims to reduce the residual in the linear system:

$$\mathbf{r}_h^n = \mathbf{b}_h - \mathbf{A}_h\mathbf{x}_h^n. \tag{3.15}$$

Alternatively, we can view the current error in the system as the difference between the true solution and our current approximation:

$$\mathbf{e}_h^n = \mathbf{x}_h - \mathbf{x}_h^n. \tag{3.16}$$

If we solve for the error in the system, we can simply offset the approximate solution to recover the true solution. However, since both the error and the solution are unknown, we are unable to solve for the error directly. Instead, by applying the differential operator to both sides,

$$\mathbf{A}_h\mathbf{e}_h^n = \mathbf{A}_h\mathbf{x}_h - \mathbf{A}_h\mathbf{x}_h^n = \mathbf{b}_h - \mathbf{A}_h\mathbf{x}_h^n = \mathbf{r}_h^n, \tag{3.17}$$

we can now solve for the error in terms of the current residual. Solving the linear system for the error is unfortunately just as expensive as solving the original linear system. However, because the error is smooth for elliptic PDEs, solving for the error at a coarse resolution will yield a close approximation to the fine resolution error vector and will, ideally, be much less expensive to solve.

Before sampling the residual $\mathbf{r}_h^n$ to a coarse representation, we want to smooth out any high frequency modes by applying Jacobi smoothing to $\mathbf{A}_h\mathbf{x}_h^n = \mathbf{b}_h$. Next, we compute the residual and downsample (restrict) it onto a coarse grid of size $2h$. Using a coarse discretization of the differential operator, the coarse error is determined by solving $\mathbf{A}_{2h}\mathbf{e}_{2h}^n = \mathbf{r}_{2h}^n$. The coarse error is then upsampled (prolonged) to the fine grid and applied as a correction to our approximate solution. Finally, we apply Jacobi smoothing to the updated solution to remove low frequency artifacts from the coarse error. The steps of this Multigrid *V-cycle* are highlighted in Algorithm 1.

Algorithm 1 Two-level Multigrid V-cycle

1: Apply smoothing to $\mathbf{A}_h\mathbf{x}_h^n = \mathbf{b}_h$
2: Compute residual $\mathbf{r}_h = \mathbf{b}_h - \mathbf{A}_h\mathbf{x}_h^n$
3: Restrict residual $\mathbf{r}_h$ to coarse grid, $\mathbf{r}_{2h}$
4: Solve coarse system $\mathbf{A}_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$
5: Prolong error $\mathbf{e}_{2h}$ to fine grid, $\mathbf{e}_h$
6: Add error correction to approximate solution $\mathbf{x}_h^{n+1} = \mathbf{x}_h^n + \mathbf{e}_h$
7: Apply smoother to solution $\mathbf{A}_h\mathbf{x}_h^{n+1} = \mathbf{b}_h$

In practice, solving for the error at a grid size $2h$ is also slow. Typically, many levels of coarsening will be applied, where steps 1-3 are iterated until the desired coarseness is achieved. After solving the coarse linear system, steps 5-7 are repeated, propagating the error correction vector back to the fine grid.

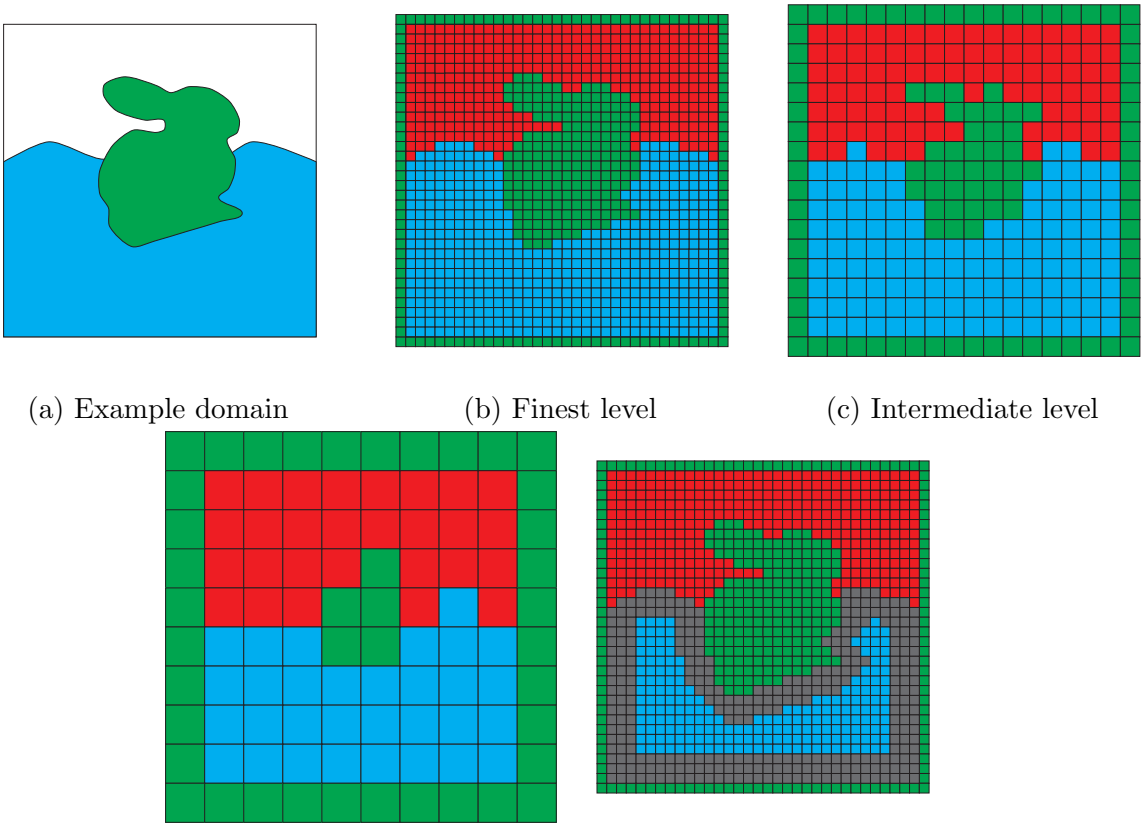
Irregular Boundaries For free surface liquids and simulations with irregular solid boundaries, extra care must be taken to ensure that coarsening and smoothing near boundaries do not introduce spurious errors. We follow the geometric Multigrid design of McAdams et al. (2010) that proposes the following modifications to the canonical in-a-box Multigrid method:

1. Label coarse cells based on the material labels of their fine-cell children,
2. Modify restriction and prolongation operators along boundaries,
3. Set ghost fluid and cut-cell weights for coarse grid differential operators,
4. Add additional smoothing along boundaries to reduce sampling errors, and
5. Apply Multigrid as a preconditioner to Conjugate Gradients.

The hierarchy of grids employed in a V-cycle are aligned such that each coarse parent cell perfectly contains eight fine children cells. A coarse cell is then labelled using the following criteria:

1. If any fine-cell children are *air* cells, the coarse-cell parent is labelled as an *air* cell,
2. Else, if any children are *liquid* cells, the parent is labelled as a *liquid* cell,
3. Else, the parent is labelled as a *solid* cell.

Figure 3.5 illustrates an example domain (Figure 3.5a) discretized at the finest grid resolution in the Multigrid hierarchy (Figure 3.5b) and subsequently coarsened (Figures 3.5c-3.5d) using McAdams et al. (2010). In order to simplify implementation and avoid out-of-bounds checks, every level in the grid hierarchy is padded with an outer layer of *solid* cells. Using this coarsening scheme, if a restriction or prolongation operator includes non-liquid cells, the contribution of that cell is set to zero. We do not apply the ghost fluid or cut-cell method at coarse resolutions; instead the coarse discrete Poisson systems are simply voxelized as if the boundary occurred at the face of the liquid cell. As the three levels in Figure 3.5 illustrate, small-scale features of the liquid are eroded away by the air region and small-scale features of the solid are eroded away by the liquid region. This discrepancy in geometry leads to a poor coarse-scale approximation of the correction error at irregular boundaries. In order to correct for this discrepancy, we apply additional Jacobi smoothing iterations along the liquid boundary before and after the usual smoothing pass



(a) Example domain (b) Finest level (c) Intermediate level

(d) Coarsest level (e) Boundary smoothing cells

Figure 3.5: A solid *green* bunny is partially submerged in the surrounding (*blue*) liquid (a). The domain is discretized at the finest grid resolution in the Multigrid hierarchy (b) with air cells labelled *red*. The Multigrid hierarchy is progressively coarsened using the strategy of McAdams et al. (2010) (c-d). (e) At each level, additional smoothing operations are applied within a thin layer of cells along the liquid boundary (*grey*).

over the entire liquid domain (steps 1 and 7 in the V-cycle) for every level in the hierarchy. We use a three-voxel wide layer of cells at the liquid boundary (see Figure 3.5e) and for each boundary smoothing operation, we apply three iterations of Jacobi smoothing. We note that other smoothing methods can alternatively be employed with the caveat that they must be symmetric operations because a preconditioner to Conjugate Gradients must be SPD.

3.3 Solving for Viscosity

The operator splitting method described in §3.1 isolates the force acting on the fluid velocity due to viscous stress,

$$\rho \frac{\partial \boldsymbol{u}}{\partial t} = \nabla \cdot \mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathsf{T}}), \tag{3.18}$$

where μ, the viscosity coefficient, may vary smoothly in space.

Initial methods for incorporating viscosity into the fluid simulation pipeline assumed a constant viscosity coefficient and, by incorporating the divergence-free constraint for fluid velocity, reduced the spatial derivatives to a standard, component-wise, Laplacian (Foster and Metaxas, 1996). Because the Laplacian operator no longer includes the cross derivative terms, $(\nabla \mathbf{u})^{\mathsf{T}}$, each set of velocity components can be solved independently rather than in a single, coupled system. This simplification is effective for viscous fluids that are entirely contained within a solid boundary (e.g., a smoke simulation in a box) but, as Batty and Bridson (2008) demonstrate, this decoupled system cannot represent the boundary stress at free surfaces. Improperly handling free surface boundary conditions can lead to artifacts in fluid motion (Carlson et al., 2002) or fail to account for stress-induced rotational motion (Fält and Roble, 2003). As illustrated in Figure 3.6, the Laplacian form fails to capture the rotational motion of the canonical viscous beam example.

Variational Method

In our propopsed octree viscosity framework, we adopt the variational method of Batty and Bridson (2008) to solve for viscous forces and capture proper rotational motion and buckling modes,

$$\int_{\Omega} \left(\frac{\rho}{2\Delta t} \|\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n}\|_2^2 + \mu \left\| \frac{\nabla \boldsymbol{u}^{n+1} + (\nabla \boldsymbol{u})^{\mathsf{T}}}{2} \right\|_F^2 \right) dV, \tag{3.19}$$

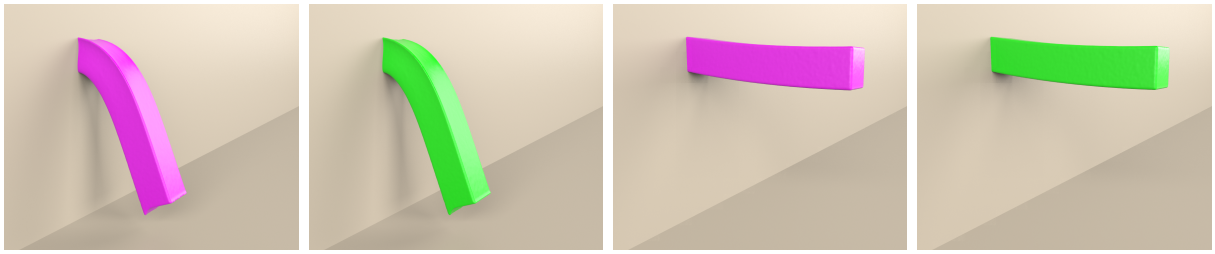


Figure 3.6: The same frame for different discretizations of an initially horizontal viscous beam collapsing. Left pair: Our chosen viscosity model (Batty and Bridson, 2008) naturally supports rotation and bending: (magenta) regular grid; (green) our proposed octree framework. Right pair: The natural boundary conditions of simpler Laplacian models ($\mu(\nabla \boldsymbol{u})\boldsymbol{n} = 0$) inhibit bending, leading instead to excessively stiff shearing. Nevertheless, our proposed octree framework is effective when applied to this model as well.

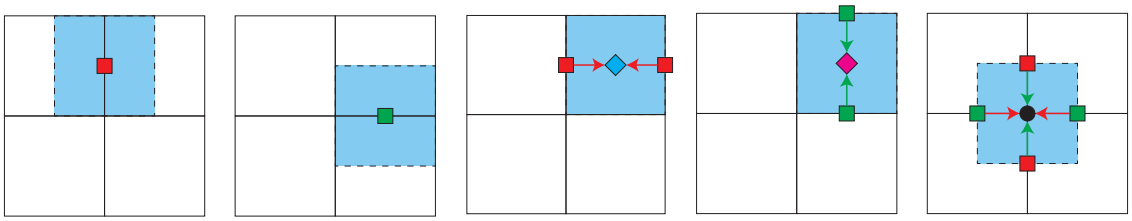
where Ω is the liquid domain and $\|\cdot\|_F$ indicates the Frobenius norm. At solid boundaries we apply a no-slip condition given by $\boldsymbol{u} = \boldsymbol{u}_S$. We apply a zero traction condition given by $\boldsymbol{t} = \boldsymbol{\tau}\boldsymbol{n} = \mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\mathsf{T})\boldsymbol{n} = \mathbf{0}$ at free surfaces, where $\boldsymbol{t}$ is the surface traction vector and $\boldsymbol{n}$ is the free surface normal[3]. Two important features of the variational method are that the free surface boundary condition is enforced automatically and the velocity that minimizes the energy functional is the solution to the viscosity PDE (3.18). This method is also time-implicit, allowing for large time steps, and yields an SPD linear system, allowing the use of faster iterative solvers.

The continuous viscosity energy functional is discretized into volume elements around each component of velocity and stress,

$$\underset{\mathbf{u}^{n+1}}{\operatorname{argmin}} \frac{1}{2\Delta t}(\mathbf{u}^{n+1} - \mathbf{u}^n)^\mathsf{T}\mathbf{P}\mathbf{W}_u(\mathbf{u}^{n+1} - \mathbf{u}^n) - (\mathbf{D}\mathbf{u}^{n+1})^\mathsf{T}\mathbf{K}\mathbf{M}\mathbf{W}_\tau\mathbf{D}\mathbf{u}^{n+1}, \tag{3.20}$$

where $\mathbf{W}_u$ and $\mathbf{W}_\tau$ are diagonal matrices representing the volume of fluid inside each volume element. $\mathbf{D}\mathbf{u} \approx (\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\mathsf{T})$ is the deformation rate operator that maps discrete velocity samples to discrete stress components and $\mathbf{D}^\mathsf{T}\mathbf{K}$ is the discrete divergence operator. $\mathbf{M}$ is the diagonal matrix containing the dynamic viscosity coefficients, and $\mathbf{K}$ is a diagonal matrix with the necessary coefficients to represent the Frobenius norm in (3.19). Taking the gradient of (3.20) with respect to $\mathbf{u}^{n+1}$ gives the discrete system for solving viscosity

[3]This assumption reduces the range of the simulation to the trivial boundary conditions $p = 0$ and $\boldsymbol{\tau}\mathbf{n} = \mathbf{0}$, instead of the full stress-traction relationship $(\boldsymbol{\tau} - p\mathbf{I})\mathbf{n} = \mathbf{0}$. This simplification can still recreate viscous rotation and buckling, but it cannot recreate continuous coiling phenomena (Larionov et al., 2017).



(a) (b) (c) (d) (e)

Figure 3.7: Staggered regular grid variable locations for velocities, (a) u, (b) v, and stresses, (c) $\tau_{xx} = 2\mu\frac{\partial u}{\partial x}$, (d) $\tau_{yy} = 2\mu\frac{\partial v}{\partial y}$, (e) $\tau_{xy} = \mu(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})$, in two spatial dimensions, with corresponding control volumes shaded in blue and axis-coloured velocity gradient stencils for stresses.

implicitly:

$$(\mathbf{P}\mathbf{W}_u + \Delta t\mathbf{D}^\mathsf{T}\mathbf{K}\mathbf{W}_\tau\mathbf{M}\mathbf{D})\mathbf{u}^{n+1} = \mathbf{P}\mathbf{W}_u\mathbf{u}^n. \tag{3.21}$$

This discrete optimization problem is quadratic with positive material properties (i.e., positive diagonal matrices) and the linear system will be SPD by construction. We emphasize that although the viscous stiffness term can contain a null-space for ballistic liquids and is therefore only positive semi-definite, the diagonal mass term acts as a regularizer that enforces positive-definiteness of the combined linear system. This SPD property holds independent of the discretization of the deformation rate operator $\mathbf{D}$ and eliminates the requirement for directly discretizing a symmetric divergence operator. We briefly review the regular grid discretization of Batty and Bridson (2008).

2D Discretization

Batty and Bridson discretize their viscosity system using the same staggered grid layout as the pressure projection. Velocity control volumes, $\mathbf{W}_u$, are centered around each active velocity sample and provide a measure of how much liquid is present in the volume element (Figures 3.7a-3.7b). The viscous stress components, τ_{xx} and τ_{yy}, are placed at the center of grid cells and τ_{xy} is placed at the corner of grid cells (since the stress tensor is symmetric, $\tau_{xy} = \tau_{yx}$). Because stress components are derivatives of velocity, they are strategically placed between their necessary velocity samples in accordance with our centered finite difference approximation. Figures 3.7c-3.7e illustrate the placement of the τ_{xx} (*cyan*), τ_{yy} (*magenta*), τ_{xy} (*black*), stress components, along with their associated control volumes, $\mathbf{W}_\tau$, and colour-coded velocity gradient stencils.

3D Discretization

Stress components in 3D are similarly positioned between their necessary velocity samples. Diagonal stress components, τ_{xx} (*cyan*), τ_{yy} (*magenta*), and τ_{zz} (*yellow*), still reside in cell centers with axis-aligned velocity gradients (Figures 3.8a-3.8c). Given that velocity samples are placed at grid face centers in 3D, off-diagonal stress components are equivalently moved to grid edges. These off-diagonal stresses also expand to three different sets, τ_{xy}, τ_{xz}, and τ_{yz}, with each set residing on a corresponding set of axis-aligned grid edges (Figures 3.8d-3.8f). Control volumes are similarly extended to 3D, forming cubes at grid faces for velocity samples, and cell centers and edges for stress components (Figure 3.9).

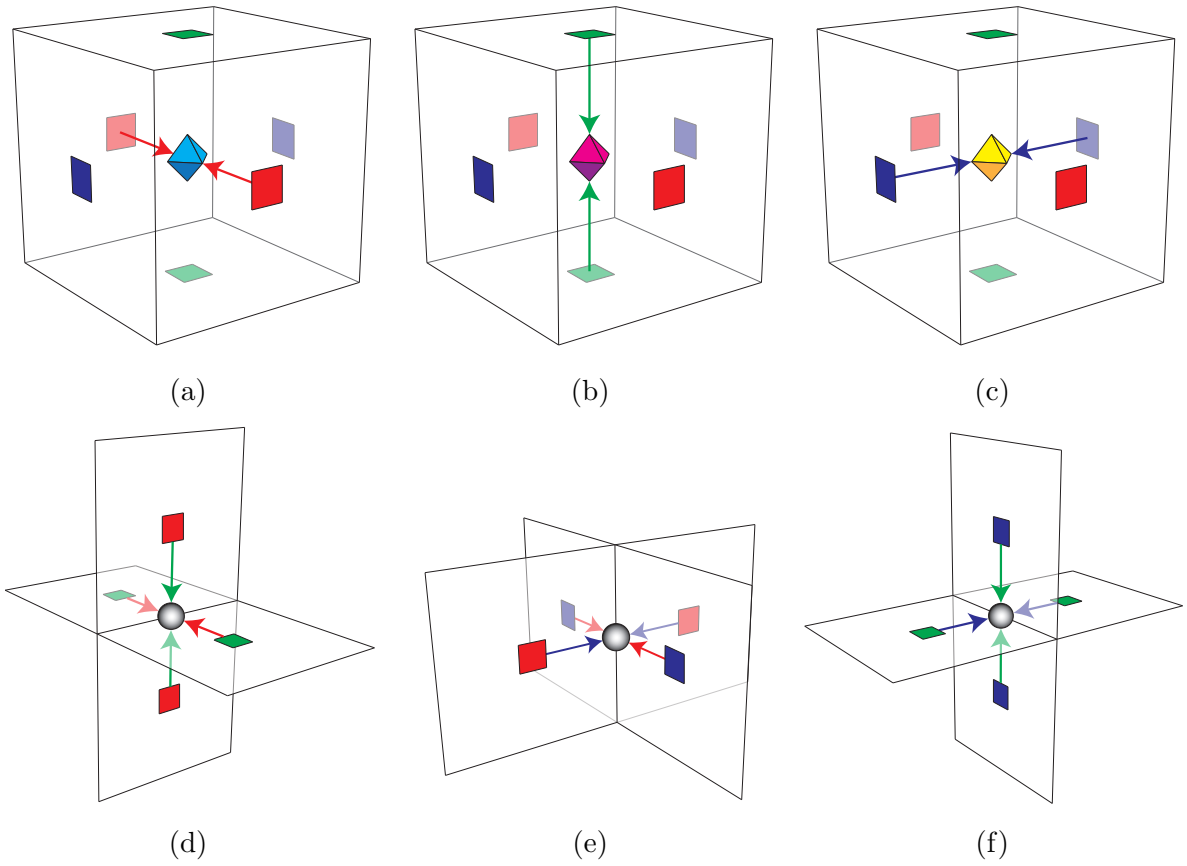
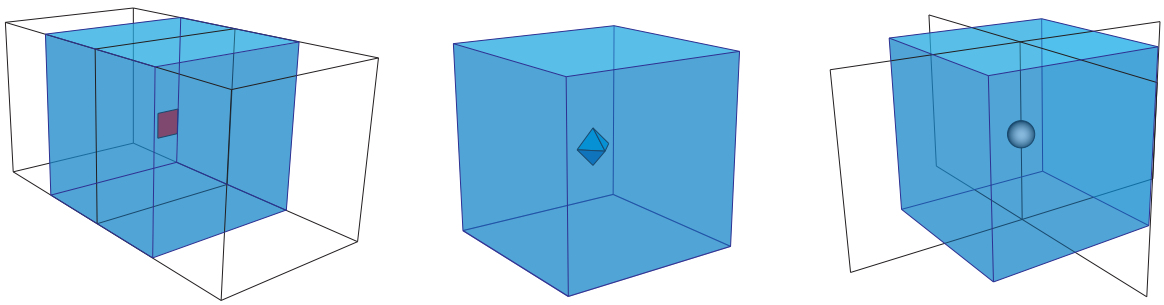


Figure 3.8: Staggered regular grid variable locations for face velocities in red, green, blue, cell-centered stresses in cyan (a) ($\tau_{xx} = 2\mu\frac{\partial u}{\partial x}$), magenta (b) ($\tau_{yy} = 2\mu\frac{\partial v}{\partial y}$), yellow (c) ($\tau_{zz} = 2\mu\frac{\partial w}{\partial z}$), and edge-centered stresses in gray (d) ($\tau_{xy} = \mu(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})$), (e) ($\tau_{xz} = \mu(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x})$), (f) ($\tau_{yz} = \mu(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y})$). Velocity gradient stencils for each stress component are coloured by axis of differentiation.



(a) (b) (c)

Figure 3.9: Control volumes for velocities (a) center stresses (b) and edge stresses (c) in 3D. The entire set of control volumes is easily generalized from these three examples.

Chapter 4

Reduced Fluid Models

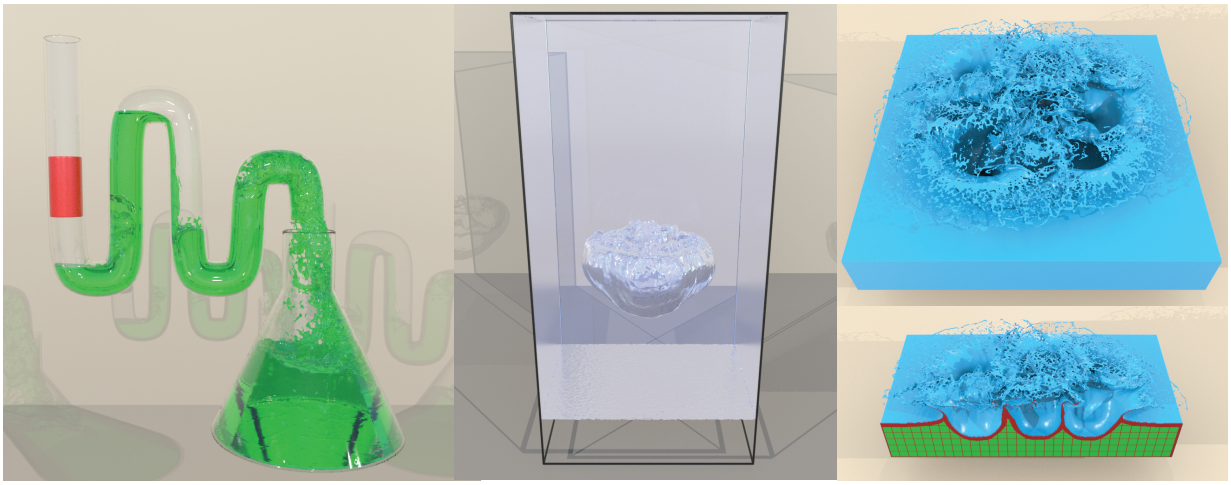


Figure 4.1: (Left) Our *constraint bubble* model allows distinct liquid (green) and solid (red) bodies to physically interact across completely unsimulated air gaps. (Middle) Immersed bubbles denser than water correctly sink, despite the interior degrees of freedom being radically reduced with our *affine region* model. (Right) Our affine region model also enables a convenient and flexible approach to liquid adaptivity with irregularly shaped coarse tiles (green).

The two-phase flow method presented in §3.2.2 for simulating air-liquid interaction is computationally expensive due to the additional pressure DOFs required throughout the air volume and the large density ratios between the two fluids that lead to ill-conditioned linear systems. We propose our constraint model that instead applies a single constraint

per air volume to the single-phase liquid free surface method: the net velocity flow into and out of an air bubble must be zero. Our constraint method effectively models incompressible, zero-density air bubbles with no air density terms appearing in the linear system. However, the constraint model breaks down for simulating fluids with non-neglible densities. We instead propose our affine region method to model the velocity field of a bubble with a single pointwise affine vector field. For large bubble volumes, a single affine vector field is not expressive enough to capture all the large-scale modes of fluid motion. In this case, we introduce a tiling strategy to subdivide a single monolithic affine region into smaller subregions, connected by active pressure cells. Additionally, our affine regions using this tiling strategy also generalize as an acceleration technique for single-phase free surface flows.

4.1 Constraint Bubbles

4.1.1 Continuous Setting

Our constraint-based bubble model augments the standard single-material pressure projection, (3.4), with support for incompressible air bubbles. As shown in Figure 3.1b, we divide the simulation volume into three material domains, Ω_A, Ω_S, and Ω_L, corresponding to air, solid and liquid regions, respectively. We will refer to any closed continuous air region as a "bubble". A single liquid region may contain zero or more bubbles within it. A liquid region may also be entirely surrounded by a single "bubble"; that is, we make no distinction between exterior air and submersed air, viewing all as bubbles. Bubble and liquid regions may also be arbitrarily nested.

Our desired behavior is that each bubble preserves its total volume. For the i^{th} bubble, we can express this as a linear velocity constraint,

$$\int_{\partial\Omega_{A_i}} \boldsymbol{u}_A \cdot \boldsymbol{n}\, dA = 0, \tag{4.1}$$

which we can separate into liquid and solid parts,

$$\underbrace{\int_{\Omega_L \cap \partial\Omega_{A_i}} \boldsymbol{u} \cdot \boldsymbol{n}\, dA}_{\mathsf{B}_i(\boldsymbol{u})} + \underbrace{\int_{\Omega_S \cap \partial\Omega_{A_i}} \boldsymbol{u}_S \cdot \boldsymbol{n}\, dA}_{b_{S_i}} = 0. \tag{4.2}$$

That is, the integrated flow through the entire boundary of a single continuous bubble region, Ω_{A_i}, must be zero. Enforcement of this constraint involves information about the

velocity field everywhere on the bubble's boundary (i.e., either liquid or solid velocities touching the bubble), so we have denoted the contribution of the liquid surface to the i^{th} bubble as the linear operator $\mathsf{B}_i(\boldsymbol{u})$ and the prescribed solid's contribution as b_{S_i}. Crucially however, no information about velocities *interior* to the bubble is required. In some ways this is unsurprising; since we have assigned zero mass to the bubble, its momentum is negligible and therefore a model for the velocity interior to the bubble can be completely avoided.

