Simulating Viscous Incompressible Fluids with Embedded Boundary Finite Difference Methods

by

Christopher Batty

B.C.Sc., The University of Manitoba, 2004

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

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2010

Abstract

The behaviour of liquids and gases ranks among the most familiar and yet complex physical phenomena commonly encountered in daily life. To create a seamless approximation of the real world, it is clear that we must be able to accurately simulate fluids. However, a crucial element of what makes fluid behaviour so complex and compelling is its interactions with its surroundings. To simulate the motion of a fluid we cannot consider the Navier-Stokes equations in isolation; we must also examine the boundary conditions at the point where the fluid meets the world.

Enforcing these boundary conditions has traditionally been a source of tremendous difficulty. Cartesian grid-based methods typically approximate the world in an unrealistic, axis-aligned block representation, while conforming mesh methods frequently suffer from poor mesh quality and expensive mesh generation. This thesis examines the use of embedded boundary finite difference methods to alleviate these shortcomings by providing a degree of sub-grid information that enables more efficient, flexible, accurate, and realistic simulations.

The first key contribution of the thesis is the use of a variational approach to derive novel embedded boundary finite difference methods for fluids, by exploiting the concept of natural boundary conditions. This idea is applied first to animate the interaction between incompressible fluids and irregularly shaped dynamic rigid bodies. I then apply a similar technique to properly handle viscous free surfaces, enabling realistic buckling and coiling in viscous flows. Lastly, I unify these ideas

to simulate Stokes flows in the presence of both free surface and solid boundaries, and demonstrate the method's convergence on a range of examples.

The second main contribution is a study of embedded boundary methods for pressure projection in the context of unstructured Delaunay and Voronoi meshes. By eliminating the need for boundary-conforming meshes, this work enables efficient high-quality adaptive mesh generation and improves simulation accuracy. Furthermore, it demonstrates that by placing simulation samples at Voronoi sites, and choosing these sites intelligently with respect to liquid geometry, one can eliminate surface noise, improve the realism and stability of surface tension, and plausibly simulate nearly arbitrarily thin sheets and droplets.

Table of Contents

List of Tables

List of Figures

Notation

The following symbols and letters are used throughout chapter 4, with units given for dimensionful quantities:

μ : coefficient of dynamic viscosity, $Pa \cdot s$

ρ : density coefficient, kg/m^3

Ω_F : fluid domain

Ω_S : solid domain (the complement of Ω_F)

Ω_L : liquid domain

Ω_A : air domain (the complement of Ω_L)

$\vec{u}$: velocity vector, m/s

p : pressure, Pa

τ : deviatoric stress tensor, Pa

Δt : time step, s

$\vec{T}$: traction vector, Pa

D : discrete deformation rate matrix

G : discrete gradient matrix

M : diagonal matrix of viscosity coefficients, per velocity sample

P : diagonal matrix of density coefficients, per pressure/stress sample

W_F : diagonal matrix of fluid (non-solid) fraction weights

W_S : diagonal matrix of solid fraction weights, complementing W_L

W_L : diagonal matrix of liquid (non-air) fraction weights

W_A : diagonal matrix of air fraction weights, complementing W_L

W^u, W^p, W^τ : superscripts on weight matrices indicate the associated sample position. ie. velocity u (cell faces), pressure p (cell centres), stresses τ (cell centres and edges).

Acknowledgements

There are many people to thank for helping me along the way to the completion of this thesis.

First and foremost, my supervisor Dr. Robert Bridson has been a supportive teacher, a dedicated collaborator, and an inspiring role model. His intelligence, kindness, and humour have all helped in making my time as a PhD student smoother and more enjoyable than I could have dared hope. In my future research, I have no doubt that when confronted with a particularly challenging research question, I will ask myself, what would Robert do?

I am deeply indebted to the talented researchers with whom I have collaborated over the years, and who shared with me the intense and rapidly alternating periods of joy and despair that comprise a paper deadline. Dr. Florence Bertails-Descoubes, Stefan Xenos, Ben Houston, and Tyson Brochu have all contributed greatly to aspects of the research in this thesis, and lost sleep in the process. Thank you!

I would like to thank my examining committee, consisting of Dr. Wolfgang Heidrich, Dr. Eldad Haber, Dr. Dominik Schötzau, and Dr. Dinesh Pai. Your thorough reading, insightful comments, attention to detail, and challenging questions have improved the content of this thesis. Thanks also to my external examiner, Dr. Frederic Gibou, for reading and reporting on my thesis.

Over the course of my PhD I took two short leaves of absence for internships,

first at Intel in 2007, and then at Weta Digital in 2009. I thank these organizations for giving me the opportunity to visit and work with them. In particular, I'd like to acknowledge Anastasio Rodriguez and Holger Spill from Weta, and Dr. Mikhail Smelyanskiy and Dr. Jatin Chhugani from Intel. Likewise, I thank Exocortex Technologies, and Ben Houston, for hiring me as a consultant, supporting my desire to publish the resulting research, and for flying me to Sweden to present it. Thanks also to NSERC for providing me with the graduate scholarships that allowed me to pursue my studies.

The students who shared lab space with me in Robert's research group have taught me a great deal, and often made sitting indoors in front of a computer on sunny summer days rather more tolerable than it ought to be. In no particular order: David, Tyson, Dinos, Mike, Hagit, David, Brent, Ives, Landon, and Essex. Outside the lab, there were also many friends who contributed to making these five years in Vancouver a real joy, especially on those occasions when I "took a weekend off" from research. I won't even attempt to list you all, but I thank you for your friendship. In particular, I have shared innumerable good times with Thomas and Derek, and I am grateful that we had the opportunity to pursue doctoral studies alongside one another.

There are also several individuals who can take some credit for my eventual pursuit of a PhD involving math and computer science. I thank Mr. Spivak for amusing (and educational) high school math classes, Dr. John Anderson and Dr. Helen Cameron for first getting me interested in academic research, and Mark Wiebe and Ben Houston for hiring me on at Frantic Films, where my journey into the world of fluid simulation began.

No words are adequate to express my appreciation for the unfailing love and support of my parents, Dorothy and Craig, and my brother Adam. Though you might not read all of the pages that follow, know that not a single one would have

been possible without you.

Finally, I'd like to thank Jennifer Silver - for teaching me that science is social, for tolerating my annual SIGGRAPH deadline disappearing act, for nudging me towards a PhD, and for sharing so many amazing adventures with me over these last four years. I looked forward to the many still to come.

Dedication

To my parents, Dorothy and Craig, and my brother, Adam.

Statement of Co-Authorship

This thesis is manuscript-based, and hence the core chapters are comprised of independent papers that have either been published or are currently in submission. Some parts of the research were carried out in collaborative fashion with other members of the Imager Lab for Graphics, Visualization, and HCI at UBC, and with employees of Exocortex Technologies, Inc., an Ottawa-based company focusing on software development for visualization and simulation. My interactions and discussions with these individuals have helped tremendously in shaping and influencing my work; in particular Dr. Robert Bridson was integral in guiding the course of my thesis research along the path to its current form. All of the work presented was carried out in collaboration with Robert, with the exception of Chapter 5.

I was the lead author on the work presented in Chapter 2. It grew out of an individual course project, and as such I carried out the main research and development effort. In the months leading up to the paper submission deadline, Dr. Bridson's postdoc Dr. Florence Bertails joined the project. She contributed to several aspects, including implementing the C++ version of our 2D rigid body coupling method, adding support for user interaction and complex meshes to the 3D code, and preparing various examples. The writing was lead by Dr. Bridson, with contributions from Dr. Bertails and myself.

Chapters 3 and 4 consist of work that is primarily my own, with supervision from Dr. Bridson.

Chapter 5 was a result of collaboration with Stefan Xenos and Ben Houston from Exocortex Technologies, Inc. I was the leader on the research aspects of the project, carrying out most of the experimental effort, writing the novel elements of the code, developing the main arguments for the method, and writing the paper. Ben Houston made the initial proposal to combine embedded boundaries with unstructured meshes. Both Ben Houston and Stefan Xenos were instrumental in developing the basic unstructured tetrahedral mesh infrastructure that the research relied on, and in helping to prepare examples and videos for the paper submission.

Chapter 6 was a joint project with Tyson Brochu, another of Dr. Bridson's doctoral students, and the research effort was divided relatively even between us. I proposed the initial idea of using Voronoi-based simulation in concert with Tyson's prior surface tracking research in order to capture thin features. I coded the initial 2D Voronoi simulation method, which Tyson integrated with his *El Topo* surface tracking code. Tyson worked out the specifics of the intelligent sampling scheme that appeared in the paper and coded the majority of the full 3D liquid simulator. He also experimented with various alternative interpolation schemes for advection. Lastly, I developed and implemented the surface tension model and the interpolation scheme that appeared in the final paper.

Chapter 1

Introduction

Fluid flow is an omnipresent element of human life. From the air we breathe, to the water we drink; from the oceans and skies through which we travel, to the blood flowing in our veins; fluids surround and pervade us every day. As we have sought to better understand and manipulate our environment, the desire to computationally reproduce fluid phenomena is a natural one, with countless practical applications. Fluid simulation has improved our ability to design aerodynamic ships, automobiles, and airplanes. It has benefited numerous industrial processes, such as container filling and food processing. Biology is another important application area; some of the earliest methods for solid-fluid interaction focused on simulating blood flow in the heart. The list goes on and on.

In the last two decades, there has also arisen a great deal of interest in adapting fluid simulation techniques for computer graphics applications. This was initially motivated by high-end visual effects needs within the film industry, in order to provide realistic and inexpensive depictions of phenomena such as liquids, smoke, explosions, and fire. Traditionally, the alternative to simulation has been either the use of practical on-set effects and scale models, or painstaking artist-driven animation. The former provides a high degree of realism, but fairly limited control and flexibility. In contrast, the latter provides essentially complete control, but remains extremely challenging due to the vast number of degrees of freedom present in fluid phenomena, the limited range of tools available, and the ability of human

viewers to perceive relatively small errors in fluid motion. Fluid simulation lies in the centre of this continuuum, providing a greater measure of control along with guaranteed repeatability, while offering more convincing animations than artists can typically achieve by hand. As fluid simulation techniques have improved they have seen wide adoption in major effects-driven films, including *The Day After Tomorrow*, *Terminator 3: Rise of the Machines*, *Star Wars: Episode III*, *Pirates of the Caribbean: At World's End*, *Poseidon*, various Pixar films, and most recently *Avatar*. Furthermore, commodity graphics hardware has also steadily improved, and as such computer gaming and interactive applications are now beginning to make use of fluid animation. This trend shows no signs of abating, suggesting that seeking practical algorithms for realistic fluid animation will continue to be an important priority for computer graphics researchers.

Driven by the desire to provide better and more general tools for fluid animation, the research community has continued to seek advances in primarily three broad areas. First, technical improvements have been considered to increase apparent realism, efficiency, and robustness, such as the development of adaptive approaches [LGF04, KFCO06], the use of better numerical integration schemes [SB08, SFK$^+$08], or the introduction of alternative methods to track liquid surfaces [BOGS06, WTGT09]. Secondly, a wide range of extensions and closely related new phenomena have been addressed, including viscous flows and melting [CMVT02], compressible flows and shock waves [YOH00, SGTL08], coupling to rigid and deformable solids and shells [CMT04, GSLF05], surface tension [HK05, ZYP06], multiphase fluids [HK05, LSSF06], sub-grid turbulence models [KTJG08, NSCL08, SB08], viscoelasticity [GBO04], and more. Thirdly, various control strategies [FL04, SY05, TKPR06] and procedural tools [BHN07] have been introduced in an ongoing effort to provide technical artists the ability to direct and influence the behaviour of animated fluids in meaningful ways.

The work in this thesis straddles the first two categories, introducing novel techniques to both improve on previously studied phenomena and enable simulation of entirely new phenomena. The focus is on viscous, incompressible liquids, possibly with surface tension, and their interaction with static, kinematic, and dynamic solid objects. Although these are certainly among the most common types of fluid encountered in our daily experience, fundamental difficulties remain in our understanding of how to simulate them. For example, no prior fluid animation method is able to reproduce the familiar coiling and folding behaviour of strongly viscous liquids like honey and syrup. Similarly, existing methods for solid-fluid coupling either fail to entirely prevent fluid from flowing through solids or require an expensive, high-quality conforming mesh to be generated at each step of a simulation. I strive to address these shortcomings with methods that are simple to implement and provide realistic results, while being based on sound physical, numerical, and geometric principles.

In developing an effective fluid simulator, many of the greatest difficulties one faces arise in the application of appropriate conditions at the physical boundaries of the domain. With this in mind, the unifying theme of this thesis is the use of *embedded boundary methods* to address these difficulties for a number of related simulation problems. This term comprises a wide range of techniques that allow simulations to be computed on an underlying grid or mesh that need not conform to the geometry of the physical domain. For example, a curved liquid surface will often cut arbitrarily and irregularly through a uniform Cartesian grid. In this sense, the real liquid-air boundary is embedded into the simulation mesh, rather than being required to align with it. There are many reasons why this approach may be preferred. First, it usually increases the apparent or effective resolution of a simulation, without requiring an attendant increase in the actual mesh resolution; this provides a valuable computational savings and in many cases improves accuracy

as well. Secondly, it allows the user greater flexibility in the construction of the simulation mesh, by relaxing the boundary conforming requirements on the mesh generation tool. This can be exploited to ensure higher mesh quality, to accelerate and simplify mesh generation, to design a mesh that better resolves important physical or geometric details, or to avoid remeshing altogether by using cost-effective regular grids. Third, it can enable convenient multiphysics coupling between simulations whose mesh discretizations may not conform to one another. An example that arises frequently in practice is the interaction between Lagrangian solid models and Eulerian fluid models. To date, embedded boundary methods have found numerous applications in computer graphics and computational physics; the current work develops several new additions to this family in the context of finite difference simulation of viscous incompressible liquids.

1.1 Background

This section provides an overview of some of the relevant concepts and literature. Because each chapter of the thesis is relatively self-contained, including its own bibliography and discussion of related work, this section will not attempt to be entirely comprehensive. The goal will instead be to provide an overall context for the work that follows.

1.1.1 Influences from Computational Fluid Dynamics

The Navier-Stokes equations are a set of complex, partial differential equations describing the motion of fluids. A classic text by Batchelor provides a good introduction to the mathematics underlying fluid dynamics [Bat67]. The inherent nonlinearity of these equations makes analytical solutions extremely challenging to derive for general situations, and in many cases the only way to mathematically

study and reproduce fluid behaviour is to make use of numerical methods.

Much of the work in this thesis, and in computer animation in general, extends a few fundamental numerical techniques for incompressible fluid simulation introduced decades ago. Pressure projection (or fractional step) methods, introduced by Chorin [Cho68], are among the most popular general techniques for solving the Navier-Stokes equations. Such methods split the Navier-Stokes equations into multiple sequential steps in order to simplify the computations. Typically a first step applies advection and viscosity terms, along with any external forces such as gravity; this may yield a velocity field with non-zero divergence (ie. featuring sources and sinks, and failing to preserve volume). A second step, referred to as the pressure projection, projects this velocity field back into the space of divergence-free (or incompressible) velocity fields. The projection operation is done by solving for a pressure field whose gradient constitutes the divergent component, and then subtracting it out. This computation takes the form of a Poisson equation, an extremely well-studied problem in its own right. (Note that since gravity forces are incorporated into the velocity field in the first step, the computed pressure will therefore include the hydrostatic pressure which counteracts gravity.) Chorin's original approach provides only first order accuracy in time; follow-up work has sought to improve on this despite difficulties in determining appropriate boundary conditions for intermediate steps, as reviewed by Guermond et al. [GMS06]. The computer graphics literature primarily uses the first order scheme, and my work follows suit.

Perhaps the most obvious discretization of the relevant Poisson problem on a regular grid would be to co-locate velocity and pressure variables at grid nodes, and apply second order, centred finite differences to approximate the divergence and gradient operators. Unfortunately, neighbouring pressure samples do not interact under this discretization, giving rise to a null space in the equations and

checkerboard-like artifacts in the pressure field. Harlow & Welch instead suggested placing pressures at cell centres and distributing velocity components to lie across appropriate cell faces [HW65]. This staggering approach allows shorter centred differencing and elegantly eliminates the spurious null space. The method was presented in the context of a two dimensional viscous incompressible flow solver that used passive marker particles to track the presence of the liquid as it migrates between computational grid cells. This Marker-And-Cell (MAC) method has had a tremendous impact on the field of incompressible flow simulation; variants remain in use today, both in industry and academic research, and it underpins the majority of recent research in animation of fluids. McKee et al. [MTF+08] provide an overview, including discussion of its history, impact, and applications.

1.1.2 An Overview of 3D Fluid Animation

The first researchers in graphics to solve the full 3D Navier-Stokes equations for liquids were Foster & Metaxas, with an Eulerian finite difference method based heavily on the traditional MAC scheme of Harlow & Welch. They applied this framework to liquids and gases and presented some simple control techniques useful for graphics scenarios [FM96, FM97b, FM97a]. However, the method suffered from dramatic computation times, primarily due to the use of explicit integration and the associated small time steps. Stam addressed this with an implicit integration technique for viscosity and a semi-Lagrangian advection scheme, ensuring unconditional stability for arbitrary time step sizes [Sta99]. This advection method, already popular in atmospheric sciences [SC91], quickly saw near-universal adoption within graphics, despite the fact that the averaging behaviour of the interpolation step of advection leads to substantial artificial damping of vorticity. Fedkiw et al. were the first to tackle this particular issue, proposing monotonic higher order

interpolation to reduce damping and a "vorticity confinement" method to amplify small vortices [FSJ01]. They also added semi-Lagrangian advection of temperature and density to efficiently simulate the specific behaviour of smoke.

Returning to liquid animation, Fedkiw and collaborators combined the benefits of Lagrangian particles and Eulerian level set methods to generate some of the earliest near-photorealistic animations of splashing water [FF01, EMF02]. They also introduced improvements for moving boundaries and a velocity extrapolation technique to provide extension velocities near the evolving surface. Carlson et al. extended the implicit viscosity integration method used by Stam to three dimensional liquids with high degrees of viscosity, as well as adding spatially varying viscosity and temperature-dependent melting effects [CMVT02].

Another important application area for fluid techniques is the simulation of fire and explosions. Nguyen et al. proposed a simple model of fuel tracking and burning, and included expansion terms across the flame front to achieve physically plausible results [NFJ02]. Yngve et al. simulated explosions using a compressible flow model including shock waves [YOH00], but the time step restriction required for stability made the method relatively impractical for most graphical situations. Feldman et al. therefore dispensed with shock waves and instead used an incompressible flow model for explosions, locally enforcing non-zero divergence regions in the projection step to approximate explosive volume change [FOA03].

These methods for liquids, smoke, fire, and explosions comprise the core fluid animation tool set that researchers have since built upon to develop many of the more complex extensions noted earlier. The book by Bridson provides a good practical overview of the basic methods and their implementation [Bri08a], along with some more recent advances. It is however worth mentioning that there are two fundamentally different approaches that have also made an impact, but which this thesis will not address. The first is purely Lagrangian particle-based models; computer

graphics has a relatively long history of using particle methods [Ree83], making approaches like smoothed particle hydrodynamics an attractive choice [MCG03]. The second is methods based primarily on vorticity [AN05, ETK+07], rather than the primitive variables, pressure and velocity. Essentially, a solution is sought in a fundamentally divergence-free space, rather than temporarily allowing drift and repeatedly projecting back into that space. Both particle methods and vorticity methods have also influenced (and been hybridized with) more typical Eulerian, pressure-projection schemes (eg. [ZB05, LTKF08, SRF05]).

1.1.3 Embedded Boundary Methods

I will use the term embedded boundary method to refer to any approach that uses one discretization to compute motion or deformation, and a second, typically higher resolution discretization, to represent the physical geometry. In many computer graphics applications, the geometry is embedded into the deforming lower resolution simulation through simple interpolation. This yields results that have more visual detail, without incurring the cost that would be required to discretize the true geometry at the higher resolution. The prototypical example of such a method is free-form deformation, in which some warp or deformation is applied to the ambient space, and the visible geometry warps accordingly [SP86]. These ideas have similarly been extended to scenarios where the underlying deformation is dictated by a physical simulation, while the simulation may still be unaware of the geometry embedded in it [FPT97, CGC+02]. The basic marker-and-cell method can also be viewed in this light: the passive marker particles dictate which grid cells are active, but their sub-grid position information is not propagated back into the fluid velocities.