Collecting all of the bubble constraints into a single vector operator B, our modified pressure PDE takes the following form

$$\begin{aligned} \rho\frac{\partial \boldsymbol{u}}{\partial t} &= -\nabla p - \frac{\partial \mathsf{B}^{\mathsf{T}}}{\partial \boldsymbol{u}}\lambda, \\ \nabla\cdot\boldsymbol{u} &= 0, \\ \mathsf{B}(\boldsymbol{u}) &= -b_S, \end{aligned} \tag{4.3}$$

where λ is the vector of Lagrange multipliers having one component per bubble.

4.1.2 Discrete Setting

We begin by directly discretizing the single-phase pressure PDE following the finite volume method described in §3.2.1, yielding the following indefinite linear system:

$$\begin{pmatrix} \frac{1}{\Delta t}\mathbf{M} & \mathbf{D}^{\mathsf{T}} \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t}\mathbf{M}\mathbf{u}^{n} \\ \mathbf{0} \end{pmatrix}. \tag{4.4}$$

Here $\mathbf{p}$ is the vector of discrete pressures, and $\mathbf{u}^n$ and $\mathbf{u}^{n+1}$ are the vectors of velocity components before and after projection, respectively. $\mathbf{M}$ and $\mathbf{D}$ are the usual diagonal fluid mass matrix and discrete divergence operator, which incorporate irregular free surfaces via the ghost fluid method (Enright et al., 2003) and irregular solid walls via cut-cells (Batty et al., 2007; Ng et al., 2009). (Note that diagonal entries of $\mathbf{M}$ are zero for entirely air and solid faces, so the corresponding rows and columns drop out.)

We use a row-vector $\mathbf{B}_i$ to represent the discretization of the i^{th} bubble constraint from (4.2), which sums the net flow across the bubble's incident liquid faces such that:

$$\mathbf{B}_i\mathbf{u} = \sum_{\text{liquid faces of } \partial\Omega_{A_i}} \mathbf{u}\cdot\mathbf{n}_{\text{face}}\,\mathsf{dA}_{\text{face}}. \tag{4.5}$$

In this expression, $\mathbf{n}_{\text{face}}$ is the cell face normal oriented out of the bubble region, and $\mathsf{dA}_{\text{face}}$ is the area of the relevant face. (Because we are employing the cut-cell methodology

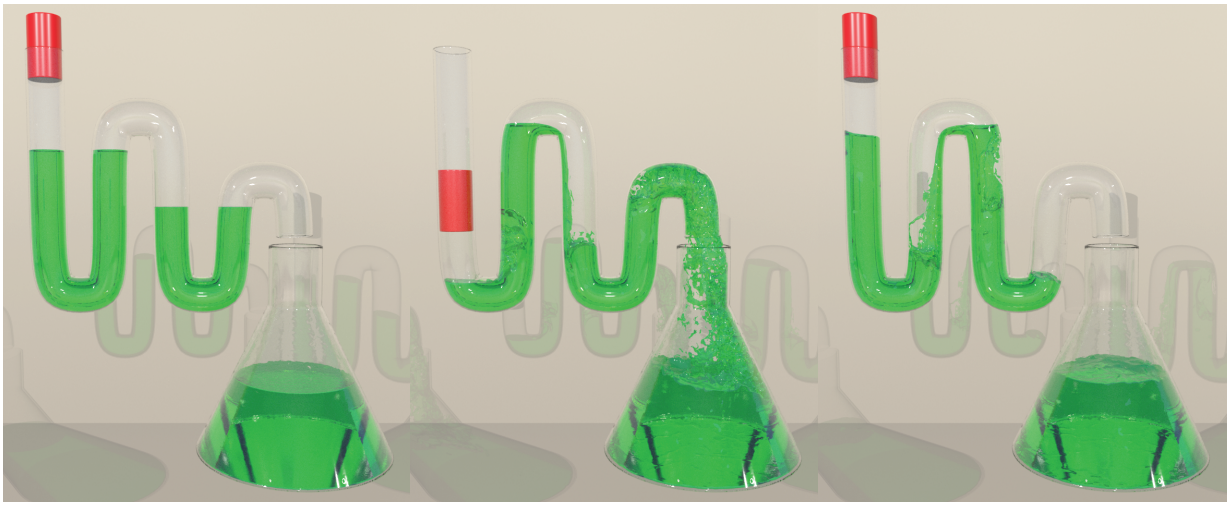


Figure 4.2: Our constraint-based approach allows two distinct liquid regions to correctly interact across completely unsimulated air. A moving piston pushes liquid through the tube indirectly through constraint-bubble regions.

(Batty et al., 2007; Ng et al., 2009), we account for only the partial area outside of solids.) Effectively, this constraint measures the aggregate discrete divergence for the entire bubble; the corresponding Lagrange multiplier λ will act as a collective pseudo-pressure enforcing the bubble's volume to be unchanging. Since $\mathbf{B}_i$ only involves liquid velocities touching the bubble, the discretization is relatively sparse.

If the bubble touches any kinematically scripted moving solids, we appropriately modify the right hand side of (4.5) to add contributions from the surfaces of those solids, i.e.,

$$b_{S_i} = \sum_{\text{solid faces of } \partial\Omega_{A_i}} \mathbf{u}_S \cdot \mathbf{n}_{\text{face}} \, \mathsf{dA}_{\text{face}}. \tag{4.6}$$

Doing so allows moving solids to affect even liquid surfaces that they are *not in direct physical contact with*, such as when an air bubble in an enclosed tube separates a liquid from a moving piston: the force is communicated through the bubble, as expected (see Figure 4.2). The same ideas can be extended in a straightforward fashion to model interactions with strongly coupled, dynamic rigid or deformable bodies (Batty et al., 2007; Robinson-Mosher et al., 2009).

Stacking the bubble constraints into a single wide matrix $\mathbf{B}$, and incorporating them into (4.4) yields a large sparse symmetric indefinite linear system that is the discrete version

of (4.3):

$$\begin{pmatrix} \frac{1}{\Delta t}\mathbf{M} & \mathbf{D}^\mathsf{T} & \mathbf{B}^\mathsf{T} \\ \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ \mathbf{p} \\ \lambda \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t}\mathbf{M}\mathbf{u}^n \\ \mathbf{0} \\ -\mathbf{b}_S \end{pmatrix}. \tag{4.7}$$

This indefinite system includes the pressure, constraint, and velocity degrees-of-freedom and is therefore quite large. However, since $\mathbf{M}$ is diagonal and hence trivially invertible, we can take the Schur complement (i.e., solve the first row for $\mathbf{u}^{n+1}$ and substitute into the latter two rows) to eliminate velocity and arrive at a smaller SPD system in terms of pressure and the bubbles' Lagrange multipliers:

$$\begin{pmatrix} \Delta t\mathbf{D}\mathbf{M}^{-1}\mathbf{D}^\mathsf{T} & \Delta t\mathbf{D}\mathbf{M}^{-1}\mathbf{B}^\mathsf{T} \\ \Delta t\mathbf{B}\mathbf{M}^{-1}\mathbf{D}^\mathsf{T} & \Delta t\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^\mathsf{T} \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{D}\mathbf{u}^n \\ \mathbf{B}\mathbf{u}^n + \mathbf{b}_S \end{pmatrix}. \tag{4.8}$$

After solving this linear system for $\mathbf{p}$ and λ, the final velocity $\mathbf{u}^{n+1}$ can be obtained using the first row of (4.7). Since $\mathbf{M}$ is diagonal, this amounts to a simple matrix-vector multiply. The upper-left block of (4.8) is the usual pressure Poisson system and the remaining blocks account for interaction with the bubble constraints. Compared to the standard pressure solve, the extra Lagrange multipliers have added one row and one column per bubble.

Our system has a similar structure to the one that arises in the compressible flow method of Aanjaneya et al. (2013), but ours assumes zero density bubbles, does not require terms related to bubble expansion or compression, and supports scripted moving objects. Most importantly, we do not require a second advection step or pressure solve to determine the (visually imperceptible) interior air motion, which allows us to cheaply simulate large regions of empty air. For small air densities, our constraint method also qualitatively agrees with a full two-phase flow simulation (see Figure 4.4).

Identifying the set of individual bubble regions can be done by determining connected components through a simple flooding approach over air cells that share faces. The flooding must be done over the air *volume*, rather than simply over connected *surfaces*, so that any nesting of regions is properly identified and handled. For example, a single bubble might contain a disconnected interior droplet; in this case the bubble should have a single volume constraint accounting for both of these disconnected bubble-liquid surfaces.

For the purposes of the advection phase, we perform standard extrapolation of the liquid velocity field into the empty air region (Enright et al., 2002). This form of extrapolation does not preserve the divergence-free velocity constraint inside air regions, however, a second pressure projection inside the air region (as done in Aanjaneya et al. (2013)) adds significant additional cost. We explored approximately enforcing divergence-free velocities

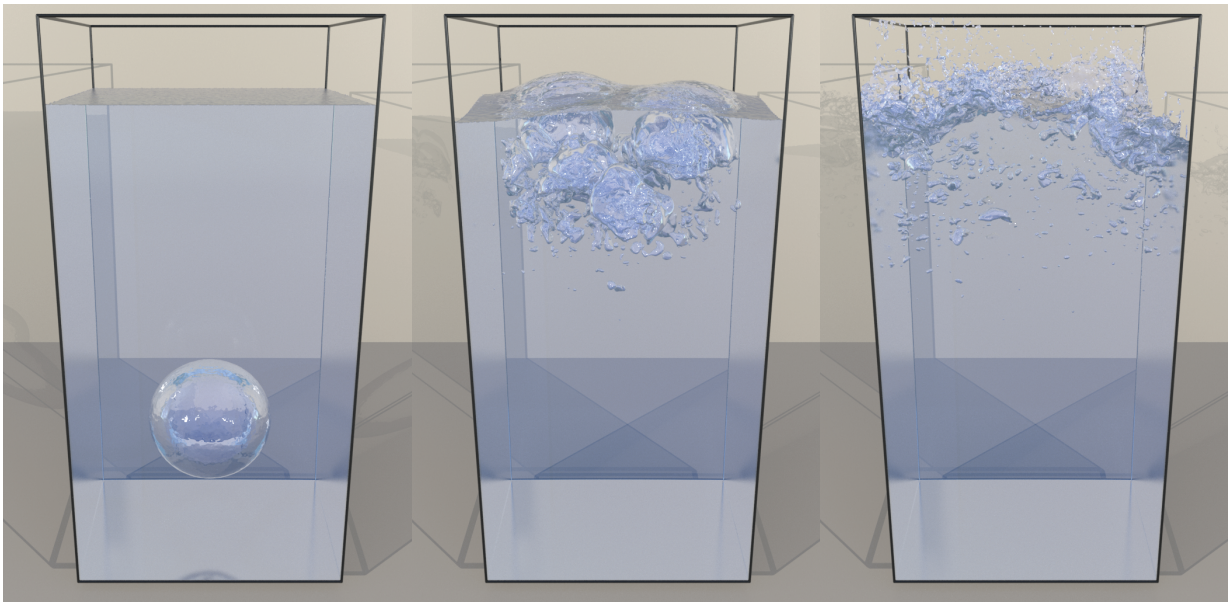
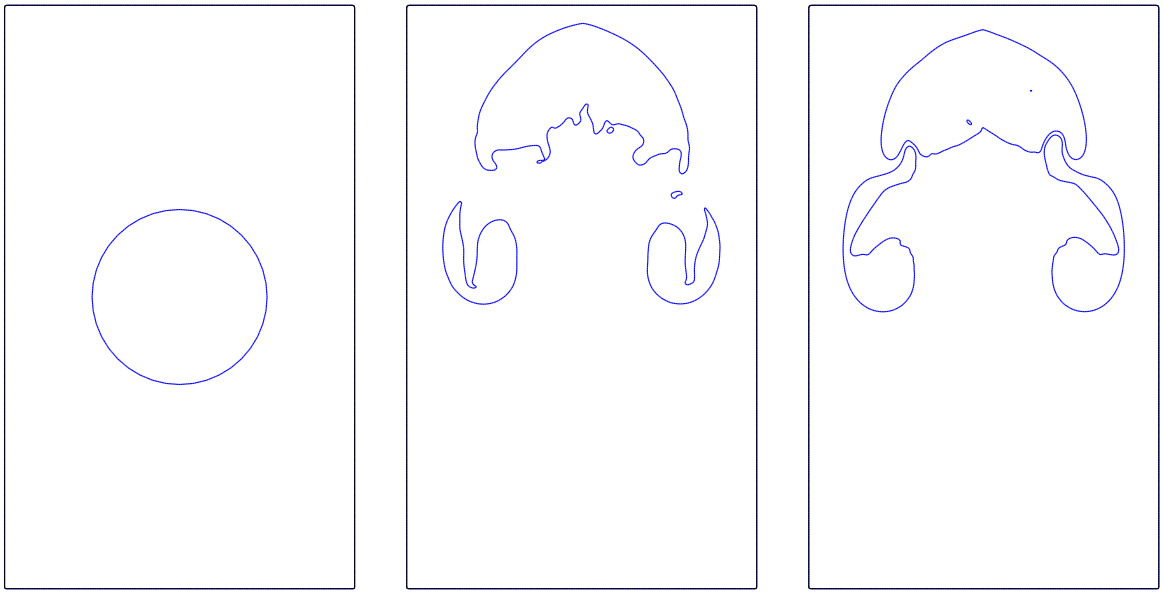


Figure 4.3: A zero-density air bubble inside a liquid column rises and breaks apart into many small bubbles. Our per-bubble volume tracking and correction framework allows small bubbles to persist over long periods of time, across complex topology changes, and without explicit air particles.



(a) Initial single bubble (b) Full two-phase (c) Constraint bubble

Figure 4.4: Our zero-density constraint-based simulation closely resembles a two-phase simulation with a small air density of $\rho = 1$. The liquid density is $\rho = 1000$ for both simulations.

by first extrapolating and then applying several Jacobi smoothing passes of the pressure Poisson system throughout the air region, but ultimately found this to be unstable.

4.2 Affine Regions

4.2.1 Motivation

As our results will demonstrate, the preceding reduced model is ideal for the common case of (approximately) zero-density bubbles, since it requires only one extra degree-of-freedom per bubble and no model whatsoever for the interior air. However, if one is interested in animating two-phase flows with more general density coefficients, so that the immersed phase rises more slowly, remains neutrally buoyant, or even sinks, this choice is insufficient. That is because its Lagrange multipliers are equivalent to one constant pressure value on each bubble's interior, even if one incorporates the bubble's desired density into the

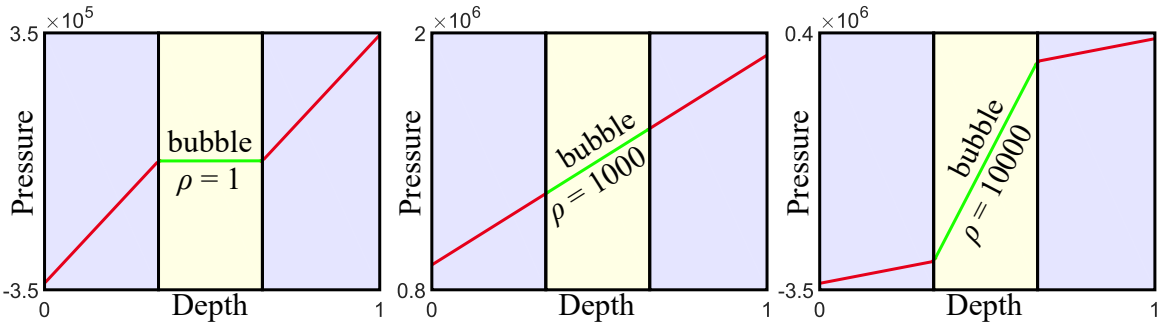
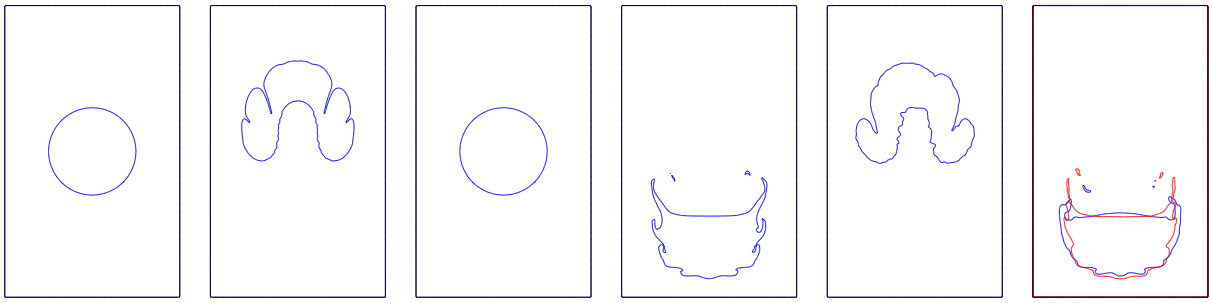


Figure 4.5: Ideal pressure profiles for vertical one-dimensional multiphase fluids in hydrostatic balance with each other for three different density ratios. The pale yellow region is the immersed bubble at $\rho = 1$, $\rho = 1000$ and $\rho = 10000$, while the pale magenta region has constant density $\rho = 1000$. A single constant pressure is a good approximation for the bubble region in the low-density (left) case, but would be inaccurate for the other cases.

equations (as the more complex compressible model of Aanjaneya et al. (2013) attempts to do). Examining the expected pressure profile of a 1D two-phase flow scenario in a vertical column at hydrostatic balance (Figure 4.5) provides useful intuition. For low bubble densities, a single constant interior pressure value is a good approximation of the true interior pressure field. However, for moderate to high bubble density coefficients, a constant pressure cannot produce the correct velocity update.

This observation initially led us to consider imposing a *linear* model of interior pressure based on the liquid pressures surrounding the bubble. Since a linear pressure field induces a constant pressure gradient (i.e., velocity update) it can correct the global translation of the bubble, and for example, recover hydrostatic balance for neutrally buoyant bubbles. However, a constant velocity correction is still too limited in the vector fields it can describe. For example, a bubble rising to collide with a boundary cannot spread out in opposing directions. While a higher order pressure model might yield better results, we also found this framework somewhat unwieldy and had difficulty preserving symmetry. Moreover, the fact that the bubble density is no longer negligible in this setting suggests that a model for interior momentum is necessary.

Therefore, rather than focusing on pressure and ignoring the interior velocity field, we adopt an explicit model for the *velocity* inside a coalesced region by assuming it to be both incompressible and affine. This turns out to have several attractive properties, including ease of symmetry preservation, an analytically divergence-free interior flow, and a structure very similar to well-known solid-fluid coupling models, while still enabling us to abstract



(a) Full two-phase ($\rho_B = \rho_L = 10^3$)
(b) Constraint bubble ($\rho_B = \rho_L = 10^3$)
(c) Affine bubble ($\rho_B = \rho_L = 10^3$)
(d) Full two-phase ($\rho_B = 10^4$, $\rho_L = 10^3$)
(e) Constraint bubble ($\rho_B = 10^4$, $\rho_L = 10^3$)
(f) Affine bubble ($\rho_B = 10^4$, $\rho_L = 10^3$)

Figure 4.6: **Model Comparison in 2D**: When the immersed fluid (i.e., bubble) has non-negligible density, constraint bubbles yield incorrect motion compared to full two-phase simulations; however, affine region-based bubbles exhibit correct buoyancy. Initial conditions are a circular bubble; each image shows the same time instant shortly into the simulation. Left trio: A neutrally buoyant bubble (i.e., bubble density ρ_B matches liquid density ρ_L) should remain stationary. Right trio: A higher density bubble ($\rho_B > \rho_L$) should sink. In (f), the blue simulation used a single interior affine region and the red simulation used interior tiling (Section 4.3) for improved fidelity.

away large collections of adjacent fluid cells.

4.2.2 Continuous Setting

Consider a region of fluid Ω_B that is constrained to possess an incompressible affine velocity field. Such a body's velocity $\boldsymbol{u}_B$ at position $\boldsymbol{x}$ can be described by the relation,

$$\boldsymbol{u}_B(\boldsymbol{x}) = \boldsymbol{u}_{\text{const}} + \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}_{\text{COM}}), \tag{4.9}$$

where $\mathbf{A} = \nabla \boldsymbol{u}_B$. That is, velocity at any point in the affine velocity field is computed from the constant, translational velocity at the center of mass $\boldsymbol{x}_{\text{COM}}$ plus a linear correction dictated by the velocity gradient. (The center of mass is used for convenience, but any fixed reference point suffices.) Some representative examples (computed using (4.9)) are shown in Figure 4.7 to highlight the perhaps surprising expressiveness of divergence-free affine vector fields.

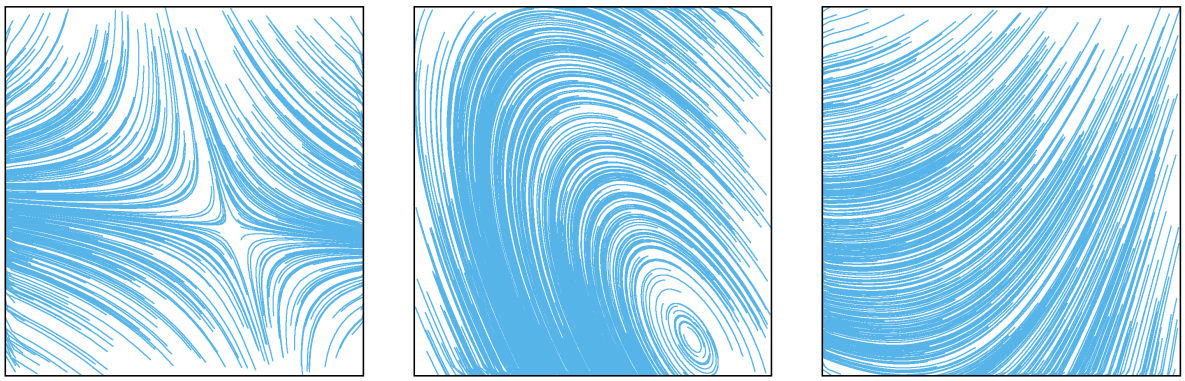


Figure 4.7: Representative divergence-free affine velocity fields (plotted as streamlines) generated by our model, highlighting its expressiveness.

To enforce incompressibility, the field must satisfy the usual condition $\nabla \cdot \boldsymbol{u}_B = 0$. Applying this constraint to (4.9) shows that the velocity gradient must be trace-free, i.e., $\mathrm{Tr}(\mathbf{A}) = 0$. This constraint leads to $\mathbf{A}$ having the following reduced form:

$$\mathbf{A}_{2D} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & -a_{11} \end{bmatrix}, \quad \mathbf{A}_{3D} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & -(a_{11}+a_{22}) \end{bmatrix}. \tag{4.10}$$

Our task now is to develop an understanding of the dynamics of a fluid body with such a velocity field. Notice that if we were to instead require the velocity field to satisfy $\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathsf{T}} = \mathbf{0}$, i.e., zero strain rate (e.g., (Carlson et al., 2004)), we would recover a rigid body motion, where the velocity consists of translational and *rotational* parts. This is a useful analogy for understanding our model, similar to the relationship between the rigid (RPIC) and affine (APIC) particle-in-cell models proposed by Jiang et al. (2015).

For simplicity we consider the 2D case and denote our generalized velocity vector as $\mathbf{v}_B = [u, v, a_{11}, a_{12}, a_{21}]$. We can then define the matrix $\mathbf{C}(\boldsymbol{x})$ that extracts the Euclidean velocity $\boldsymbol{u}_B$ at a point $\boldsymbol{x}$:

$$\boldsymbol{u}_B = \mathbf{C}(\boldsymbol{x})\mathbf{v}_B \tag{4.11}$$

$$= \begin{bmatrix} 1 & 0 & x - x_{\mathrm{COM}} & y - y_{\mathrm{COM}} & 0 \\ 0 & 1 & -(y - y_{\mathrm{COM}}) & 0 & x - x_{\mathrm{COM}} \end{bmatrix} \mathbf{v}_B. \tag{4.12}$$

The 3D case follows straightforwardly. Under our rigid body analogy, we will also require a generalized mass matrix for the affine fluid body, which dictates how the body's generalized

velocity $\mathbf{v}_B$ changes under applied forces. Since the kinetic energy of the fluid body can be described by the integral

$$\int_{\Omega_B} \frac{\rho_B}{2} \|\boldsymbol{u}_B\|^2 \, dV = \int_{\Omega_B} \frac{\rho_B}{2} \|\mathbf{C}\mathbf{v}_B\|^2 \, dV \tag{4.13}$$

$$= \tfrac{1}{2}\mathbf{v}_B^{\mathsf{T}} \left(\int_{\Omega_B} \rho_B \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C} \, dV \right) \mathbf{v}_B, \tag{4.14}$$

the symmetric positive definite matrix $\mathbf{M}_B = \int_{\Omega_B} \rho_B \mathbf{C}^{\mathsf{T}}\mathbf{C} \, dV$ is exactly our desired generalized mass matrix.

Our intention is to immerse this affine fluid body within a regular grid-based fluid solver, and enforce two-way coupling between them; we model our approach loosely on the rigid-body coupling framework of Batty et al. (2007). Therefore, our next task is to determine how the affine vector field of this fluid body is affected by the surrounding pressure p of the regular fluid. As in the rigid body case, the net translational component of the pressure force is simply the integral of pressure acting on the body's surface:

$$\boldsymbol{f}_{\text{const}} = \int_{\partial\Omega_B} p\boldsymbol{n} \, dA, \tag{4.15}$$

where $\boldsymbol{n}$ is the affine fluid body's surface normal. We can similarly account for the effect on the velocity gradient components of $\mathbf{v}_B$ using

$$\boldsymbol{F}_{\text{linear}} = \int_{\partial\Omega_B} p\boldsymbol{n}(\boldsymbol{x} - \boldsymbol{x}_{\text{COM}})^{\mathsf{T}} \, dA, \tag{4.16}$$

which plays a role analogous to the net torque on a rigid body. (The expressions above can be derived, for example, by differentiating the power that the surrounding fluid pressure applies on the boundary, $\int_{\partial\Omega_B} \boldsymbol{u}_B \cdot (p\boldsymbol{n}) \, dA$, by the affine velocity degrees of freedom.) By flattening $\boldsymbol{F}_{\text{linear}}$ into a vector to correspond with $\mathbf{v}_B$ and combining these two contributions, we form a linear operator $\mathbf{J}$ that transforms (regular grid) boundary fluid pressures into generalized forces on the affine fluid body's degrees of freedom. Together with our generalized mass matrix, we have an (Eulerian) update rule for our affine fluid body in terms of our reduced degrees of freedom: $\mathbf{M}_B \frac{\partial \mathbf{v}_B}{\partial t} = \mathbf{J}p$. (We have handled advection separately, hence this expression does not use the material derivative, $\frac{D\mathbf{v}_B}{Dt}$.)

Finally, if $\boldsymbol{u}_F$ denotes the velocity field of the surrounding regular fluid, the combined

pressure projection PDE that we seek to solve is

$$\begin{aligned} \rho \frac{\partial \boldsymbol{u}_F}{\partial t} &= -\nabla p \\ \nabla \cdot \boldsymbol{u}_F &= 0 \\ \mathbf{M}_B \frac{\partial \mathbf{v}_B}{\partial t} &= \mathbf{J} p \end{aligned} \tag{4.17}$$

subject to $\boldsymbol{u}_F = \boldsymbol{u}_B$ on their shared boundary. Incompressibility of Ω_B does not appear explicitly in these expressions because it is enforced implicitly through the reduced velocity model. This PDE will take tentative values for $\boldsymbol{u}_F$ and $\mathbf{v}_B$ after advection and application of forces, and simultaneously ensure incompressibility of both regions while handling the exchange of forces between them.

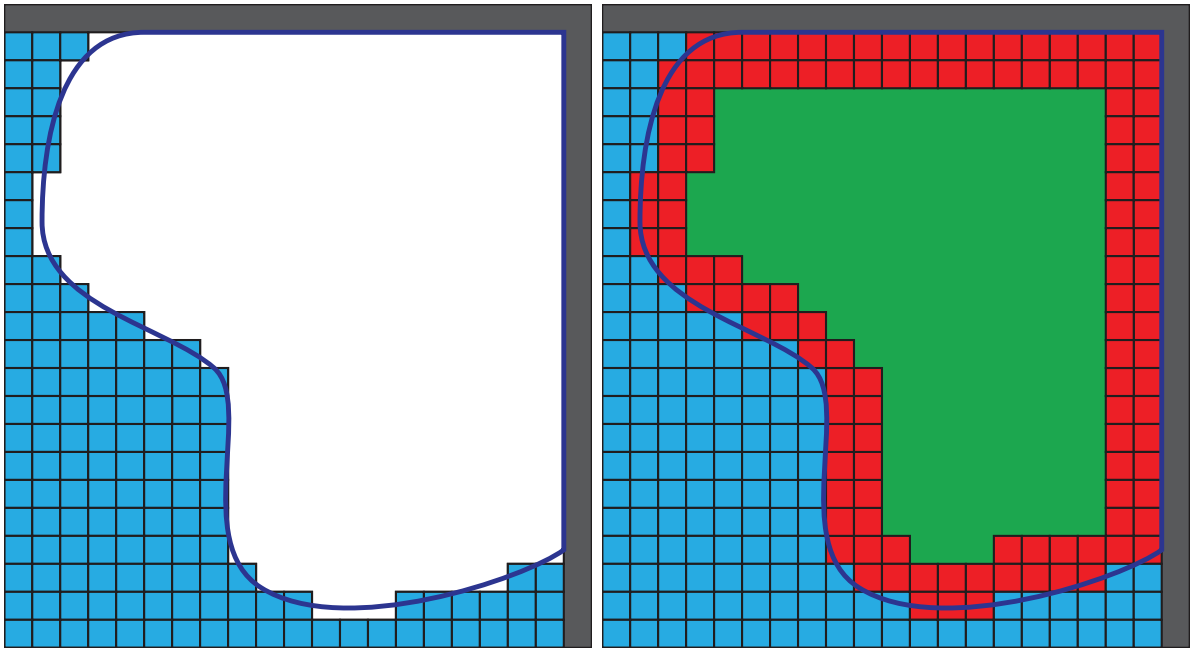
The motivating application for our reduced model is simulating entrained bubbles, but bubbles tend to be highly deformable near their interface, especially with low surface tension. As such, the affine model alone is inadequate for this case because it lacks the necessary flexibility to capture the rapidly emerging surface details formed during bubble break-up. However, the affine approximation *is* often effective for the smoother (and invisible) interior air motion away from the interface itself. We therefore adopt a standard ghost-fluid two-phase flow model (Hong and Kim, 2005) to handle the interface conditions and a narrow interior band of air, while replacing only the slightly deeper interior air region(s) with our affine model. Figure 4.8 compares our grid setup for constraint bubbles and affine bubbles.

4.2.3 Discrete Setting

To discretize the coupling problem above, we adopt a standard finite volume approach for the surrounding regular grid fluid. We assume that the boundary between the affine and regular grid regions lies exactly on grid-aligned faces, recalling that it does not correspond to a physical boundary but rather a change of representations. (However, one could adopt a cut-cell formulation (Ng et al., 2009) to generalize to irregular shapes, if this was deemed beneficial.)

Directly discretizing the PDE (4.17) yields a symmetric indefinite linear system:

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_F & \mathbf{D}^\mathsf{T} & \mathbf{0} \\ \mathbf{D} & \mathbf{0} & \mathbf{J}^\mathsf{T} \\ \mathbf{0} & \mathbf{J} & -\frac{1}{\Delta t}\mathbf{M}_B \end{bmatrix} \begin{bmatrix} \mathbf{u}_F^{n+1} \\ \mathbf{p} \\ \mathbf{v}_B^{n+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_F \mathbf{u}_F^n \\ \mathbf{0} \\ -\frac{1}{\Delta t}\mathbf{M}_B \mathbf{v}_B^n \end{bmatrix}. \tag{4.18}$$



(a) Constraint bubble (b) Affine bubble

Figure 4.8: Constraint bubbles use no interior fluid cells; affine bubbles pad the interior affine region with a thin band of uniform cells to capture detailed surface deformations. Legend: ● = liquid cells, ○= constraint bubble region, ● = exterior bubble cells, ● = interior affine region, ● = solid boundary.

The first row is the usual velocity update for the regular fluid region Ω_F, where $\mathbf{M}_F$ is a diagonal mass matrix with entries corresponding to regular fluid faces and $-\mathbf{D}^\mathsf{T}$ is the pressure gradient operator across those faces. The second row represents the discrete divergence of regular fluid cells, with contributions from regular fluid faces given by $\mathbf{D}\mathbf{u}_F^{n+1}$ and from coupled affine fluid faces given by $\mathbf{J}^\mathsf{T}\mathbf{v}_B^{n+1}$. Finally, the third row gives the effect of the pressure on the affine region's velocity. The variables $\mathbf{u}_F^n$ and $\mathbf{v}_B^n$ represent the provisional velocities after applying external forces and advection.

It remains only to define the discrete mass matrix $\mathbf{M}_B$ and discrete pressure force operator $\mathbf{J}$. The discrete mass matrix is

$$\mathbf{M}_B = \sum_{a,i} \rho \mathbf{C}_a(\mathbf{x}_i)^\mathsf{T} \mathbf{C}_a(\mathbf{x}_i)\, \mathsf{dV}, \tag{4.19}$$

where a iterates over the axis directions (x, y, z), i iterates over all faces in that axis having one or both of its two incident cells inside the affine region, $\mathbf{x}_i$ is the midpoint of the face, and dV is the volume of a cell. The notation $\mathbf{C}_a(\mathbf{x})$ indicates the row of $\mathbf{C}$ corresponding to the axis a, evaluated at $\mathbf{x}$. The discrete $\mathbf{J}$ operator (matrix) is

$$\mathbf{J}\mathbf{p} = \sum_{a,j} \mathbf{C}_a(\mathbf{x}_j)^\mathsf{T} p_j \mathbf{n}_j \mathsf{dA}, \tag{4.20}$$

where a iterates over the three axis directions, j iterates over the faces *on the boundary* of the affine region, $\mathbf{x}_j$ is the center of the associated face, p_j is the pressure at the center of the incident regular fluid cell, $\mathbf{n}_j$ is the normal to that face, and dA is the area of a grid face.

To reduce the system size of (4.18), we can form the Schur complement to eliminate fluid velocity $\mathbf{u}_F^{n+1}$ as we did for constraint bubbles:

$$\begin{bmatrix} -\Delta t \mathbf{D}\mathbf{M}_F^{-1}\mathbf{D}^\mathsf{T} & \mathbf{J}^\mathsf{T} \\ \mathbf{J} & -\frac{1}{\Delta t}\mathbf{M}_B \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{v}_B^{n+1} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\mathbf{u}^n \\ -\frac{1}{\Delta t}\mathbf{M}_B\mathbf{v}_B^n \end{bmatrix}. \tag{4.21}$$

This system is still symmetric indefinite, with a form that closely mirrors the rigid-body coupling approaches of Robinson-Mosher et al. (2008; 2009); as such, it will require an indefinite solver such as MINRES or QMR. However, if desired, a second Schur complement can be applied with respect to the block-diagonal lower-right block, which eliminates $\mathbf{v}_B^{n+1}$ and yields a smaller symmetric positive definite system in terms of fluid pressure alone,

$$-\Delta t(\mathbf{D}\mathbf{M}_F^{-1}\mathbf{D}^\mathsf{T} - \mathbf{J}^\mathsf{T}\mathbf{M}_B^{-1}\mathbf{J})\mathbf{p} = -\mathbf{D}\mathbf{u}^n - \mathbf{J}^\mathsf{T}\mathbf{v}_B^n. \tag{4.22}$$

This system's structure matches that of Batty et al. (2007) and, being SPD, is amenable to solution with Conjugate Gradients. However, this transformation comes at the cost of introducing the dense block $\mathbf{J}^\mathsf{T}\mathbf{M}_B^{-1}\mathbf{J}$, which mutually couples all pressures incident on the boundary of the affine region. Depending on the numerical strategies chosen to solve this problem, one of these two forms may be preferable. We defer discussion of our numerical solver to §4.4.2.

At each application of the above projection, we first recover the input $\mathbf{v}_B^n$ for the affine region using a simple least squares fit over the region's fluid grid faces. These are computed from standard advection on the grid (e.g., by semi-Lagrangian, particle-based, etc.) The least squares problem has the form

$$\underset{\mathbf{v}_B^n}{\operatorname{argmin}} \sum_{a,i} \left(u_{a,i} - \mathbf{C}_a(\mathbf{x}_i)\mathbf{v}_B^n\right)^2, \tag{4.23}$$

where a iterates over the axes, i iterates over all grid faces of the affine region, and $\mathbf{x}_i$ is the face midpoint (velocity sample point).

We conclude this section by observing that, despite our original motivation being bubbles, our divergence-free affine fluid model makes no assumptions that are specific to the two-phase setting. It can therefore also be applied to the interior of a free-surface flow with the same benefit of reducing its active degrees-of-freedom.

4.3 Tiled Affine Regions for Adaptivity

The preceding affine model provides an effective approximation for the interior of fluid bodies with modest sizes, both for single-phase free-surface and multiphase flow scenarios. At the same time, a single affine region is inherently limited in the complexity of velocity fields that it can describe. Therefore, to further extend our model's usefulness to even larger interior regions, we propose a simple tiling strategy that uses multiple medium-sized affine regions, stitched together by thin layers of uniform regular grid cells. As can be seen in Figure 4.10, this leads to fine uniform cells in a band around interfaces, irregularly shaped affine tiles next to them, and finally large rectangular tiles filling the deeper regions. On the interior, a one-layer band of uniform cells is placed between affine tiles. Our tiling approach drastically reduces the degrees-of-freedom needed to capture a large domain, while providing a good qualitative match to fully uniform regular grid flows, as in Figures 4.11 and 4.12.

There are several additional advantages to this approach. Compared to an octree, tetrahedral mesh, or Chimera grid approach, it is much simpler to integrate into a regular

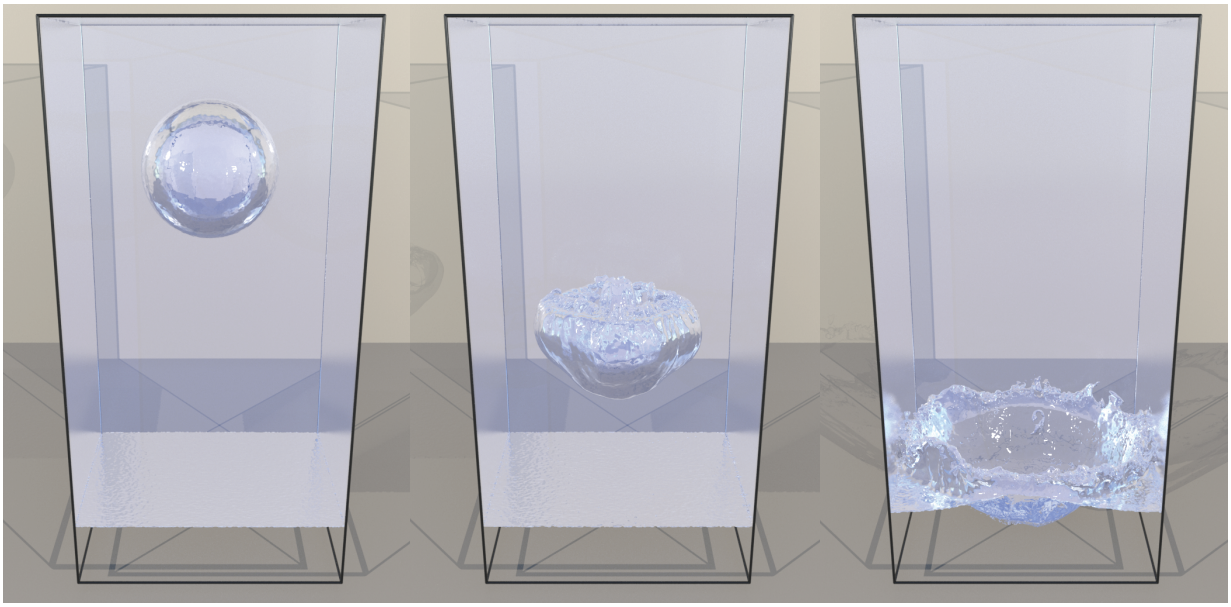


Figure 4.9: A bubble with higher density than the surrounding liquid correctly sinks in 3D. Its interior is tiled with multiple affine regions, as in §4.3.

grid simulation pipeline, since it requires no elaborate data structures or complex (re)meshing. It provides more effective and general coarsening than tall cells or stretched grids. Notably, it provides analytically divergence-free fields in the coarsened regions, a feature which may be useful for some applications that no competing adaptivity method for fluid animation provides. The $\mathbf{J}$ and $\mathbf{M}_B$ matrices needed for perfectly rectangular tiles can be precomputed and reused for efficiency.

4.4 Simulator Design

4.4.1 Volume Tracking and Correction

To foster adoption in existing industrial pipelines, we have intentionally chosen to extend the standard free-surface FLIP method (Zhu and Bridson, 2005). Whereas previous Eulerian (or hybrid) methods for bubbles employ multi-material level sets or particles to directly track *both* the liquid and bubble regions, we instead extend basic FLIP by simply labeling any simultaneously non-solid and non-liquid region as a bubble. However, it is well-known that accumulated numerical advection errors cause liquid FLIP particles to separate or condense

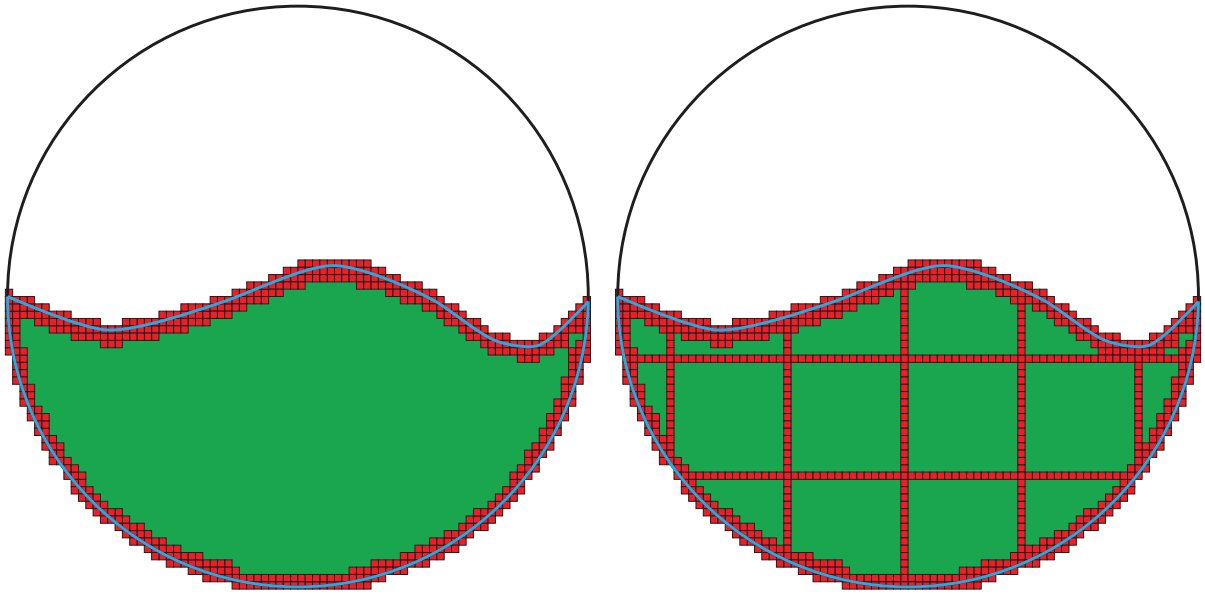


Figure 4.10: Left: A free-surface flow with a single interior affine region (green). Right: The same flow subdivided into a mix of regular (square) and irregular affine tiles for greater flexibility.

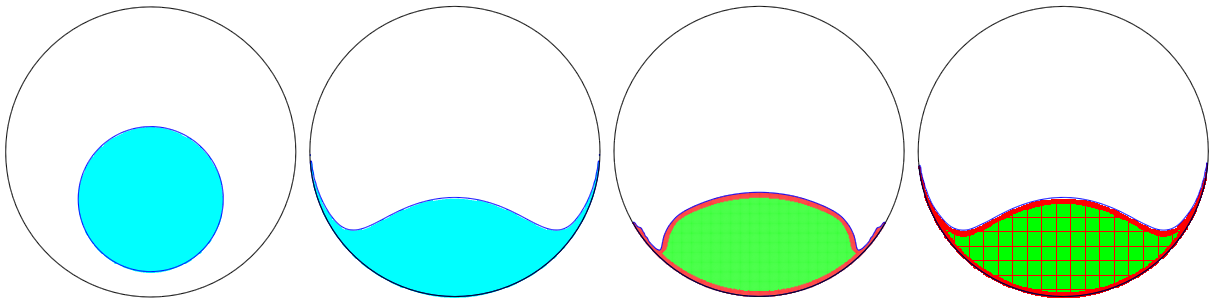


Figure 4.11: (a) A 2D ball of free-surface fluid is released under gravity, and falls to collide with the circular domain boundary. (b) Ground truth regular grid simulation. (c) A simulation with a *single* interior affine region (green) is insufficiently flexible for this scenario. (d) A simulation using *tiled* affine regions provides a much closer match to the ground truth.

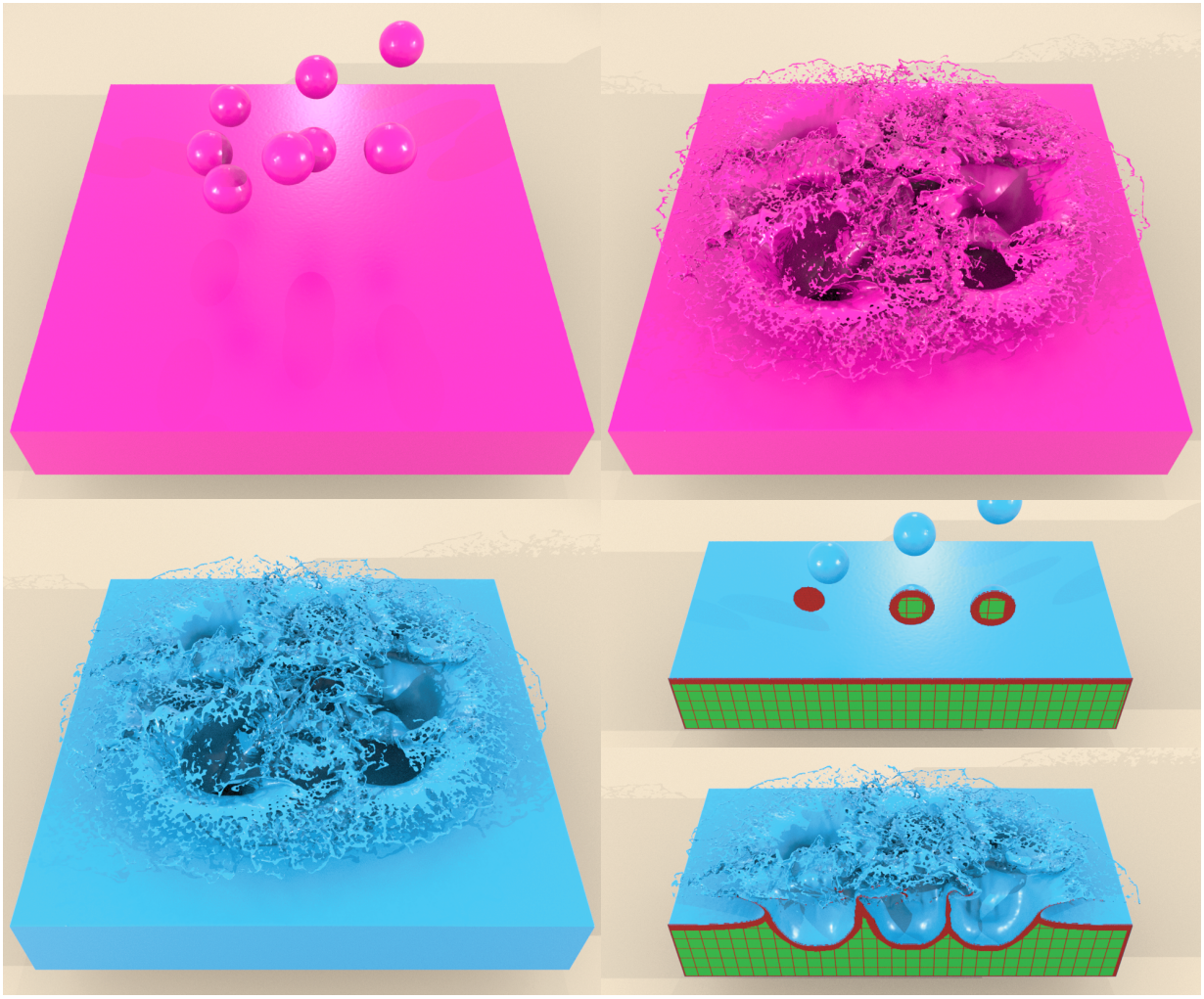


Figure 4.12: A high resolution splashing scenario. Top-left: Initial conditions. Top-right: Regular grid simulation. Bottom-left: A simulation with interior tiled affine regions offers a close qualitative match. Bottom-right: Cutaway view of the interior tiling.

over time (Kugelstadt et al., 2019), inducing erroneous volume change and occasionally creating spurious empty gaps or *voids*. Since we do not explicitly track the air geometry, liquid volume drift destructively modifies the implied bubble volume with it, while artificial voids give birth to false bubbles that begin to rise. In turbulent simulations along solid boundaries, this can even give the appearance of solid surfaces *boiling*. Explicit tracking of the bubble material could prevent voids and somewhat reduce volume change, but would also add nontrivial overhead. To address these issues, we propose to track bubbles implicitly by augmenting each FLIP particle with a new bubble ID attribute.

We build a mapping of bubble identities from one time step to the next using the *old* bubble IDs stored on the FLIP particles from the previous step and the bubble IDs assigned to *new* bubble regions that those particles end up next to in the current step. In Figure 4.13, particles are initially assigned the ID of their adjacent bubble and, after advection, the particle IDs are used to map the old bubble IDs to the new bubble IDs. This mapping forms a bipartite graph, where nodes represent bubble regions and edges indicate whether bubbles have simply advected or undergone more complex merging and splitting.

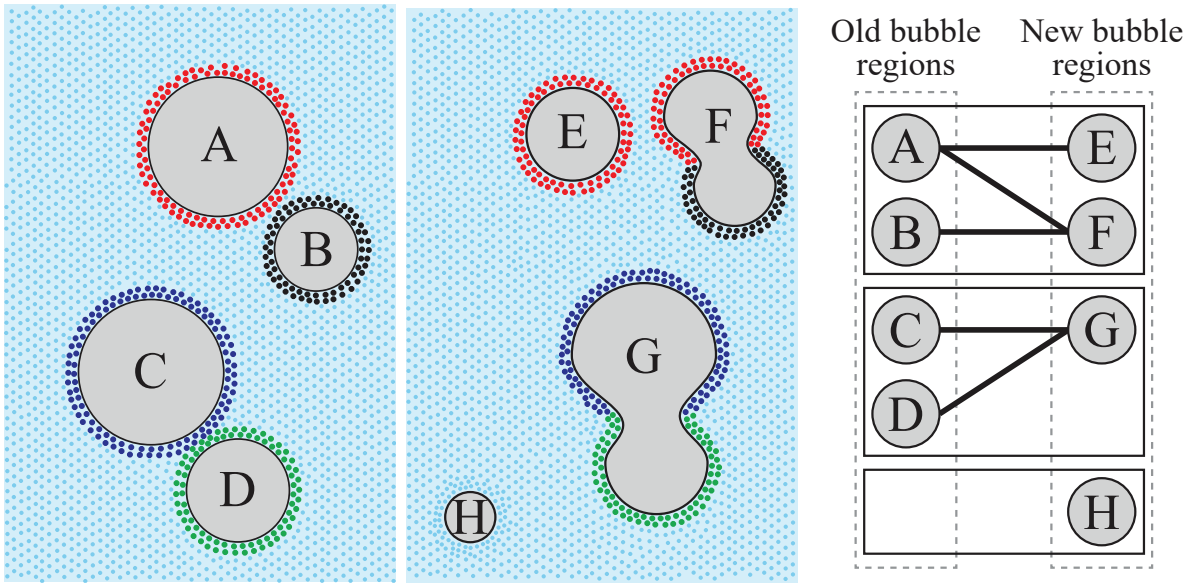
Removing Spurious Void Bubbles Using this mapping, *void* bubbles correspond to new bubble ID nodes with no incoming edges. We seek to collapse them away by applying a negative (rather than zero) divergence (Feldman et al., 2003). A bubble's volume change relates to its divergence by

$$\int_{\Omega_B} \nabla \cdot \boldsymbol{u} dV \approx \frac{V_B^{n+1} - V_B^n}{\Delta t}, \tag{4.24}$$

where a bubble is driven to collapse by setting $V_B^{n+1} = 0$.

Volume Correction Equation 4.24 can also be used to correct volume drift by setting a bubble's target volume to its initial *rest* volume, $V_B^{n+1} = V_B^0$. For old-new bubble pairs having a one-to-one map, we simply copy the rest volume from the old bubble to the new. For more complex scenarios where bubbles have possibly split and/or merged, it is not possible to directly assign rest volumes. Instead, we find connected components in the bipartite graph of bubble mappings and redistribute the current rest volumes for all old bubble regions in the set. Updated rest volumes are assigned according to the volume of each new bubble relative to the accumulated volume of each new bubble region in the set,

$$V_{B_i}^0 = \frac{V_{B_i}^n}{\sum_{k \in G_i^n} V_{B_k}^n} \sum_{j \in G_i^0} V_{B_j}^0, \tag{4.25}$$



(a) Start of time step. (b) After advection. (c) Bubble graph.

Figure 4.13: Based on empty regions and their surrounding particle labels (a) before and (b) after advection, we construct a bipartite graph (c) to conservatively redistribute target rest volumes. Two cases of note are: (1) bubble A splits while partially merging with B; (2) bubble H is a spurious void formed by minor particle advection errors, which should be collapsed away.

where $V_{B_i}^0$ and $V_{B_i}^n$ are the rest and current volumes of the i^{th} new bubble, and G_i^n and G_i^0 are the set of new and old bubbles in the connected component containing the i^{th} bubble, respectively.

This componentwise volume tracking and correction approach is very general in nature, and so can also be applied to the liquid-phase to good effect, or even free surface FLIP simulations without bubbles.

4.4.2 Optimized Linear Solver

Faster Matrix-Vector Multiplication As with rigid-body coupling, the affine contribution $\mathbf{J}^\intercal\mathbf{M}^{-1}\mathbf{J}$ in the SPD system (4.22) is a dense block that mutually couples all cells on an affine region's boundary, quickly becoming a bottleneck. We adopt the suggestion of Bridson (2015) to exploit this block's low rank (Batty et al., 2007) by storing it in factored

form and, when needed during PCG, performing three sparse matrix-vector multiplications in place of a single much denser one.

Multigrid Preconditioning Both constraint bubbles and affine fluid regions are reduced models that represent immersed bubbles either with one Lagrange multiplier, or a small layer of exterior cells with an interior affine velocity field. This reduction in the total number of degrees of freedom increases simulation performance. However, our experiments using a parallelized diagonal preconditioned Conjugate Gradients solver indicate that the major computational bottleneck is ultimately the much larger liquid domain. Recent work by Aanjaneya (2018) proposed an efficient solver for multi-domain systems using a Schur-complement method that applies a direct solver on the reduced model and a geometric Multigrid V-cycle for the liquid domain. Because our affine bubble model includes exterior cells at the bubble's boundary, the cost of a direct solve per solver iteration can be prohibitively expensive. Instead, we propose a simplified preconditioner that couples a lightweight smoothing routine over the bubble's domain and a Multigrid V-cycle over the liquid domain.