Embedded boundary methods in computational physics also embed geometry

into a non-conforming simulation mesh, but more often attempt to feed this sub-grid information into the simulation to achieve more accurate results. A very early example is the Shortley-Weller discretization of the Laplace equation with irregular boundaries [SW38]. While they used a regular grid to solve the equations, the knowledge of the sub-grid positions at which grid lines intersect boundaries is used to retain second order accuracy, much like in recent ghost fluid methods for the Poisson equation (eg. [GFCK02]). There are a host of techniques that fall into this category, including immersed boundary methods [Pes02], cut cell methods [SBCL06], immersed interface methods [LL94], ghost fluid methods [FAMO99], and many that defy straightforward classification. A fairly recent review of these methods is provided by Mittal and Iaccarino [MI05].

Some recent graphics methods have also begun to exploit sub-grid information in the design of embedded boundary methods, because in many scenarios achieving greater realism fundamentally requires the use of more accurate methods. For example, in their method for large viscoplastic deformation, Wojtan et al. calculated exact partial masses embedded within each finite element, since erroneous centre of mass positions can yield obvious visible errors in the motion of small fragments [WT08]. My thesis extends this trend, seeking embedded boundary methods that improve accuracy, which in turn enhances visual fidelity.

1.1.4 Variational Principles and Natural Boundary Conditions

Another of the themes that recur throughout this work is the use of variational principles. A variational principle is an expression of the solution of a physical or mathematical problem in terms of the stationary point of a given functional over a space of possible functions. Variational principles have played a fundamental role both in the development of our understanding of physics and in our approaches to

numerical simulation of physics. For example, in classical mechanics the Euler-Lagrange equations express the traditional laws of motion in terms of a particular energy functional of a physical system (see eg. [Bri08b]). This idea has also been used to develop numerical time integration schemes with certain conservative properties, through the development of a set of Discrete Euler-Lagrange (DEL) equations. These variational integrators have also been proposed for use in graphics, because they guarantee certain physical invariants that can be useful in ensuring plausible physical motion [KYT$^+$06].

An example familiar to most computer graphics practitioners is cubic spline interpolation. Cubic splines model a smooth curve by finding a piecewise cubic polynomial that interpolates a given set of input points. Splines originally arose in the context of drafting in which an actual piece of flexible wood was used to create a smooth curve [Mal77]. Mathematically, the cubic spline is often interpreted as the curve that minimizes the integral of the squared second derivative over all possible interpolating curves (which is closely related to its curvature). If no specific additional conditions are assigned to this variational problem, it will yield the curve with zero curvature at the endpoints. This is referred to as a free or natural boundary condition, and in many common situations this "natural" behaviour is the result one is interested in. Similarly, a key motivation for my use of variational principles to re-express fluid problems is that the natural boundary conditions turn out to have useful physical meanings for the problems I address.

Variational principles are also fundamental to the modern finite element method (eg. [BS02]). In this method, a given partial differential equation is typically multiplied by a particular test function, and integrated over the problem domain to arrive at a variational problem. The stationary point of this problem will be the solution to the original PDE, restricted to the space of test functions used. This is known as the weak form of the problem, since loosely speaking the PDE is enforced on

average and with respect to the space of test functions, rather than point-wise. A finite dimensional space can then be used to discretize the weak form, allowing one to find a particular solution. This weak form will also possess certain natural boundary conditions that need not be enforced directly, in contrast to "essential" boundary conditions that must be built into the solution space. Though I exploit finite difference rather than finite element methods, the early chapters of the thesis will illustrate that natural boundary conditions can still be exploited for their convenience and simplicity in handling otherwise challenging boundaries.

There are also a number of finite difference methods that rely on variational or integral forms as well. In particular, the ideas I propose share a number of common threads with work on support-operator methods and mimetic finite difference methods, by authors such as Shashkov, Hyman, Lipnikov, Morel, Steinberg and others. The essential idea underpinning the support operator method is to express the problem as a first order system, then discretize either the gradient or divergence operator (called the "prime" operator), and finally use an integration by parts identity to determine the corresponding "derived" operator. This ensures that the discrete divergence and gradient operators retain many of the properties of their continuous counterparts, including being the negative adjoints of each other. As with many of the schemes in this thesis, this approach also ensures positive-definiteness of the resulting linear systems [SS95]. These methods have been applied to highly irregular Cartesian, polygonal, and non-conforming meshes, to problems with rapidly varying coefficients, and to Stokes flow (eg. [GL04, SS96, BGLM09]). Furthermore, Hyman and Shashkov have specifically considered the issue of boundary conditions for elliptic problems, including showing how natural boundary conditions can be exploited [HS98]. However, my work differs in a key way from these schemes; all mimetic methods of which I am aware use strictly conforming grids or meshes to handle complex boundaries, whereas I rely on embedded boundary

methods. In practice, this difference amounts to choosing an alternate definition of the inner product that can account for partial areas of cells intersected by physical boundaries. This interesting connection may also be useful in future work for making more concrete statements about convergence and accuracy.

1.1.5 Unstructured Mesh Methods

Because regular grids ease the computation of finite difference derivative estimates and are comparatively efficient and simple to work with, many fluid simulation techniques, including several in this thesis, have been developed specifically for uniform Cartesian grids. However, it has long been recognized that meshes possessing less rigid structure can also provide important benefits. These include triangle and tetrahedral meshes, irregular quadrilateral and hexahedral meshes, and even more general polygonal and polyhedral meshes. The simulation mesh plays a key role in determining the accuracy of any method that uses it, and the flexibility inherent in unstructured meshes makes them better able to exploit this fact. In practice accessing these benefits can be rather more difficult than it sounds, and the challenges posed in constructing high quality meshes under the variety of constraints that arise in practice has given rise to an entire field of research, known as mesh generation.

As an example of how mesh characteristics can play a role, adaptively refined meshes place many more elements in regions where greater accuracy or resolution are required [BO83]. Similarly, anisotropic meshes feature elements that may be stretched or compressed to align with characteristics of the problem at hand. Although long skinny triangles often imply a poor quality mesh, in the context of a well-designed anisotropic mesh they can appreciably reduce the error with which a function or partial differential equation may be approximated [LS03]. Unstruc-

tured meshes can also be designed to conform to physical geometry, so that mesh faces coincide with boundaries or internal interfaces, thereby drastically simplifying the enforcement of certain boundary conditions (eg. [FOK05]). However, conforming meshes also require more work on the part of the mesh generator to provide this convenience, and even modern conforming mesh generators can require tens of minutes for relatively small meshes [ACSYD05, TWAD09]. This suggests a tradeoff between the complexity of generating appropriate high quality meshes and the complexity of designing a numerical method for a particular problem. While the focus of this thesis is not on mesh generation itself, the use of embedded boundary methods has clear ramifications for the meshes that can be used. In the early chapters of the thesis it is shown how they allow simple grids to be used, sidestepping meshing entirely. The later chapters illustrate that embedded boundaries have benefits even for unstructured mesh simulation: they reduce the difficulty of mesh generation, allowing the use of either simple unmodified lattices [BTMF05, LS07, WT08] or a basic Delaunay triangulation (eg. [Si06]).

1.2 Thesis Contributions

Conceptually, the thesis is divided into two main topics. First, I consider variational (or energy minimization) approaches to deriving finite difference methods for simulations of viscous, incompressible fluids (chapters 2 through 4). This yields novel embedded boundary methods that are consistent in many ways with existing discretizations, but which drastically simplify the handling of complex boundary conditions that arise along liquid surfaces and dynamic solid boundaries. The latter chapters develop techniques to apply embedded boundary methods in combination with unstructured and semi-structured tetrahedral meshes (chapters 5 and 6). In practice this enables more flexible adaptivity without loss of accuracy, improved

realism in simulation of thin liquid structures, a more stable and accurate model for surface tension, and greatly simplified mesh generation. I will now outline the purpose and specific contributions of each chapter.

A Fast Variational Framework for Accurate Solid-Fluid Coupling: Chapter 2 introduces a variational expression of the classic pressure projection problem which enforces incompressibility in fluid flow. This novel formulation greatly simplifies the handling of complex irregular solid boundaries on Cartesian grids through the addition of simple weighting terms, and without the need for elaborate special-case code. By minimizing the integral of kinetic energy of the updated fluid velocities with respect to pressure over the fluid region, solid boundaries are enforced automatically through the natural boundary conditions of the minimization problem. This eliminates the traditional "stairstep" artifacts of previous approaches, while still yielding a symmetric positive-definite linear system with a stencil similar to existing schemes. The method is easily extended to support two-way solid coupling, simply by adding the kinetic energy of the rigid body itself to the minimization problem. Furthermore, a new inequality boundary condition is introduced which prevents liquid from flowing into solid objects while allowing it to separate freely, rather than artificially adhering to surfaces. The paper was presented at the 2007 SIGGRAPH conference on Computer Graphics and Interactive Techniques, and published in the journal ACM Transactions on Graphics [BBB07].

Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids: Chapter 3 takes the idea of using variational principles to enforce difficult irregular boundary conditions, and applies it to the problem of accurately simulating viscous flows with free surfaces on Cartesian grids. Free surface boundary conditions are particularly challenging in this scenario because they imply a traction constraint that relates the fluid stresses to the surface normal. Previous approaches to this problem artificially eliminate rotational motion, and either yield non-symmetric

linear systems or require semi-implicit splitting to incorporate spatially varying viscosity. The minimization form introduced by this work is fully implicit and unconditionally stable, handles rotation and variable viscosity, and generates a symmetric positive-definite linear system. Moreover, it enables simulation of the fascinating buckling and coiling motions that arise when pouring highly viscous fluids like honey or syrup. This paper was presented at the 2008 Symposium on Computer Animation [BB08].

A Variational Finite Difference Method for Time-Dependent Stokes Flow on Irregular Domains: Chapter 4 builds on the previous two chapters to develop a unified variational method for Stokes flows that supports both free surface and solid boundaries simultaneously. The Stokes equations model slow flows for which advective terms are negligible in comparison to viscous forces; this is a reasonable model for a number of common flows, and can also be a key component in algorithms for simulating the full Navier-Stokes equations. This model resolves viscosity and pressure forces simultaneously, making it a combination of the above problems where each was considered in isolation. The free surface conditions are further complicated in this setting because the zero traction constraint tightly couples pressure and viscous shear forces together. The method I developed relates pressure, deviatoric stress, and velocities within a single optimization problem that yields a symmetric positive-definite linear system, while handling arbitrary, non-conforming physical boundaries in two and three dimensions. I provide a range of analytical experiments that demonstrate first order convergence in velocity, as well as a practical application to viscous jet buckling. This work is currently in submission to the Journal of Computational Physics [BB10].

Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids: In Chapter 5 I explore the combination of embedded boundary methods with unstructured and semi-structured Delaunay tetrahedral meshes for adap-

tive fluid simulation. While tetrahedral methods are useful for adaptive discretizations of the pressure projection problem, previous approaches suffer from expensive mesh generation, poor quality meshes, and consistency errors that cause still and slow-moving fluid to behave unnaturally. By adapting existing Cartesian grid embedded boundary methods for free surfaces and solid boundaries to unstructured tetrahedral meshes, we eliminate the need for the mesh to conform to boundaries. This in turn greatly simplifies the task of mesh generation, so that we can accelerate remeshing, allow adaptivity even across domain boundaries, and ensure the meshes are of high quality and possess the Delaunay property. Furthermore, since small movements of physical boundaries no longer require a completely new mesh, we can exploit temporal coherence and reuse meshes to further save on meshing costs. This work was presented at the 2010 Eurographics conference, and published in the journal Computer Graphics Forum [BXH10].

Matching Fluid Simulation Elements to Surface Geometry and Topology: Chapter 6 uses the same embedded boundary ideas as the previous chapter, except that they are applied to simulation on a Voronoi mesh, rather than its dual Delaunay tetrahedral mesh. When using tetrahedral meshes, pressure samples are placed at tetrahedral circumcentres, whose position we have no direct control over. In contrast, when using the Voronoi mesh the pressures are located at Voronoi sites, whose placement we can choose explicitly in order to better capture the geometric features of the liquid being simulated. By combining an intelligent, geometry-aware pressure sampling method with recent triangle-mesh-based Lagrangian surface tracking methods, we can convincingly simulate thin sheets and droplets with far fewer sample points than would be required with existing methods. This approach also eliminates surface noise that usually accumulates when the resolution of the simulation mesh fails to match that of the surface mesh. Furthermore, we develop a novel method for applying surface tension, by estimating curvatures di-

rectly from the surface triangle mesh and applying the resulting forces sharply at the interface using the ghost fluid method. Detailed capillary waves can be animated in this fashion with improved stability and greatly reduced damping. Lastly, this chapter introduces a modified approach to velocity interpolation on Delaunay meshes that maintains the simplicity and efficiency of standard barycentric interpolation on tetrahedra while producing results that are qualitatively consistent with more complex, generalized barycentric interpolation over convex polyhedra. This research was presented at the 2010 SIGGRAPH conference, and published in ACM Transactions on Graphics [BBB10].

Bibliography

[ACSYD05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):617–625, 2005.

[AN05] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *Symposium on Computer Animation*, pages 87 – 96, 2005.

[Bat67] G. K. Batchelor. *An Introduction to Fluid Dynamics*. 1967.

[BB08] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*, pages 219–228, 2008.

[BB10] Christopher Batty and Robert Bridson. A variational finite difference method for time-dependent Stokes flow on irregular domains. *In submission*, 2010.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):100, 2007.

[BBB10] Tyson Brochu, Christopher Batty, and Robert Bridson. Matching

fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH)*, 29(3), 2010.

[BGLM09] L. Beirao Da Veiga, V. Gyryra, K. Lipnikov, and G. Manzini. Mimetic finite difference method for the Stokes problem on polygonal meshes. *Journal of Computational Physics*, 228(19):7215–7232, 2009.

[BHN07] Robert Bridson, Jim Hourihan, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):46, 2007.

[BO83] Marsha J. Berger and Joseph E. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations, 1983.

[BOGS06] Adam W. Bargteil, James F. O'Brien, Tolga G. Goktekin, and John A. Strain. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1):19–38, January 2006.

[Bri08a] Robert Bridson. *Fluid Simulation for Computer Graphics*. 2008.

[Bri08b] Alain J. Brizard. *An Introduction to Lagrangian Mechanics*. World Scientific, 2008.

[BS02] Susan C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, 2nd edition, 2002.

[BTMF05] Robert Bridson, Joseph Teran, Neil Molino, and Ronald Fedkiw. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers*, 21(1):2–18, November 2005.

[BXH10] Christopher Batty, Stefan Xenos, and Ben Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics)*, 29(2):695–704, 2010.

[CGC+02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):586–593, 2002.

[Cho68] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[CMT04] Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):377–384, August 2004.

[CMVT02] Mark Carlson, Peter J. Mucha, R. Van Horn, and Greg Turk. Melting and flowing. In *Symposium on Computer Animation*, pages 167–174, 2002.

[EMF02] Doug Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):736–744, 2002.

[ETK+07] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[FAMO99] Ronald Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comp. Phys.*, 152(2):457–492, 1999.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH*, pages 23–30. ACM Press, 2001.

[FL04] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):441–448, 2004.

[FM96] Nick Foster and Dimitris Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[FM97a] Nick Foster and Dimitris Metaxas. Controlling fluid animation. In *Computer Graphics International*, page 178, 1997.

[FM97b] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH*, pages 181–188, 1997.

[FOA03] Bryan E. Feldman, James F. O'Brien, and Okan Arikan. Animating suspended particle explosions. *ACM Trans. Graph. (SIGGRAPH)*, 22(3):708–715, 2003.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):904–909, July 2005.

[FPT97] Petros Faloutsos, Michiel Van De Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE TVCG*, 3(3):201–214, 1997.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH*, pages 15–22, 2001.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A

method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):463–468, August 2004.

[GFCK02] Frédéric Gibou, Ron Fedkiw, L.-T. Cheng, and Myungjoo Kang. A second order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comp. Phys.*, 176(1):205–227, 2002.

[GL04] V. Gyryra and K. Lipnikov. Mimetic finite difference methods for diffusion equations on non-orthogonal non-conformal meshes. *Journal of Computational Physics*, 199(2):589–597, 2004.

[GMS06] J. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6011–6045, September 2006.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):973–981, 2005.

[HK05] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):915–920, July 2005.

[HS98] J. M. Hyman and M. Shashkov. Approximation of boundary conditions for mimetic finite-difference methods. *Computes Math. Applic.*, 36(5):79–99, 1998.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and

James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):820–825, 2006.

[KTJG08] Theodore Kim, Nils Thuerey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):50, 2008.

[KYT+06] L. Kharevych, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E. Marsden, Peter Schröder, and Mathieu Desbrun. Geometric, variational integrators for computer animation. In *Symposium on Computer Animation*, pages 43–51, 2006.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):457–462, August 2004.

[LL94] Randall J. LeVeque and Zhilin Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31(4):1019 – 1044, 1994.

[LS03] François Labelle and Jonathan Richard Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Annual Symposium on Computational Geometry*, pages 191–200, 2003.

[LS07] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):57, 2007.

[LSSF06] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw.

Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):812–819, 2006.

[LTKF08] Frank Losasso, Jerry Talton, Nipun Kwatra, and Ronald Fedkiw. Two-way coupled SPH and particle level set fluid simulation. *IEEE TVCG*, 14(4):797–804, 2008.

[Mal77] Michael A. Malcolm. On the computation of nonlinear spline functions. *SIAM J. Numer. Anal.*, 14(2):254 – 282, 1977.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation*, pages 154–159, 2003.

[MI05] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annual review of fluid mechanics*, 37:239–261, 2005.

[MTF+08] Sean McKee, Murilo F. Tomé, Valdemir G. Ferreira, José A. Cuminato, Antonio Castelo, F. S. de Sousa, and Norberto Mangiavacchi. The MAC method. *Computers & Fluids*, 37(8):907–930, September 2008.

[NFJ02] Duc Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):721–728, 2002.

[NSCL08] Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph. (SIGGRAPH Asia)*, 27(5):166, 2008.

[Pes02] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[Ree83] W. T. Reeves. Particle systemsa technique for modeling a class of fuzzy objects. In *SIGGRAPH*, volume 2, pages 359–375, 1983.

[SB08] Hagit Schechter and Robert Bridson. Evolving sub-grid turbulence for smoke animation. In *Symposium on Computer Animation*, pages 1–7, 2008.

[SBCL06] Peter Schwartz, Michael Barad, Phillip Colella, and Terry Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comp. Phys.*, 211(2):531–550, 2006.

[SC91] Andrew Staniforth and Jean Cote. Semi-Lagrangian integration schemes for atmospheric modelsa review. *Monthly Weather Review*, 119(9):2206–2223, 1991.

[SFK+08] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable MacCormack method. *SIAM J. Sci.Comput.*, 35(2-3):350–371, 2008.

[SGTL08] Jason Sewall, Nico Galoppo, Georgi Tsankov, and Ming C. Lin. Visual simulation of shockwaves. In *Symposium on Computer Animation*, pages 126–138, 2008.

[Si06] Hang Si. *TetGen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator*, 2006.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH*, volume 20, pages 151–160, 1986.

[SRF05] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):910–914, 2005.

[SS95] M. Shashkov and S. Steinberg. Support-operator finite-difference algorithms for general elliptic problems. *Journal of Computational Physics*, 118(1):131–151, 1995.

[SS96] M. Shashkov and S. Steinberg. Solving equations with rough coefficients on rough grids. *Journal of Computational Physics*, 129(2):383–405, 1996.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999.

[SW38] G. H. Shortley and R. Weller. The numerical solution of Laplace's equation. *Journal of Applied Physics*, 9(5):334, 1938.