The preconditioner computes an approximate solution to $\mathbf{As} = \mathbf{r}$, where $\mathbf{A}$ is the linear system from either (4.8) for constraint bubbles, or (4.22) for affine fluid regions. We partition the degrees of freedom into liquid regions Ω_L, and bubble regions Ω_B, which we denote by $\mathbf{s}_L$ and $\mathbf{s}_B$, respectively. We found that extending the bubble domain to overlap with the liquid domain in a three voxel-wide narrow band gave faster convergence and better performance, so the minor additional cost was a worthwhile trade-off.

We first apply several iterations of damped Jacobi smoothing over the bubble region, denoted by the approximate inverse operator $\mathbf{A}_B^{\ddagger}$ in Algorithm 2. Then, we apply a geometric Multigrid V-cycle to Ω_L, denoted by $\mathbf{A}_L^{\dagger}$, closely following McAdams et al. (2010). To account for the intermediate values in the bubble domain, we modified the right-hand side vector as $\mathbf{r}_L - \mathbf{A}_{LB}\mathbf{s}_B^{n-1}$, where $\mathbf{A}_{LB}$ is the subsystem of $\mathbf{A}$ that couples the bubble regions with the liquid domain. After applying a V-cycle, the same smoothing routine $\mathbf{A}_B^{\ddagger}$ is applied to each bubble region, this time accounting for intermediate values from the liquid domain, by modifying the right-hand side vector as $\mathbf{r}_B - \mathbf{A}_{BL}\mathbf{s}_L^n$ (where $\mathbf{A}_{BL}$ is the transpose of $\mathbf{A}_{LB}$). Note that each V-cycle $\mathbf{A}_L^{\dagger}$ is sandwiched between two applications of the smoother $\mathbf{A}_B^{\ddagger}$, preserving the symmetry of our preconditioner.

Throughout the Multigrid V-cycle, we employ a variation of red-black Gauss-Seidel to apply smoothing over the liquid domain in parallel. We found that a red-black coloring scheme applied at the cell level offered no meaningful improvement in convergence compared to a Jacobi smoother. However, applying a red-black scheme at the *tile level* of Houdini's

sparse grids resulted in 2× faster convergence than Jacobi smoothing (SideFX, 2021). A tile is a 16^3 collection of grid cells and we apply red-black coloring to divide tiles between threads. Gauss-Seidel smoothing is then applied to each cell within a single tile in serial per thread. We found this strategy offers the full benefit of parallelism with the improved convergence of Gauss-Seidel when compared to Jacobi smoothing.

Intuitively, Algorithm 2 is trying to solve a two-way coupled system by performing outer iterations on a partitioned solver until convergence. This is also similar in spirit to the iterative scheme proposed by Aanjaneya (2018) to approximate the solution to the Schur-complement system. Moreover, like Aanjaneya (2018), we observed that increasing the total number of iterations N inside the preconditioner improved solver convergence. However, in our experiements we found that the additional cost of applying subsequent V-cycles outweighed the reduction in the total number of PCG iterations. For this reason, we used $N = 1$ for all our simulations.

Algorithm 2 Coupled Preconditioner $\mathbf{As} = \mathbf{r}$

$\mathbf{s}_B^0 = \mathbf{A}_B^{\ddagger}\mathbf{r}_B$
for $n = 1 \dots N$ **do**
  $\mathbf{s}_L^n = \mathbf{A}_L^{\dagger}(\mathbf{r}_L - \mathbf{A}_{LB}\mathbf{s}_B^{n-1})$
  $\mathbf{s}_B^n = \mathbf{A}_B^{\ddagger}(\mathbf{r}_B - \mathbf{A}_{BL}\mathbf{s}_L^n)$
end for

4.5 Simulations Results

To demonstrate that our proposed models can be implemented as direct extensions to a standard single-phase FLIP-based liquid simulator, we developed our models as plugins to Houdini's FLIP solver (SideFX, 2021). All our 3D examples were simulated on a 16-core Ryzen 1950x CPU. The linear systems were solved using a custom Conjugate Gradients method using double precision, with a relative tolerance of 10^{-5} for all examples. All our timings are listed in Table 4.1.

4.5.1 Water Cooler

Figure 4.14 illustrates the familiar glugging effect of a *water cooler* scenario. The traditional single-phase free surface approach simply allows the liquid to pour rapidly through as

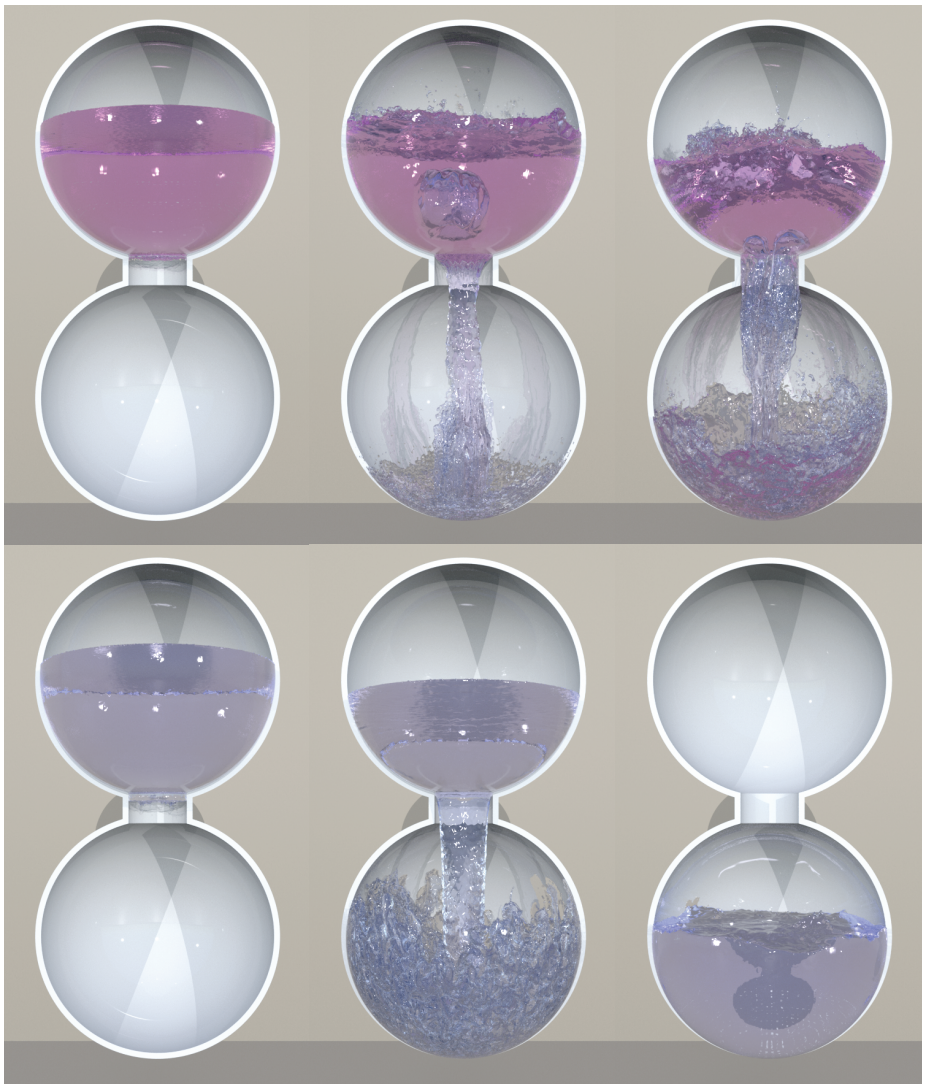


Figure 4.14: (Top) Our constraint-bubble approach recovers the familiar *glugging* motion of liquid pouring through the neck of the water cooler. (Bottom) A standard free surface liquid simulation fails to recreate the expected motion.

Table 4.1: **Timing breakdowns:** Timings for all our 3D examples. "Linear Solve Time" refers strictly to the time to solve the pressure projection linear system itself; "Pressure Projection Overhead" involves all the components of our extended constraint bubble / affine region pressure projection process that are not the linear system solver, i.e., flood-filling to find regions, setting up tiling structures, etc.

Simulation	Resolution	Method	Linear Solve Time	Pressure Projection Overhead	Total Simulation Time
Water Cooler	200x450x200	Free Surface	1h21m	10m	4h07m
		Constraint	1h49m	15m	4h42m
Rising Bubbles	200x400x200	Constraint	8h44m	40m	12h03m
Bubble Tube	350x390x180	Constraint	1h40m	15m	2h42m
Sinking Bubbles	200x400x200	Two-phase	40h38m	1h58m	45h49m
		Affine (bubble only)	13h06m	2h13m	18h35
		Affine (both fluids)	9h59m	1h47m	15h02
Splash Tank	500x350x500	Regular Grid	29h04m	51m	33h04m
		Affine (16^3 tiles)	6h29m	1h39m	11h54m
		Affine (32^3 tiles)	3h46m	1h32m	8h37m

if both chambers were open to the outside air. By enforcing the incompressible bubble constraints, the downward flow of liquid must match the upward flow of air, generating a sequence of rising bubbles that are constantly being created and pinched off.

Because of the large air regions in the simulation, our constraint method greatly reduces the number of DOFs in the pressure projection when compared to a full two-phase simulation. For the first frame of this example, an equivalent two-phase simulation would contain 8.7M DOFs. For the same substep, our constraint method reduces the total DOFs in the linear system to 2.5M unreduced DOFs and 2 bubble DOFs. Although the bubble region DOF count fluctuates throughout the simulation due to small bubbles being entrained, it is only in the hundreds.

Despite this example containing 3× more air volume than liquid volume, the solver overhead associated with adding bubbles to the linear system, flood-filling, and bubble ID tracking is insignificant. For example, solving the linear system comprises 87% of the total pressure projection time. Flood-filling to build bubble connected components and tracking bubble IDs both comprise less than 1% of the total pressure projection. The lion's share of overhead was dedicated to building the linear system, which would be significantly slower if a full two-phase linear system was constructed with 3.5× more DOFs.

A possible shortcoming of the constraint method is that the row and column corresponding to a given bubble's constraint can be relatively dense depending on the bubble's surface area. This is because each bubble constraint involves all liquid face velocities incident on that bubble; elimination of the velocity variables leads to coupling between the bubbles Lagrange multiplier and the pressures of all its incident cells. This adds overhead compared to a pure free-surface flow solver (in addition to the cost of identifying bubble regions), but we found it to be relatively minor. The (Multigrid preconditioned) free-surface method took a total of 4h07m to simulate the water cooler vs. our constraint-bubble method at 4h42m, i.e., bubbles were 14% slower. Although the free-surface and constraint bubble water coolers exhibit wildly differing behaviors which affects their efficiency (notably, the former settles down quickly), this data nevertheless suggests that artists can expect to add air bubble effects to scenes for only a small additional cost.

4.5.2 Rising Bubbles

In Figure 4.3, we simulate a zero-density constraint bubble immersed in water. The incompressibility constraint applied to the bubble prevents the surrounding liquid from collapsing it. As the surrounding liquid flows downwards under gravity, the massless bubble is forced to rise. Subsequently, turbulent velocities cause it to break apart into many small bubbles. Although we use FLIP particles only for the liquid, and the bubbles undergo frequent splitting and merging, our method is able to track these bubble volumes and correct their volume drift.

This example contains a large liquid body consisting of 13.7M liquid DOFs, compared to only a would-be 2.6M bubble DOFs needed for proper a two-phase simulation. Although reducing 2.6M DOFs to only a few constraint DOFs is a significant reduction, it is still relatively small compared to the remaining size of the liquid-region pressure Poisson linear system. This example motivates our coupled Multigrid preconditioner (see §4.4.2) in an effort to optimize the residual reduction for the large liquid body. A single substep comparison shows that our coupled Multigrid preconditioner outperforms a traditional diagonal preconditioner for Conjugate Gradients by 6.3×, converging in 15 iterations over 15.9s, compared to 750 iterations over 1m38s using the diagonal preconditioner. The large density ratio between liquid and air regions, 1000:1, for an equivalent two-phase flow solver leads to poorly conditioned linear systems. We observed that over the first five frames, a standard two-phase method required an average of 1752 iterations using the diagonal preconditioner and was 17.6× slower than our Multigrid-preconditioned constraint method. A two-phase Multigrid preconditioner such as Wan and Liu (2004) could offer improved convergence but would still require pressure DOFs throughout the bubble volume.

4.5.3 Bubble Tube

Figure 4.2 demonstrates how kinematically scripted moving solid boundaries interact with our constraint-based air bubbles. When the *red* piston pushes down through the glass tube, it creates a net flux at the solid-air boundary (and corresponding right-hand-side terms in the pressure solve) that must be compensated by an opposing flux at the air-liquid boundary. This flux on the liquid surface pushes the liquid volume through the tube, despite it never having come into direct contact with the solid. Similarly, the fully disconnected liquid regions interact through the second air pocket and the resulting chain of interactions, driven by the moving piston, forces liquid out of the spout and into the beaker below. As the piston pulls back to its initial position, it creates a negative flux that draws the liquid back into the tube along with it.

The ambient air region surrounding the glass tube is 21× larger than the volume of liquid in the simulation implying a huge potential speed-up; including this body of air in a full-two phase simulation would clearly be prohibitively expensive. Fortunately in this case, one could choose to treat the ambient region as a Dirichlet condition and remove it from the two-phase pressure system, since it does not affect the tube's inner dynamics (though the interior air gaps would still require full two-phase air DOFs). However, if two liquid bodies were to be hydraulically connected across a potentially huge ambient air region, *only* our approach will suffice. To identify connected air components on which to apply our constraint bubbles, a flood-fill operation across all such *potentially* active cells is required, which could hypothetically be a bottleneck. We found, though, that despite the ambient region being very large in this problem our reasonably optimized/parallel flood-fill comprised only 2% of the total pressure projection time.

We observed that our geometric Multigrid preconditioner did not perform well for this example. This is a well-known limitation of Multigrid schemes like that of McAdams et al. (2010) for simulations with *maze*-like solid boundary conditions. Employing a topology-aware coarsening scheme could alleviate this problem (Dick et al., 2016).

4.5.4 Sinking Bubbles

As demonstrated in Figure 4.6, our proposed affine velocity method is able to maintain hydrostatic equilibrium for bubbles with density coefficients matching the surrounding liquid, as well as capturing the expected sinking motion of bubbles with even higher density. This same sinking behavior is further illustrated in Figure 4.9, in which a dense bubble falls through the surrounding liquid into a pool of dense bubble material initially at rest at the bottom of the tank.

Similar to the *rising bubble* example, the liquid volume is significantly larger than the bubble volume and simply reducing the DOF count in the bubble regions would offer only a small relative performance increase. We employ a Multigrid preconditioner similar to the constraint method, but with a small layer of exterior bubble cells included in the bubble smoothing routine. In a single time step comparison, a diagonal preconditioned Conjugate Gradients solve converges in 1274 iterations and takes 3m01s, compared to 50 iterations in 1m08s for our coupled Multigrid preconditioner, giving a performance improvement of 2.7×. We believe the lower performance improvement compared to the constraint-based examples is in part due to the use of a pointwise smoother in the exterior bubble region. Although a Multigrid V-cycle is incredibly effective at reducing the residual in the linear system for the liquid domain, a pointwise smoother over the exterior domain is not. We believe a *box* smoother, that would perform local direct solves over a group of DOFs, might offer improved performance (Aanjaneya et al., 2019).

We also experimented with applying the tiled affine method to *both* materials. Similar to the bubble-only affine method, we keep a layer of exterior cells at the two-phase and solid boundaries. We found that using the same layer thickness as the bubble-only method resulted in subtle grid artifacts as the bubble sank. Doubling the thickness from three voxels to six voxels on each side of the two-phase boundary was enough to remove these artifacts. This approach offered a moderate performance improvement of 1.2× over the bubble-only affine coupled Multigrid method.

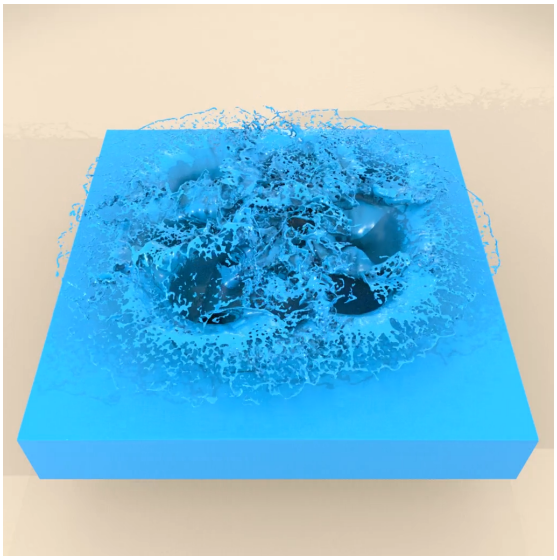
Both affine methods significantly outperform the two-phase equivalent by 3.1× for the affine method on the bubble only, and 4.1× for the affine method on both fluids. However, the performance increase is not as dramatic as the *rising bubbles* example because the density ratio, 10:1, of the two fluids is not as extreme.

4.5.5 Splash Tank

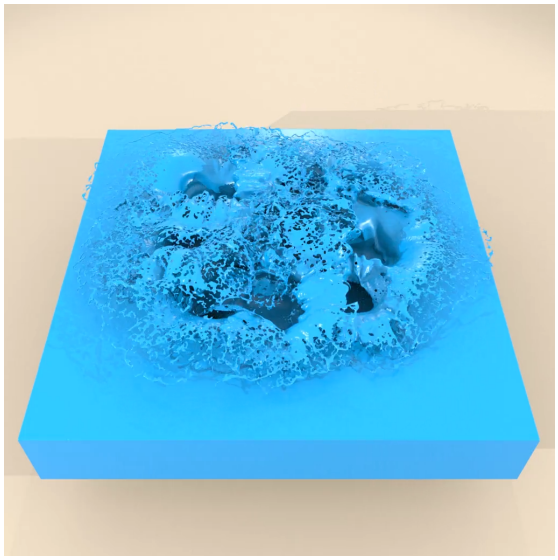
Figure 4.12 demonstrates that our affine model generalizes from the two-phase flows to the more common single-phase free-surface case, and can be used to create highly detailed results. In this example, both the spheres and liquid pool maintain a 9-voxel thick band at the liquid-air boundary, a 2-voxel thick band at the liquid-solid boundary, and their interiors are filled with tiled affine regions that are (at most) 16^3 voxels in size with a single layer of voxels between each tile. The affine regions lead to a reduction from 21.5M DOFs in the regular grid setting to 6.4M unreduced cells and 5.2K affine interior regions. The linear solve time of our affine free-surface model outperforms the regular grid method by 3.7× using a diagonal preconditioned Conjugate Gradients solver for both. We further

compared methods at 2× resolution, doubling the affine region size and boundary thickness, and observed a 6.1× performance improvement for a single time step. This suggests that our proposed method offers much better scaling than the regular grid for high resolutions. We emphasize that existing methods for improving the performance of a regular grid method, such as Multigrid or octree adaptivity, often involve complex data structures and implementations. By comparison, our proposed affine method requires minimal additional implementation.

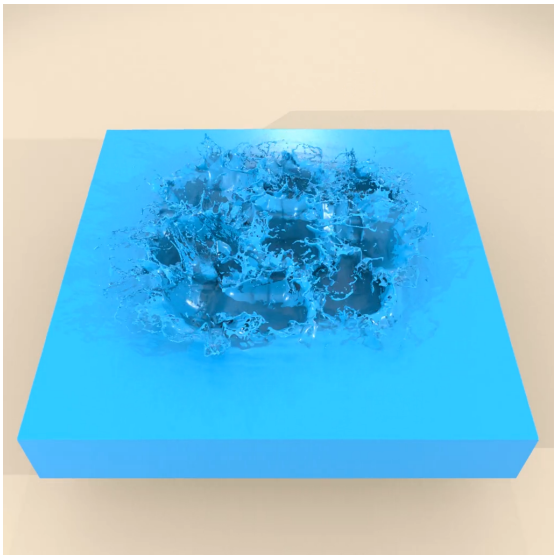
We also investigated the effect of changing the size of the voxel layer at the free surface and the size of the affine tiles (see Figure 4.15). Reducing the voxel layer the free surface from a 9-voxel thick band to only 3 voxels thick offers a moderate performance improvement of 1.2×, however, the thin band resulted in grid artifacts (see Figure 4.15d). Instead, maintaining a 9-voxel thick band and doubling the tile size to 32^3 offers a 5.6× performance improvement over the regular grid and 1.5× over the 16^3 tile-sized affine method with only slightly damped motion and no surface artifacts.



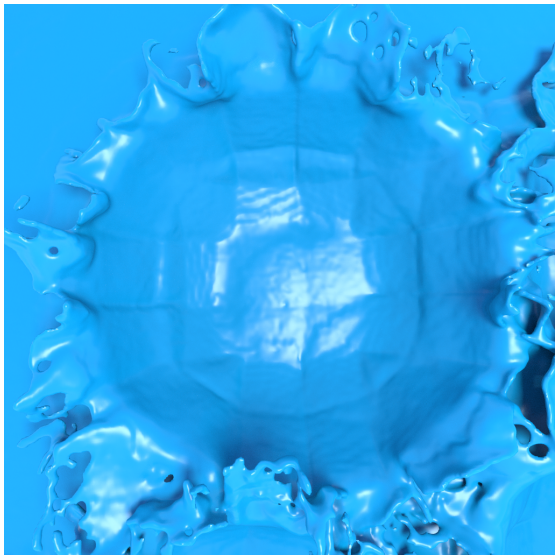
(a) 9-voxel surface layer. 16^3 tile size.



(b) 9-voxel surface layer. 32^3 tile size.



(c) 3-voxel surface layer. 16^3 tile size.



(d) Grid artifacts due to a thin voxel layer.

Figure 4.15: Our baseline configuration (a) employs a 9-voxel thick layer at the free surface and an affine region tile size of 16^3. Increasing the tile size (b) to 32^2 offers a performance improvement of 1.5×, whereas reducing the free-surface voxel layer to 3-voxels instead (c) only offers a performance improvement of 1.2× and results in grid artifacts at the surface (d).

Chapter 5

Adaptive Viscosity

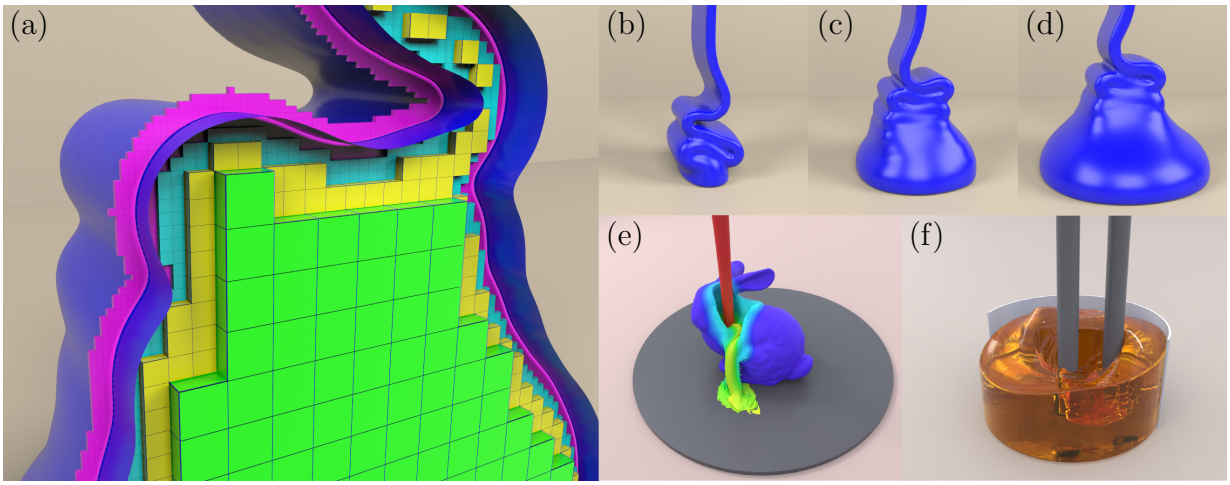


Figure 5.1: Our adaptive viscosity discretization is constructed on a graded octree structure (a), and achieves speed-up factors for the linear solve ranging from 3.8× to 9.4× compared to the regular grid approach. Our method supports rotational effects observed with a buckling sheet of viscous liquid (b-d), spatially varying viscosity coefficients (e), and kinematic objects (f).

In §3.3, we introduced the variational method of Batty and Bridson (2008) to solve for viscous forces,

$$\underset{\boldsymbol{u}^{n+1}}{\operatorname{argmin}} \int_{\Omega} \left(\frac{\rho}{2\Delta t} \|\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n}\|_2^2 + \mu \left\| \frac{\nabla \boldsymbol{u}^{n+1} + (\nabla \boldsymbol{u}^{n+1})^{\mathsf{T}}}{2} \right\|_F^2 \right) dV, \tag{5.1}$$

where the discrete linear system of its solution is SPD by construction:

$$(\mathbf{P}\mathbf{W}_u + \Delta t \mathbf{D}^\mathsf{T}\mathbf{K}\mathbf{W}_\tau \mathbf{M}\mathbf{D})\mathbf{u}^{n+1} = \mathbf{P}\mathbf{W}_u\mathbf{u}^n. \tag{5.2}$$

Most importantly, the SPD property holds independent of the actual discretization, allowing us to define the deformation rate operator $\mathbf{D}$ and volume weights $\mathbf{W}_u$ and $\mathbf{W}_\tau$ using our proposed adaptive framework.

Solving for viscosity is often substantially slower than solving for pressure, in part, because the viscosity linear system is significantly larger due to the three velocity vector components that are necessarily coupled by the free surface boundary condition. Therefore, as our results will show, adaptively coarsening the simulation grid interior to liquid offers a significant reduction in both the velocity DOFs of the linear system as well as simulation time. Because viscous forces dampen the liquid motion in the interior, our coarse approximation still closely matches equivalent regular grid simulations.

The variational finite difference framework was proposed for pressure and solid-fluid coupling problems (Batty et al., 2007), and has since been applied to viscosity, stream functions, granular flow, and more (Batty and Bridson, 2008; Narain et al., 2010; Larionov et al., 2017; Ando et al., 2015a,b). The typical advantage of this approach is its simpler handling of difficult irregular boundary conditions on regular Cartesian grids, while reducing to standard staggered finite differences on the interior and preserving symmetric positive-definiteness. Rather than only using it to handle boundary conditions, however, we propose to further apply this variational finite difference perspective to support octree-based adaptivity. Doing so hinges on discretizing the integrals of the variational form in the presence of transitions between grid levels, which in practice involves two key changes near T-junctions: first, selecting appropriate variable locations and control volumes to integrate over, and second, designing finite difference operators to approximate derivative terms. While the variational form ensures symmetry and hints that a valid numerical scheme should exist, these discretization choices must nevertheless be approached carefully if one wishes to achieve consistent, accurate solutions and compact stencils that will be efficient in three dimensions.

To form the discrete viscous energy (3.20), the only derivative is the deformation rate operator $\mathbf{D}$, which implies that we must modify the finite difference velocity gradient (or stress) stencils. Since symmetry follows from the variational form, the corresponding discrete tensor divergence operator arises implicitly as the (scaled) transpose of $\mathbf{D}$. Similarly, the final linear system for octree viscosity remains SPD with no additional effort. By contrast, typical direct staggered finite difference/volume approaches on octrees almost invariably lead to asymmetry even for simple Poisson problems, as discussed by previous authors

(Losasso et al., 2004; Batty, 2017). Prior derivations of *symmetric* octree schemes for Poisson problems relied on a combination of subtle intuition and trial and error (Losasso et al., 2004, 2006; Aanjaneya et al., 2017). Our more systematic adaptive variational finite difference framework instead preserves symmetry naturally and, with judicious discretization choices, yields accurate, efficient solutions on the challenging variable viscosity PDE.

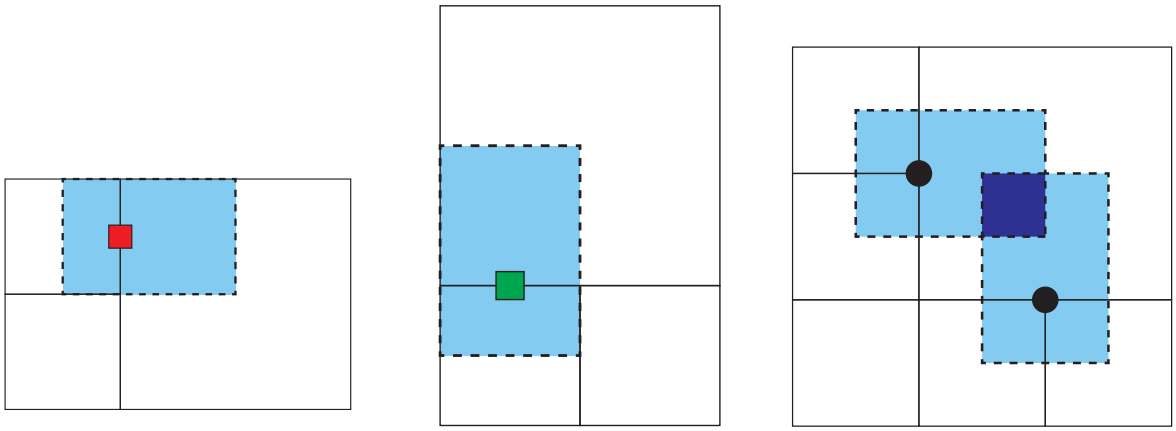
For simplicity of implementation, similar to prior work (Ferstl et al., 2014; Aanjaneya et al., 2017), we assume that the octree is *graded*, i.e., cells sharing a face differ by no more than one level of resolution. (Note that this does not preclude cells that share only a node or edge from differing by more than one level, which allows for slightly more rapid coarsening.) We further assume that the free surface occurs only within the finest grid resolution and that resolution changes occur only in the interior of the liquid; this avoids any potential interactions between T-junctions and boundary conditions. In regions of the octree without resolution changes, the variable locations, control volumes, and stencil structures all follow the template of the regular grid method described in §3.3, adjusting for the cell size in the corresponding level of the octree. We now adapt this regular grid template to discretize (3.21) near T-junctions, beginning with the simpler two dimensional case.

5.1 Two Dimensions

5.1.1 Variable Locations and Control Volumes

Our 2D quadtree layout is designed to follow naturally from the 2D regular grid setting presented in Figure 3.7. Similar to Losasso et al. (2004; 2006), we place velocity samples on each fine grid face at level transitions (see Figure 5.2). Similarly, we place stresses at cell centers and nodes, including the new T-junction ("dangling") nodes arising at level transitions.

To ensure the velocity and stress control volumes for each variable separately cover the entire integrable fluid volume, we stretch the control volumes into the adjacent coarse-grid cells at T-junctions (see Figures 5.2 and 5.3). Unlike the regular grid case in which absolute cell volumes can be completely factored out such that volume fractions suffice, all quadtree control volumes must be appropriately scaled to reflect the relative grid cell size at the corresponding level of the quadtree. The rectangular control volumes shown in Figure 5.2c can partially overlap in one case at a T-junction. We tested a correction that modifies the control volume shapes to avoid this double-counting, but it had no discernible effect



(a) (b) (c)

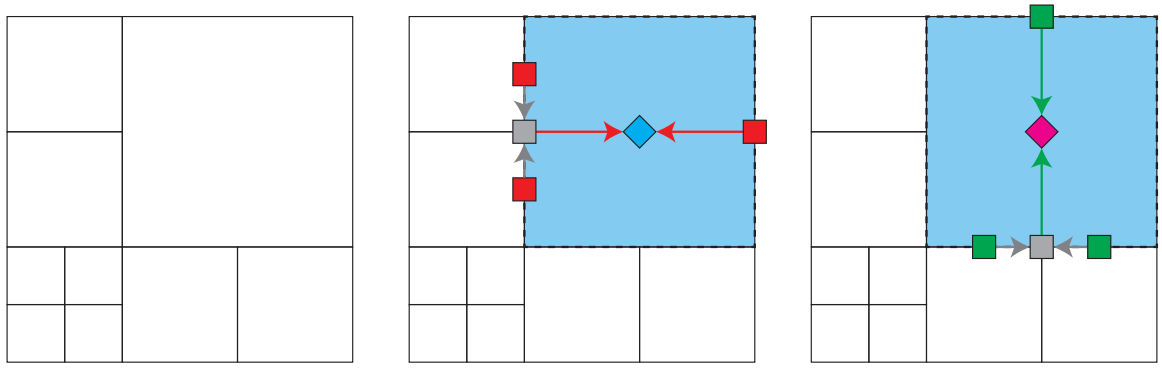
Figure 5.2: Quadtree variable locations for velocities: (a) u, and (b) v, with corresponding control volumes shaded in *blue*. At T-junctions (c) the control volumes partially overlap (*dark blue*). Our investigation determined that this double-counting had no effect on visual results or convergence rates.

on either the observed convergence rate or the visual results, so we prefer the rectangular volumes for simplicity.

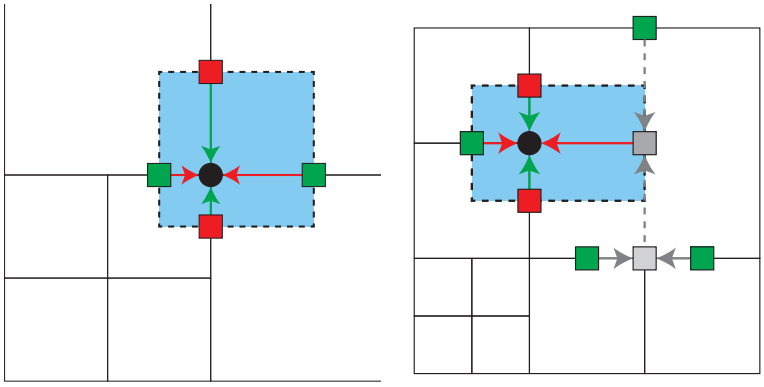
5.1.2 Finite Difference Stencils

To compute stresses at level transitions, we desire adaptive grid finite difference velocity gradient stencils that mimic the regular grid stencils of Figure 3.7; once again, we need not explicitly discretize the tensor divergence operator, as it arises implicitly through symmetry. We will consider cell-centered and node-centered stress stencils in turn.

Since only fine velocity samples are present on transition faces, coarse cell-centered stresses (τ_{xx}, τ_{yy}) incident on T-junctions will not have coarse velocity samples that align with their regular finite difference stencil. As shown in Figures 5.3b and 5.3c, we address this by creating a coarse "ghost" velocity sample on the T-junction, which can be used in the regular stencil. This ghost sample does not exist as a real degree of freedom, but is simply an interpolated value constructed as the average of the two fine velocity samples. The remaining cross-derivative node-centered stresses (τ_{xy}) at transitions are of two types: *irregular* junctions, where different resolutions meet diagonally (Figure 5.3d), and *T-junctions* (Figure 5.3e).



(a) Example layout. (b) τ_{xx} (c) τ_{yy}



(d) τ_{xy}, irregular junction. (e) τ_{xy}, T-junction.

Figure 5.3: Quadtree stress placement, stencils, and control volumes in two spatial dimensions. Gray squares indicate ghost samples constructed by averaging.

Irregular junction stencil The irregular junction stencil in Figure 5.3d is similar to the regular grid case (Figure 3.7e), except that the velocity samples are no longer evenly spaced. We simply adjust the denominator of the finite difference estimate to reflect the distance between the sample points. While no longer a true centered difference, this does not pose an issue: our linear system remains symmetric, and the 2D numerical results in §5.5.3 confirm the convergence of our discretization.

T-junction stencil The T-junction stresses encounter a problem similar to the cell-centered stresses: the finite difference stencil required to compute the $\frac{\partial v}{\partial x}$ component in Figure 5.3e does not find a velocity sample when reaching out into the neighboring coarse cell. Once again, we create a ghost velocity sample by averaging the v-velocity faces of the coarse cell. However, a further wrinkle can arise if either one or both of the cells above and below the coarse cell are also subdivided. Figure 5.3e shows the case of one coarse face and two fine faces in the v-velocity direction. In such cases, we average the velocity values at the two fine faces to create an *intermediate* ghost velocity sample, and then average this ghost sample with the opposing coarse velocity sample to create a final ghost sample that completes the stencil.

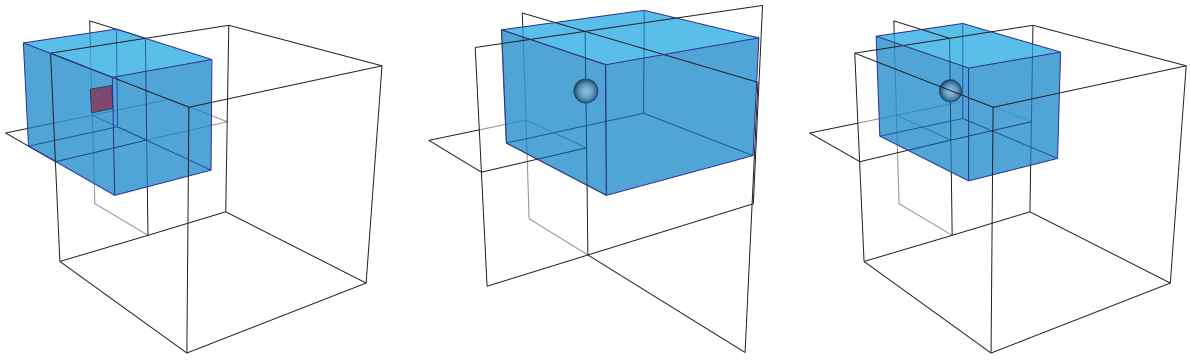
Discussion Because only a discrete velocity gradient $\mathbf{D}$ is required, our symmetry-preserving method possesses an attractive degree of conceptual simplicity. By contrast, Gerya et al. (2013) applied direct finite difference constructions separately to the vector gradient and tensor divergence operators on 2D quadtrees and arrived at a variety of possible viscosity discretizations, all of which exhibit asymmetry.

5.2 Three Dimensions — Basic Method

We now describe the essentials of our new discretization in three dimensions (i.e., on octrees), deferring an additional key accuracy enhancement to §5.3. Despite the apparent complexity of the task, we will show that the necessary stencil set reduces to a few classes of 3D configurations, which are unique up to appropriate rotations and reflections.

5.2.1 Variable Locations and Control Volumes

Similar to the quadtree case, velocity samples are placed at all fine faces surrounding T-junctions (Figure 5.4a). Stresses are placed at all cell centers and (fine) edge midpoints



(a) u (b) τ_{xz}, irregular junction. (c) τ_{xz}, T-junction.

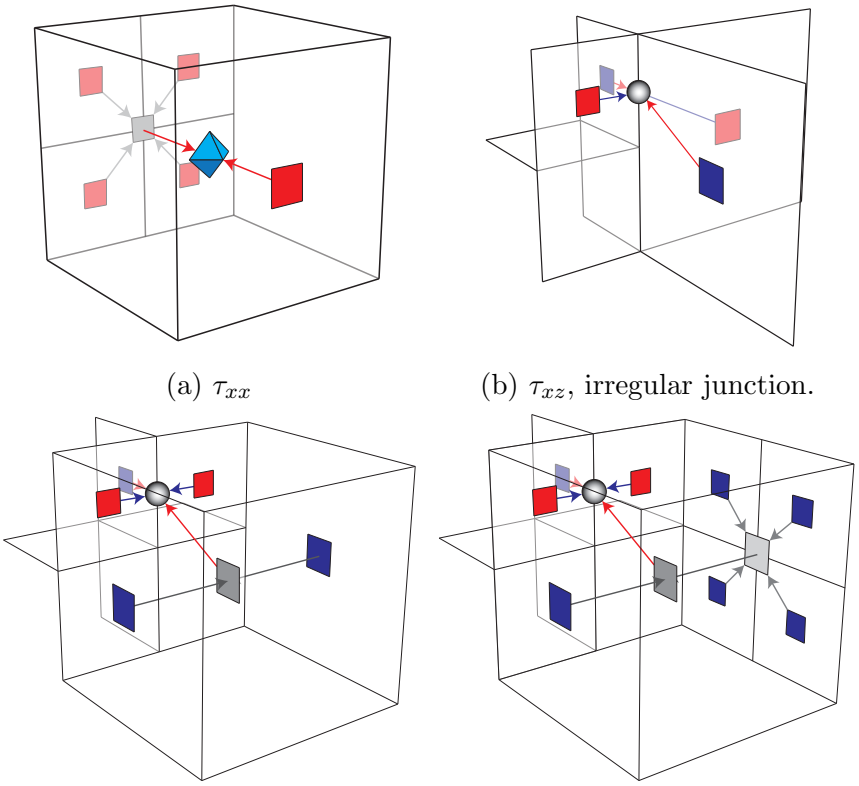
Figure 5.4: Octree velocity samples, stress placement, and control volumes.

(Figures 5.4b and 5.4c); this creates four *edge-based* stresses per T-junction in 3D, as compared to the single nodal stress per T-junction in 2D. These figures also illustrate how the rectangular cuboid control volumes for fine cell samples are expanded into the adjacent coarse cells. As in 2D, these control volumes overlap slightly in some cases, but modifying the control volumes to correct for this did not affect the observed motion or convergence rate.

5.2.2 Finite Difference Stencils

In constructing our 3D velocity gradient stencils, we aim to prioritize simplicity and compactness for the sake of ease of implementation and computational efficiency. Our stencil for the cell-centered stress τ_{xx} at a level transition is shown in Figure 5.5a. This stencil requires creating ghost velocity samples at coarse velocity positions on T-junctions by averaging the four inset fine velocity samples. For cross-derivative edge-centered stresses, τ_{xz}, there are two basic scenarios, with similarities to their 2D counterparts: irregular junction stresses, on edges that are shared *diagonally* between coarse and fine cells (Figure 5.5b), and T-junction stresses, for edges that lie on a subdivided coarse face (Figure 5.5c).

Irregular junction stencil Figure 5.5b shows the stress stencil for an irregular junction. In contrast to the 2D case, the stress sample does *not* lie in the same plane as the necessary velocity samples, leading to sloped velocity gradient estimates. We revisit this issue in §5.3, but for now we note that this is consistent with the sloped gradients for pressure adopted by Losasso et al. (2004) and that it does converge in practice. This is the only irregular



(a) τ_{xx} (b) τ_{xz}, irregular junction.

(c) τ_{xz}, T-junction. (d) τ_{xz}, T-junction, alternate case.

Figure 5.5: Octree stress stencils for cell-centered and edge-centered locations. Gray squares indicate ghost velocity samples constructed by averaging.

junction case to consider; although face-grading allows cells sharing an edge diagonally to differ by two levels, the faces adjacent to the edge can only differ by one, which is covered by our stencil.

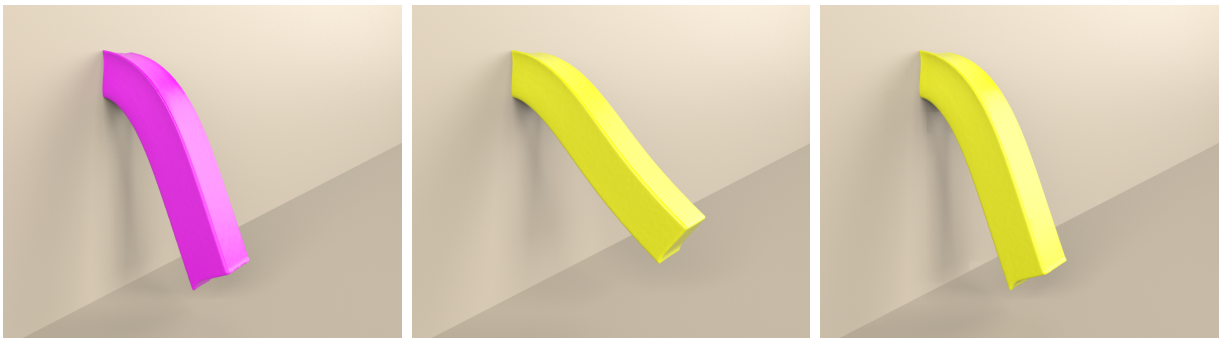
T-junction stencil Figure 5.5c shows the τ_{xz} stress stencil on a T-junction face; it likewise has one sample out of alignment so we treat it with a similar sloped stencil. However, there is also no coarse velocity sample at the center of the coarse cell, as required for the $\frac{\partial w}{\partial x}$ finite difference stencil. Therefore, we create a ghost velocity sample by averaging the two opposing w-velocity faces of the coarse cell. In Figure 5.5d, we illustrate a slight variation in which one of the two coarse faces is also subdivided. In this case, we create another intermediate ghost velocity sample at the coarse face, which is subsequently averaged with the other coarse w-velocity sample (which may itself be a ghost sample as well) to generate the final cell-centered ghost sample. This completes our 3D stencil set.

Losasso et al. (2004) discussed a few choices for the distance value used for the denominator of their sloped gradient estimates, and observed that they are all equally effective; we used the in-plane distance between the sample points. Despite these sloped gradient estimates, our 3D numerical results exhibit approximately first order convergence under grid refinement (§5.5.3).

5.3 Three Dimensions — Enhanced Gradients

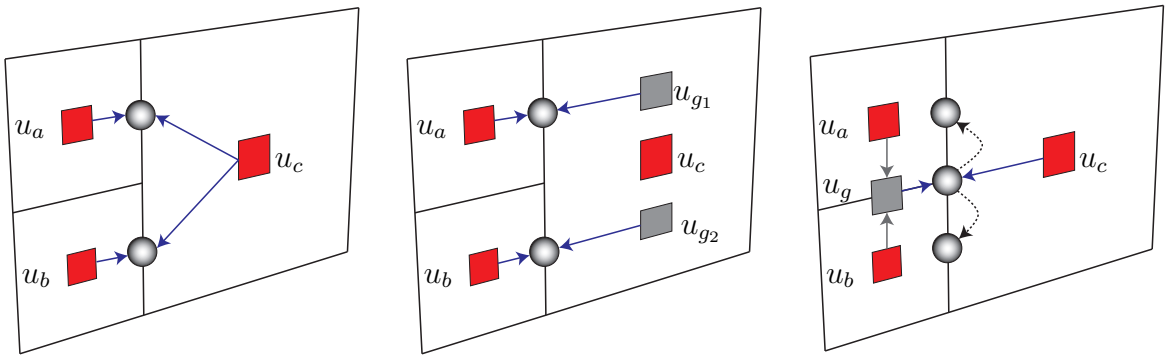
Using the method described so far we observed both numerical convergence and plausible viscous flows; however, we can improve the discretization further still. Close inspection of the fluid motion shows artificial damping of bending and rotation compared to regular grid simulations (see Figures 5.6a vs. 5.6b), and we traced the source of this issue back to the sloped gradient estimates. Prior authors studying adaptive discretizations for the Poisson problem also observed that orthogonality of *pressure* gradient stencils with respect to grid faces is critical to accuracy, especially in hydrostatic cases (Losasso et al., 2006; Batty et al., 2010; Aanjaneya et al., 2017). These observations motivate our enhanced orthogonality-preserving construction for discrete *velocity* gradients.

Sloped gradients occur for stress variables on the midpoint of fine edges at resolution transitions, both in the irregular and T-junction cases (i.e., Figures 5.5b-5.5d). A 2D cross-section is shown in Figure 5.7a. Letting Δ_L be the large cell width, Δ_s be the small cell width, and $\Delta = (\Delta_L + \Delta_s)/2$, then our sloped estimate for the top vertical edge is $\frac{\partial u}{\partial z} \approx (u_c - u_a)/\Delta$.



(a) Reference. (b) Sloped gradients. (c) Enhanced gradients.

Figure 5.6: The same frame of animation from three discretizations of a horizontal viscous beam released under gravity. (a) A regular grid reference simulation. (b) Our *basic* octree discretization with a two-level coarsened interior shows artificial stiffness due to sloped gradients. (c) Our *enhanced* octree discretization with corrected gradients matches the motion and detail of the reference (a), but its linear solve is 4.2× faster than that of the regular grid discretization.



(a) Basic sloped. (b) Linear ghosts. (c) Enhanced (ours).

Figure 5.7: Possible velocity gradient stencils at irregular junctions and T-junctions, shown as 2D slices of the 3D geometry. (a) Our basic method (§5.2) uses a low-quality sloped estimate. (b) One improved option would be to construct axis-aligned fine ghost points; however, the necessary additional linear interpolations (not shown) would involve several more cases and neighbor cell data. (c) In our proposed enhanced method (§5.3), fine face components are averaged together to create a coarse ghost; an axis-aligned estimate is then constructed at the node and used for *both* fine edges.

A direct approach to form the desired axis-aligned gradients would be to apply additional (and in some cases nested) linear interpolations to yield fine ghost values u_{g_1} and u_{g_2} directly opposite each existing fine velocity sample (Figure 5.7b). Although this would provide an accurate estimate $(u_{g_1} - u_a)/\Delta$ for the top edge and symmetry would be preserved by construction, forming u_{g_1} requires complicated stencils involving additional neighboring velocities and several geometric cases depending on the local refinement pattern. We offer a simpler and more compact solution.

We average the two fine face velocity components u_a and u_b together to create a coarse ghost velocity $u_g = (u_a + u_b)/2$ at the midpoint between them (Figure 5.7c); this enables a standard finite difference gradient estimate $\frac{\partial u}{\partial z} \approx (u_c - u_g)/\Delta$ between the existing coarse velocity u_c and the new ghost u_g. Because they lie in the same plane, this estimate is properly axis-aligned and measures (only) the correct component of the velocity gradient, unlike the sloped estimate. The stencil is also nearly as compact as the sloped estimate, requiring just a single additional velocity sample.

The velocity gradient components assigned to the two fine edges in this way share the same axis-aligned gradient estimate from the node. This approach is therefore conceptually similar to the method that Losasso et al. (2006) adopted to replace their earlier sloped pressure gradients, i.e., they construct a single axis-aligned gradient estimate at the center of the T-junction face, and assign it to all four surrounding fine faces. (However, it differs in that we construct orthogonal *edge-based velocity* gradients, whereas Losasso et al. construct *face-based pressure* gradients.)

To implement this change, we modify the $\mathbf{D}$ operator accordingly and our method again guarantees symmetry by construction. This small but vital enhancement effectively eliminates spurious discretization-dependent stiffness, as shown in Figures 5.6c and 5.15, while the spatial convergence rate improves to second order in the L_1 norm (Figure 5.8). Applications in computer graphics are often balancing between visual realism and computational efficiency. This a demonstrable example of a quantitative accuracy improvement in our numerical method directly improving the visual quality of the simulation and therefore a meaningful improvement in the computer graphics setting.

5.4 Implementation and Applications

We integrated our proposed octree-based viscosity solver into two completely independent liquid simulators. To demonstrate that our method can significantly accelerate existing regular grid simulation pipelines, we implemented it as a plugin for Houdini (SideFX, 2021);

it acts as a drop-in replacement for Houdini's viscosity step, leaving the rest of its regular grid simulator untouched. To demonstrate that our method can alternatively be used to add viscous effects to existing inviscid, purely octree-based liquid simulators, we incorporated it into the method of Aanjaneya et al. (2017).

In both implementations, we follow Setaluri et al. (2014) in constructing our octree as an aligned pyramid of regular grids instead of a conventional pointer-based tree. This allows querying neighboring elements (cells, faces, edges, nodes) at differing refinement levels with simple grid index offsets and scaling, eschewing pointer-based traversals. This structure also provides localized memory accesses within a single level. We use sparse grid structures to restrict memory allocations to active regions of each level. For our Houdini implementation, we use its internal compressed-tile design as a sparse grid structure, and for our implementation of the Aanjaneya et al. (2017) method, we use SPGrid.

Both implementations also employ the standard optimization of constructing the final system (3.21) in a single pass, rather than forming and multiplying the individual sparse matrices that compose it. A single pass is more involved for the adaptive viscosity system than for regular grids, since the transposition $\mathbf{D}^\mathsf{T}$ requires "reversing" the velocity gradient stencils whose transposes implicitly yield the discrete tensor divergence. Nevertheless, we found it to be much faster than the matrix multiplication approach.

5.4.1 Accelerating Regular Grid Simulators

Viscosity is a significant bottleneck in regular grid simulators: for some scenes using Houdini the viscosity step was up to two orders of magnitude slower than pressure projection. Fortunately, viscous flows possess smooth interior velocity fields, making this an ideal setting for spatial adaptivity without noticeable quality loss. This motivated us to accelerate Houdini's regular grid fluid solver by replacing its viscosity step with our octree-based version. To do so, we sandwich our method between a pair of interpolation operators that transfer velocities between the regular and octree grids. The smoothness of the interior flow also enables a relatively simple refinement strategy: we set a band of finest resolution regular grid cells near interfaces and boundaries, and coarsen into the interior as rapidly as possible while respecting the face-grading rule.

5.4.2 Interpolation

We align the finest octree level with the source regular grid, allowing finest level velocity samples to be directly copied into the octree. We construct interior coarsened velocities

in the octree by recursive weighted averaging of the next finer level samples, using the restriction operator proposed by Zhu et al. in the context of Multigrid elasticity ((Zhu et al., 2010), §6). To transfer the octree velocity data back to the regular grid, we adopt the low-dissipation octree velocity interpolant of Setaluri et al. (2014). We confirmed that interpolating to and from the octree induces a negligible amount of additional artificial dissipation for viscous flows by adding a round-trip grid-to-octree-to-grid transfer(-only) step to a regular grid solver, and comparing it against a standard regular grid solver on a beam bending scenario; any differences were visually indiscernible. The close match between our octree and regular grid results (e.g., Figure 5.6) further confirms this fact.

5.4.3 Additional Details

We use Houdini's multithreaded implementation of Jacobi-preconditioned Conjugate Gradients to solve the linear system in equation (3.21). Likewise, we used Houdini's multithreaded library to iterate over its sparse grid structure in parallel. Lastly, again following Houdini, we exploited parallelism throughout our implementation, including during octree adaptation, both interpolation steps, and linear system construction.

5.4.4 Laplacian model

The speed of the simpler Laplacian model of viscosity (Carlson et al., 2002; Fält and Roble, 2003) may be desirable in some cases, despite the clear limitations illustrated in Figure 3.6. Hong and Kim applied this model to octrees using Losasso's discrete octree Laplacian (Losasso et al., 2004), but this requires interpolating staggered face velocities to cell centres and back to faces. Our methodology enables a purely face-based Laplacian viscosity discretization by applying it to the simplified smooth energy

$$\int_{\Omega} \left(\frac{\rho}{\Delta t} \|\mathbf{u}^{n+1} - \mathbf{u}^{n}\|_2^2 + \mu \left\|\nabla \mathbf{u}^{n+1}\right\|_F^2 \right) dV, \tag{5.3}$$

which can be compared with (3.19). This form leads to one SPD linear system per velocity component.

5.4.5 Adding Viscosity to an Octree Liquid Simulator

We also incorporated our octree viscosity solver as a plugin into the fully adaptive *inviscid* simulator of Aanjaneya et al. (2017). This required the minor modification that velocity

values be interpolated from the slanted faces of the power diagram to the regular octree faces, and then interpolated back again after the viscous update.

5.5 Convergence Studies

Even though our variational finite difference approach is SPD by construction and the minimum of the energy functional (3.19) is the solution to the viscosity PDE (Batty and Bridson, 2008), we have only discretized the deformation rate operator $\mathbf{D}$. It is not intuitively obvious that $\mathbf{D}^\mathsf{T}$ is a consistent discretization of the tensor divergence operator and therefore is it essential that we study the convergence properties of our method. To quantitatively evaluate the spatial accuracy of our octree method, we constructed analytical tests representing a single time step of the viscosity PDE (3.18) on a closed box domain in 2D and 3D for a particular vector field and spatially varying viscosity function. Beginning from an irregular initial refinement pattern, we recursively subdivided every cell, evaluating the velocity error at each refinement step. The results are plotted in Figure 5.8; the specific test cases, their derivations and data for the plots are given below.