[SY05] Lin Shi and Yizhou Yu. Taming liquids for rapidly changing targets. In *Symposium on Computer Animation*, pages 229–236, 2005.

[TKPR06] Nils Thuerey, Richard Keiser, Mark Pauly, and Ulrich Rüde. Detail-preserving fluid control. In *Symposium on Computer Animation*, pages 7–12, 2006.

[TWAD09] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):75, 2009.

[WT08] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):47, 2008.

[WTGT09] Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):76, 2009.

[YOH00] Gary D. Yngve, James F. O'Brien, and Jessica K. Hodgins. Animating explosions. In *SIGGRAPH*, pages 29–36, 2000.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):965–972, July 2005.

[ZYP06] Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. Simulation of bubbles. In *Symposium on Computer Animation*, pages 325–333, 2006.

Chapter 2

A Fast Variational Framework for Accurate Solid-Fluid Coupling

2.1 Introduction

Physical simulation is an increasingly popular approach for producing animations of fluid. It holds out the promise of automatically generating detailed and physically-plausible motion for phenomena such as smoke, water and explosions, and their complex interactions with dynamic solid objects. However, current techniques encounter problems with both efficiency and robustness. In this paper, we propose a new variational framework which allows robust and accurate solid-fluid coupling on relatively coarse Cartesian grids, providing potentially orders of magnitude faster simulation.

For the purposes of this work, we focus on splitting/projection algorithms using fluid velocity and pressure as primary variables to solve the incompressible Euler

A version of this chapter has been published. Batty, C., Bertails, F. and Bridson, R. (2007) A Fast Variational Framework for Accurate Solid-Fluid Coupling, ACM Transactions on Graphics (Proc. SIGGRAPH) 26(3):100

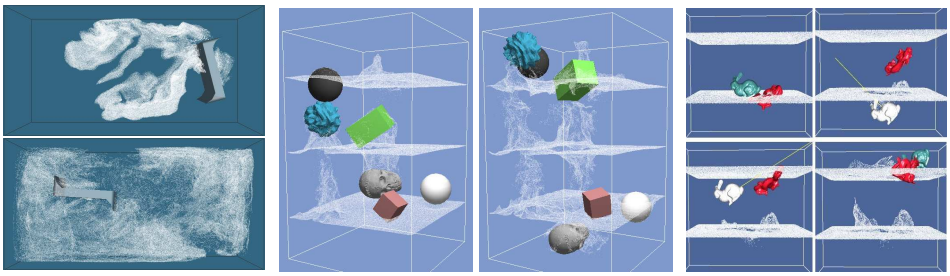


Figure 2.1: Left: A solid stirring smoke runs at interactive rates, two orders of magnitude faster than previously. Middle: Fully coupled rigid bodies of widely varying density, with flow visualized by marker particles. Right: Interactive manipulation of immersed rigid bodies.

equations:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} + \frac{1}{\rho}\boldsymbol{\nabla} p = \mathbf{f} \\ \boldsymbol{\nabla} \cdot \mathbf{u} = 0 \end{cases}$$

We use the standard notation ($\mathbf{u}$ for fluid velocity, p for pressure, ρ for density, $\mathbf{f}$ for acceleration due to body forces such as gravity, etc.); Bridson et al.'s course notes [MBG06] provide further background. The essential steps in such an algorithm are advection, corresponding to updating the positions of fluid elements (analogous to updating positions in a solid simulation), and projection, corresponding to solving for pressure to make the velocity field incompressible (analogous to updating velocities from forces in a solid simulation). We use the near-zero-dissipation FLIP method of Zhu and Bridson [ZB05] for advection but do not use vorticity confinement [FSJ01] in our examples. The contributions of this paper are concerned with the pressure projection step.

We do note there are several compelling alternatives to this class of methods, such as vorticity approaches (e.g. [ANSN06]), Smoothed Particle Hydrodynamics (e.g. [KAG+05]) and the Lattice Boltzmann Method (e.g. [TIR06]), which are beyond the scope of this paper.

2.1.1 Previous Work

Solid Wall Boundary Conditions

The dominant approach has been to discretize the fluid equations on a regular Cartesian grid, using the staggered "MAC-grid" arrangement of pressure and velocity unknowns [HW65]. Foster and Metaxas [FM96] pioneered its use in computer graphics, implementing solid boundary conditions by voxelizing obstacles onto the grid. This method, which we shall refer to as the "voxelized pressure solve", works very well if all object boundaries happen to be aligned with the grid, but otherwise introduces significant stair-step artifacts which unfortunately do not converge to zero as grid resolution is increased. In figure 2.2, observe that the streamlines of the flow match the voxelized version, not the original geometry; for comparison we show that our method matches the correct streamlines regardless of orientation with respect to the grid. These artifacts are particularly objectionable in water simulations, where water will pool on the steps rather than freely flowing down a slope. However, even in smoke simulations they are noticeable as undue numerical viscosity: all components of velocity (including those tangential to the true surface) are driven to zero. The best that can be done is to excessively increase grid resolution until the affected grid cells are small enough to be visually negligible.

Foster and Fedkiw [FF01] attempted to mitigate these problems by using an accurate normal to enforce the solid wall velocity boundary condition (allowing the fluid to slip past tangentially), and then constraining the voxelized pressure solve not to touch these velocities. Houston et al. [HBW03] later introduced the constrained velocity extrapolation method to further improve on this, extrapolating the tangential component of fluid velocity into solids accurately and thus eliminating grid artifacts from fluid advection. Rasmussen et al. [REN+04] further elaborated

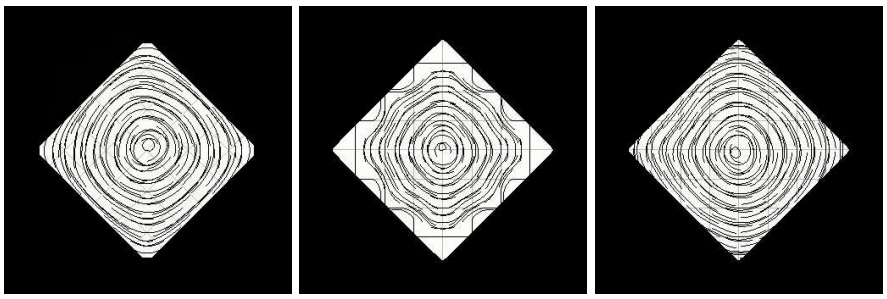


Figure 2.2: Streamlines of a vortex-in-a-box. From left to right: grid-aligned geometry ground truth, voxelized pressure solve, variational pressure solve. Note the stair-step artifacts in the voxelized solve, eliminated with the variational method.

the approach for level set advection. Unfortunately, all these methods suffer from artifacts due to the voxelized pressure solve, which is highly visible on coarse grids, necessitating high resolutions. Furthermore, though the approach works well for highly dynamic splashing scenarios, it fails in the simple hydrostatic case of fluid sitting still in a container: unphysical currents develop and unstably blow up, since the pressure cannot cancel the tangential component of force due to gravity on oblique boundaries.

Here we take an aside to discuss grid resolution. To be able to simulate a flow with features such as vortices as small as some length h, grid cells must be no larger than h. Since convergence to a quantitatively accurate solution is generally irrelevant to animation, it would be ideal to have grid spacing in fact equal to h, much coarser than typically used in scientific simulations. Making the most of coarse grids is particularly important since memory scales as $O(n^3)$, for an $n \times n \times n$ grid, and the time required for simulation can scale as badly as $O(n^5)$ if the typical time step restriction $\Delta t \sim 1/n$ is taken and the typical Incomplete Cholesky Level 0 Preconditioned Conjugate Gradient algorithm is used for solving linear systems. Increasing grid resolution just to account for the failure of an algorithm to faithfully reflect the physics at the appropriate resolution is clearly a very steep price to pay.

Octree methods [LGF04] allow one to use high resolution only where needed, partially overcoming the efficiency problem of the voxelized approach. However, the node-averaging used for octrees gives results inferior to the MAC grid at the same resolution [IGLF06] requiring even finer resolution (and more stringent restriction on time step); the considerable computational overhead of the pointer structure compared to a regular grid reduces the potential benefit further.

Another promising approach is introduced by Feldman et al. [FOK05] where unstructured tetrahedral meshes, which can align with arbitrary geometry, are used in lieu of grids. These have the side benefit of easy adaptivity, and recent advances in mesh generation mean only a fraction of the simulation time is spent on making meshes even if rebuilt nearly from scratch each time step [KFCO06]. However, matching fine-scale geometry or porous objects requires an impractically large mesh, and the extensive averaging used in interpolation introduces more numerical dissipation, demanding higher resolution meshes than comparable grid simulations. Furthermore, the overhead of unstructured meshes compared to regular grids of similar resolution is considerable.

Roble et al. [RbZF05] take an intermediate approach, using an underlying regular grid, but modifying only boundary cells to align with objects. A growing body of similar work exists within the computational fluid dynamics community [JC98, UMRK01, KAK03, MKLU05, KLMU05, SBCL06]. While both finite volume and finite difference techniques are represented in this sampling, a common implementation difficulty is the complexity of robustly handling the fully three-dimensional geometry and/or stencils required to capture the non-grid-aligned Neumann boundary condition.

Solid-Fluid Coupling

Takahashi et al. [THK02] presented a simple method for coupling fluids and rigid bodies, with the same problematic voxelization as before, where the rigid bodies provide velocity boundary conditions to the fluid and the resulting pressure subsequently provides a net force and torque on the rigid bodies. We note that in certain situations, such as a rigid stopper resting on fluid in a closed tube, the alternating nature of the coupling results in failure: the fluid may be constrained to compress by rigid body velocities, giving an inconsistent linear system for pressure.

Génevaux et al. [GHD03] introduced coupling between a free surface fluid simulation using marker particles, and elastic solid simulation using masses and springs. The coupling is achieved by attaching the solid with *ad hoc* damped springs to nearby fluid marker particles (averaging the force down onto the grid for the fluid simulator), then using the voxelized pressure solve.

Carlson et al. [CMT04] simulated coupled fluid and rigid bodies with Distributed Lagrange Multipliers, conceptually considering rigid bodies as fluid on a grid, solving for pressure, then projecting velocity in those regions back to rigid motion with careful additions to the body force to account for density differences between solids and fluids. However, the method cannot stably handle light solids (less than ~ 0.45 the fluid density), and fails in some cases, allowing fluid to erroneously leak through a rigid plug supporting it.

Guendelman et al. [GSLF05] returned to the alternating voxelized approach of Takahashi et al., generalized to include octree grids, thin solids and arbitrary solid dynamics. To improve upon the noisy pressure resulting from voxelization to cell faces, a second pressure solve, doubling the expense of the calculation, is done with solid masses added to the fluid grid density in the style of the Immersed Boundary Method [Pes02]; this smoother pressure is used to calculate force on solids. Incon-

sistent linear systems arising in enclosed fluid regions are handled by projecting out the null-space components from the right-hand-side of the pressure equation; this has the unfortunate effect of allowing closed regions to change volume as the solid dictates, meaning fluid-filled balloons, for example, cannot be simulated.

Full simultaneous coupling between fluids and solids was achieved by Klingner et al. for rigid bodies [KFCO06] and by Chentanez et al. for deformable objects [CGFO06], with an approach that produces similar discretizations to ours. While focusing mostly on tetrahedral meshes that align with solid boundaries Chentanez et al. do note that this approach can be applied on regular grids, though the accompanying animation of fluid pouring on a bunny model shows apparent grid artifacts.

Of course, many scientific works in computational fluid dynamics address fluid-solid coupling. We highlight Peskin's Immersed Boundary Method [Pes02], which averages solid properties onto the fluid grid (and thus cannot stop fluid leaking through solids, for example); the ALE approach of Hirt et al. [HAC74], which requires tetrahedral meshes fully resolving all solid boundaries; and Le et al.'s use of the Immersed Interface Method to couple fluid with rigid and elastic bodies [LKP06], whose expenses each time step include a solve with an unsymmetric matrix and a Singular Value Decomposition of a matrix with size proportional to the number of points used to discretize solids.

2.1.2 Contributions

We introduce a new variational interpretation of the pressure equation for coupled fluids in section 2.2, namely that pressure minimizes the kinetic energy of the system. The pressure update and total kinetic energy may be easily discretized on a regular grid with arbitrary immersed solid geometry. Then the discrete pressure which minimizes this discrete kinetic energy is found; this reduces automatically

to a well-posed, sparse, symmetric positive semi-definite linear system. The solution is free of grid artifacts, permitting fast solution on coarse grids. It enforces simultaneous coupling correctly, handling all problem cases mentioned in the previous section, and can be applied with arbitrary solid dynamics. In addition, we automatically account for sub-grid-resolution solid geometry in our discrete estimate of kinetic energy; this gives us approximate sub-grid accuracy in coupling, allowing robust handling of fluid flowing through thin gaps that could not be efficiently resolved with octrees or tetrahedral meshes. We work out the details of our solver for rigid bodies in section 2.3.

Finally, the variational framework highlights an analogy between fluid pressure and modern treatment of inelastic contact forces between rigid bodies: they both are based on kinetic energy minimization. Inspired by this connection, we introduce a new free surface/solid boundary condition in section 2.4, expressed as an inequality constraint on our minimization. This allows fluid to freely separate from solid walls, similar to rigid bodies separating from contact, fixing some enduring artifacts seen in previous fluid simulation work where fluid unnaturally crawls along walls and ceilings.

2.2 A Variational Interpretation of Pressure

Consider a fluid and immersed solids. In continuous space variables the pressure update for fluid velocity from time n to $n+1$ is

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\nabla p \tag{2.1}$$

where $\tilde{\mathbf{u}}$ is the intermediate velocity field resulting from advection and integration of body forces such as gravity. The accompanying update to solid velocities is

$$\mathbf{V}^{n+1} = \mathbf{V}^n + \Delta t \mathbf{M}_S^{-1} \mathbf{J} p \tag{2.2}$$

where $\mathbf{V}$ is the generalized velocity of the solid (possibly a continuous field for a deformable object, or a six-dimensional vector for a single rigid body, etc.), $\mathbf{M}_S$ is the mass linear operator (convolution with solid density for a deformable object, the usual 6×6 matrix containing the inertia tensor and the total mass for each rigid body, etc.), and $\mathbf{J}$ is a linear operator converting pressure on the boundary of the solid to generalized forces.

The pressure enforces the incompressibility condition $\nabla \cdot \mathbf{u}^{n+1} = 0$ inside the fluid, and the boundary condition $\mathbf{u}^{n+1} \cdot \hat{\mathbf{n}} = \mathbf{v}^{n+1} \cdot \hat{\mathbf{n}}$ on the solid boundary, with $p = 0$ on the free surface. Here $\mathbf{v}$ is the velocity of the solid evaluated at a point on the boundary, which must be $\mathbf{v} = \mathbf{J}^* \mathbf{V}$ from basic physical principles, with $\mathbf{J}^*$ the adjoint or transpose of $\mathbf{J}$. The solid boundary condition is simply stating the fluid may flow neither in nor out of a solid. Substituting the update equations in these conditions gives the PDE form of the coupled pressure equations.

The total kinetic energy of the system is

$$\text{KE} = \iiint_{\text{fluid}} \frac{1}{2} \rho ||\mathbf{u}||^2 + \frac{1}{2} \mathbf{V}^* \mathbf{M}_S \mathbf{V} \tag{2.3}$$

where the solid term may be a double convolution integral for continua or just a finite quadratic form for rigid bodies. It is a straightforward exercise in variational calculus to show that the pressure PDE is in fact the Euler-Lagrange equation for minimizing kinetic energy with respect to pressure (see Bridson et al.'s course notes [MBG06] for a proof of the simpler case with motionless solids). Note that the ki-

netic energy is bounded below by zero and depends only quadratically on pressure, so this minimization is well-posed. Put another way, the pressure solve is computing a projection of velocities onto the space of divergence-free fluid velocities and compatible solid velocities. Projection is equivalent to finding the closest point on that space, and the metric defining "closest" is kinetic energy. This statement of the pressure problem is also in fact exactly the Lagrange Multiplier approach to constrained mechanics (e.g. [Bar96]), with pressure playing the role of Lagrange Multiplier.

Scripted or stationary solids can be incorporated by taking the limit as their mass goes to infinity, and instead of total kinetic energy using the difference in energy of the open system (excluding infinite masses) from one time step to the next. The coupling with the scripted objects reduces to calculating the work done by pressure on their boundaries, i.e. the exchange of energy between the open system and the scripted objects. Once discretized and reduced to a linear system, it is in fact easier to take the limit as mass goes to infinity, i.e. as $\mathbf{M}_S^{-1}$ goes to zero, which has the effect of just eliminating terms from the matrix.

2.2.1 Fluid Discretization

Instead of discretizing the local pressure PDE with boundary conditions, we discretize the global variational principle. This avoids directly discretizing the tricky velocity boundary condition at non-grid-aligned solid boundaries, instead relying on the easier task of estimating the kinetic energy. Moreover, it reduces to the standard PDE discretizations for grid-aligned geometry, and leads to simulation code largely the same or simpler than previous techniques.

We discretize the fluid variables on a standard MAC grid, and use the regular finite difference approximation to the gradient for the pressure update. For example,

the $x-$ component update is

$$u^{n+1}_{i+1/2,j,k} = \tilde{u}_{i+1/2,j,k} - \frac{\Delta t}{\rho}\left(\frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}\right)$$

We use the standard second order accurate ghost fluid boundary condition for pressures lying on the other side of a free surface [GFCK02].

The kinetic energy integral of the fluid decouples into the sum of the kinetic energies from the $x-$, $y-$, and $z-$ components of fluid velocity, which we approximate individually:

$$\begin{aligned}\mathrm{KE}_F \approx \sum_{i,j,k} \frac{1}{2} m_{i+1/2,j,k} u^2_{i+1/2,j,k} + \frac{1}{2}\sum_{i,j,k} m_{i,j+1/2,k} v^2_{i,j+1/2,k} \\ + \frac{1}{2}\sum_{i,j,k} m_{i,j,k+1/2} w^2_{i,j,k+1/2}\end{aligned} \tag{2.4}$$

Here the m's are estimates of the mass of the fluid in the Δx^3 cube surrounding the appropriate MAC velocity sample point: e.g. $m_{i+1/2,j,k}$ is ρ times the volume of fluid inside $[x_i, x_{i+1}] \times [y_{j-1/2}, y_{j+1/2}] \times [z_{k-1/2}, z_{k+1/2}]$. See figure 2.3 for a 2D example. These volumes can easily and efficiently be calculated exactly from a polygonal representation of the geometry, or they may be approximated by Δx times the area of the associated cell face (giving rise to a method related to Finite Volumes), or even just Δx^2 times the extent of the fluid on the line segment between pressure samples (calculated trivially from a level set representation). The problematic voxelized pressure solve corresponds to setting masses equal to $\rho \Delta x^3$ or 0, losing all sub-grid information about the boundary.