Approximately first order convergence is consistently observed in the L_∞ norm, with our enhanced approach exhibiting lower error than the sloped approach in 3D. However, under the L_1 norm the enhanced approach achieves full second order accuracy, whereas the sloped approach remains first order. Note that first order accurate velocity solutions are consistent with previous staggered octree schemes for pressure (Losasso et al., 2006; Aanjaneya et al., 2017). While these methods offer second order accuracy in the *pressure*, the pressure gradient and velocity field remain first order.

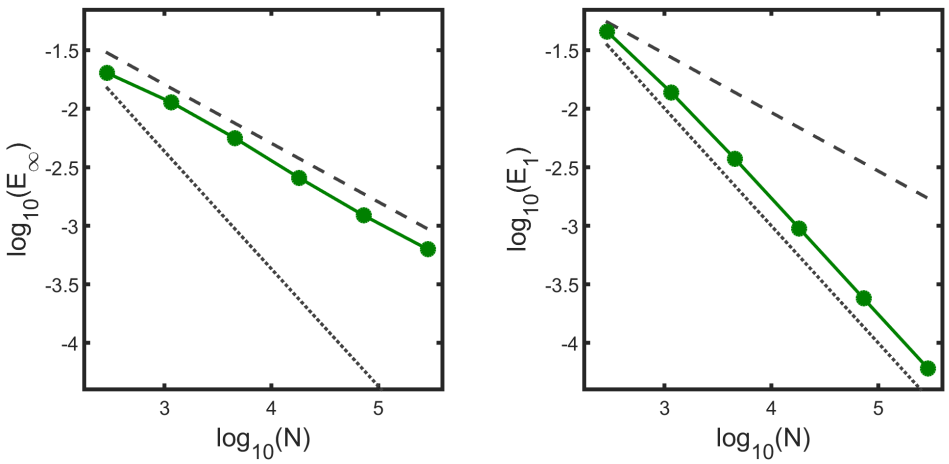
5.5.1 2D Test Case

Consider the square domain $[0, \pi]^2$. Our methodology will be to choose an end-of-step velocity field at time $t = \Delta t$ that satisfies the no-slip boundary condition $\boldsymbol{u} = \mathbf{0}$ on the domain boundaries. There are many valid options; to simplify boundaries, we use a product of sinusoids as the final velocity field,

$$u(x, y, t = \Delta t) = \sin(x)\sin(y),$$
$$v(x, y, t = \Delta t) = \sin(x)\sin(y).$$

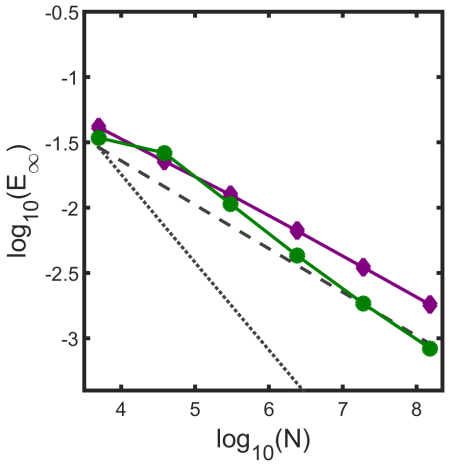
Next, we choose a spatially varying function to determine the viscosity coefficient,

$$\mu(x) = \frac{x}{\pi} + \frac{1}{2}.$$

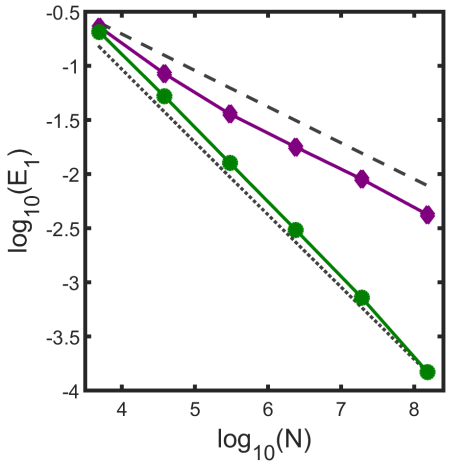


(a) 2D L_∞ error

(b) 2D L_1 error



(c) 3D L_∞ error



(d) 3D L_1 error

Figure 5.8: Log-log plots of L_1 and L_∞ velocity error (labelled E_1 and E_∞, respectively) vs. cell count N under refinement in two (top) and three (bottom) dimensions. Dashed and dotted guide lines indicate ideal first and second order slopes. In 3D, comparing the sloped (purple diamonds) and enhanced (green circles) gradient discretization error behavior confirms that the enhanced method experiences significantly improved convergence.

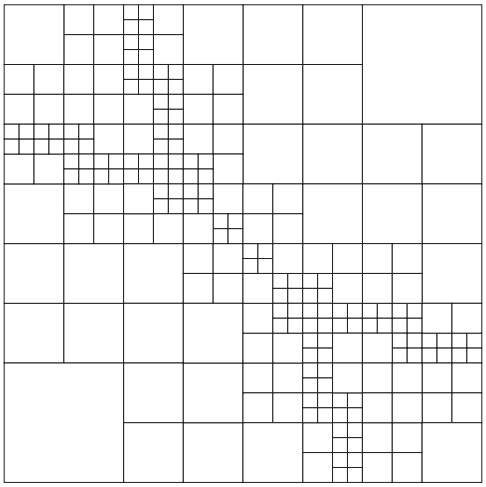


Figure 5.9: The initial tree refinement pattern in 2D.

Once again, many options are possible, provided the function remains non-negative over the domain. Finally, using analytical differentiation, we can take one step *backwards* in time to $t = 0$ following the viscosity equations, (semi-)discretized in time as in our method,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \Delta t \frac{1}{\rho} \nabla \cdot \mu(\nabla \boldsymbol{u}^n + (\nabla \boldsymbol{u}^n)^\mathsf{T}). \tag{5.4}$$

This yields the initial conditions for our problem:

$$\begin{aligned} u(x, y, t = 0) =& \sin(x)\sin(y) - \Delta t \left(\frac{2}{\pi} \cos(x)\sin(y) + \right. \\ & \left. \left(\frac{x}{\pi} + \frac{1}{2} \right) (\cos(x + y) - 2\sin(x)\sin(y)) \right), \\ v(x, y, t = 0) =& \sin(x)\sin(y) - \Delta t \left(\left(\frac{x}{\pi} + \frac{1}{2} \right) (\cos(x)\cos(y) - 3\sin(x)\sin(y)) \right. \\ & \left. + \frac{1}{\pi} \sin(x + y) \right). \end{aligned}$$

We used a time step of $\Delta t = 1$. The initial refinement pattern of the 2D domain is shown in Figure 5.9.

5.5.2 3D Test Case

Our 3D problem follows the same general approach, but over the domain $[0, \pi]^3$. The final velocity field is

$$\begin{aligned} u(x, y, z, t = \Delta t) &= \sin(x)\sin(y)\sin(z), \\ v(x, y, z, t = \Delta t) &= \sin(x)\sin(y)\sin(z), \\ w(x, y, z, t = \Delta t) &= \sin(x)\sin(y)\sin(z), \end{aligned}$$

and the associated viscosity function is

$$\mu(x, y) = \frac{x}{\pi} + y + 1.$$

The resulting initial velocity field is given by:

$$\begin{aligned} u(x, y, z, t = 0) = &\sin(x)\sin(y)\sin(z)(1 + 2\Delta t\mu(x, y)) - \\ &\Delta t(\sin(z)(\cos(x)\sin(y) + \sin(x)\cos(y) + \\ &\frac{2}{\pi}\cos(x)\sin(y)) + \mu(x, y)(\cos(x + y)\sin(z) + \cos(x + z)\sin(y))), \\ v(x, y, z, t = 0) = &\sin(x)\sin(y)\sin(z)(1 + 2\Delta t\mu(x, y)) - \\ &\Delta t(2\sin(x)\cos(y)\sin(z) + \\ &\frac{1}{\pi}\sin(x + y)\sin(z) + \mu(x, y)(\cos(x + y)\sin(z) + \sin(x)\cos(y + z))) \\ w(x, y, z, t = 0) = &\sin(x)\sin(y)\sin(z)(1 + 2\Delta t\mu(x, y)) - \\ &\Delta t(\sin(x)(\cos(y)\sin(z) + \sin(y)\cos(z)) + \\ &\frac{1}{\pi}\sin(x + z)\sin(y) + \mu(x, y)(\cos(x + z)\sin(y) + \sin(x)\cos(y + z))). \end{aligned}$$

We used a time step of $\Delta t = 1$. The initial refinement pattern of the domain (Figure 5.10) is generated from a union of two implicit spheres placed at $(0, 0, 0)$ and (π, π, π), with radii of $\frac{\sqrt{3}}{2}\pi$. Grid cells at the finest level are activated if their cell centers are within a distance $\frac{\Delta x}{2}$ of the surface of the spheres. We then proceed to grade the octree using our face-grading criteria. We chose this refinement pattern because it exercises all the relevant octree stencils.

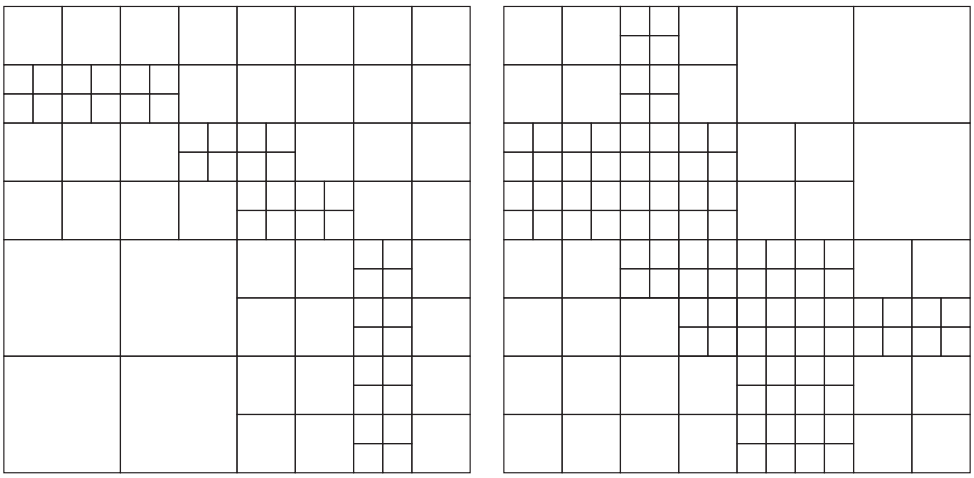


Figure 5.10: Two slices of the initial tree refinement pattern in 3D. Both slices are on a yz-plane, at $x = 0$ and $x = \frac{\pi}{2}$, respectively.

Table 5.1: Convergence of 2D discretization in L_∞ on a quadtree grid, exhibiting approximately first order behavior.

Grid	$\|u - u^h\|_\infty$	Order	$\|v - v^h\|_\infty$	Order
32^2	1.8929 E -2		2.0159 E -2	
64^2	1.1169 E -2	0.76	1.1377 E -2	0.83
128^2	5.6016 E -3	1.00	5.4032 E -3	1.07
256^2	2.5636 E -3	1.13	2.5145 E -3	1.10
512^2	1.2273 E -3	1.06	1.1995 E -3	1.07
1024^2	6.2867 E -4	0.97	6.2013 E -4	0.95

5.5.3 Detailed Numerical Results

Beginning from the initial refinement pattern, we recursively subdivided every cell, evaluating the L_∞ and L_1 velocity errors at each step. The 2D results are presented in Tables 5.1 and 5.2, and 3D results for sloped and enhanced gradients in Tables 5.3, 5.5, 5.4, and 5.6, respectively; the corresponding convergence plots are illustrated in Figure 5.8. The grid size listed in these tables corresponds to the finest virtual grid size of the tree structure (i.e. the effective grid resolution).

Table 5.2: Convergence of 2D discretization in L_1 on a quadtree grid, exhibiting approximately second order behavior.

Grid	$\|u - u^h\|_1$	Order	$\|v - v^h\|_1$	Order
32^2	4.2504 E -2		4.5429 E -2	
64^2	1.3526 E -2	1.65	1.3805 E -2	1.72
128^2	3.6910 E -3	1.87	3.7281 E -3	1.89
256^2	9.4728 E -4	1.96	9.5028 E -4	1.97
512^2	2.3910 E -4	1.99	2.3933 E -4	1.99
1024^2	6.0214 E -5	1.99	6.0514 E -5	1.98

Table 5.3: Convergence of 3D octree discretization in L_∞ with *sloped* gradients, exhibiting approximately first order behavior.

Grid	$\|u - u^h\|_\infty$	Order	$\|v - v^h\|_\infty$	Order	$\|w - w^h\|_\infty$	Order
16^3	4.0994 E-2		3.6799 E-2		4.1257 E-2	
32^3	2.2438 E-2	0.87	2.2182 E-2	0.73	2.2709 E-2	0.86
64^3	1.2464 E-2	0.85	1.2338 E-2	0.85	1.2538 E-2	0.86
128^3	6.6379 E-3	0.91	6.5528 E-3	0.91	6.6554 E-3	0.91
256^3	3.4793 E-3	0.93	3.4317 E-3	0.93	3.4818 E-3	0.93
512^3	1.8087 E-3	0.94	1.7859 E-3	0.94	1.8082 E-3	0.95

Table 5.4: Convergence of 3D octree discretization in L_∞ with *enhanced gradients*, exhibiting approximately first order behavior.

Grid	$\|u - u^h\|_\infty$	Order	$\|v - v^h\|_\infty$	Order	$\|w - w^h\|_\infty$	Order
16^3	3.4294 E-2		2.8760 E -2		3.4065 E -2	
32^3	2.5290 E-2	0.44	2.6265 E -2	0.13	2.4756 E -2	0.46
64^3	1.0322 E-2	1.30	1.0698 E -2	1.30	1.0301 E -2	1.27
128^3	4.1865 E-3	1.30	4.3085 E -3	1.31	4.2229 E -3	1.29
256^3	1.8039 E-3	1.21	1.8412 E -3	1.23	1.8227 E -3	1.21
512^3	8.2871 E-4	1.13	8.3664 E -4	1.14	8.3273 E -4	1.13

Table 5.5: Convergence of 3D octree discretization in L_1 with *sloped* gradients, exhibiting approximately first order behavior.

Grid	$\|u - u^h\|_1$	Order	$\|v - v^h\|_1$	Order	$\|w - w^h\|_1$	Order
16^3	2.2728 E -1		2.2728 E -1		2.2846 E -1	
32^3	8.3192 E -2	1.45	8.4643 E -2	1.43	8.3151 E -2	1.46
64^3	3.4742 E -2	1.26	3.5813 E -2	1.24	3.4676 E -2	1.26
128^3	1.7170 E -2	1.02	1.7768 E -2	1.01	1.7126 E -2	1.02
256^3	8.6849 E -3	0.98	8.9916 E -3	.98	8.6589 E -3	.98
512^3	4.1313 E -3	1.07	4.2207 E -3	1.09	4.1245 E -3	1.07

Table 5.6: Convergence of 3D octree discretization in L_1 with *enhanced gradients*, exhibiting approximately *second* order behavior.

Grid	$\|u - u^h\|_1$	Order	$\|v - v^h\|_1$	Order	$\|w - w^h\|_1$	Order
16^3	2.0364 E-1		2.0128 E -1		2.0504 E -1	
32^3	5.2047 E-2	1.97	5.1960 E -2	1.95	5.2332 E -2	1.97
64^3	1.2568 E-2	2.05	1.2622 E -2	2.04	1.2617 E -2	2.05
128^3	3.0287 E-3	2.05	3.0522 E -3	2.05	3.0355 E -3	2.06
256^3	7.1526 E-4	2.08	7.2298 E -4	2.08	7.1639 E -4	2.08
512^3	1.4614 E-4	2.29	1.4797 E-4	2.29	1.4630 E -4	2.29

Table 5.7: Timing breakdowns for our simulations on regular and octree grids. Only active voxels are included in the voxel counts.

Scene	Regular grid voxel count	Octree grid voxel count	Grid type	Linear solve	Solve speed-up	Viscosity total	Viscosity speed-up	Simulation total	Simulation speed-up
Viscous Buckling	137K (initial) - 2.7M (final)	76K (initial) - 587K (final)	Regular	3h47m		3h50m		4h32m	
			Octree	59m48s	3.8×	1h05m	3.5×	1h47m	2.5×
Viscous Beam	757K	227K	Regular	2h30m		2h31m		2h40m	
			Octree	35m34s	4.2×	37m20s	4×	46m35s	3.4×
Melting Bunny	2.2M (initial) - 3.2M (final)	422K (initial) - 1.3M (final)	Regular	50h05m		50h19m		52h43m	
			Octree	13h11m	3.8×	13h32m	3.7×	15h25m	3.4×
Letter Mixer	139K (initial) - 4.8M (final)	65K (initial) - 952K (final)	Regular	71h20m		72h59m		96h23m	
			Octree	12h56m	5.5×	15h27m	4.7×	39h02m	2.5×
Gooey Armadillo	4.8M	976K	Regular	26h20m		26h28m		27h55m	
			Octree	2h49m	9.4×	3h00m	8.8×	4h25m	6.3×
Bunny Drop	9.0M	1.1M	Regular	—		—		—	
			Octree	73h01m	N/A	76h32m	N/A	96h00m	N/A

5.6 Simulation Results

5.6.1 Timings

To evaluate the efficiency and effectiveness of our octree-based viscosity plugin for Houdini, we simulated a variety of viscous scenarios involving buckling, rotation, variable viscosity, and moving solids. The *Bunny Drop* (§5.6.7) and *Viscous Buckling* (§5.6.2) examples and the performance comparison in Figure 5.17 were simulated with a 32-core Ryzen 2990wx CPU; all remaining examples were simulated with a 16-core Xeon E5-2630 CPU. The Conjugate Gradients routine used double precision and a relative tolerance of 10^{-3} for all examples. We emphasize that only Houdini's viscosity step was replaced with an octree method; every other step remains on a regular grid using Houdini's standard implementation. Four-level octrees were used in all cases except where stated otherwise, since additional coarse levels yielded minimal benefit (see §5.6.5). To eliminate extraneous factors and ensure a consistent baseline, we implemented and used our own purely regular grid viscosity step (Batty and Bridson, 2008) for all comparisons. (Performance profiling showed that our regular grid implementation was approximately 2% slower than Houdini's on a representative scene; the simulation results were visually indistinguishable.)

Performance numbers for the octree and regular grid are presented in Table 5.7, with the octree linear solve being faster by factors ranging from 3.8× to 9.4×. As expected in comparing the *overall* time between the regular grid and octree implementations, the additional overhead of the octree slightly reduces its net benefit: the total viscosity solve, including octree adaptation, interpolation, and matrix construction, yields speed-up factors

of 3.5× to 8.8×. Although viscosity is just one component of the complete liquid simulator, *overall* speed-up factors ranged from 2.4× to 6.3×. In one case the simulation time dropped from more than a full day to just over four hours, which is a far more practical turn-around time. It should be noted that these speed-up factors are only available for scenes where the regular grid simulation succeeded; nevertheless, the octree enables simulations at even higher effective resolutions than are possible with the regular grid, as demonstrated by the *Bunny Drop* (§5.6.7) example.

5.6.2 Viscous Buckling

Figure 5.11 shows that our method can reproduce the familiar viscous buckling phenomenon. The octree and regular grid results match closely throughout the simulation despite the octree's lower computational cost, with only slight drifts out of phase. We investigated increasing the width of the fine grid cell layer at the liquid surface and found it offers a small perceptible improvement towards matching the regular grid example, but it comes with a significant computational cost. The viscosity steps for two-, three-, and four-voxel wide fine layer examples took 1h05m, 2h05m, and 2h25m with a DOF count in the final frame of 1.9M, 2.2M, and 2.7M, respectively.

This example also illustrates that the degree of timing improvement is inherently problem- and geometry-dependent, since coarsening becomes possible only with volumes of sufficient depth. Because the volume of the pile gradually increases, the benefit increases in correspondence. In the initial frame, the two-voxel fine layer octree reduced the active DOFs in the viscosity solve from 412K to 277K, while the last frame went from 8.2M velocity samples to 1.9M.

5.6.3 Melting Bunny (Variable Viscosity)

Next, we melt a highly viscous bunny by pouring hot liquid onto it, showcasing our variable viscosity support (Figure 5.12). We define the viscosity coefficient as a function of per-particle temperatures. We mimic heat diffusion at each time step by transferring temperature from the particles to the grid, applying a simple blurring pass over the grid-based temperature field, and finally transferring the updated temperature back to the particles (more intricate thermodynamics could be incorporated if desired, e.g., (Carlson et al., 2002; Stomakhin et al., 2014)).

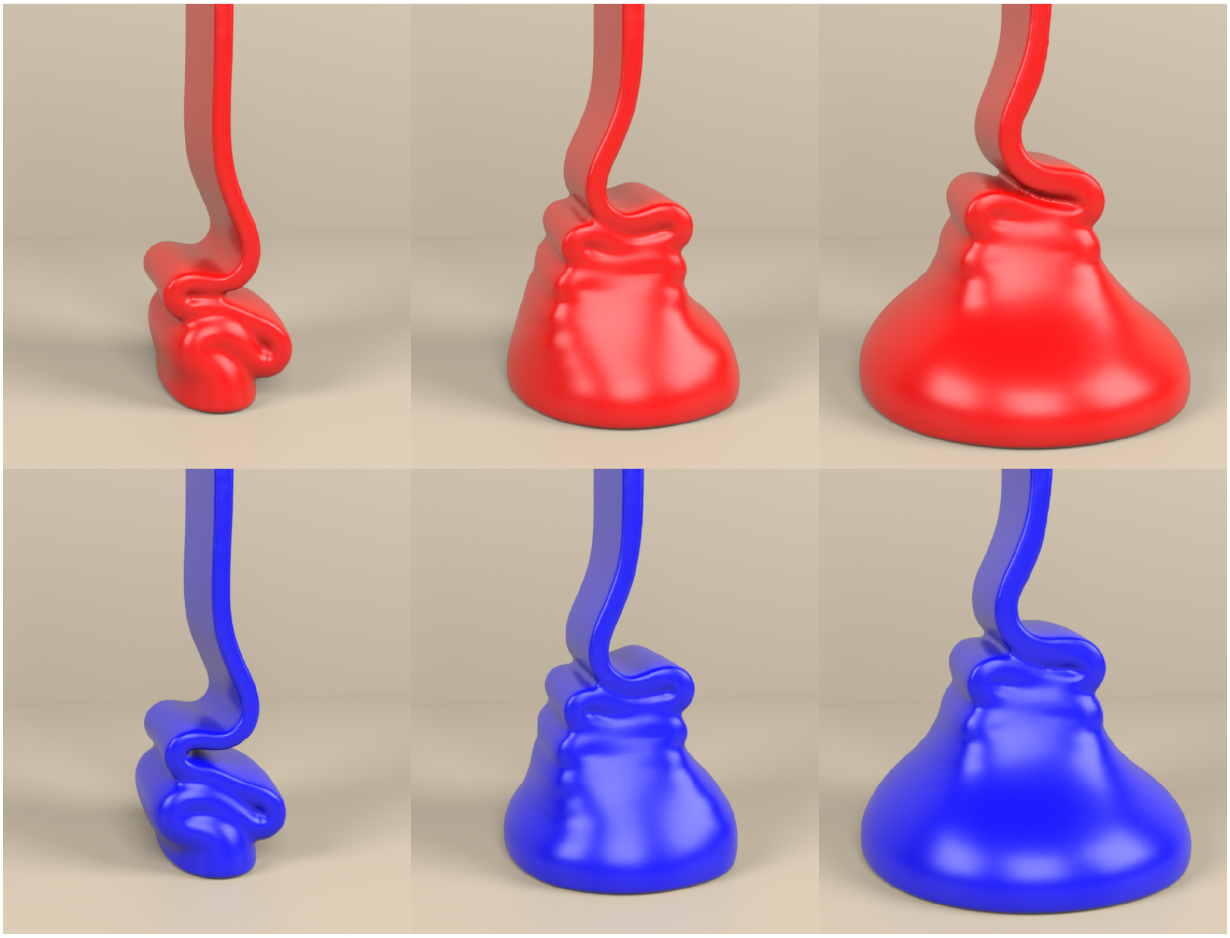


Figure 5.11: **Viscous Buckling**: A buckling viscous sheet exhibits qualitatively consistent motion using a regular grid (red) and our octree-based (blue) viscosity solver. The motion matches closely over a long period.

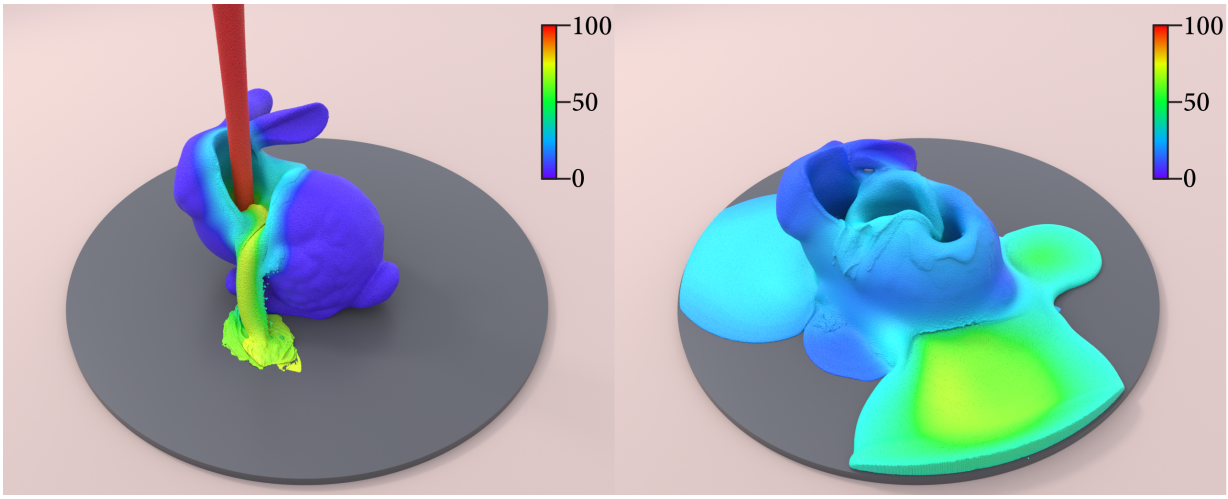


Figure 5.12: **Melting Bunny**: Hot liquid is poured onto a viscous bunny, melting holes into it. The viscosity is a function of the liquid temperature, visualized with a pseudocolor map. Linear solve speed-up factor: 3.8×.

5.6.4 Letter Mixer

Viscous letters are piled up and mixed together with scripted moving solids in Figure 5.13. A standard no-slip condition applied to the discrete viscosity equations forces the fluid velocity to match the solid, causing the viscous liquid to be dragged alongside moving solids.

This example also highlights conditions under which we see large benefits (see Figure 5.14). Since strong viscous forces are working hard to oppose gravity when all of the letters are fully stacked up, these earlier frames are approximately three times as costly for the regular grid (red) and twice as costly for the octree (blue) compared to when the letters have settled into the container. At that point, the velocity field is no longer changing dramatically between time steps so *warm-starting* the solver allows both regular grid and octree methods to reach convergence more quickly. Regardless, the octree-based simulation's total time is faster by a factor of 2.5×.

5.6.5 Viscous Beam

High frequency velocity modes are quickly damped out in viscous liquids, resulting in a smooth field with small variation across many grid cells. Because of this effect, we observe

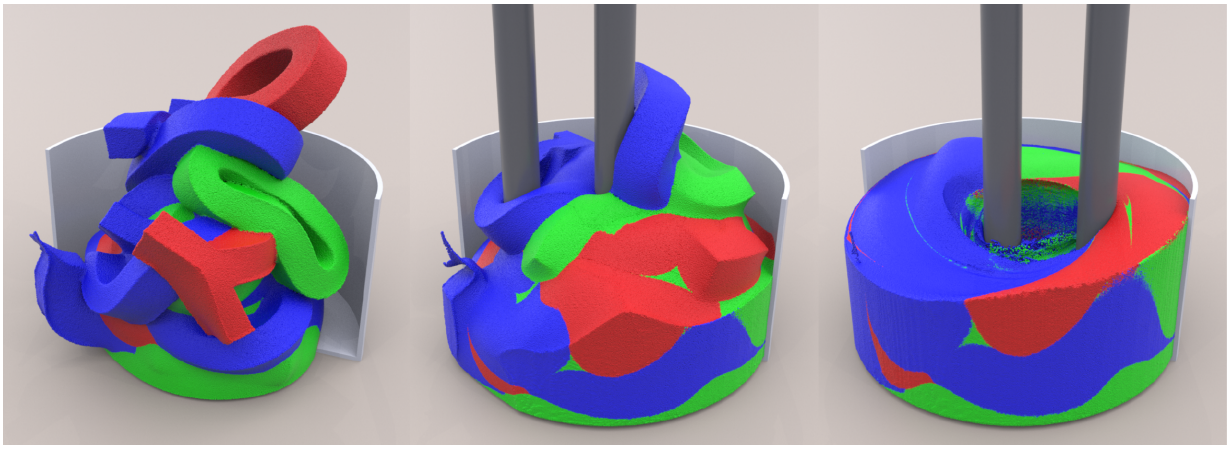


Figure 5.13: **Letter Mixer**: Viscous letters are piled into a container and stirred by scripted moving solids. Linear solve speed-up factor: 5.5×.

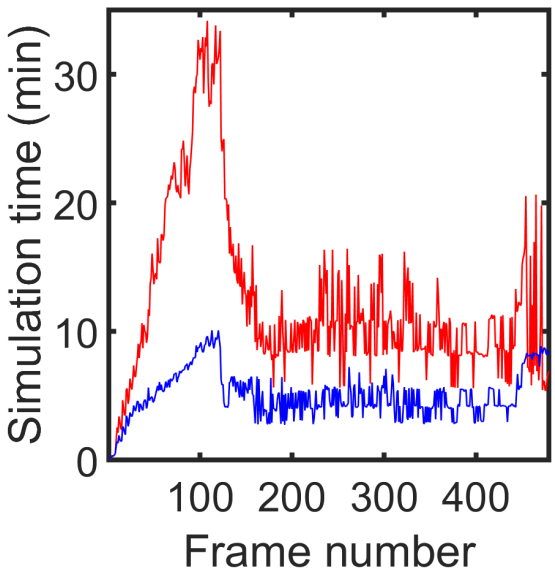
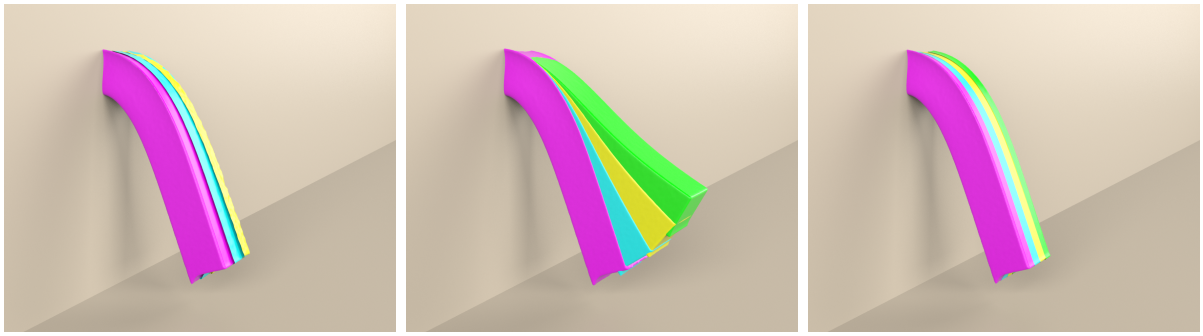


Figure 5.14: Simulation time per frame for the *Letter Mixer* (§5.6.4) example. The early frames of letters piling up and deforming are significantly slower (up to 3× slower) for the regular grid (*red*) simulation compared to the octree simulation (*blue*). Even after the letters have settled into the container, the octree simulation is 2× faster than the regular grid.



(a) Regular grids. (b) Sloped gradients. (c) Enhanced gradients.

Figure 5.15: **Viscous Beam**: (Left) Overlaid strictly regular grid viscous beams released under gravity exhibit no significant differences in bending rates at different resolutions (●=base simulation, ●=half-resolution, ●=quarter-resolution). (Middle) With our basic sloped gradient discretization, interior-coarsened octrees yield noticeable over-stiffening (●, ●, ●,● indicate 0, 1, 2, and 3 levels of interior coarsening). (Right) With our enhanced axis-aligned gradients (same color-coding), the octree simulations are dramatically improved, closely matching the reference regular grid results in magenta. Linear solve speed-up factor for the coarsest octree: 4.2×.

in Figure 5.15 that our enhanced octree discretization experiences no visually significant difference in rotation rates compared to regular grids on a viscous beam test. The basic (sloped) discretization is significantly more damped due to the poor gradient estimates.

Examining the timings for these beam tests reveals that the majority of the performance improvement is often achieved after only a single level of interior coarsening. The viscosity step in the magenta (regular grid) simulation averaged 75.8 seconds per frame, cyan (1-level coarsened) averaged 27.6 seconds, yellow (2-levels) averaged 19.8 seconds, and green (3-levels) averaged 18.7 seconds. This reflects a natural diminishing of returns: every subsequent step leaves far fewer active DOFs available to be coarsened.

5.6.6 Gooey Armadillo

In Figure 5.16 a strongly viscous armadillo is released from a standing position and slowly collapses. The large yet compact liquid volume, strong viscous forces, and high degree of deformation under gravity make this our most accelerated example: we see a speed-up factor of 9.4× for the linear solve and 8.8× for the entire viscosity solve. Furthermore,

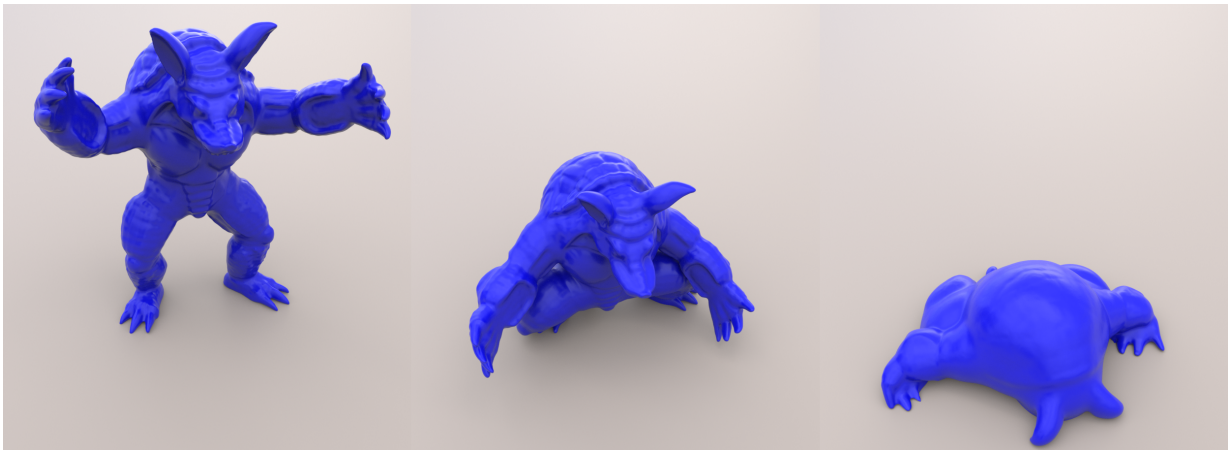


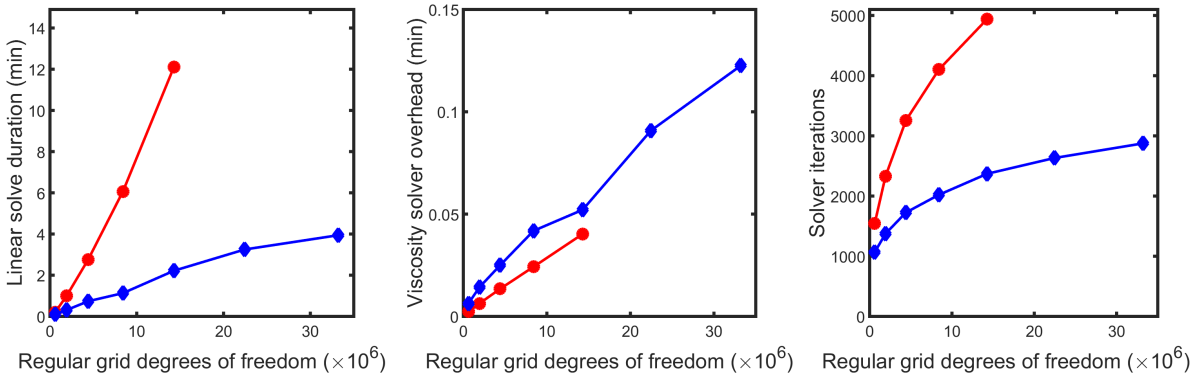
Figure 5.16: **Gooey Armadillo**: A highly viscous armadillo released from a standing position slowly collapses. Linear solve speed-up factor: 9.4×.

because the viscosity solve is a significant bottleneck in this scenario, the entire simulation performance improved by a factor of 6.3×.

Figure 5.17a plots the linear solve time as a function of fine-grid (effective) resolution for a single frame in the armadillo example for the regular grid (red) and octree (blue) methods, illustrating that our method is more beneficial for higher resolution problems. For the same resolution the smaller systems generated by our method also require fewer iterations, as shown in Figure 5.17c. Convergence of the Conjugate Gradients routine stalled completely at around 14M DOFs for the regular grid, even when periodically recomputing the residual to reduce round-off effects. By contrast, the octree system succeeded up to an equivalent of 140M regular DOFs, i.e., even beyond the range of the graphs. If the regular grid viscosity had converged at higher resolutions, these performance trends suggests that octree viscosity would yield even larger speed-up factors. Figure 5.17b shows that the overhead of both discretizations entails only a small additional computational cost compared to that of the linear solver and is not much worse for the octree.

5.6.7 Bunny Drop

Our proposed method can simulate viscous liquids at higher (effective) grid resolutions than the regular grid. In Figure 5.18, a high resolution fine grid is necessary to capture the collision between the falling viscous bunny and the two stationary thin wires. The corresponding linear system for the regular grid is so poorly conditioned that Conjugate



(a) Linear solve cost. (b) Solver overhead. (c) CG iteration counts.

Figure 5.17: Performance of the first frame for *Gooey Armadillo* (§5.6.6). The X-axis indicates the number of active DOFs for the regular grid; for the corresponding octree, this is the *effective* resolution. (a) The linear system solve for the regular grid (red) requires significantly more compute time than for the equivalent octree (blue). The regular grid also failed at higher resolutions for which the octree succeeds. (b) The overhead for the octree solve (matrix setup, interpolation, etc.) is slightly larger than for the regular grid, but still inexpensive relative to the large improvement of the solve time. (c) The number of Conjugate Gradients iterations also increases more slowly for the octree than the corresponding regular grid.

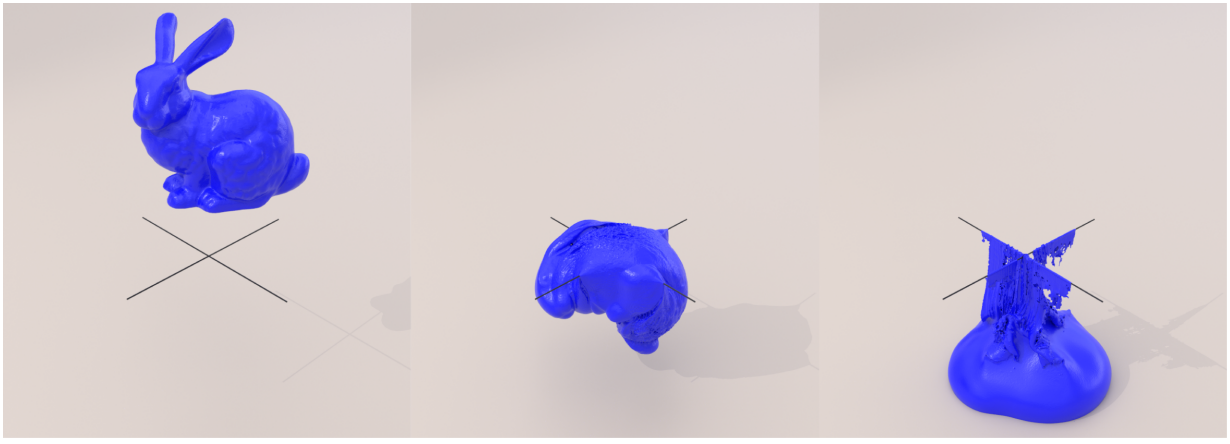


Figure 5.18: **Bunny Drop**: A viscous bunny is dropped on two thin wires. The regular grid viscosity failed to solve at this grid resolution.

Gradients fails to converge. Our method significantly reduces the size of the linear system from 27.9M DOFs for the regular grid to 4.1M for the octree grid and allows Conjugate Gradients to succeed. Because our adaptive method maintains a fine grid resolution at the liquid surface, it is also able to capture the bunny's collisions along the wires.

5.6.8 Pure Octree Simulator Examples

Figure 5.19 demonstrates that our method also works seamlessly in a purely adaptive setting, as a viscosity plugin for the inviscid octree framework of Aanjaneya et al. (2017). Figure 5.19 (left) shows a source pouring (Newtonian) ketchup. Figure 5.19 (right) shows four armadillos initially stacked together falling under gravity and collapsing into a pile. We use our basic (sloped gradient) octree viscosity method here, illustrating that even this simpler approach can yield qualitatively plausible motion in many cases. Both examples use an effective finest octree resolution of 512^3 with 4 levels, and a level-set based interface tracking resolution of 2048^3. Table 5.8 gives a timing breakdown.

Table 5.8: Average timing breakdown (in seconds) for pure octree examples.

	Source	Armadillos
Level set advection	7.2	17
Reinitialization	6.2	18
Velocity advection	1.9	1.98
Viscous update	18.4	50.3
Projection	9.6	22.7
Velocity extrapolation	1.2	2.2
Grid adaptation	3	4.2
Total time step	48	117

5.6.9 Degrees of Freedom and Matrix Sparsity

Our method achieves its efficiency by reducing the number of active DOFs. For example, the octree viscosity DOF count for the *Gooey Armadillo* (§5.6.6) was 7.7× smaller than for the regular grid. Although velocity samples at transition regions have more non-zeros per matrix row than in uniform regions, T-junctions are a relatively small subset of the

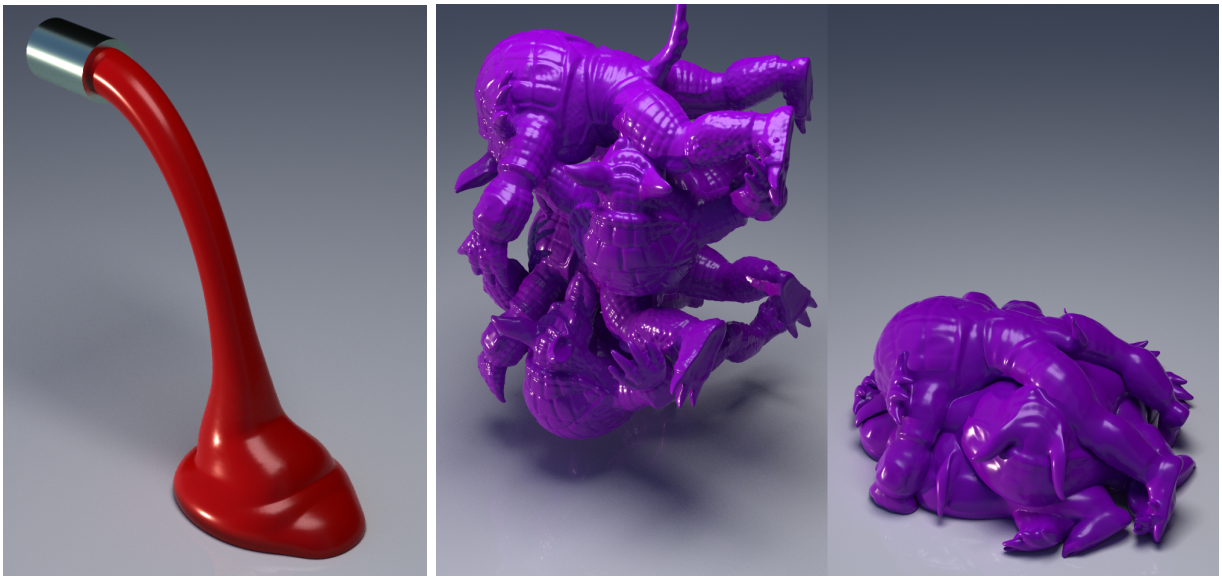


Figure 5.19: **Pure Octree Simulator**: (Left) A source pours (Newtonian) ketchup. (Right) A stack of armadillos falls into a pile.

domain and our stencils are compact so the average number of non-zeros per DOF is still comparable to a regular grid. For example, in the first frame of Figure 5.16 each matrix row contained, on average, 14.67 non-zeros for the regular grid and 16.03 non-zeros for the octree. Hence, the sparsity of our octree linear systems is not appreciably worse than for regular grids.

If one were to derive a hexahedral finite element viscosity discretization on an octree, it could also provide a DOF reduction. However, a typical node-based linear FEM would lead to much denser matrices than ours. In the first frame of Figure 5.16 our octree consists of 3.39M active face DOFs, and assuming roughly 15 non-zeros per DOF, this yields about 50.85M non-zeros for our proposed method. The same octree has 1.06M active nodes, and each FEM node would contain three velocity DOFs, resulting in 3.18M node-based DOFs. Because regular grid FEM would require 81 non-zeros per DOF on uniform regions (Zhu et al., 2010) (and T-junctions would worsen this) the linear system would contain approximately 257.8M non-zeros, i.e., more than a factor of five more matrix non-zeros than our method for about the same number of unknowns.

Chapter 6

Conclusion

Our research focuses primarily on improving fluid simulation performance by strategically reducing computational effort in fluid regions that do not meaningfully contribute to visual quality. These efficiency gains are motivated by the visual effects industry's perpetual drive for better quality simulations and faster turn around times to iterate on a director's vision. To emphasize this point, in collaboration with SideFX, our *constraint bubbles* and *adaptive viscosity* contributions have been implemented as features in their industry standard software, Houdini, versions 16.5 (SideFX, 2017) and 18.5 (SideFX, 2020), respectively. Additionally, we have released all of our research code as free, open source plug-ins to Houdini to both encourage researchers to build upon our work and visual effects artists to utilize our research tools in industry (Goldade, 2019, 2020).

6.1 Summary

The goal of our work is to significantly improve simulation efficiency for viscous liquids and entrained bubbles. Our novel variational finite difference discretization of the viscosity PDE on adaptive grids is convergent in analytical tests, SPD by construction, and up to 9.4× faster to solve than an equivalent regular grid discretization. The significant reduction in degrees-of-freedom, up to 7.7×, also allows for higher resolution simulations than is achievable with regular grids, due to Conjugate Gradients failing to converge. We demonstrate that our adaptive method still closely matches the regular grid, confirming our hypothesis that interior flows are dampened due to viscosity and can be approximated coarsely without impacting visual quality.

Our novel reduced models for bubbles consists of a constraint-based approach to remove the interior DOFs for zero-density bubbles and affine regions to model the interior, pointwise divergence-free, velocity field of non-negligible density bubbles. Our constraint method allows for interesting effects like solid-liquid coupling at-a-distance through monolithic constraint regions without incurring the substantial increase in air volume DOFs required for a standard two-phase method. Our bubble volume tracking method enables the use of the standard liquid-only FLIP implementation that correctly removes spurious voids and corrects volume drift caused by numerical errors during advection. Our affine regions correctly achieve hydrostatic equilibrium for matching bubble-liquid densities and for bubbles with greater density than the liquid, bubbles correctly sink. The affine method offers performance improvements of 4.1× over a full two-phase method. We extend our affine region method to act as a generalized coarsening strategy to accelerate free surface liquid simulations, offering a performance improvement of 6.1× while remaining qualitatively similar to the regular grid equivalent.

6.2 Discussion and Future Work

6.2.1 Reduced Model Liquids

We have proposed two new reduced fluid models that integrate with and extend the widely adopted staggered grid paradigm, and we have fruitfully applied these models in a range of compelling two-phase and free-surface flow scenarios. Affine regions, while powerful, cannot always be applied in the same situations as constraint bubbles. For example, a single constraint bubble suffices for each air gap in the winding tube geometry (§4.5.3), but multiple affine regions would be needed for comparable non-zero density flows because their explicit interior velocity field cannot represent several twists. Unsurprisingly, we also observed that a too-thin layer of regular cells between the free surface and the affine tiles can cause grid-dependent motion artifacts (see Figure 4.15c), similar to prior adaptive schemes (e.g., (Irving et al., 2006) discuss the notion of problem-dependent "optical depth").

Uniform affine tiling is perhaps the simplest adaptivity strategy that one might consider; because of the geometric flexibility inherent in our affine regions, alternative grading/shaping/sizing patterns could yield even further speed-ups. For example, quickly ramping tile sizes away from the surface, as in octrees, is an obvious next step. Because we already compute a least-squares fit of the provisional affine velocity field, the error of this fit is a potential oracle to determine if affine tiles can be locally coarsened. We assumed voxelized affine regions for simplicity, but as with rigid bodies, irregular regions could be

supported via cut-cells (Ng et al., 2009). We also assumed that affine tiles were separated by a layer of regular cells, for simplicity and to avoid velocity discontinuities between tiles — it would be interesting to explore direct tile-tile coupling.

Coupling instead to a boundary element-type (e.g., *harmonic*) velocity field (Da et al., 2016) is also an exciting avenue, though such models have many velocity degrees of freedom that cover the entire surface, rather than being coefficients of a low order polynomial. Surface tension effects could be added with a standard ghost-fluid approach (Enright et al., 2003; Hong and Kim, 2005). Since we copy the affine velocity field back to the regular grid, if a closer match to regular-grid pressure-projected velocities was desired, we could additionally apply several Jacobi smoothing iterations of the pressure projection system over the resulting velocity field.

Compressible Affine Fluids

Our constraint bubbles assume incompressibility for simplicity. However, if air compressibility is desired, our bubble-tracking could straightforwardly be extended with a per-bubble mass variable to inform a density-based equation of state model. Such terms are used by Aanjaneya et al. (2013), but our approach would again circumvent their explicit interior projection or (mass-conserving) advection. Relatedly, it would be interesting to introduce divergence-control for affine regions, which could be useful for explosion effects (Feldman et al., 2003) and volume correction, as well as the extension to equation of state models for compressible bubbles. This could be incorporated through a modified constraint of $\nabla \cdot \boldsymbol{u}_B = \mathrm{Tr}(\mathbf{A}) = const$, leading to additional terms in the right hand side vector.

Affine Region Advection

We only exploited affine regions during pressure projection (because it is often the dominating cost for inviscid fluids in practice) and used standard grid-based advection in their interiors. This might become a bottleneck for sufficiently large regions. However, it may be possible to radically reduce this cost, perhaps through a *reduced affine advection* model, in the vein of prior model reduced advection treatments (Wicke et al., 2009; Cui et al., 2018), or by carefully recovering (approximate) affine velocities from surrounding grid velocities, in the spirit of our constraint bubbles.

6.2.2 Reduced Model Framework

Affine regions for two-phase and free surface flows are interesting proof-of-concepts for a more general reduced model framework. The flexiblity of handling irregular geometric regions and the simplicity of the implementation suggest a promising new framework to quickly add adaptivity to other simulations. Recently, Panuelos et al. (2020) extended our affine regions to unsteady-Stokes (Larionov et al., 2017) by modelling velocities with a quadratic vector field. Their investigation determined that for viscous stresses, an affine field was insufficient. To that end, adopting higher order polynomial vector fields, as in PolyPIC (Fu et al., 2017), would improve the velocity field's flexibility at the cost of (rapidly) increasing degree-of-freedom counts per region. We are also interested in applying our reduced model to MPM simulations. The flux splitting method (Stomakhin et al., 2014; Fang et al., 2020; Bonet and Wood, 2008) for modelling nearly incompressible elastic materials employs a pressure Poisson problem similar to Aanjaneya et al. (2013) which, as discussed above, is a potential extension of our affine method. A more general extension of our reduced model for MPM is to model the deformation gradient tensor as a polynomial tensor field over a reduced region and therefore solve elastic forces in a reduced framework.

6.2.3 Adaptive Viscosity

Our performance numbers for our adaptive viscosity method confirm its efficiency relative to regular grids, our numerical experiments verify that our adaptive discretization converges to analytical viscosity PDE solutions and our visual experiments demonstrate both the key importance of orthogonal gradients and the method's ability to achieve high-quality results. A commonly held belief of adaptive methods was that the performance benefits are outweighed by both their overhead cost and complexity. We hope our work, like that of Setaluri et al. (2014), will help to further dispel this perception. We consciously designed a new method that can work as a drop-in replacement for an existing regular grid viscosity solver which eschews the need for an entire simulation pipeline to be redeveloped to support adaptive grids.

We retained Houdini's parallelized Jacobi-preconditioned Conjugate Gradients for our tests because, despite the large number of iterations it required, alternative off-the-shelf solvers that we tried (e.g., algebraic Multigrid) did not exhibit competitive performance. Aanjaneya et al. (2019) investigated geometric Multigrid methods for solving the regular grid viscosity problem and discovered that Jacobi smoothing performs poorly. Instead, they employed a local box smoother over each velocity DOF. Our guarantee of symmetric positive definiteness is essential in extending their preconditioned Multigrid method to the

adaptive setting. Adapting the distributive smoothing concepts from staggered Multigrid elasticity schemes (e.g., (Zhu et al., 2010)) may also be a viable avenue.

For simplicity, we adopted a uniformly refined grid at the liquid's surface and a (face-) graded interior of coarse grid cells. Lifting these restrictions could offer greater flexibility and faster coarsening, but at the cost of increased stencil complexity. Additionally, as illustrated in Figure 5.1, the face-grading scheme rapidly transitions from the fine magenta grid cells at the liquid surface to the much coarser green grid cells. A non-graded scheme would only achieve at most one extra layer of green cells.

Relatedly, compared to a more classical finite difference/volume approach, our method has two potential error sources: first, our various stencils are often not centered differences, and second, near level transitions the discrete tensor divergence operator constructed by symmetry does not have an obvious finite difference/volume interpretation. More complex but *non-symmetric* stencils would likely enable second order convergence in L_∞ (e.g., Horesh and Haber (2011) describe a non-symmetric second order octree scheme for Maxwell's equations).

6.2.4 Adaptive Variational Finite Difference Framework

We consider our octree discretization for viscous liquids as a demonstration of a more general novel adaptive variational finite difference framework that is symmetric positive definite by construction. For example, the unsteady-Stokes method of coupling pressure and viscous stresses (Larionov et al., 2017) imposes a large trade-off of simulation performance for better viscous coiling effects and surface details. This method is already presented in the variational form, requiring only the adaptive finite difference discretizations to extend the regular grid method to octrees. It would be interesting to compare solving the Stokes method using our adaptive variational framework against the quadratic regions of Panuelos et al. (2020) in terms of both implementation complexity and overall performance improvements. Looking beyond viscosity, regular grid variational finite differences have also been applied to solid-fluid interaction (Batty et al., 2007), granular flow (Narain et al., 2010), and stream functions (Ando et al., 2015a) among other problems; Zhu et al. (2010) similarly used staggered finite differences for elasticity, which can likewise be expressed in a variational form. Our work naturally opens the door to *systematic, symmetry-preserving* extensions of all such staggered finite difference methods to octrees.