Expressing the vector of all fluid velocity components as $\mathbf{u}$ and pressures as $\mathbf{p}$, the gradient finite difference operator as matrix $\mathbf{G}$ and the diagonal matrix of all

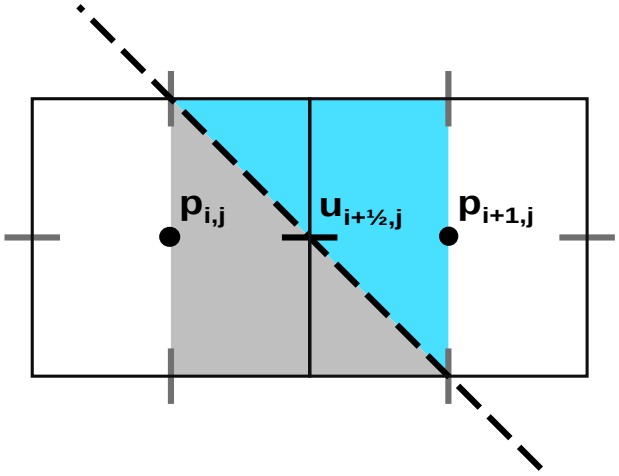


Figure 2.3: Area weights in 2D: The shaded region, in blue (fluid) and grey (solid), indicates the staggered cell surrounding velocity sample $u_{i+\frac{1}{2},j}$ on a standard MAC grid. The area used to compute the corresponding mass, $m_{i+\frac{1}{2},j}$, is that of the fluid region.

the fluid cell masses as matrix $\mathbf{M}_F$, the discrete pressure update is

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\mathbf{G}\mathbf{p} \tag{2.5}$$

and the discrete fluid's kinetic energy is

$$\begin{aligned} \mathrm{KE}_F^{n+1} &= \frac{1}{2}\mathbf{u}_{n+1}^T \mathbf{M}_F \mathbf{u}_{n+1} \\ &= \frac{1}{2}(\tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\mathbf{G}\mathbf{p})^T \mathbf{M}_F (\tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\mathbf{G}\mathbf{p}) \end{aligned}$$

We will add the terms corresponding to solids in later sections. For now notice that minimizing KE with respect to pressure is a weighted linear least-squares problem. Since the weights in $\mathbf{M}_F$ are non-negative masses, it is automatically well-posed (up to addition of pressure differences in the null-space of $\mathbf{G}$, which of course have

no influence on velocities). The normal equations are automatically a consistent, symmetric positive semi-definite linear system:

$$\frac{\Delta t}{\rho^2}\mathbf{G}^T\mathbf{M}_F\mathbf{G}\mathbf{p} = \frac{1}{\rho}\mathbf{G}^T\mathbf{M}_F\tilde{\mathbf{u}} \tag{2.6}$$

For binary voxel weights it is straightforward to see this is exactly the same as the traditional discrete pressure equation; in general we have the same 7-point-stencil sparsity structure, but with coefficients based on cell masses. For the sake of space we explicitly write only the 5-point two-dimensional version for cell i, j:

$$\begin{aligned}\frac{\Delta t}{\rho^2\Delta x^2}\begin{pmatrix}(m_{i+1/2j}+m_{i-1/2j}+m_{ij+1/2}+m_{ij-1/2})p_{ij}\\ -m_{i+1/2j}p_{i+1j}-m_{i-1/2j}p_{i-1j}\\ -m_{ij+1/2}p_{ij+1}-m_{ij-1/2}p_{ij-1}\end{pmatrix} = \\ \frac{1}{\rho\Delta x}\begin{pmatrix}-m_{i+1/2j}u_{i+1/2j}+m_{i-1/2j}u_{i-1/2j}\\ -m_{ij+1/2}v_{ij+1/2}+m_{ij-1/2}v_{ij-1/2}\end{pmatrix}\end{aligned} \tag{2.7}$$

Note that we have multiplied both sides by -1 to make the system positive semi-definite. This is still a symmetric M-matrix, and thus may be solved efficiently with Modified Incomplete Cholesky Preconditioned Conjugate Gradient using exactly the same code as a traditional voxelized solver.

Figure 2.2 shows a comparison of a 2D vortex-in-a-box, simulated with the box grid-aligned (our ground truth), the box rotated and classic voxelized weights used, and the box rotated with our new scheme. The bumpy stair-step grid artifacts of the voxelized scheme are essentially eliminated with the variational approach. We also note that unlike previous partial fixes we are guaranteed to be stable, and for the hydrostatic case the exact hydrostatic pressure $p = -\rho gy$ is the solution to our minimization, perfectly canceling out gravitational acceleration (giving $\mathbf{u} = \mathbf{0}$).

2.3 Coupling Fluids and Rigid Bodies

2.3.1 Pressure Discretization

To couple a rigid body, we just approximate the **J** operator that maps pressure to net force and torque on the body for the pressure update, and add the kinetic energy of the solid to our minimization.

In continuous variables, the translational part of the J operator is defined by the net force equation:

$$\mathbf{F} = \mathbf{J}_{trans}\mathbf{p} = -\iint_S p\hat{\mathbf{n}} \tag{2.8}$$

where S is the full surface of the solid, $\hat{\mathbf{n}}$ is the outward pointing normal (hence the negative sign), and recalling $p = 0$ on dry parts of the solid surface. From the fundamental theorem of calculus this is equivalent to a volume integral:

$$\mathbf{J}_{trans}\mathbf{p} = -\iiint_{\text{solid}} \nabla p \tag{2.9}$$

where p is conceptually smoothly extended into the volume (all interior values cancel, thus we never actually refer to pressures beyond a grid cell into the interior). This can be discretized in a manner consistent with the fluid pressure solve. For example, for the horizontal component we have

$$J_x\mathbf{p} = -\sum_{i,j,k} vol_{i+1/2,j,k}\frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x} \tag{2.10}$$

The volume weights are the volume of the rigid body occupying the cell centered on each particular MAC grid velocity sample position, exactly analogous to the mass weights used to define kinetic energy of the fluid. For fully submerged cells, we can in fact compute the fluid weights by subtracting off the solid volumes (however they are approximated) from the volume of a full cell. We note that in the

interior of the solid, where all the volumes are full, the sum telescopes and cancels out pressure unknowns in the interior.

The torque part of the J operator is likewise defined:

$$\mathbf{T} = \mathbf{J}_{rot}\mathbf{p} = -\iint_S (\mathbf{x} - \mathbf{X}_{com}) \times p\hat{\mathbf{n}} \tag{2.11}$$

where $\mathbf{X}_{com}$ is the center of mass of the object. Again we transform this into a volume integral:

$$\mathbf{J}_{rot}\mathbf{p} = \iiint_{\text{solid}} \boldsymbol{\nabla} \times p(\mathbf{x} - \mathbf{X}_{com}) \tag{2.12}$$

For each component of torque we approximate this with a sum, using volume weights, as for translation.

Note that if approximations to the volume weights are used in defining $\mathbf{J}$, the 6×6 mass matrix $\mathbf{M}_S$ used to compute the rigid body's kinetic energy from translation and rotation should ideally be consistent with those volumes, multiplied by rigid body density, rather than the exact mass matrix. However, this only becomes an issue for achieving perfect hydrostatic rest with neutrally buoyant rigid bodies, and may be ignored in more dynamic scenes.

Once $\mathbf{J}$ and $\mathbf{M}_S$ have been computed, we add the rigid body's terms to the kinetic energy minimization:

$$\text{KE}_S^{n+1} = \frac{1}{2}(\mathbf{V}^n + \Delta t \mathbf{M}_S^{-1}\mathbf{J}\mathbf{p})^T \mathbf{M}_S (\mathbf{V}^n + \Delta t \mathbf{M}_S^{-1}\mathbf{J}\mathbf{p}) \tag{2.13}$$

This modifies the linear system for pressure, equation (2.6), by adding $\Delta t \mathbf{J}^T \mathbf{M}_S^{-1}\mathbf{J}$ to the matrix and $-\mathbf{J}^T\mathbf{V}^n$ to the right-hand side. The sparsity of this addition depends on how many grid cells the solid boundary overlaps; we currently naïvely use a general sparse matrix data structure to handle it, but note that the addition is low rank (rank 6) which could be gainfully exploited by more sophisticated numerical

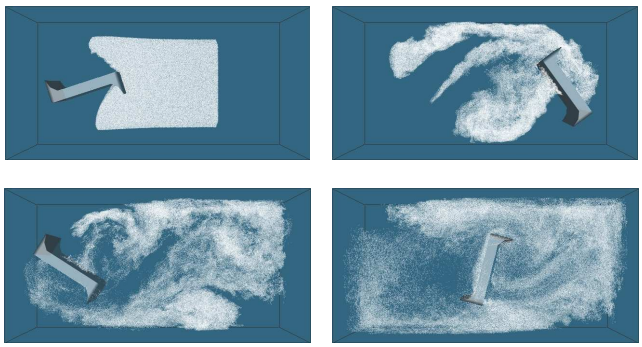


Figure 2.4: Simulation of a paddle rotating through smoke on a $80 \times 40 \times 60$ grid, running at 3 seconds per frame. Note the fine-scale turbulent vortices captured by our approach.

linear algebra.

We do highlight an assumption used in this derivation: rigid bodies are thick enough to have an interior sampled on the grid. For thin rigid bodies, shells in particular, this is violated and the above approach does not work as described. There we need some method for encoding the unknown discontinuous pressure jump from one side of the rigid body to the other. We expect, in future work, to define ghost pressures on either side of such bodies, where we use the ghost pressure rather than the real pressure on the other side. A similar approach was successfully adopted by Tam et al., who simulated fluid interaction with thin rods, albeit for high-speed compressible flows [TRS05].

To handle scripted rigid bodies (objects with prescribed motion unaffected by the fluid) we let $\mathbf{M}_S^{-1}$ drop to zero, removing that term from the matrix but keeping the contribution to the right-hand side of the linear system. Note that if a scripted motion constrains the fluid to compress (e.g. in a piston), the linear system becomes inconsistent. If the user insists on this scenario, we remove the null-space component of the right-hand side as in Guendelman et al.'s work and allow the

fluid to change volume [GSLF05].

2.3.2 Time Integration

For time integration, we use the following scheme at each time step:

- Advance fluid positions (advection) and rigid body positions/orientations independently with current velocities.
- Process collisions.
- Add Δt times body forces to all velocities.
- Solve the energy minimization problem for pressure.
- Update fluid and solid velocities with pressure.

Note that for advection the tangential fluid velocity should be extrapolated into sample points with zero fluid mass, similar to Houston et al. [HBW03]. In future work we plan to add frictional contact forces to the energy minimization problem, which extends it to a Quadratic Program (QP) with constraints; currently our simulations use the simpler rigid body algorithm of Guendelman et al. [GBF03].

2.3.3 Results

We ran our simulator on several examples comparable to previous papers, on an older 2.8 GHz Pentium 4 desktop. We begin with the paddle wheel by Klingner et al. [KFCO06]: they report simulation times of approximately one minute per frame. On a strictly larger $80 \times 40 \times 60$ grid that approximately matches their smallest tetrahedra, our code runs at 3 seconds per frame, a factor of 20 speed-up (presumably due to the overhead of the unstructured mesh). However, this grid contains more velocity samples than the tetrahedral mesh, and due to the sharper interpolation possible on a regular grid gives significantly more detailed results. If

we instead find a grid size, $40 \times 20 \times 30$, that better matches the look of the original (albeit still ending with a much higher degree of turbulent mixing from more finely resolved vortices than the original—even though we do not use vorticity confinement in our simulation) our simulator runs interactively at 5 frames per second, a factor of 300 speed-up. We observe here the critical importance of methods which can accurately capture details on coarse grids. Figure 2.1, left, shows frames from the coarse grid, and figure 2.4 from the higher resolution grid for comparison.

Figure 2.1, middle, shows a simulation of a variety of rigid bodies of differing densities in a fluid-filled container. The asteroid object (in cyan), however, has a density 0.1 times that of the fluid which Carlson et al.'s algorithm cannot stably handle [CMT04]. Our simulation on a $60 \times 90 \times 60$ grid, ran at 25 seconds per frame. Figure 2.1, right, shows an interactive simulation of 2 complex solids immersed in a fluid, running at 2 frames per second on a $20 \times 20 \times 20$ grid. From top to down and left to right: the user interactively selects the heavy blue bunny (selection in white), drags it up, and launches it so that it collides with the much lighter red dragon.

We believe that by exploiting the low rank of the rigid body additions to the matrix in the future, we will achieve further significant improvements, particularly since the most time is spent on easily optimized matrix-vector multiplies.

Finally, to illustrate more clearly the ability of our model to capture sub-grid details, figure 2.5 shows frames from a 2D animation of a heavy rigid box nearly blocking a fluid channel. The gaps on either side of the box are only half a grid cell wide, yet fluid convincingly flows past, jostling the box from side to side.

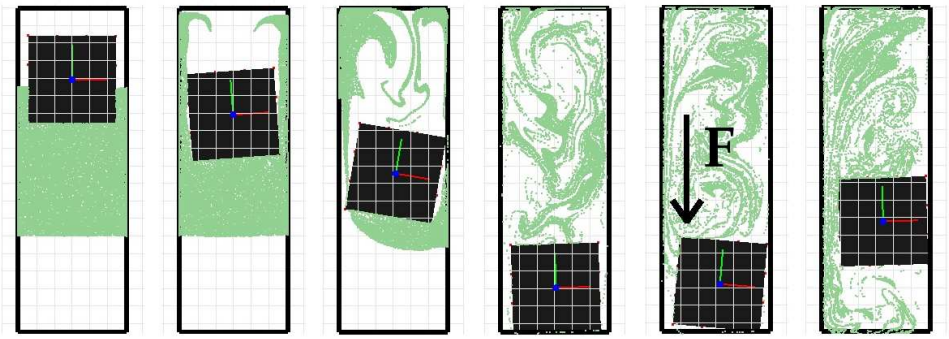


Figure 2.5: Our variational framework gives sub-grid resolution in this rigid body flow example, allowing efficient and plausible solution on a coarse grid. The box sinks in a slightly larger tube, jostled from side to side by the fluid; later an applied force **F** drives fluid in sub-grid gaps to push the box upwards. Fluid flow is visualized with marker particles.

2.4 Wall Separation

A common numerical artifact seen in free-surface water simulations is fluid "crawling" up walls and even along ceilings, eventually dripping down or crossing over to another wall to descend. The source of this problem is the $\mathbf{u} \cdot \hat{\mathbf{n}} = \mathbf{v}_{solid} \cdot \hat{\mathbf{n}}$ boundary condition, which states that fluid cannot flow into or out of a solid. While this is well established physically for many flow situations, it has the unfortunate side-effect of not allowing fluid to separate from a wall, a phenomena which is readily observed in everyday life. Without getting into the physical chemistry of molecular-scale interactions that actually govern surface wetting/drying (and that are not captured by continuum mechanics) we argue heuristically that in reality only a thin film of fluid is left on the wall in these situations. This film, at most a wet patch, is far too small to be resolved on an animation grid. Simulations using this boundary condition instead enforce a layer of thickness unrealistically proportional to the grid cell size that sticks to the wall, and rely on numerical error in

advection to eventually separate it.

Foster and Fedkiw [FF01] (with extensions by Houston et al. [HBW03] and Rasmussen et al. [REN+04]) observed this problem and offered a fix which works well in certain highly dynamic splashing situations. After advecting and applying forces, if fluid velocities are found to be separating from solids ($\tilde{\mathbf{u}} \cdot \hat{\mathbf{n}} > \mathbf{v}_{solid} \cdot \hat{\mathbf{n}}$), that separation velocity is enforced in the pressure solve. However, it becomes unstable and physically implausible in more static conditions with oblique boundaries (as mentioned in section 2.1.1), and completely fails for closed or nearly closed fluid-filled containers. We instead exploit our variational framework to arrive at a robust, physically consistent solution.

Essentially we want to enforce the boundary condition:

$$\mathbf{u} \cdot \hat{\mathbf{n}} \geq \mathbf{v}_{solid} \cdot \hat{\mathbf{n}} \tag{2.14}$$

allowing the fluid to separate from the wall but not flow into it. If it separates from the wall, it becomes a free surface, $p = 0$, but if not we argue one appropriate condition on pressure is $p > 0$: we rule out suction from keeping the fluid stuck. This is then a complementarity condition:

$$0 \leq p \perp \mathbf{u} \cdot \hat{\mathbf{n}} - \mathbf{v}_{solid} \cdot \hat{\mathbf{n}} \geq 0 \tag{2.15}$$

This is equivalent to turning our kinetic energy minimization problem into an inequality-constrained QP, with just the linear constraint $p \geq 0$ on solid boundaries: the complementarity is automatically enforced for us by the KKT conditions. Thus we can again avoid discretizing the boundary condition, relying on the discretization of the variational principle to automatically capture it.

Parenthetically, this makes the analogy between solving for pressure and solv-

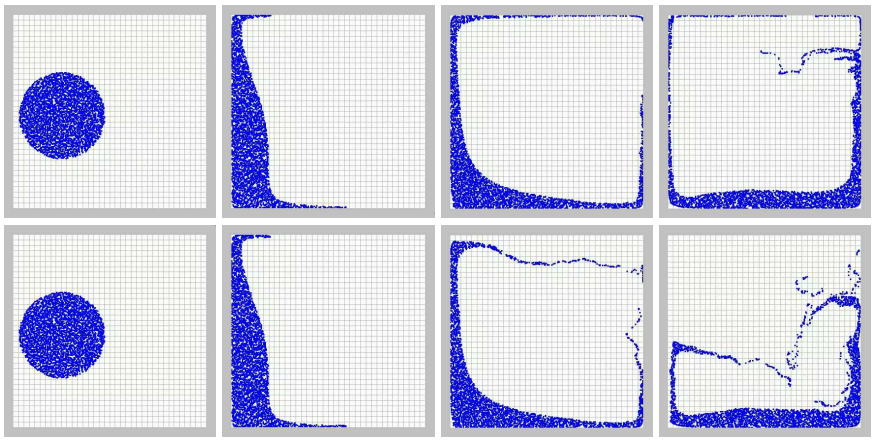


Figure 2.6: A ball of water splashes against the left wall. In the top row, the standard solid wall boundary condition is used, resulting in fluid unnaturally sticking to walls. In the bottom row, our new wall separation condition lets the fluid peel off plausibly.

ing for rigid body contact even closer. In rigid body contact, contact forces or impulses are constrained to be non-negative with a complementarity condition on relative velocity, allowing bodies to separate but not interpenetrate.

Figure 2.6 shows a 2D comparison of the standard boundary condition and our wall separation condition. We used the PATH solver [FM98] to solve the equivalent KKT Linear Complementarity Problem, whose performance limits us to relatively small problem sizes; in future work we plan to investigate more scalable QP solvers.

2.5 Conclusion

We introduced a variational framework for pressure in fluid flow, allowing easy coupling to solids with arbitrary geometry not aligned with the grid. By exploiting the demonstrated sub-grid accuracy of this approach, the desirable properties of the resulting linear system and the efficiency of Cartesian grid-based simulation,

we achieve a performance gain of one or two orders of magnitude over existing techniques, and overcome many limitations associated with previous methods. In addition we introduced a novel wall separation boundary condition, which fits naturally in the variational framework and robustly eliminates unwanted sticky artifacts which have plagued free surface simulations in the past.

Due to the conceptual simplicity of our framework, we believe that extending the coupling mechanism to arbitrary deformable bodies should be straightforward, and preliminary results indicate this to be the case. In future work we also plan to properly account for thin solids with ghost pressure values, exploit the low rank of rigid body matrix additions to improve performance, and use a more scalable QP solver to better handle frictional rigid body contacts and wall separation.

Bibliography

[ANSN06] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *Symposium on Computer Animation*, pages 25–32, 2006.

[Bar96] David Baraff. Linear-time dynamics using Lagrange multipliers. In *SIGGRAPH*, pages 137–146, 1996.

[CGFO06] Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Symposium on Computer Animation*, pages 83–89, 2006.

[CMT04] Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):377–384, August 2004.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH*, pages 23–30. ACM Press, 2001.

[FM96] Nick Foster and Dimitris Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[FM98] Michael C. Ferris and Todd S. Munson. Complementarity problems

in GAMS and the PATH solver. Technical report, Computer Sciences Dept., University of Madison, 1998.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):904–909, July 2005.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH*, pages 15–22, 2001.

[GBF03] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH)*, 22(3):871–878, 2003.

[GFCK02] Frédéric Gibou, Ron Fedkiw, L.-T. Cheng, and Myungjoo Kang. A second order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comp. Phys.*, 176(1):205–227, 2002.

[GHD03] Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. Simulating fluid-solid interaction. In *Graphics Interface*, pages 31–38, 2003.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):973–981, 2005.

[HAC74] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comp. Phys.*, 14(3):227–253, 1974.

[HBW03] Ben Houston, Chris Bond, and Mark Wiebe. A unified approach for modeling complex occlusions in fluid simulations. In *SIGGRAPH Sketches*, 2003.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[IGLF06] Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):805–811, 2006.

[JC98] Hans Johansen and Phillip Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comp. Phys.*, 147(1):60–85, November 1998.