References

Mridul Aanjaneya. 2018. An Efficient Solver for Two-way Coupling Rigid Bodies with Incompressible Flow. *Computer Graphics Forum* 37, 8 (2018), 59–68. https://doi.org/10.1111/cgf.13512

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids. *ACM Trans. Graph.* 36, 4, Article Article 140 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073625

Mridul Aanjaneya, Chengguizi Han, Ryan Goldade, and Christopher Batty. 2019. An Efficient Geometric Multigrid Solver for Viscous Liquids. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article Article 14 (July 2019), 21 pages. https://doi.org/10.1145/3340255

Mridul Aanjaneya, Saket Patkar, and Ronald Fedkiw. 2013. A monolithic mass tracking formulation for bubbles in incompressible flow. *J. Comput. Phys.* 247 (2013), 17 – 61. https://doi.org/10.1016/j.jcp.2013.03.048

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively Sampled Particle Fluids. *ACM Trans. Graph.* 26, 3 (July 2007), 48es. https://doi.org/10.1145/1276377.1276437

Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. 1998. A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations. *J. Comput. Phys.* 142, 1 (May 1998), 146. https://doi.org/10.1006/jcph.1998.5890

Ryoichi Ando and Christopher Batty. 2020. A Practical Octree Liquid Simulator with Adaptive Surface Resolution. *ACM Trans. Graph.* 39, 4, Article 32 (July 2020), 17 pages. https://doi.org/10.1145/3386569.3392460

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015a. A Stream Function Solver for Liquid Simulations. *ACM Trans. Graph.* 34, 4, Article Article 53 (July 2015), 9 pages. https://doi.org/10.1145/2766935

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article Article 103 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461982

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2015b. A Dimension-Reduced Pressure Solver for Liquid Simulations. *Comput. Graph. Forum* 34, 2 (May 2015), 473480. https://doi.org/10.1111/cgf.12576

Inc Autodesk. 2021. Maya.

Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH 07)*. Association for Computing Machinery, New York, NY, USA, 16es. https://doi.org/10.1145/1275808.1276397

Christopher Batty. 2017. A cell-centred finite volume method for the Poisson problem on non-graded quadtrees with second order accurate gradients. *J. Comput. Phys.* 331 (2017), 49 – 72. https://doi.org/10.1016/j.jcp.2016.11.035

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-Fluid Coupling. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH 07)*. Association for Computing Machinery, New York, NY, USA, 100es. https://doi.org/10.1145/1275808.1276502

Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 08)*. Eurographics Association, Goslar, DEU, 219228.

Christopher Batty and Ben Houston. 2011. A Simple Finite Volume Method for Adaptive Viscous Liquids. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 11)*. Association for Computing Machinery, New York, NY, USA, 111118. https://doi.org/10.1145/2019406.2019421

Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. 2012. Discrete Viscous Sheets. *ACM Trans. Graph.* 31, 4, Article 113 (July 2012), 7 pages. https://doi.org/10.1145/2185520.2185609

Christopher Batty, Stefan Xenos, and Ben Houston. 2010. Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids. *Comput. Graph. Forum* 29 (05 2010), 695–704. https://doi.org/10.1111/j.1467-8659.2009.01639.x

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete Viscous Threads. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH 10)*. Association for Computing Machinery, New York, NY, USA, Article 116, 10 pages. https://doi.org/10.1145/1833349.1778853

Javier Bonet and Richard D. Wood. 2008. (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511755446

Andrea Bonito, Marco Picasso, and Manuel Laso. 2006. Numerical simulation of 3D viscoelastic flows with free surfaces. *J. Comput. Phys.* 215, 2 (2006), 691 – 716. https://doi.org/10.1016/j.jcp.2005.11.013

Landon Boyd and Robert Bridson. 2012. MultiFLIP for Energetic Two-phase Fluid Simulation. *ACM Trans. Graph.* 31, 2, Article 16 (April 2012), 12 pages. https://doi.org/10.1145/2159516.2159522

J.U. Brackbill and H.M. Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.* 65, 2 (1986), 314 – 343. https://doi.org/10.1016/0021-9991(86)90211-1

Robert Bridson. 2015. *Fluid simulation for computer graphics, 2nd edition.* AK Peters / CRC Press.

Tyson Brochu, Christopher Batty, and Robert Bridson. 2010. Matching Fluid Simulation Elements to Surface Geometry and Topology. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH 10)*. Association for Computing Machinery, New York, NY, USA, Article Article 47, 9 pages. https://doi.org/10.1145/1833349.1778784

Oleksiy Busaryev, Tamal K. Dey, Huamin Wang, and Zhong Ren. 2012. Animating Bubble Interactions in a Liquid Foam. *ACM Trans. Graph.* 31, 4, Article 63 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185559

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. A Multiresolution Framework for Dynamic Deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 02)*. Association for Computing Machinery, New York, NY, USA, 4147. https://doi.org/10.1145/545261.545268

Mark Carlson, Peter J. Mucha, and Greg Turk. 2004. Rigid Fluid: Animating the Interplay between Rigid Bodies and Fluid. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH 04)*. Association for Computing Machinery, New York, NY, USA, 377384. https://doi.org/10.1145/1186562.1015733

Mark Carlson, Peter J. Mucha, R. Brooks Van Horn, and Greg Turk. 2002. Melting and Flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 02)*. Association for Computing Machinery, New York, NY, USA, 167174. https://doi.org/10.1145/545261.545289

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. OBrien, and Jonathan R. Shewchuk. 2007. Liquid Simulation on Lattice-Based Tetrahedral Meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 07)*. Eurographics Association, Goslar, DEU, 219228.

Nuttapong Chentanez and Matthias Müller. 2011. Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid. *ACM Trans. Graph.* 30, 4, Article Article 82 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964977

A. J. Chorin and J.E. Marsden. 1993. *A Mathematical Introduction to Fluid Mechanics* (third ed.). Springer-Verlag New York. https://doi.org/10.1007/978-1-4612-0883-9

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. OBrien. 2013. Simulating Liquids and Solid-Liquid Interactions with Lagrangian Meshes. *ACM Trans. Graph.* 32, 2, Article 17 (April 2013), 15 pages. https://doi.org/10.1145/2451236.2451243

Armando Coco and Giovanni Russo. 2018. Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface. *J. Comput. Phys.* 361 (2018), 299 – 330. https://doi.org/10.1016/j.jcp.2018.01.016

R.K. Crockett, P. Colella, and D.T. Graves. 2011. A Cartesian grid embedded boundary method for solving the Poisson and heat equations with discontinuous coefficients in three dimensions. *J. Comput. Phys.* 230, 7 (2011), 2451 – 2469. https://doi.org/10.1016/j.jcp.2010.12.017

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable Laplacian Eigenfluids. *ACM Trans. Graph.* 37, 4, Article 87 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201352

Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2015. Double Bubbles sans Toil and Trouble: Discrete Circulation-Preserving Vortex Sheets for Soap Films and Foams. *ACM Trans. Graph.* 34, 4, Article 149 (July 2015), 9 pages. https://doi.org/10.1145/2767003

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-Only Liquids. *ACM Trans. Graph.* 35, 4, Article 78 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925899

Gilles Daviet and Florence Bertails-Descoubes. 2016. A Semi-Implicit Material Point Method for the Continuum Simulation of Granular Materials. *ACM Trans. Graph.* 35, 4, Article 102 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925877

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power Particles: An Incompressible Fluid Solver Based on Power Diagrams. *ACM Trans. Graph.* 34, 4, Article 50 (July 2015), 11 pages. https://doi.org/10.1145/2766901

Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid Simulation Using Laplacian Eigenfunctions. *ACM Trans. Graph.* 31, 1, Article 10 (Feb. 2012), 11 pages. https://doi.org/10.1145/2077341.2077351

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96*, Ronan Boulic and Gerard Hégron (Eds.). Springer Vienna, Vienna, 61–76.

C. Dick, M. Rogowsky, and R. Westermann. 2016. Solving the Fluid Pressure Poisson Equation Using MultigridEvaluation and Improvements. *IEEE Transactions on Visualization and Computer Graphics* 22, 11 (2016), 2480–2492. https://doi.org/10.1109/TVCG.2015.2511734

Essex Edwards and Robert Bridson. 2014. Detailed Water with Coarse Grids: Combining Surface Meshes and Adaptive Discontinuous Galerkin. *ACM Trans. Graph.* 33, 4, Article 136 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601167

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, Circulation-Preserving, Simplicial Fluids. *ACM Trans. Graph.* 26, 1 (Jan. 2007), 4es. https://doi.org/10.1145/1189762.1189766

R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera Grids for Water Simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on*

Computer Animation (SCA 13). Association for Computing Machinery, New York, NY, USA, 8594. https://doi.org/10.1145/2485895.2485897

Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and Rendering of Complex Water Surfaces. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 02)*. Association for Computing Machinery, New York, NY, USA, 736744. https://doi.org/10.1145/566570.566645

Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. 2003. Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free Surface Flows *(Fluids Engineering Division Summer Meeting)*. 337–342. https://doi.org/10.1115/FEDSM2003-45144

Henrik Fält and Doug Roble. 2003. Fluids with extreme viscosity. In *SIGGRAPH Sketches*. 1.

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. 2020. IQ-MPM: An Interface Quadrature Material Point Method for Non-Sticky Strongly Two-Way Coupled Nonlinear Solids and Fluids. *ACM Trans. Graph.* 39, 4, Article 51 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392438

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 1522. https://doi.org/10.1145/383259.383260

Bryan E. Feldman, James F. OBrien, and Okan Arikan. 2003. Animating Suspended Particle Explosions. *ACM Trans. Graph.* 22, 3 (July 2003), 708715. https://doi.org/10.1145/882262.882336

F. Ferstl, R. Westermann, and C. Dick. 2014. Large-Scale Liquid Simulation on Adaptive Hexahedral Grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417.

Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 2330. https://doi.org/10.1145/383259.383261

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Models Image Process.* 58, 5 (Sept. 1996), 471483. https://doi.org/10.1006/gmip.1996.0039

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A Polynomial Particle-in-Cell Method. *ACM Trans. Graph.* 36, 6, Article 222 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130878

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials. *ACM Trans. Graph.* 36, 6, Article 223 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130879

T. V. Gerya, D. A. May, and T. Duretz. 2013. An adaptive staggered grid finite difference method for modeling geodynamic Stokes flows with strongly variable viscosity. *Geochemistry, Geophysics, Geosystems* 14, 4 (2013), 1200–1225. https://doi.org/10.1002/ggge.20078 arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/ggge.20078

Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *J. Comput. Phys.* 176, 1 (2002), 205 – 227. https://doi.org/10.1006/jcph.2001.6977

Frdric Gibou and Chohong Min. 2012. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.* 231, 8 (2012), 3246 – 3263. https://doi.org/10.1016/j.jcp.2012.01.009

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (Nov. 2012), 9 pages. https://doi.org/10.1145/2366145.2366167

Ryan Goldade. 2019. *Adaptive Viscosity Solver.* https://github.com/rgoldade/AdaptiveViscositySolver

Ryan Goldade. 2020. *Reduced Fluids.* https://github.com/rgoldade/ReducedFluids

Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. 2020. Constraint Bubbles and Affine Regions: Reduced Fluid Models for Efficient Immersed Bubbles and Flexible Spatial Coarsening. *ACM Trans. Graph.* 39, 4, Article 43 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392455

Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. 2019. An Adaptive Variational Finite Difference Framework for Efficient Symmetric Octree Viscosity. *ACM Trans. Graph.* 38, 4, Article 94 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3322939

S. T. Greenwood and D. H. House. 2004. Better with Bubbles: Enhancing the Visual Realism of Simulated Fluid. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 04)*. Eurographics Association, Goslar, DEU, 287296. https://doi.org/10.1145/1028523.1028562

Eitan Grinspun, Petr Krysl, and Peter Schröder. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. Graph.* 21, 3 (July 2002), 281290. https://doi.org/10.1145/566654.566578

Arthur Guittet, Maxime Theillard, and Frédéric Gibou. 2015. A Stable Projection Method for the Incompressible Navier-Stokes Equations on Arbitrary Geometries and Adaptive Quad/Octrees. *J. Comput. Phys.* 292, C (July 2015), 215238. https://doi.org/10.1016/j.jcp.2015.03.024

Karl Gustafson and Robert Hartman. 1983. Divergence-Free Bases for Finite Element Schemes in Hydrodynamics. *SIAM J. Numer. Anal.* 20, 4 (1983), 697–721. http://www.jstor.org/stable/2157235

Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of TimeDependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids* 8, 12 (1965), 2182–2189. https://doi.org/10.1063/1.1761178

Jeong-Mo Hong and Chang-Hun Kim. 2005. Discontinuous Fluids. *ACM Trans. Graph.* 24, 3 (July 2005), 915920. https://doi.org/10.1145/1073204.1073283

Jeong-Mo Hong, Ho-Young Lee, Jong-Chul Yoon, and Chang-Hun Kim. 2008. Bubbles Alive. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 14. https://doi.org/10.1145/1360612.1360647

Lior Horesh and Eldad Haber. 2011. A Second Order Discretization of Maxwell's Equations in the Quasi-Static Regime on OcTree Grids. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2805–2822. https://doi.org/10.1137/100798508 arXiv:https://doi.org/10.1137/100798508

Hikaru Ibayashi, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2018. Simulating Liquids on Dynamically Warping Grids. *IEEE Transactions on Visualization and Computer Graphics* in press (2018). https://doi.org/10.1109/TVCG.2018.2883628

Markus Ihmsen, Julian Bader, Gizem Akinci, and Matthias Teschner. 2011. Animation of Air Bubbles with SPH. *GRAPP 2011 - Proceedings of the International Conference on Computer Graphics Theory and Applications*, 225–234.

M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner. 2014. Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2014), 426–435. https://doi.org/10.1109/TVCG.2013.105

Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH 06)*. Association for Computing Machinery, New York, NY, USA, 805811. https://doi.org/10.1145/1179352.1141959

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Trans. Graph.* 34, 4, Article Article 51 (July 2015), 10 pages. https://doi.org/10.1145/2766996

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The Material Point Method for Simulating Continuum Materials. In *ACM SIGGRAPH 2016 Courses (SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 24, 52 pages. https://doi.org/10.1145/2897826.2927348

Myungjoo Kang, Ronald P. Fedkiw, and Xu-Dong Liu. 2000. A Boundary Condition Capturing Method for Multiphase Incompressible Flow. *Journal of Scientific Computing* 15, 3 (01 Sep 2000), 323–360. https://doi.org/10.1023/A:1011178417620

T. Keeler and R. Bridson. 2014. Ocean Waves Animation Using Boundary Integral Equations and Explicit Mesh Tracking. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 14)*. Eurographics Association, Goslar, DEU, 1119.

Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. 2007. Simulation of Bubbles in Foam with the Volume Control Method. In *ACM SIGGRAPH 2007 Papers (SIGGRAPH 07)*. Association for Computing Machinery, New York, NY, USA, 98es. https://doi.org/10.1145/1275808.1276500

Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. 2010. A Practical Simulation of Dispersed Bubble Flow. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 70, 5 pages. https://doi.org/10.1145/1833349.1778807

Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-Prager Elastoplasticity for Sand Animation. *ACM Trans. Graph.* 35, 4, Article 103 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925906

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. OBrien. 2006. Fluid Animation with Dynamic Meshes. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH 06)*. Association for Computing Machinery, New York, NY, USA, 820825. https://doi.org/10.1145/1179352.1141961

T. Kugelstadt, A. Longva, N. Thurey, and J. Bender. 2019. Implicit Density Projection for Volume Conserving Liquids. *IEEE Transactions on Visualization and Computer Graphics* (2019).

Timothy R. Langlois, Changxi Zheng, and Doug L. James. 2016. Toward Animating Water with Complex Acoustic Bubbles. *ACM Trans. Graph.* 35, 4, Article 95 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925904

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational Stokes: A Unified Pressure-Viscosity Solver for Accurate Viscous Liquids. *ACM Trans. Graph.* 36, 4, Article Article 101 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073628

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids* 35, 10 (2006), 995–1010. https://doi.org/10.1016/j.compfluid.2005.01.006

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 457462. https://doi.org/10.1145/1015706.1015745

Wenlong Lu, Ning Jin, and Ronald Fedkiw. 2016. Two-Way Coupling of Fluids to Reduced Deformable Bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 16)*. Eurographics Association, Goslar, DEU, 6776.

S.P. MacLachlan, J.M. Tang, and C. Vuik. 2008. Fast and robust solvers for pressure-correction in bubbly flow problems. *J. Comput. Phys.* 227, 23 (2008), 9742 – 9761. https://doi.org/10.1016/j.jcp.2008.07.022

Gianmarco Manzini, Alessandro Russo, and N. Sukumar. 2014. New perspectives on polygonal and polyhedral finite element methods. *Mathematical Models and Methods in Applied Sciences* 24, 08 (2014), 1665–1699. https://doi.org/10.1142/S0218202514400065 arXiv:https://doi.org/10.1142/S0218202514400065

Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. 2008. Polyhedral Finite Elements Using Harmonic Basis Functions. In *Proceedings of the*

Symposium on Geometry Processing (SGP 08). Eurographics Association, Goslar, DEU, 15211529.

A. McAdams, E. Sifakis, and J. Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 10)*. Eurographics Association, Goslar, DEU, 6574.

Viorel Mihalef, Betul Unlusu, Dimitris Metaxas, Mark Sussman, and M. Y. Hussaini. 2006. Physics Based Boiling Simulation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Marie-Paule Cani and James O'Brien (Eds.). The Eurographics Association. https://doi.org/10.2312/SCA/SCA06/317-324

Chohong Min and Frédéric Gibou. 2006. A Second Order Accurate Projection Method for the Incompressible Navier-Stokes Equations on Non-Graded Adaptive Grids. *J. Comput. Phys.* 219, 2 (Dec. 2006), 912929. https://doi.org/10.1016/j.jcp.2006.07.019

Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Brentzen, and Franois Anton. 2010. Optimization-based Fluid Simulation on Unstructured Meshes. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)*, Kenny Erleben, Jan Bender, and Matthias Teschner (Eds.). The Eurographics Association. https://doi.org/10.2312/PE/vriphys/vriphys10/011-020

M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. B. Christensen, J. Andreas Brentzen, and R. Bridson. 2014. Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes. *IEEE Transactions on Visualization and Computer Graphics* 20, 1 (2014), 4–16.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-Preserving Integrators for Fluid Animation. *ACM Trans. Graph.* 28, 3, Article Article 38 (July 2009), 8 pages. https://doi.org/10.1145/1531326.1531344

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Goslar, DEU, 154159.

Rahul Narain, Abhinav Golas, and Ming C. Lin. 2010. Free-Flowing Granular Materials with Two-Way Solid Coupling. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA 10)*. Association for Computing Machinery, New York, NY, USA, Article Article 173, 10 pages. https://doi.org/10.1145/1866158.1866195

Rahul Narain, Jonas Zehnder, and Bernhard Thomaszewski. 2019. A Second-Order Advection-Reflection Solver. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 16 (July 2019), 14 pages. https://doi.org/10.1145/3340257

Yen Ting Ng, Chohong Min, and Frédéric Gibou. 2009. An Efficient Fluid-Solid Coupling Algorithm for Single-Phase Flows. *J. Comput. Phys.* 228, 23 (Dec. 2009), 88078829. https://doi.org/10.1016/j.jcp.2009.08.032

Michael B. Nielsen and Robert Bridson. 2016. Spatially Adaptive FLIP Fluid Simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks (SIGGRAPH 16)*. Association for Computing Machinery, New York, NY, USA, Article 41, 2 pages. https://doi.org/10.1145/2897839.2927399

Kirill Nikitin and Yuri Vassilevski. 2008. Free surface flow modelling on dynamically refined hexahedral meshes. *Russian Journal of Numerical Analysis and Mathematical Modelling* 23 (11 2008), 469–485. https://doi.org/10.1515/RJNAMM.2008.027

K. D. Nikitin, M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski. 2011. A numerical method for the simulation of free surface flows of viscoplastic fluid in 3D. *Journal of Computational Mathematics* (2011), 605–622.

Maxim A. Olshanskii, Kirill M. Terekhov, and Yuri V. Vassilevski. 2013. An octree-based solver for the incompressible NavierStokes equations with enhanced stability and low dissipation. *Computers & Fluids* 84 (2013), 231 – 246. https://doi.org/10.1016/j.compfluid.2013.04.027

Jonathan Panuelos, Ryan Goldade, and Christopher Batty. 2020. Efficient Unified Stokes using a Polynomial Reduced Fluid Model. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*, Dominik L. Michels (Ed.). The Eurographics Association. https://doi.org/10.2312/sca.20201214

Saket Patkar, Mridul Aanjaneya, Dmitriy Karpman, and Ronald Fedkiw. 2013. A Hybrid Lagrangian-Eulerian Formulation for Bubble Generation and Dynamics. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 13)*. Association for Computing Machinery, New York, NY, USA, 105114. https://doi.org/10.1145/2485895.2485912

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An Implicit Viscosity Formulation for SPH Fluids. *ACM Trans. Graph.* 34, 4, Article 114 (July 2015), 10 pages. https://doi.org/10.1145/2766925

A. Peer and M. Teschner. 2017. Prescribed Velocity Gradients for Highly Viscous SPH Fluids with Vorticity Diffusion. *IEEE Transactions on Visualization and Computer Graphics* 23, 12 (2017), 2656–2662. https://doi.org/10.1109/TVCG.2016.2636144

Stphane Popinet. 2018. Numerical Models of Surface Tension. *Annual Review of Fluid Mechanics* 50, 1 (2018), 49–75. https://doi.org/10.1146/annurev-fluid-122316-045034 arXiv:https://doi.org/10.1146/annurev-fluid-122316-045034

Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. 2015. A Material Point Method for Viscoelastic Fluids, Foams and Sponges. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA 15)*. Association for Computing Machinery, New York, NY, USA, 157163. https://doi.org/10.1145/2786784.2786798

N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. 2004. Directable Photorealistic Liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 04)*. Eurographics Association, Goslar, DEU, 193202. https://doi.org/10.1145/1028523.1028549

Sander Rhebergen and Garth N Wells. 2018. A hybridizable discontinuous Galerkin method for the Navier–Stokes equations with pointwise divergence-free velocity field. *Journal of Scientific Computing* 76, 3 (2018), 1484–1501. https://doi.org/10.1007/s10915-018-0671-4

Avi Robinson-Mosher, R. Elliot English, and Ronald Fedkiw. 2009. Accurate Tangential Velocities for Solid Fluid Coupling. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 09)*. Association for Computing Machinery, New York, NY, USA, 227236. https://doi.org/10.1145/1599470.1599500

Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-Way Coupling of Fluids to Rigid and Deformable Solids and Shells. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 19. https://doi.org/10.1145/1360612.1360645

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans. Graph.* 33, 6, Article Article 205 (Nov. 2014), 12 pages. https://doi.org/10.1145/2661229.2661269

L Shi and Y Yu. 2004. Visual smoke simulation with adaptive octree refinement. *Proceedings of the Seventh IASTED International Conference on Computer Graphics and Imaging*, Article 2-s2.0-10444232010 (2004), Article 2-s2.0-10444232010. http://hdl.handle.net/10722/151846

SideFX. 2017. *SideFX releases Houdini 16.5.* https://www.sidefx.com/community/sidefx-releases-houdini-165/

SideFX. 2020. *Houdini 18.5 What's New, VFX.* https://www.sidefx.com/products/whats-new/18_5_vfx/

SideFX. 2021. Houdini.

Barbara Solenthaler and Markus Gross. 2011. Two-Scale Particle Simulation. *ACM Trans. Graph.* 30, 4, Article 81 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964976

B. Solenthaler and R. Pajarola. 2008. Density Contrast SPH Interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 08)*. Eurographics Association, Goslar, DEU, 211218.

B. Solenthaler and R. Pajarola. 2009. Predictive-Corrective Incompressible SPH. *ACM Trans. Graph.* 28, 3, Article 40 (July 2009), 6 pages. https://doi.org/10.1145/1531326.1531346

Oh-Young Song, Hyuncheol Shin, and Hyeong-Seok Ko. 2005. Stable but Nondissipative Water. *ACM Trans. Graph.* 24, 1 (Jan. 2005), 8197. https://doi.org/10.1145/1037957.1037962

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121128. https://doi.org/10.1145/311535.311548

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461948

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for Phase-Change and Varied Materials. *ACM Trans. Graph.* 33, 4, Article Article 138 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601176

Alexey Stomakhin, Joel Wretborn, Kevin Blom, and Gilles Daviet. 2020. Underwater Bubbles and Coupling. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 2, 2 pages. https://doi.org/10.1145/3388767.3407390

Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications* 87, 1 (1995), 236 – 252. https://doi.org/10.1016/0010-4655(94)00170-7 Particle Simulation Methods.

Mark Sussman. 2003. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.* 187, 1 (2003), 110 – 136. https://doi.org/10.1016/S0021-9991(03)00087-1

Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C. Lin. 2015. Implicit Formulation for SPH-Based Viscous Fluids. *Comput. Graph. Forum* 34, 2 (May 2015), 493502. https://doi.org/10.1111/cgf.12578

Tetsuya Takahashi and Ming C. Lin. 2019. A Geometrically Consistent Viscous Fluid Solver with Two-Way Fluid-Solid Coupling. *Computer Graphics Forum* 38, 2 (2019), 49–58. https://doi.org/10.1111/cgf.13618 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13618

Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. 2010. A Multiscale Approach to Mesh-Based Surface Tension Flows. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH 10)*. Association for Computing Machinery, New York, NY, USA, Article 48, 10 pages. https://doi.org/10.1145/1833349.1778785

Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model Reduction for Real-Time Fluids. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH 06)*. Association for Computing Machinery, New York, NY, USA, 826834. https://doi.org/10.1145/1179352.1141962

Justin W. L. Wan and Xu-Dong Liu. 2004. A Boundary Condition–Capturing Multigrid Approach to Irregular Boundary Problems. *SIAM Journal on Scientific Computing* 25, 6 (2004), 1982–2003. https://doi.org/10.1137/S1064827503428540

Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. 2018. A Physically Consistent Implicit Viscosity Solver for SPH Fluids. *Computer Graphics Forum* 37, 2 (2018), 145–155. https://doi.org/10.1111/cgf.13349 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13349

Jeremy D. Wendt, William Baxter, Ipek Oguz, and Ming C. Lin. 2007. Finite volume flow simulations on arbitrary domains. *Graphical Models* 69, 1 (2007), 19 – 32. https://doi.org/10.1016/j.gmod.2006.05.004

Martin Wicke, Matt Stanton, and Adrien Treuille. 2009. Modular Bases for Fluid Dynamics. *ACM Trans. Graph.* 28, 3, Article 39 (July 2009), 8 pages. https://doi.org/10.1145/1531326.1531345

Chris Wojtan and Greg Turk. 2008. Fast Viscoelastic Behavior with Thin Features. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 18. https://doi.org/10.1145/1360612.1360646

Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum Foam: A Material Point Method for Shear-Dependent Flows. *ACM Trans. Graph.* 34, 5, Article 160 (Nov. 2015), 20 pages. https://doi.org/10.1145/2751541

Yonghao Yue, Breannan Smith, Peter Yichen Chen, Maytee Chantharayukhonthorn, Ken Kamrin, and Eitan Grinspun. 2018. Hybrid Grains: Adaptive Coupling of Discrete and Continuum Simulations of Granular Media. *ACM Trans. Graph.* 37, 6, Article 283 (Dec. 2018), 19 pages. https://doi.org/10.1145/3272127.3275095

Omar Zarifi and Christopher Batty. 2017. A Positive-Definite Cut-Cell Method for Strong Two-Way Coupling between Fluids and Deformable Bodies. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA 17)*. Association for Computing Machinery, New York, NY, USA, Article Article 7, 11 pages. https://doi.org/10.1145/3099564.3099572

Changxi Zheng and Doug L. James. 2009. Harmonic Fluids. *ACM Trans. Graph.* 28, 3, Article 37 (July 2009), 12 pages. https://doi.org/10.1145/1531326.1531343

Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. 2006. Simulation of Bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 06)*. Eurographics Association, Goslar, DEU, 325333.

Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. 2015. Codimensional Non-Newtonian Fluids. *ACM Trans. Graph.* 34, 4, Article 115 (July 2015), 9 pages. https://doi.org/10.1145/2766981

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A New Grid Structure for Domain Extension. *ACM Trans. Graph.* 32, 4, Article Article 63 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461999

Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH 05)*. Association for Computing Machinery, New York, NY, USA, 965972. https://doi.org/10.1145/1186822.1073298

Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An Efficient Multigrid Method for the Simulation of High-Resolution Elastic Solids. *ACM Trans. Graph.* 29, 2, Article Article 16 (April 2010), 18 pages. https://doi.org/10.1145/1731047.1731054