[KAG+05] Richard Keiser, Bart Adams, Dominique Gasser, Paolo Bazzi, Philip Dutré, and Markus Gross. A unified Lagrangian approach to solid-fluid animation. In *Eurographics Symposium on Point-Based Graphics*, pages 125–148, 2005.

[KAK03] M. P. Kirkpatrick, S. W. Armfield, and J. H. Kent. A representation of curved boundaries for the solution of the Navier-Stokes equations on a staggered three-dimensional Cartesian grid. *J. Comp. Phys.*, 184(1):1–36, 2003.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):820–825, 2006.

[KLMU05] S. Krishnan, Hao Liu, S. Marella, and H. S. Udaykumar. Sharp interface Cartesian grid method II: A technique for simulating droplet in-

teractions with surfaces of arbitrary shape. *J. Comp. Phys.*, 210(1):32–54, 2005.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):457–462, August 2004.

[LKP06] D. V. Le, B. C. Khoo, and J. Peraire. An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries. *J. Comp. Phys.*, 220(1):109–138, 2006.

[MBG06] Matthias Müller, Robert Bridson, and Eran Guendelman. Fluid Simulation. In *ACM SIGGRAPH Course Notes*, 2006.

[MKLU05] S. Marella, S. Krishnan, Hao Liu, and H. S. Udaykumar. Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations. *J. Comp. Phys.*, 210(1):1–31, 2005.

[Pes02] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[RbZF05] Doug Roble, Nafees bin Zafar, and Henrik Falt. Cartesian grid fluid simulation with irregular boundary voxels. In *SIGGRAPH Sketches*, 2005.

[REN+04] Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, N. Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. Directable photorealistic liquids. In *Symposium on Computer Animation*, pages 193–202, 2004.

[SBCL06] Peter Schwartz, Michael Barad, Phillip Colella, and Terry Ligocki. A Cartesian grid embedded boundary method for the heat equation and

Poisson's equation in three dimensions. *J. Comp. Phys.*, 211(2):531–550, 2006.

[THK02] Tsunemi Takahashi, Ueki Heihachi, and Atsushi Kunimatsu. The simulation of fluid-rigid body interaction. In *SIGGRAPH Sketches*, 2002.

[TIR06] Nils Thuerey, Klaus Iglberger, and Ulrich Rüde. Free surface flows with moving and deforming objects with LBM. In *Vision, Modeling, and Visualization*, 2006.

[TRS05] D. Tam, R. Radovitzky, and R. Samtaney. An algorithm for modelling the interaction of a flexible rod with a two-dimensional high-speed flow. *International Journal for Numerical Methods in Engineering*, 64(8):1057–1077, 2005.

[UMRK01] H. S. Udaykumar, Rajat Mittal, P. Rampunggoon, and A. Khanna. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *J. Comp. Phys.*, 174(1):345–380, 2001.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):965–972, July 2005.

Chapter 3

Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids

3.1 Introduction

Viscous liquids are a common feature of the world around us. Household examples include honey, syrup, paints, cake batter, and molasses; the unique behaviour exhibited by these liquids is therefore extremely familiar to most of us. Film and games often make use of increasingly exotic examples including wet mud, tar, lava, quicksand, or goo. The distinguishing characteristic of these liquids is their resistance to shearing flow, resulting in extremely slow, damped motion that, in the interior of the fluid, is not terribly compelling to watch. However, at the interface between air and liquid a host of complex and distinctive effects can arise. When viscous fluid is poured onto a surface it will often begin to coil or fold over upon itself, generating intricate surface details. The unwieldy technical names for such

A version of this chapter has been published. Batty, C., and Bridson, R. (2008) Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids, Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation.

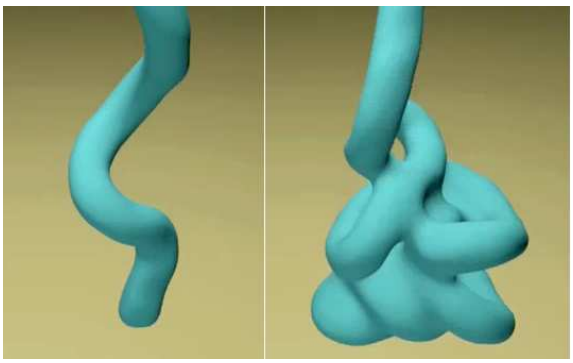


Figure 3.1: An initially straight stream of viscous fluid buckles and coils as it falls.

phenomena are cylindrical and planar viscous jet buckling, respectively; however, they can readily be understood by considering that liquid will prefer the path of least resistance. The falling fluid above and the viscous pile below apply opposing forces, but the surrounding air applies little to no resistance, causing the fluid to bend or bow out to one side. This and many more subtle behaviours are generated by the delicate coupling of air and liquid, and the resulting motion may provide important visual cues to a fluid's material properties. A recent example comes from the makers of *Bee Movie* [Rui07], who met with difficulties attempting to model honey with standard viscous fluid simulators. Although they resorted to a (non-physical) viscoelastic model, we postulate that the true root of the problem lies not in the constitutive law, but in the free surface boundary conditions. We present a new method that enforces these conditions easily and accurately for the first time, using a novel fully implicit time integration scheme. This new method allows for the efficient simulation of a variety of complex viscous liquid phenomena that were previously extremely difficult or impossible to reproduce.

3.1.1 Contributions

We now summarize our primary contributions. First, we point out that in order to achieve convincing viscous behaviour it is in fact vital to enforce the traction-free boundary conditions on the liquid free surface, which requires full coupling between the components of velocity. We then proceed to develop a fully implicit variational interpretation of the viscosity update which relates the total viscous dissipation to an energy term reflecting the change in fluid velocity. We prove its equivalence to the standard PDE form and note that since the minimization form is quadratic in velocity, the problem is automatically well-posed and its discretization is symmetric semi-definite, allowing efficient solution using conjugate gradient. Furthermore, it leads to a simple volume-weighting scheme on the MAC grid which implicitly enforces the difficult free surface boundary condition, greatly simplifying implementation. Finally, we illustrate how to combine this type of variational Neumann boundary condition with traditional Dirichlet boundary conditions, allowing us to handle both free surfaces and solid walls. This is useful for our viscous solve as well as the variational pressure projection introduced by Batty et al. [BBB07]. We provide examples illustrating that this method is unconditionally stable, eliminates artifacts in rotation and bending, conserves angular momentum, supports variable viscosity without modification, and provides more accurate modeling of free surface viscous liquids than previously seen in graphics.

3.2 Related Work

We will focus on demonstrating that correct free surface boundary conditions are important for properly simulating viscous liquids, and will use viscous buckling and coiling as our key example. This phenomenon was first studied by physi-

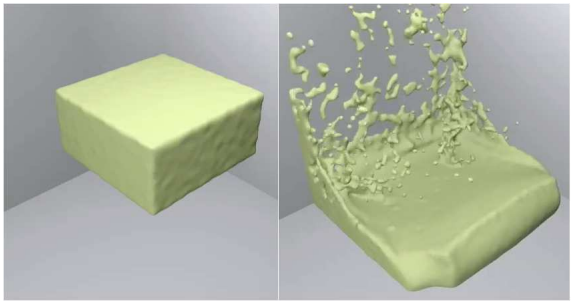


Figure 3.2: A block of fluid whose viscosity varies smoothly along its length is dropped onto a flat plane; the far end splashes in an inviscid manner, while the near end deforms only slightly.

Figure 3.3: Three different simulations of a long sheet of fluid falling under gravity demonstrating the influence of viscosity on buckling; from left to right viscosity values are 0.2, 1, and 5.

cist G. I. Taylor [Tay68], and a thorough experimental study was carried out by Cruikshank & Munson [CM81]. Bejan later penned a review article on the subject [Bej87], which also issued a rallying cry to the computational fluid mechanics community to tackle this "new frontier".

Viscous fluids were introduced to computer graphics by Miller & Pearce [MP89], who extended particle systems with inter-particle forces to approximate melting and flowing of viscous substances. Similarly, Terzopoulos et al. [TPF89] demonstrated the ability to melt finite element solids into collections of interacting particles.

The first work in computer graphics to simulate viscous fluids using the 3D Navier-Stokes equations was Foster & Metaxas [FM96], who adapted the classic MAC method of Harlow & Welch [HW65]. Though quite effective, it required small time steps due to the use of explicit integration. Stam [Sta99] introduced an implicit viscosity solve (along with semi-Lagrangian advection) which enabled much larger time steps, greatly improving simulation efficiency. By assuming constant viscosity, this method decouples the components of velocity allowing each to be solved independently. The resulting three linear systems are symmetric positive definite with a Poisson-like form and can be conveniently solved with a conjugate gradient method. We will refer to this method as the classic decoupled solve.

Carlson et al. [CMVT02] adapted this model to handle free surface liquids and variable viscosity; by further adding a heat diffusion model they generated an impressive animation of a wax bunny steadily melting due to a nearby heat source. However, their simplification of both the variable viscosity term and the free surface boundary condition introduced artifacts such as nonphysical damping of ballistic motion, which they partially rectified by directly adding back in the expected net translational motion (albeit choosing to neglect rotation). Falt & Roble [FR03] later corrected the translational error (though again, not the rotational error) by

enforcing Neumann boundary conditions of the form $(\nabla\vec{u})\cdot\vec{n}=0$ at grid-aligned air-fluid interfaces.

Rasmussen et al. [REN+04] also studied the case of free surface variable viscosity, but rather than dropping terms they eliminated the coupling between velocity components by proposing a combined implicit-explicit (IMEX) integration scheme. Under this scheme the dimensionally coupled components are first integrated explicitly, and the remaining decoupled, symmetric components are integrated implicitly. For constant viscosity regions the explicit components exactly cancel (assuming the input velocities are incompressible) leaving behind the same three linear systems as before. This technique was used to creating a stunning melting robot sequence for the third *Terminator* film.

Hong et al. [HK05] demonstrated two-phase fluids with discontinuous jumps in viscosity across the interface between constant viscosity fluids, simplifying earlier work by Kang et al. [KFL00] and adapting it to the octree discretization of Losasso et al. [LGF04]. Losasso et al. [LSSF06] extended this approach to multiple *immiscible* liquids, but still used constant viscosity for a given fluid to avoid the time step restrictions of the IMEX integration scheme.

Several papers have examined non-Newtonian fluids, ie. fluids whose stress is non-linearly related to the strain rate, and whose behaviour lies on the continuum between fluid and solid. Zhu & Bridson [ZB05] added a simplified frictional plasticity model to a fluid simulator to animate the motion of sand. To simulate large viscoplastic flow Bargteil et al. [BHWT07] started instead from the Lagrangian finite element viewpoint, and added remeshing and basis updates to the invertible finite element method of Irving et al. [ITF04]. Wojtan & Turk subsequently extended this scheme with an embedded deformation method and an explicit surface tracker to retain thin features and speed up meshing [WT08].

Goktekin et al. introduced an explicit method for simulating *viscoelastic* liq-

uids [GBO04], by adding an elasticity step to a fluid simulator based on an estimate of accumulated strain. They captured the complex elastic behavior of such fluids, including a small degree of buckling. However, our work differs from theirs in a few key points. First, our method is fully implicit and unconditionally stable, and properly handles rotation. Secondly, and more importantly, we demonstrate that by correctly capturing the true free surface boundary condition, we can capture the buckling of purely viscous Newtonian fluids. For example, our method can simulate honey or molasses without introducing spurious (nonphysical) elastic effects. In fact, it is complementary to their method and could be used as a drop-in replacement for their standard viscous step, which is entirely orthogonal to the elastoplastic components of the work.

There are also examples of SPH methods [CBP05], vorticity-based methods [ETK+07], and Lattice Boltzmann methods [Thu07] that support viscous fluids, though none in graphics have displayed viscous buckling. In computational physics, a few papers have successfully tackled this phenomenon including the SPH method of Rafiee et al. [RMH07] and the unstructured mesh finite element method of Bonito et al. [BPL06]. We will instead focus on Eulerian, Cartesian grid-based simulation.

In computational physics, the classic MAC scheme has been adapted to handle highly viscous (low Reynolds number) free surface fluids. A pair of papers by Hirt & Shannon [HS68] and Nichols & Hirt [NH71] looked at enforcing the full traction-free surface boundary conditions in 2D, the former examining the normal stress condition, the latter the tangential stress condition. They assume each cell is either full or empty, approximate the resulting surface normals as either grid-aligned or at 45 degrees, and derive discrete conditions for each case. Pracht used these same conditions in an implicit approach [Pra71] that solves a large linear system for pressure and velocity simultaneously.

The various incarnations of the GENSMAC method of Tomé, McKee, and co-workers [TM94, TM99, TFC+01] extended the general MAC framework to three dimensions including explicit traction-free surface boundary conditions. To our knowledge, GENSMAC is the first and only MAC-type scheme to successfully simulate viscous jet buckling. The free surface is again enforced using a case-based analysis, assuming incompressibility and a small set of possible surface normals. More recent work of de Sousa et al. [dSMN+04] used an accurate normal extracted from a surface mesh, but it is unclear how this is used in applying the boundary conditions. Noting difficulties with simulating low Reynolds flow, Oishi et al. [OCF+06] adapted GENSMAC to an implicit solve in 2D, but with decoupled pressure and velocity (in contrast to Pracht's work). They present results showing that to achieve reasonable time step sizes, it is necessary to solve both the equations of motion *and* the boundary conditions implicitly. They have since extended this method to 3D [OTCM08], enabling the simulation of 3D coiling for quite viscous fluids. However, this technique requires the solution of a large asymmetric linear system as well as unwieldy derivation and implementation of 26 cases of discrete surface orientation arising in 3D.

There are also techniques that more accurately enforce the boundary conditions in an explicit manner. These approaches perform a least-squares estimate of the velocity gradient near the surface using several sample points and an SVD operation, and then apply an extrapolation while enforcing the boundary conditions [PZ02, HP04]. This contrasts with the simple constant extrapolation prevalent in computer graphics (eg. [EMF02]).

3.3 Variable Viscosity Flow

We wish to simulate highly viscous incompressible fluids, possibly with varying viscosity. In this setting, the Navier Stokes equations have the following form:

$$\vec{u}_t = -\vec{u}\cdot\nabla\vec{u} + \tfrac{1}{\rho}\nabla\cdot\tau - \tfrac{1}{\rho}\nabla p + \vec{g} \tag{3.1}$$

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

$$\tau = \mu(\nabla u + (\nabla u)^T) \tag{3.3}$$

where, $\vec{u}$ is velocity, μ is dynamic viscosity, p is pressure, ρ is density, $\vec{g}$ is external accelerations (eg. gravity), and τ is the viscous shear stress tensor. We take the standard approach in graphics of using operator splitting to solve for viscous forces independently. In a given timestep we first apply advection and external forces, project the velocities to be divergence free, solve for viscosity, and finally project the velocities to be divergence free a second time (see eg. [LSSF06]). (Two pressure projections are needed because operator-split viscosity formulations typically assume an incompressible velocity field.) This leaves us with the following PDE for integrating viscosity alone:

$$\vec{u}_t = \frac{1}{\rho}\nabla\cdot\left(\mu(\nabla\vec{u} + (\nabla\vec{u})^T)\right) \tag{3.4}$$

Previous approaches discretized this PDE form directly, using explicit, IMEX [REN+04], or implicit schemes, giving the following:

$$\vec{u} = \vec{u}^{old} + \frac{\Delta t}{\rho}\nabla\cdot\left(\mu(\nabla\vec{u^*} + (\nabla\vec{u^\dagger})^T)\right) \tag{3.5}$$

For the sake of brevity, we use $\vec{u}$ to refer to the updated velocity, while $\vec{u}^{old}$ refers to the input velocity. To define a particular integration scheme, $\vec{u}^*$ and $\vec{u}^\dagger$ are chosen

to be either $\vec{u}^{old}$ or $\vec{u}$. A fully explicit scheme sets $\vec{u}^* = \vec{u}^\dagger = \vec{u}^{old}$, a fully implicit scheme sets $\vec{u}^* = \vec{u}^\dagger = \vec{u}$, and the IMEX scheme can be arrived at by setting $\vec{u}^* = \vec{u}$ and $\vec{u}^\dagger = \vec{u}^{old}$. (For constant viscosity $\nabla \cdot (\nabla \vec{u})^T = 0$ due to incompressibility, decoupling the components of velocity and leaving the Poisson-like form usually given.)

The explicit scheme tends to require a small time step for stability; one can employ sub-cycling, taking many viscous sub-steps per overall time step, but for moderately viscous fluids this quickly becomes untenable. Rasmussen et al. partially addressed this with the IMEX scheme, whose implicit part somewhat lessens the time step restriction. It also decouples the three velocity components in the implicit part, giving the usual three systems of the classic decoupled solve. However, their primary reason for choosing an IMEX scheme over a fully implicit one which would eliminate the time step restriction entirely was that for their finite differencing method the implicit scheme generates an asymmetric linear system. Such a system cannot be solved with the usual conjugate gradient method, requiring instead a more expensive and potentially less robust solver such as GMRES. We will show in section 3.5 that we actually can solve this problem efficiently in a fully implicit way, by exploiting a variational principle that guarantees symmetry.

3.4 Viscous Free Surface Boundary Conditions

Neglecting the effects of surface tension, the true free surface boundary conditions for Navier-Stokes dictate that there is zero traction $\vec{t}$ applied on the plane of the surface. From the definition of Cauchy stress, this gives us:

$$\vec{t} = \sigma \vec{n} = 0 \tag{3.6}$$

where σ is the full Cauchy stress tensor and $\vec{n}$ is the normal to the free surface. Splitting σ into components of pressure p and shear stress τ, we have:

$$\sigma\vec{n} = (-p\mathbf{I} + \tau)\vec{n} = 0 \tag{3.7}$$

Since we have decoupled the velocity and pressure solves in our method, we do the same with the boundary conditions. If we assume as usual that the free surface pressure is zero during the pressure solve, we're left with the boundary condition $\tau\vec{n} = 0$. This implies:

$$\mu(\nabla u + (\nabla u)^T)\vec{n} = 0 \tag{3.8}$$

A key point to note is that this couples together the components of velocity *even for constant viscosity* [LIRO07]. To correctly enforce it we must solve the full system (3.4) even if decoupling occurs on the interior of the fluid.

Methods for enforcing this constraint fall into two categories: explicit and implicit. The explicit approach first extrapolates the current surface velocities into the nearby air region, possibly subject to the zero-traction constraint. During the viscous solve these air velocities are held fixed as Dirichlet boundary conditions. In graphics, simple constant extrapolation of velocity without constraint enforcement is typical. The complexity of this approach can increase almost arbitarily depending on the desired spatial accuracy for both the extrapolation and the zero-traction constraint. However, because it uses the input velocities as the starting point, its temporal accuracy and stability are ultimately still limited even in an otherwise fully implicit solve [OCF+06]. In practice, this means that if the viscosity would otherwise dictate a large change in surface velocity, it cannot occur because the old extrapolated boundary velocities remain unchanged.

In contrast, the implicit approach uses a Neumann boundary condition, so that

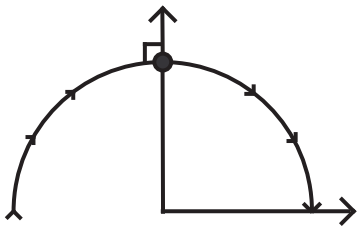


Figure 3.4: A rotational velocity field $\vec{u} = (y, -x)$. The zero-traction boundary condition must be correctly enforced in order to preserve angular momentum.

boundary velocities need not be known in advance. The key difficulty encountered with this approach is that the full complexity of the extrapolate/constrain process above must effectively be built into the linear system, greatly increasing implementation complexity.

Considered naïvely, either of these approaches requires estimating the normal direction, and determining exactly where and how to discretize the constraint onto the simulation grid. However, this boundary condition is in a sense a "natural" or homogeneous Neumann boundary condition, and finite element methods commonly exploit this property to circumvent the need to enforce such conditions explicitly. For example, Bonito et al. used this idea in their finite element simulations of viscous buckling [BPL06]. Our variational interpretation accomplishes the same goal within the finite difference scheme, with an approach closely related to that of Batty et al. [BBB07].

Before presenting the details, we emphasize that correctly enforcing this boundary condition is not merely an esoteric exercise, but crucial in animating the most attractive aspects of viscous flow. A common and seemingly reasonable boundary condition one might apply to the classic decoupled solve has the form $(\nabla \vec{u}) \cdot \vec{n} = 0$. This is the Neumann boundary condition used by Falt & Roble (assuming grid-

aligned surfaces), and also corresponds to the constant extrapolation of velocity used by Enright et al. [EMF02]. However, consider the simple 2D rigid rotational velocity field defined by $\vec{u} = (y, -x)$, as seen in Figure 3.4. At a location on the positive y-axis with surface normal $\vec{n} = (0, 1)$, equation (3.8) becomes $\frac{\partial u}{\partial y} = 0$ and $\frac{\partial v}{\partial y} = 0$. The true gradients of this rotational field are $\frac{\partial v}{\partial y} = 0$ and, crucially, $\frac{\partial u}{\partial y} = 1$. We see that the incorrect boundary condition directly works to halt rotational motion, and for moderately viscous fluids the effect is that angular velocity is rapidly damped out. Our technique will correctly give $\frac{\partial v}{\partial y} = 0$ and $\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} = 1$, eliminating this artifact.

3.5 A Variational Interpretation of Viscosity

We now consider how to phrase an implicit viscosity step in terms of a minimization problem. One characterization of the true solution to a Stokes viscous flow problem (i.e. ignoring advection) is that it is the unique velocity field which minimizes the rate of viscous dissipation, subject to the constraint of incompressibility. This result, known as the minimum dissipation theorem, is originally due to Helmholtz [Bat67]. If we express the deformation rate tensor as

$$\dot{\varepsilon} = \frac{\nabla\vec{u} + (\nabla\vec{u})^T}{2} \tag{3.9}$$

then the rate of viscous dissipation is given by

$$\Phi = 2\mu\dot{\varepsilon} : \dot{\varepsilon} = 2\mu\|\dot{\varepsilon}\|_F^2 \tag{3.10}$$

Recall that the : operator refers to tensor double contraction (analogous to a matrix dot-product) and $\|\cdot\|_F$ indicates the Frobenius norm of a matrix. Unfortunately, simply minimizing this expression fails to produce the desired effect, because we

have decoupled pressure and viscosity. We no longer have a classic Stokes problem and are not strictly enforcing incompressibility during this step. Instead what we can do is try to enforce that the velocity changes as little as possible, while simultaneously seeking a velocity field that minimizes dissipation over the timestep. Putting this together we get:

$$\min_{\vec{u}} \iiint \rho \left\| \vec{u} - \vec{u}^{old} \right\|^2 + 2\Delta t \iiint \mu \left\| \frac{\nabla \vec{u} + (\nabla \vec{u})^T}{2} \right\|_F^2 \tag{3.11}$$

Here the volume integrals are taken over the fluid; no boundary integrals are required. Calculus of variations can be used to show that minimizing this expression gives us back exactly the time-discretized PDE form for the viscous update, even for the variable viscosity case—see appendix 3.A for the mathematical details. The integrals are quadratic in the new velocity and obviously bounded below by zero, so the minimization is automatically well-posed, and the discretized form will be symmetric semi-definite (as well as sparse), guaranteeing that conjugate gradient can be used to solve it efficiently. However, the most beneficial result of expressing the viscosity update in this manner is that we no longer need to handle the free surface with special cases: it is captured automatically by minimization of this volume integral.

3.5.1 Discretization of the Variational Principle

Rather than tackling the PDE form directly, which would include the complex free surface condition (3.8), we will discretize the variational principle (3.11), and then minimize this discrete form. We store the velocity components in the MAC grid configuration, so that the first integral has fractional volume weights centred on faces, exactly as in Batty et al. [BBB07]. Similarly, the viscous dissipation integral gives rise to volume terms associated with the various components of stress, which

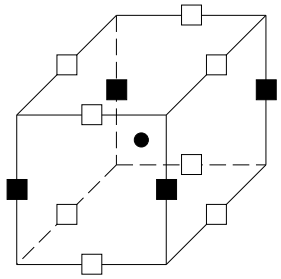


Figure 3.5: The locations of stress samples on the MAC grid. $\tau_{11}, \tau_{22}, \tau_{33}$ all sit at the central black circle. τ_{23} samples are white squares, τ_{13} samples are black squares, and τ_{12} samples are the hatched squared.

we locate on cell-centres and edges, as done by Goktekin et al. [GBO04]. Notice that centred finite differencing of adjacent MAC velocities places stress at these locations. As a result of this configuration, the volume weights for the second integral are chosen by computing the volume fraction of fluid contained within the cube of volume Δx^3 surrounding each stress sample point. The actual method of computing these volumes is dependent on the choice of surface tracker. Estimates rather than exact volumes may be used, but the volume estimates for the different locations should be consistent. In our system we splat the union of spheres around the particles onto a grid to get an approximate signed distance field, and then estimate volumes with simple quadrature.

Discretizing and minimizing, we get a new discrete velocity update that closely

mirrors the standard implicit solve:

$$u = u^{old} + \frac{\Delta t}{V_u \rho} \begin{pmatrix} (V_p 2\mu u_x)_x \\ +(V_{\tau_{12}} \mu (u_y + v_x))_y \\ +(V_{\tau_{13}} \mu (u_z + w_x))_z \end{pmatrix}$$

$$v = v^{old} + \frac{\Delta t}{V_v \rho} \begin{pmatrix} (V_{\tau_{12}} \mu (v_x + u_y))_x \\ +(V_p 2\mu v_y)_y \\ +(V_{\tau_{23}} \mu (v_z + w_y))_z \end{pmatrix}$$

$$w = w^{old} + \frac{\Delta t}{V_w \rho} \begin{pmatrix} (V_{\tau_{13}} \mu (w_x + u_z))_x \\ +(V_{\tau_{23}} \mu (w_y + v_z)_y) \\ +(V_p 2\mu w_z)_z) \end{pmatrix}$$

The V terms refer to cell-centered volumes (p, but note that $\tau_{11}, \tau_{22}, \tau_{33}$ all sit here), face-centred volumes (u, v, w), and edge-centred volumes ($\tau_{12}, \tau_{13}, \tau_{23}$). This is of course similar to the form given by Rasmussen et al. [REN+04]. The important differences are the addition of volume weights, and the use of the MAC grid so that centred differencing leaves the various discrete derivatives in the correct locations. Appendix 3.B gives a detailed discretization for a u-velocity update in 2D for the sake of brevity, but the extension to higher dimensions is straightforward. In practice we note it is often more convenient to use dimensionless volume fractions rather than actual volumes.

3.5.2 Combining Ghost Fluid and Variational Boundaries

A natural question to ask is whether this type of variational Neumann boundary can co-exist with Dirichlet boundary conditions, especially of the second order accurate ghost fluid-type employed by Enright et al. [ENGF03]. This is relevant not only to the current work, in which the air boundary is Neumann and the solid

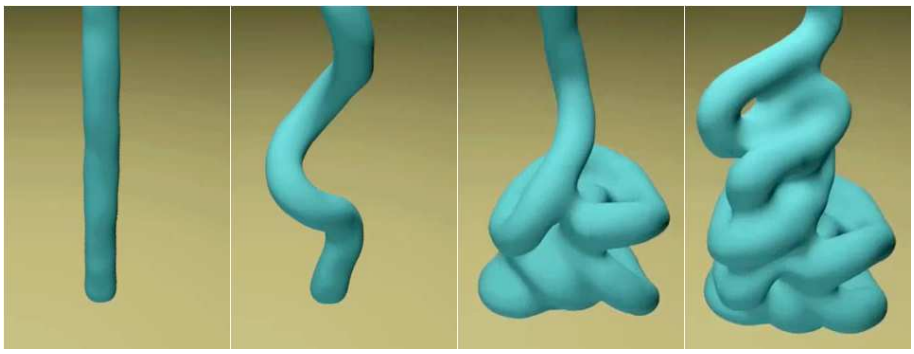


Figure 3.6: A cylinder of viscous liquid falls under gravity, and spontaneously develops a coiling and folding motion.

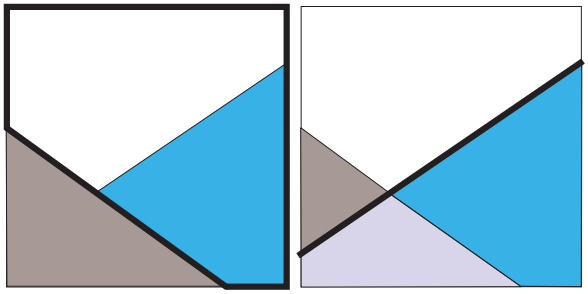


Figure 3.7: Left: A liquid-air-solid triple point for the pressure projection case. Cyan indicates liquid, white indicates air, grey indicates solid wall. The combined volume used for the fluid weights is outlined by the bold line. Right: The liquid signed distance field is extrapolated into the wall for use in the ghost fluid Dirichlet boundary condition, with the ghost-fluid interface identified by the bold line.

boundary is Dirichlet, but also to the work of Batty et al. [BBB07] who used a similar natural boundary condition to handle the Neumann pressure gradient constraint along non-grid-aligned solid walls. Their results were primarily restricted to examples lacking free surfaces, though they claimed that ghost fluid Dirichlet boundaries could straightforwardly be incorporated. This turns out to be the case, as we outline below. For concreteness we focus on the variational pressure problem, in which ghost fluid air-water interfaces must be handled carefully to avoid surface artifacts. The same general method is applied for boundary conditions in our viscous solve as well, by swapping air for solid in the following discussion.

The main uncertainty is whether the fluid volume estimates used in variational approaches ought to include the volume from a cell in which a Dirichlet condition is being applied. A "ghost fluid" point of view shows that the answer is yes. The ghost fluid method treats the air side of the interface as a smooth extension of the fluid domain, whose variables are chosen in such a way as to enforce the Dirichlet condition at the correct location. Therefore we assume the fluid volume is also extended smoothly into the air region, and so include its volume in our minimization (Figure 3.7, left). To enforce both variational solid and ghost-fluid air boundary conditions on different parts of the boundary, we simply compute the fluid volume weights by including air volume, but excluding solid volume. Then we apply the ghost fluid method on top using the liquid signed distance to determine the interface location, and ignoring the presence of weighting terms and solid walls. The discretization of equation (11) from [ENGF03] becomes (up to scaling):

$$\frac{(vol_{i+\frac{1}{2}})\frac{p_{fs}-p_i}{\theta\Delta x} - (vol_{i-\frac{1}{2}})\frac{p_i-p_{i-1}}{\Delta x}}{\Delta x} \tag{3.12}$$

The signed distance data used for determining the position of the interface should be extrapolated smoothly into the wall, much like in the work of Rasmussen

et al. [REN+04]. This ensures that the solver "sees" a smooth liquid surface right up to the (implicitly defined) solid wall, rather than one which erroneously bends away or terminates. This is illustrated in Figure 3.7, right.

3.6 Implementation

We augmented the basic FLIP approach of Bridson et al. [ZB05] with our new viscous solve. While we emphasize that our method plugs conveniently into any standard Cartesian grid-based graphics fluid simulator, an advantage of using FLIP with our method is that in combination they can seamlessly handle liquids that range continuously from almost purely inviscid to extremely viscous in a single simulation (Figure 3.2). We slightly reduced the memory footprint of FLIP by using one particle per cell with a larger radius, and transferring velocities from particles to the grid using a wider SPH-like kernel. The rendered surface is generated by wrapping a smoothed implicit surface around the underlying particles. Despite only minimal optimization, our examples typically required only a minute or two per frame for simulation. For example, the buckling sheet averaged one minute per frame on a 45x45x300 grid. Of that, about 50% is currently the viscosity step, which we solve with conjugate gradient and an incomplete Cholesky preconditioner. We note that while our method is inherently slower than that of Carlson et al. due to solving a unified system that is three times larger, we believe that the improved behaviour is worth this additional expense.

3.7 Examples

We now present a variety of examples demonstrating the validity of our approach and the range of behaviours that can be achieved. First, we illustrate the benefits of

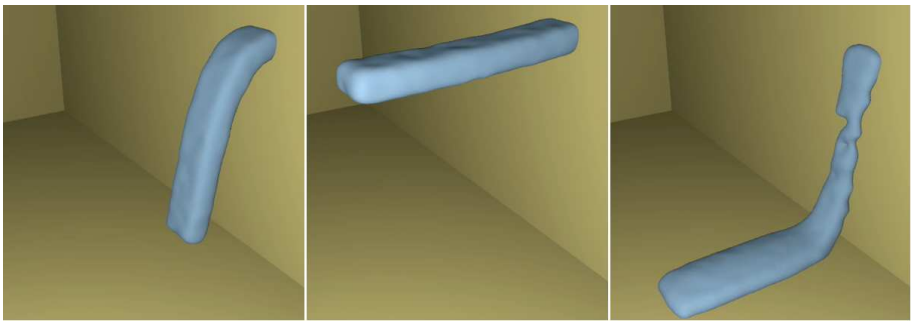


Figure 3.8: A beam-shaped blob of constant viscosity fluid is attached to a wall, and simulated with three different boundary conditions. Left: Correct variational boundary conditions allow rotation, so viscous forces cause the fluid to bend in towards the wall. Middle: Incorrect Neumann boundary conditions cannot handle rotation, so the fluid can only shear and the motion is excessively damped. Right: Incorrect Dirichlet boundary conditions cannot change to reflect large changes in velocity, so the fluid falls as if unsupported.

fully implicit viscosity integration. In a 2D simulation with a moderate coefficient of viscosity, we used the explicit, IMEX, and our fully implicit schemes to simulate a blob of initially motionless fluid falling under gravity. (Because these examples are 2D, for the explicit and IMEX schemes we implemented the tangential stress condition as proposed by Nichols & Hirt, setting $\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$, either implicitly or explicitly to match the integration scheme. In 2D $\tau\vec{n} = 0$ implies $\tau = 0$, which is relatively easy to implement, but in 3D the normal becomes important, greatly increasing the complexity.) Our fully implicit approach is perfectly stable taking one large step, whereas the explicit and IMEX approaches require approximately 28 and 14 sub-steps, respectively, to avoid blow-up.

Next we examined rotational motion of a 2D circular disk of high viscosity fluid under zero gravity conditions. The Falt & Roble Neumann conditions result in rotational motion being lost instantly. Extrapolated explicit Dirichlet conditions fare slightly better, since the boundary conditions contain lagged velocities, but

it still halts after a few timesteps. Our variational approach does a much better job at maintaining rotation without discernible artifacts, and rotates for hundreds of frames. (The remaining dissipation is primarily due to splitting errors related to the distinct advection and pressure phases of the simulator - advection alone partially transfers energy in rotational modes to divergent modes, which are then removed by pressure projection.)

A common test of elastic bending is a beam pinned at one end to a solid wall. We perform an analogous test on a chunk of constant viscosity (non-elastic) fluid, by applying no-slip boundary conditions at the wall (Figure 3.8). The implicit Neumann boundary conditions of Falt & Roble fail due to the loss of rotation at the surface. The fluid is far more damped than the viscosity would otherwise dictate, nearly halting motion altogether. Furthermore, rather than rotating, the fluid incorrectly shears and falls vertically instead of collapsing in towards the wall. The extrapolated Dirichlet boundary condition likewise results in large shearing. It has the additional problem that because the boundary velocities are set before the solve, they cannot change in response to the viscous forces propagating from the "pinned" end which ought to partially counterbalance gravity. The bulk of the fluid therefore falls under gravity as if it were not supported at all. Our technique results in the correct behaviour.

Next we drop a long thin cylinder of viscous fluid onto a plane (Figure 3.6). We successfully reproduce the strong buckling and coiling effect that is characteristic of many common purely viscous fluids, and has not been accomplished previously in graphics. To explore the effect of different coefficients of viscosity on the buckling behaviour, we drop a sheet of fluid perpendicular to the ground plane (Figure 3.3). For low viscosities no buckling occurs, while for higher viscosities the folds become much longer and more pronounced.

To demonstrate that we can handle variable viscosity, we drop a block of fluid

whose viscosity varies continuously from one end to the other (Figure 3.2). Initially the block falls uniformly under gravity, illustrating that our method introduces no erroneous rotational or translational forces. Once it collides with the flat, featureless ground plane, the inviscid end collapses and splashes up against the far wall, while the viscous end sags slightly on impact. Waves and turbulent motion occurring at the inviscid end damp out as they pass towards the viscous end, so that when the simulation concludes the initial sharp edge of the fluid block is still visible.

Lastly, we illustrate that our approach to handling Dirichlet and Neumann variational boundaries together lets us easily incorporate free surfaces into the method of Batty et al. We drop a sphere of liquid inside a hollow Stanford bunny mesh, generating complex splashing and interaction with the bunny geometry (Figure 3.9).

3.8 Conclusions and Future Work

We have shown that by considering a variational principle for the viscosity solve, we can achieve complex viscous fluid effects that have been lacking in the graphics literature to date. Nonetheless, there are several avenues for future work. First, the complete free surface boundary condition couples pressure to velocity, so a unified pressure-viscosity solve is likely needed to handle this tighter coupling. Unfortunately this requires solving a larger and more complex symmetric indefinite system, and it is unclear if this would benefit graphics applications. Similarly, we did not support surface tension, although it can play a vital role in the surface behaviour. We could easily add it to the pressure solve (see eg. [ENGF03]) or use another method from the graphics literature, but a fully unified implicit approach would be interesting to consider. Finally, although the new linear system of our method is symmetric positive definite, it is no longer an M-matrix. This is the class of matrices with positive eigenvalues and non-positive off-diagonal entries, and

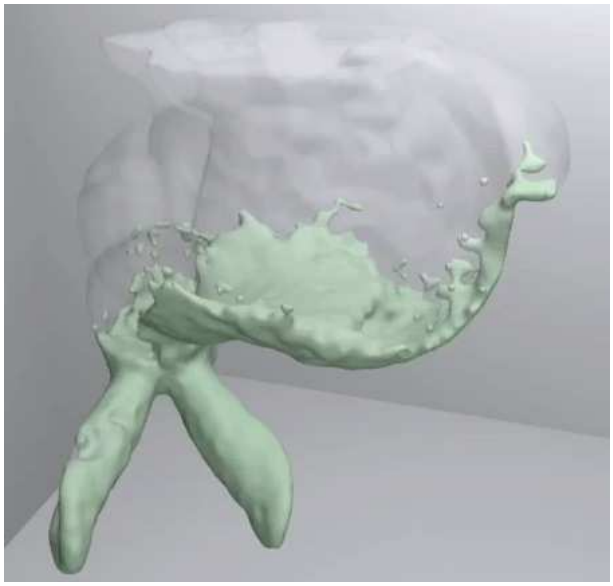


Figure 3.9: A low viscosity liquid splashes inside the Stanford bunny.

for which the modified incomplete Cholesky preconditioner is expected to perform well. Research into alternative preconditioners could therefore further accelerate our solver.

3.A Equivalence of the Minimization Form

Let $\mathbf{D}$ be the rate of deformation operator defined such that $\mathbf{D}(\vec{u}) = (\nabla\vec{u} + (\nabla\vec{u})^T)/2$. Suppose $\vec{u}$ is the minimizer of (3.11). We introduce an arbitrary vector $\vec{q}$, a scalar ε, and a scalar function $g(\varepsilon)$ such that:

$$\begin{aligned} g(\varepsilon) = \quad & \iiint_{\text{fluid}} \rho \|\vec{u} + \varepsilon\vec{q} - \vec{u}^{old}\|^2 \\ & + 2\Delta t \iiint_{\text{fluid}} \mu \mathbf{D}(\vec{u} + \varepsilon\vec{q}) : \mathbf{D}(\vec{u} + \varepsilon\vec{q}) \end{aligned}$$

This function is quadratic in ε. Since $\vec{u}$ is the minimizer, we know that $\varepsilon = 0$ is the minimizer of g, and thus $g'(0) = 0$. Thus the coefficient of the linear terms of g must be 0, so we have:

$$0 = \iiint_{\text{fluid}} \rho \vec{q}^T (\vec{u} - \vec{u}^{old}) + 2\Delta t \iiint_{\text{fluid}} \mu \mathbf{D}(\vec{u}) : \mathbf{D}(\vec{q})$$

We now require a generalized integration by parts formula. For a symmetric rank-two tensor A and a vector q the following can easily be verified:

$$\iiint_{\omega} \mathbf{D}(\vec{q}) : \mathbf{A} = \iint_{\partial\omega} \vec{q}^T \mathbf{A}\vec{n} - \iiint_{\omega} \vec{q}^T \nabla \cdot \mathbf{A}$$

This allows us to eliminate the $\mathbf{D}(\vec{q})$ term giving:

$$\begin{aligned} 0 = & \iiint_{\text{fluid}} \rho \vec{q}^T (\vec{u} - \vec{u}^{old}) \\ & - 2\Delta t \iiint_{\text{fluid}} \vec{q}^T \nabla \cdot \mu \mathbf{D}(\vec{u}) + 2\Delta t \iint_{\text{surface}} \mu \vec{q}^T \mathbf{D}(\vec{u})\vec{n} \end{aligned}$$

Since $\vec{q}$ is arbitrary, the terms multiplying it must be zero. Hence in the fluid domain we have:

$$0 = \rho(\vec{u} - \vec{u}^{old}) - 2\Delta t \nabla \cdot \mu \mathbf{D}(\vec{u})$$

therefore

$$\vec{u} = \vec{u}^{old} + \frac{\Delta t}{\rho} \nabla \cdot \mu (\nabla \vec{u} + (\nabla \vec{u})^T)$$

which is the evolution equation for viscosity (3.5). On the surface of the fluid we have

$$0 = 2\Delta t \mu \mathbf{D}(\vec{u})\vec{n}$$

or equivalently

$$\mu(\nabla \vec{u} + (\nabla \vec{u})^T)\vec{n} = 0$$

which is the boundary condition on stress (3.8). Thus minimizing this integral is equivalent to solving the PDE form.

3.B Detailed 2D Discretization

The discretization of the implicit u-velocity update in 2D is:

$$u_{i+\frac{1}{2},j} = u^{old}_{i+\frac{1}{2},j} + \frac{\Delta t}{\rho V_{i+\frac{1}{2},j}} (A + B + C)$$

where

$$A = 2 \left(\frac{(V_p \mu)_{i+1,j} \frac{u_{i+\frac{3}{2},j} - u_{i+\frac{1}{2},j}}{\Delta x} - (V_p \mu)_{i,j} \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x}}{\Delta x} \right)$$

$$B = \left(\frac{(V_{\tau_{12}} \mu)_{i+\frac{1}{2},j+\frac{1}{2}} \frac{u_{i+\frac{1}{2},j+1} - u_{i+\frac{1}{2},j}}{\Delta x} - (V_{\tau_{12}} \mu)_{i+\frac{1}{2},j-\frac{1}{2}} \frac{u_{i+\frac{1}{2},j} - u_{i+\frac{1}{2},j-1}}{\Delta x}}{\Delta x} \right)$$

$$C = \left(\frac{(V_{\tau_{12}} \mu)_{i+\frac{1}{2},j+\frac{1}{2}} \frac{v_{i+1,j+\frac{1}{2}} - v_{i,j+\frac{1}{2}}}{\Delta x} - (V_{\tau_{12}} \mu)_{i+\frac{1}{2},j-\frac{1}{2}} \frac{v_{i+1,j-\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta x}}{\Delta x} \right)$$

Figure 3.10 shows the corresponding stencil.

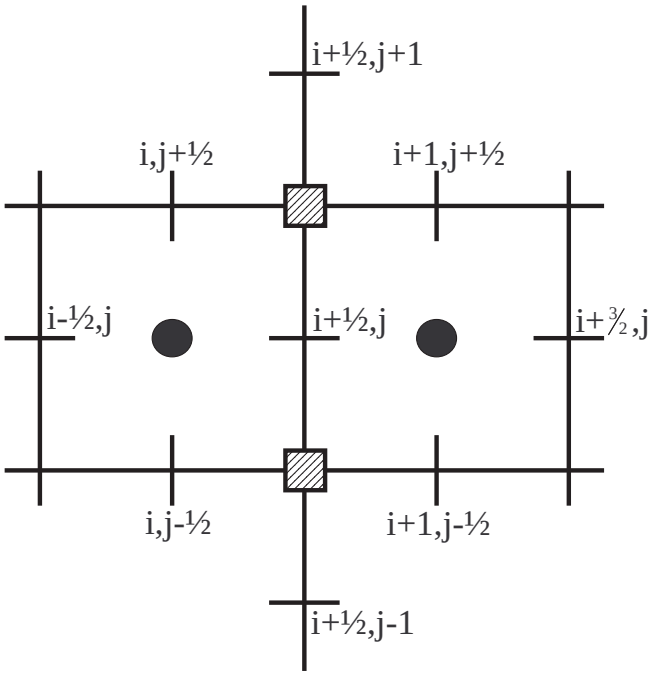


Figure 3.10: The 2D stencil for the u-velocity update.

Bibliography

[Bat67] G. K. Batchelor. *An Introduction to Fluid Dynamics*. 1967.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):100, 2007.

[Bej87] Adrian Bejan. Buckling flows: A new frontier in fluid mechanics. *Annual Reviews of Heat Transfer*, 1(1):262–304, 1987.

[BHWT07] Adam W. Bargteil, Jessica K. Hodgins, Chris Wojtan, and Greg Turk. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):16, 2007.

[BPL06] Andrea Bonito, Marco Picasso, and Manuel Laso. Numerical simulation of 3D viscoelastic flows with free surfaces. *J. Comp. Phys.*, 215(2):691–716, 2006.

[CBP05] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation*, pages 219–228, 2005.

[CM81] J. O. Cruikshank and B. R. Munson. Viscous-fluid buckling of plane and axisymmetric jets. *Journal of Fluid Mechanics*, 113:221–239, 1981.

[CMVT02] Mark Carlson, Peter J. Mucha, R. Van Horn, and Greg Turk. Melting and flowing. In *Symposium on Computer Animation*, pages 167–174, 2002.

[dSMN+04] F. S. de Sousa, Norberto Mangiavacchi, L. G. Nonato, Antonio Castelo, Murilo F. Tomé, and Sean McKee. A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces. *J. Comp. Phys.*, 198(2):469–499, 2004.

[EMF02] Doug Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):736–744, 2002.

[ENGF03] Doug Enright, Duc Nguyen, Frédéric Gibou, and Ron Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, 2003.

[ETK+07] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[FM96] Nick Foster and Dimitris Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[FR03] Henrik Fält and Doug Roble. Fluids with extreme viscosity. In *SIGGRAPH Sketches*, page 1, 2003.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):463–468, August 2004.

[HK05] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):915–920, July 2005.

[HP04] Yue Hao and Andrea Prosperetti. A numerical method for three-dimensional gas-liquid flow computations. *J. Comp. Phys.*, 196(1):126–144, 2004.

[HS68] C. W. Hirt and J. P. Shannon. Free surface stress conditions for incompressible-flow calculations. *J. Comp. Phys.*, 2(4):403–411, 1968.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[ITF04] Geoffrey Irving, Joseph Teran, and Ron Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Symposium on Computer Animation*, pages 131–140. ACM Press, 2004.

[KFL00] Myungjoo Kang, Ron Fedkiw, and Xu-Dong Liu. A boundary condition capturing method for multiphase incompressible flow. *SIAM J. Sci. Comput.*, 15(3):323–360, 2000.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):457–462, August 2004.

[LIRO07] A. Limache, S. R. Idelsohn, R. Rossi, and E. Onate. The violation of objectivity in Laplace formulations of the Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 54(6-8):639–664, 2007.

[LSSF06] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):812–819, 2006.

[MP89] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

[NH71] B. D. Nichols and C. W. Hirt. Improved free surface boundary conditions for numerical incompressible-flow calculations. *J. Comp. Phys.*, 8(3):434–448, 1971.

[OCF+06] Cassio M. Oishi, José A. Cuminato, Valdemir G. Ferreira, Murilo F. Tomé, Antonio Castelo, Norberto Mangiavacchi, and Sean McKee. A stable semi-implicit method for free surface flows. *Journal of Applied Mechanics*, 73(6):940–947, 2006.

[OTCM08] Cassio M. Oishi, Murilo F. Tomé, José A. Cuminato, and Sean McKee. An implicit technique for solving 3D low Reynolds number moving free surface flows. *J. Comp. Phys.*, 227(16):7446–7468, 2008.

[Pra71] William E. Pracht. A numerical method for calculating transient creep flows. *J. Comp. Phys.*, 7(1):46–60, 1971.

[PZ02] Stephane Popinet and Stephane Zaleski. Bubble collapse near a solid boundary: A numerical study of the influence of viscosity. *Journal of Fluid Mechanics*, 464:137–163, 2002.

[REN+04] Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, N. Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. Directable

photorealistic liquids. In *Symposium on Computer Animation*, pages 193–202, 2004.

[RMH07] A. Rafiee, M. T. Manzari, and M. Hosseini. An incompressible SPH method for simulation of unsteady viscoelastic free surface flows. *International Journal of Non-Linear Mechanics*, 42(10):1210–1223, 2007.

[Rui07] Allen Ruilova. Creating realistic CG honey. In *SIGGRAPH Posters*, page 58, 2007.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999.

[Tay68] George I. Taylor. Instability of jets, threads, and sheets of viscous fluids. In *Proc. Int. Cong. Appl. Mech.*, 1968.

[TFC+01] Murilo F. Tomé, A. C. Filho, José A. Cuminato, Norberto Mangiavacchi, and Sean McKee. GENSMAC3D: A numerical method for solving unsteady three-dimensional free surface flows. *International Journal for Numerical Methods in Fluids*, 37(7):747–796, 2001.

[Thu07] Nils Thuerey. *Physically based animation of free surface flows with the Lattice Boltzmann method*. PhD thesis, 2007.

[TM94] Murilo F. Tomé and Sean McKee. GENSMAC: A computational marker and cell method for free surface flows in general domains. *J. Comp. Phys.*, 110(1):171–186, 1994.

[TM99] Murilo F. Tomé and Sean McKee. Numerical simulation of viscous flow: Buckling of planar jets. *International Journal for Numerical Methods in Fluids*, 29(6):705–718, 1999.

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (From goop to glop). In *Graphics Interface*, pages 219–226, 1989.

[WT08] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):47, 2008.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):965–972, July 2005.

Chapter 4

A Variational Finite Difference Method for Time-Dependent Stokes Flow on Irregular Domains

4.1 Introduction

The equations of Stokes flow (or creeping flow) describe the motion of fluids at low Reynolds number, where the influence of advection is considered negligible relative to viscous forces. There are a wide variety of flows where this assumption is reasonable, making it a problem of substantial practical importance. Some examples include micro-fluidics technologies, propulsion of micro-organisms, and sedimentation. Furthermore, Stokes problems are often encountered as a sub-problem in methods for solving the full Navier-Stokes equations.

The goal of this paper is to derive and validate a fully implicit, embedded boundary finite difference method for Stokes flow problems in the presence of

A version of this chapter has been submitted for publication. Batty, C. and Bridson, R. (2010) A Variational Finite Difference Method for Time-Dependent Stokes Flow on Irregular Domains.

irregularly-shaped free surfaces and moving solid boundaries. A traditional difficulty in applying finite difference techniques to such problems is the complexity of enforcing non-grid-aligned boundary conditions on regular Cartesian grids, including the commonly used staggered grid arrangement. While such non-colocated grids are effective at eliminating the classic pressure checkerboard instability, accurately applying boundary conditions on components that are spatially scattered can be a challenge. This difficulty is further compounded when the boundary condition implies a constraint on the components of the stress tensor (which are themselves derived from the velocity), as in the case of a free surface. Our approach will be to take inspiration from finite element methods (eg. [BPL06]), and re-express the Stokes flow problem in a variational form, relating velocity, pressure, and deviatoric stress. This yields an interesting hybrid approach that is discretized with simple finite differences, but implicitly enforces complex boundaries through the natural boundary conditions of the problem.

Research into improving free surface boundary conditions on finite difference grids dates to shortly after the introduction of the classic marker-and-cell method [HW65], when it was recognized that for low Reynolds flows, a more accurate free surface boundary condition is required. Hirt & Shannon proposed a simple correction to the normal stress [HS68], which Nichols & Hirt subsequently improved to address the tangential stress condition [NH71]. Because a standard explicit discretization of viscous terms suffers from a time step restriction of $\Delta t < O(\rho \Delta x^2/\mu)$, Pracht incorporated the preceding ideas into a fully implicit integration scheme to ensure stability [Pra71]. Tomé et al. have extended this general approach to three dimensions within their GENSMAC code [TM94, TGC$^+$04, OTCM08]. A potential drawback of these schemes is the assumption that the surface is in one of a number of specific configurations (with respect to the Cartesian grid axes) for which the free surface boundary conditions can be simplified and explicitly en-

forced. This can limit the generality and accuracy of the method, while increasing the difficulty of implementation, particularly for implicit schemes. For example, in 3D there are 26 orientations to be handled, though this can be reduced by carefully exploiting symmetry.

Substantial effort has also gone towards irregular solid boundary conditions on Cartesian finite difference grids. Examples of such embedded boundary methods include the immersed boundary method (IBM) [Pes02], the immersed interface method (IIM) [LL97], the matched interface-boundary method (MIB) [ZZFW05], and cut-cell (finite volume) methods [JC98]. These methods have been effective and some achieve higher order accuracy, but they tend to be quite complex and typically do not extend straightforwardly to the full Stokes problem addressed here. They often also lead to non-symmetric systems that are more computationally intensive to solve.

Before delving into the details of our approach, we highlight a few of the most closely related techniques. First, a simple ghost fluid method has been used to enforce Dirichlet boundary conditions in Poisson-type problems [CS70, GFCK02, ENGF03] for pressure projection methods and methods for implicit integration of spatially constant viscosity. Secondly, a finite volume-like technique has been proposed for handling Neumann boundary conditions in the context of pressure projection methods for solid-fluid interaction [PB79, RbZF05, BBB07, NMG09]. Both of these methods rely on a staggered grid with weighting terms that modify the standard finite difference stencils, and can be interpreted as solving particular variational problems; our method is a natural generalization of these ideas to Stokes flow. We also build directly on ideas presented by Batty & Bridson for viscous free surface flows in computer animation [BB08].

4.2 Time-Dependent Stokes Flow

To set the stage for describing our method, we review the problem under consideration. The full governing equations for time-dependent Stokes flow are:

$$\rho \vec{u}_t = -\nabla p + \nabla \cdot \tau + \vec{F} \tag{4.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{4.2}$$

$$\tau = 2\mu \left(\frac{\nabla \vec{u} + (\nabla \vec{u})^T}{2} \right) \tag{4.3}$$

where ρ is the fluid density, $\vec{u}$ is the fluid velocity, p is the pressure, τ is the symmetric deviatoric stress tensor, μ is the dynamic viscosity coefficient, the subscript t indicates a time derivative, and $\vec{F}$ represents any external body forces. We allow both ρ and μ to vary spatially. At solid boundaries, the no-slip condition applies, which dictates that the fluid velocities match those of the solid boundary, $\vec{u}_{BC}$:

$$\vec{u} = \vec{u}_{BC} \tag{4.4}$$

At a free surface, the fluid is subject to the constraint that the traction $\vec{T}$ applied at the surface is zero:

$$\vec{T} = (-pI + \tau)\vec{n} = \vec{0} \tag{4.5}$$

In the above, $\vec{n}$ is taken to be the outward normal of the free surface and I represents the identity matrix.

Typical numerical methods for Stokes flow can be interpreted as combining a classic pressure projection with an implicit step of viscous forces. We will therefore consider each of these two sub-problems independently to illustrate the general approach, before turning to the full Stokes problem.

4.2.1 A Note on the Simplified Stokes Equations

Often, for situations where the viscosity is spatially constant, some manipulations are carried out to reduce the contribution of viscosity to a simpler form. Specifically:

$$\begin{aligned} \nabla \cdot \tau &= \nabla \cdot \left(\mu(\nabla \vec{u} + (\nabla \vec{u})^T)\right) && (4.6)\\ &= \mu \nabla \cdot \nabla \vec{u} + \mu \nabla \cdot (\nabla \vec{u})^T && (4.7)\\ &= \mu \nabla \cdot \nabla \vec{u} + \mu \nabla (\nabla \cdot \vec{u}) && (4.8)\\ &= \mu \nabla \cdot \nabla \vec{u} && (4.9) \end{aligned}$$

where a simple vector calculus identity and the divergence free property of $\vec{u}$ have been used to eliminate the second term on the right hand side. This reduces the contribution of viscosity to a simple Laplace operator applied to each component of velocity independently. However, the natural boundary conditions of this modified problem are quite different from those of the original problem, as discussed by Limache et al. [LIRO07, LSDI08]; if care is not taken this can lead to non-physical solutions, particularly for free surface flows. We therefore follow Batty & Bridson [BB08] in preferring the fully general form of the viscous terms outlined previously.

4.3 A Variational Formulation for Pressure Projection

The first sub-problem of the Stokes equation that we consider is the enforcement of incompressibility. This is exactly the classic pressure projection method introduced

by Chorin [Cho68]. After discretizing in time, pressure projection has the form:

$$\frac{\rho}{\Delta t}(\vec{u}-\vec{u}^*) = -\nabla p \tag{4.10}$$

$$\nabla \cdot \vec{u} = 0 \tag{4.11}$$

where $\vec{u}$ is the resulting divergence free velocity field, $\vec{u}^*$ is the input divergent velocity field, and Δt is the size of the time step. This system can be arrived at by extremizing the following functional:

$$\max_p \min_{\vec{u}} \iiint_{\Omega_L} \frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2 - \Delta t\, p \nabla \cdot \vec{u} \tag{4.12}$$

where the domain of integration is over the liquid (non-air) region, Ω_L. This can be interpreted as a constrained least squares problem, in which we find the nearest velocity field $\vec{u}$ to the input velocity $\vec{u}^*$ while satisfying the divergence free constraint, enforced by a Lagrange multiplier p. Calculus of variations can be used to show that the optimality conditions for this problem yield precisely the equations of the original problem, with the Dirichlet condition $p = 0$ at the free surface enforced as a natural boundary condition. Therefore, if we discretize the integral appropriately and solve the resulting optimization problem, we will arrive at the correct solution, without needing to explicitly build the boundary conditions into the discretization. This is a key element of our method, which enables the handling of complex boundary conditions with relative ease.

An alternate variational form for the pressure projection is:

$$\max_p \min_{\vec{u}} \iiint_{\Omega_F} \frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2 + \Delta t \nabla p \cdot \vec{u} \tag{4.13}$$

where the integration is now performed over the fluid (non-solid) region, Ω_F. This enforces the same equations over the domain, but instead of free surface boundary

conditions, it naturally includes the Neumann free-slip boundary condition $\vec{u} \cdot \vec{n} = 0$ along solid boundaries. Note that the two functionals differ essentially by an integration by parts on the rightmost term, except that the resulting boundary integral term is discarded.

We highlight here the relationship to the variational approach of Batty et al. [BBB07]. If we take the standard expression for the velocity update $\vec{u} = \vec{u}^* - \frac{\Delta t}{\rho} \nabla p$, and substitute it into the solid boundary formulation (4.13), we arrive at a functional equivalent to that presented by Batty et al. A similar transformation can be applied to the free surface functional (4.12); when discretized this yields a scheme similar to the ghost fluid method of Enright et al. [ENGF03].

4.3.1 Discretization

A common approach to such a problem would be to optimize the particular functional in question, and discretize the resulting PDE. However, this would require us to explicitly handle potentially difficult boundary conditions, which is precisely what we would like to avoid. Instead, we follow Batty et al. [BBB07, BB08], in discretizing the variational principle first, and only then finding the extremum, so that the boundary conditions are enforced naturally.

We discretize the derivatives with centred finite differences on the classic staggered (MAC) grid. This arrangement is illustrated in 2D and 3D in Figures 4.1 and 4.2, respectively. We then approximate the integrals by simply scaling each term by the fractional volume of material in a cell-sized region surrounding the appropriate sample point. For example, terms that lie on faces are scaled by the volume of fluid in a square (or cubic) control volume surrounding the face centre. These control volumes are illustrated in Figure 4.3. For use in free surface boundary conditions, we will need to estimate the fraction that is interior to the liquid (ie. not air), in-

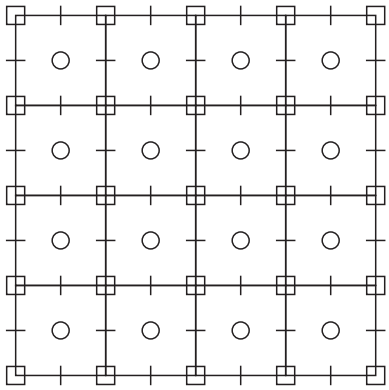


Figure 4.1: The standard staggered pressure-velocity grid layout in 2D, with stress components added. Circles indicate the locations of pressure and diagonal stress components (τ_{xx}). Dashes across cell faces indicate horizontal and vertical velocity components. Small squares at cell corners indicate the locations of off-diagonal stress components (τ_{xy}).

dicated by weights W_L. Later we will also need the corresponding air fractions, $W_A = I - W_L$. Likewise, for solid boundary conditions we estimate the fraction W_F of a cell that is inside the fluid (ie. not solid), and its complementary solid fraction, $W_S = I - W_F$. These volume fractions are illustrated in Figure 4.4, and can be straightforwardly estimated from a level set representation using a method analogous to that of Min & Gibou [MG07].

In equation (4.12), we are integrating over the liquid region Ω_L, so we must use weighting terms with subscript L. The first term of the equation consists of velocity data that lies on faces, so we estimate the integral using fractional volumes associated to each velocity face sample. We indicate this using superscripts on the weights to indicate the type of control volume being considered; in this case the weight used should be W_L^u. Similarly, the second term consists of pressures and divergences that are conceptually located at cell centres, so we use cell-centred weights, W_L^p. With these choices, the discrete pressure problem with free surface

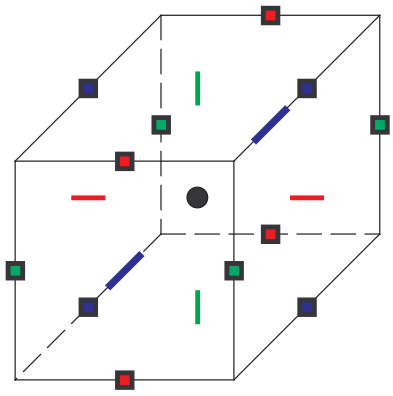


Figure 4.2: The standard staggered pressure-velocity grid layout in 3D, with stress components added. The black circle indicates the location of pressure and diagonal stress components (τ_{xx}, τ_{yy}). The colored squares on cell edges indicate the locations of off-diagonal stress components (τ_{yz} is red, τ_{xz} is green, τ_{xy} is blue). Dashes across cell faces indicate velocity components (u is red, v is green, w is blue).

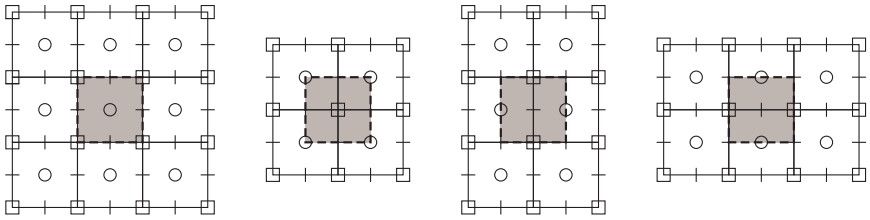


Figure 4.3: Control volumes around each sample location in 2D. From left to right: pressure and diagonal stress control volume, off-diagonal stress control volume, horizontal velocity control volume, vertical velocity control volume.

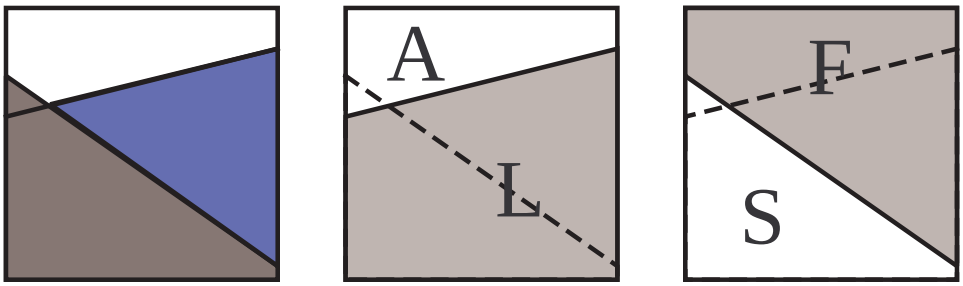


Figure 4.4: An illustration of volume fraction regions for solid and free surface weighting. Left: A geometric scenario in which a solid (dark gray) meets a body of liquid (blue) in the presence of air (white). Note the air-liquid interface extrapolated into the solid. Middle: The volume fraction region for a liquid (ie. non-air) weight, W_L, shown in gray, and its complementing air fraction W_A shown in white; the presence of the solid is ignored. Right: The volume fraction region for a fluid (ie. non-solid) weight, W_F, shown in gray, and its complementing solid fraction, W_S, shown in white; in this case the position of the liquid-air interface is ignored.

boundaries looks like:

$$\max_p \min_u \frac{1}{2}(u-u^*)^T P W_L^u (u-u^*) + \Delta t\, p^T W_L^p G^T u \tag{4.14}$$

where P is a diagonal matrix of densities per face, W_L^u is the diagonal matrix of liquid fractions per face, and W_L^p is the diagonal matrix of liquid fractions per cell. G is a matrix approximating a discrete gradient, with its negative transpose $-G^T$ being a discrete divergence approximation.

The resulting symmetric indefinite linear system comprising the optimality conditions of this problem are:

$$\begin{pmatrix} \frac{1}{\Delta t} P W_L^u & G W_L^p \\ W_L^p G^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t} P W_L^u u^* \\ 0 \end{pmatrix} \tag{4.15}$$

By discretizing (4.13) in the same manner, we arrive at the discrete problem

for the solid boundary case:

$$\max_p \min_u \frac{1}{2}(u-u^*)^T P W_F^u (u-u^*) + \Delta t u^T W_F^u G p \tag{4.16}$$

and its corresponding linear system:

$$\begin{pmatrix} \frac{1}{\Delta t} P W_F^u & W_F^u G \\ G^T W_F^u & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t} P W_F^u u^* \\ 0 \end{pmatrix} \tag{4.17}$$

where W_F^u is a diagonal matrix of fluid (ie. non-solid) fractions per face. It is vital to note that because this integral is over the fluid region Ω_F, the set of weights used here is different from the previous problem; the former uses the liquid vs. air fractions W_L, and the latter uses the fluid vs. solid fractions, W_F.

It is straightforward to combine the two systems into a single system that handles both solids and free surfaces simultaneously:

$$\begin{pmatrix} \frac{1}{\Delta t} W_F^u P W_L^u & W_F^u G W_L^p \\ W_L^p G^T W_F^u & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t} P W_L^u W_F^u u^* \\ 0 \end{pmatrix} \tag{4.18}$$

By combining the two formulations only at the discrete level, we are able to exploit the natural boundary conditions to handle *both* boundary types together, despite the fact that in each of the two continuous variational formulations only one of the two boundaries can be considered natural.

Since the upper left block of this matrix is diagonal, the Schur complement can be applied to arrive at the usual symmetric positive-definite (SPD) Poisson-type system for pressure alone. The resulting sparse SPD system in this case has the

form:

$$\left(\begin{array}{c} \end{array} \Delta t W_L^p G^T P^{-1} W_L^{u-1} W_F^u G W_L^p \begin{array}{c} \end{array} \right) \left(\begin{array}{c} \end{array} p \begin{array}{c} \end{array} \right) = \left(\begin{array}{c} \end{array} W_L^p G^T W_F^u u^* \begin{array}{c} \end{array} \right) \tag{4.19}$$

Our results indicate that this method achieves approximately first order convergence of both velocity and pressure in L^1. In L^∞, we see first order convergence of pressure, though the velocity fails to converge.

As previously noted, this approach is closely related to the works of Gibou et al. [GFCK02] and Ng et al. [NMG09], which make use of ghost fluid and finite volume ideas for free surface and solid wall boundary conditions, respectively. Our results indicate that the variational formulation does not achieve the second order convergence of pressure that these existing methods do. However, our approach does point the way to simple discretizations that are straightforwardly extensible to the more complex viscosity and Stokes problems in which ensuring symmetry, positive-definiteness, and proper boundary conditions becomes quite difficult, particularly in three dimensions.

4.4 A Variational Formulation for Viscosity

The second sub-problem of the Stokes equations that we consider is the integration of viscous forces, ignoring for the moment the influence of pressure. This can be a useful tool in its own right: there are various fractional step algorithms for the Navier-Stokes equations that solve for viscous forces in a separate step. For example, Ng et al. have used the ghost fluid method for an implicit discretization of the Laplacian form of viscosity (4.9) on irregular domains [NMG09]. Batty & Bridson [BB08] introduced a similar technique that has the added benefit of handling spatially varying viscosity and density, while also supporting correct free

surface boundary conditions. This section introduces a closely related method; the key distinctions are that our new formulation is expressed in terms of both stress and velocity, rather than velocity alone, and that it also handles solid boundary conditions naturally.

When discretized in time using first order backward Euler, the fully general viscosity problem has the form:

$$\frac{\rho}{\Delta t}(\vec{u}-\vec{u}^*) = \nabla\cdot\tau \tag{4.20}$$

$$\tau = 2\mu\left(\frac{\nabla\vec{u}+(\nabla\vec{u})^T}{2}\right) \tag{4.21}$$

where $\vec{u}$ is the updated velocity after applying viscous forces and $\vec{u}^*$ is the input velocity. Similar to the pressure projection problem, we will consider two variational formulations for this problem, which differ by an integration by parts. The first naturally enforces the free surface traction boundary condition $\vec{T}=\tau\vec{n}=\mu(\nabla\vec{u}+(\nabla\vec{u})^T)\vec{n}=\vec{0}$:

$$\max_{\tau}\min_{\vec{u}}\iiint_{\Omega_L}\frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2+\Delta t\,\tau:\left(\frac{\nabla\vec{u}+(\nabla\vec{u})^T}{2}\right)-\frac{\Delta t}{4\mu}\|\tau\|_F^2 \tag{4.22}$$

The second yields the solid boundary condition $\vec{u}=\vec{0}$ at the solid wall:

$$\max_{\tau}\min_{\vec{u}}\iiint_{\Omega_F}\frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2-\Delta t\vec{u}\cdot(\nabla\cdot\tau)-\frac{\Delta t}{4\mu}\|\tau\|_F^2 \tag{4.23}$$

If we substitute $\tau=\mu(\nabla\vec{u}+(\nabla\vec{u})^T)$ into (4.22) we can eliminate the stress variable, and arrive at the method proposed by Batty & Bridson for the free surface viscosity problem [BB08]. (Another important possibility is to eliminate velocity and solve strictly for stress. We will return to this idea when considering the discretized problem.)

4.4.1 Discretization

We will again make use of a staggered grid for our discretization. However, to handle viscosity we also need to place the components of the deviatoric stress tensor on our grid. The most natural way to do this is to place diagonal components of the stress tensor at cell centres, and off-diagonal components on cell edges (nodes in two dimensions), as also depicted in Figures 4.1 and 4.2. Straightforward centred differencing can then be used to compute the required derivatives in the correct locations. This was first proposed by Darwish et al. [DWB92] for two dimensions, and later extended to three dimensions by Mompean & Deville [MD97]. While this approach has traditionally been applied to non-Newtonian fluids in which the constitutive equations are more complex, we find its simplicity and elegance useful for the purely Newtonian flows we consider.

With these new sample locations, we will also require corresponding volume fractions to discretize the variational forms. We use the superscript τ to indicate volume fractions associated with the stress locations at both cell centres and cell edges. For example W_L^τ represents a diagonal matrix of liquid volume fractions, with one entry per stress sample.

Because we are looking at strictly the deviatoric stress τ, and ignoring pressure, we can assume $\mathrm{Tr}(\tau) = 0$ which in 2D implies that $\tau_{xx} = -\tau_{yy}$. We can therefore simplify the necessary computations by solving just for τ_{xx} rather than both quantities; in the final linear system, this will yield equations of the form $\tau_{xx} = \mu\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)$, thereby retaining symmetry. Similar transformations apply in 3D to eliminate $\tau_{zz} = -(\tau_{xx} + \tau_{yy})$, yielding $\tau_{xx} + \frac{1}{2}\tau_{yy} = \mu\left(\frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right)$ and $\tau_{yy} + \frac{1}{2}\tau_{xx} = \mu\left(\frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right)$. Naturally, the stress tensor will also be symmetric, so that $\tau_{xy} = \tau_{yx}$, $\tau_{xz} = \tau_{zx}$, and $\tau_{yz} = \tau_{zy}$, which further reduces the number of variables to be computed.

With this grid layout in mind, the discrete free surface viscosity problem is:

$$\max_{\tau}\min_{u}\frac{1}{2}(u-u^*)^T P W_L^u (u-u^*)+\Delta t\,\tau^T W_L^\tau D u-\frac{\Delta t}{4}\tau^T M^{-1} W_L^\tau \tau \tag{4.24}$$

where P is a diagonal matrix of densities per velocity sample, M is a diagonal matrix of viscosity coefficients per stress sample, and D is a matrix representing a discrete deformation rate operator, whose negative transpose is the tensor divergence (ie. $Du \approx \frac{1}{2}(\nabla u + (\nabla u)^T)$ and $-D^T\tau \approx \nabla\cdot\tau$). The resulting linear system is:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_L^u & D^T W_L^\tau \\ W_L^\tau D & -\frac{1}{2}M^{-1}W_L^\tau \end{pmatrix}\begin{pmatrix} u \\ \tau \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_L^u u^* \\ 0 \end{pmatrix} \tag{4.25}$$

Solving this system will give a solution to the viscosity problem with the difficult free surface traction boundary condition enforced naturally.

Similarly, the discrete viscosity problem with solid boundaries enforced naturally is:

$$\max_{\tau}\min_{u}\frac{1}{2}(u-u^*)^T P W_F^u (u-u^*)+\Delta t\,u^T W_F^u D^T \tau-\frac{\Delta t}{4}\tau^T M^{-1} W_F^\tau \tau \tag{4.26}$$

with corresponding linear system:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_F^u & W_F^u D^T \\ DW_F^u & -\frac{1}{2}M^{-1}W_F^\tau \end{pmatrix}\begin{pmatrix} u \\ \tau \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_F^u u^* \\ 0 \end{pmatrix} \tag{4.27}$$

As we did for the pressure problem, we can combine the two discrete systems to handle both boundary conditions at once:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_F^u W_L^u & W_F^u D^T W_L^\tau \\ W_L^\tau D W_F^u & -\frac{1}{2}M^{-1}W_L^\tau W_F^\tau \end{pmatrix}\begin{pmatrix} u \\ \tau \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_L^u W_F^u u^* \\ 0 \end{pmatrix} \tag{4.28}$$

However, this system is still indefinite; for numerical efficiency we would prefer to solve a symmetric positive-definite system. Conveniently, both the upper-left and lower-right blocks are diagonal so we can straightforwardly take the Schur complement of either to arrive at a sparse SPD system for stress or velocity alone. The SPD system for velocity is:

$$\left(\begin{array}{c} \frac{1}{\Delta t} P W_F^u W_L^u + 2 W_F^u D^T M W_F^{\tau^{-1}} W_L^\tau D W_F^u \end{array} \right) \left(\begin{array}{c} u \end{array} \right) = \left(\begin{array}{c} \frac{1}{\Delta t} P W_L^u W_F^u u^* \end{array} \right) \quad (4.29)$$

This is effectively the same system solved by Batty & Bridson [BB08] for the free surface case, except that with our modifications the solid boundaries are also enforced naturally.

The SPD system for stress is:

$$\left(\begin{array}{c} \frac{1}{2} M^{-1} W_F^\tau W_L^\tau + \Delta t W_L^\tau D P^{-1} W_L^{u^{-1}} W_F^u D^T W_L^\tau \end{array} \right) \left(\begin{array}{c} \tau \end{array} \right) = \left(\begin{array}{c} W_L^\tau D W_F^u u^* \end{array} \right) \quad (4.30)$$

The method we have just described for the viscosity problem results in different convergence orders depending on whether the boundary condition under consideration is a free surface or a solid wall. The free surface-only problem exhibits second order convergence of stresses in L^1 and first order convergence of stresses in L^∞, while velocity convergence at first order in L^1 and not at all in L^∞. The solid wall-only problem is essentially reversed, with second order convergence in velocity in L^1 and first order convergence of velocity in L^∞, but only first and zeroth order convergence of stress in L^1 and L^∞, respectively. Notice that the more accurate variable in each case is the one with the Dirichlet condition applied; in the free surface case stress is constrained while in the solid case the Dirichlet condition is on velocity.

4.5 A Variational Formulation for Stokes Flow

Given the above methods for the pressure and viscosity problems, it is straightforward to combine them in order to address the full Stokes problem, again using a backwards Euler time discretization:

$$\frac{\rho}{\Delta t}(\vec{u}-\vec{u}^*) = \nabla\cdot\tau - \nabla p \tag{4.31}$$

$$\nabla\cdot\vec{u} = 0 \tag{4.32}$$

$$\tau = \mu(\nabla\vec{u}+(\nabla\vec{u})^T) \tag{4.33}$$

Our Stokes formulation with free surface boundaries is:

$$\max_{p,\tau}\min_{\vec{u}} \iiint_{\Omega_L} \frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2 - \Delta t\, p\nabla\cdot\vec{u} + \Delta t\,\tau : \left(\frac{\nabla\vec{u}+(\nabla\vec{u})^T}{2}\right) - \frac{\Delta t}{4\mu}\|\tau\|_F^2 \tag{4.34}$$

This implicitly enforces the zero traction condition (4.5) that couples pressure and deviatoric stress together at the boundary; it would otherwise be quite difficult to handle.

Our Stokes formulation for static solid boundaries is:

$$\max_{p,\tau}\min_{\vec{u}} \iiint_{\Omega_F} \frac{\rho}{2}\|\vec{u}-\vec{u}^*\|^2 + \Delta t\vec{u}\cdot(\nabla p - \nabla\cdot\tau) - \frac{\Delta t}{4\mu}\|\tau\|_F^2 \tag{4.35}$$

This yields the same Stokes equations, except that the natural boundary condition is that of a static solid wall.

4.5.1 Discretization

We use the same staggered grid arrangement as the preceding schemes, with stress, pressure and velocity components distributed at appropriate locations. The discrete minimization forms can be arrived at in the same manner as for the pressure and

viscosity problems. The discrete Stokes problem with free surface boundaries is:

$$\max_{p,\tau}\min_{u}\frac{1}{2}(u-u^*)^T PW_L^u(u-u^*)+\Delta t\, pW_L^p G^T u+\Delta t\,\tau^T W_L^\tau Du-\frac{\Delta t}{4}\tau^T M^{-1}W_L^\tau\tau \tag{4.36}$$

The linear system for the free surface Stokes problem is:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_L^u & D^T W_L^\tau & GW_L^p \\ W_L^\tau D & -\frac{1}{2}M^{-1}W_L^\tau & 0 \\ W_L^p G^T & 0 & 0 \end{pmatrix}\begin{pmatrix} u \\ \tau \\ p \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_L^u u^* \\ 0 \\ 0 \end{pmatrix} \tag{4.37}$$

The discrete Stokes problem with solid boundaries is:

$$\max_{\tau}\min_{u}\frac{1}{2}(u-u^*)^T PW_F^u(u-u^*)+\Delta t u^T W_F^u(Gp+D^T\tau)-\frac{\Delta t}{4}\tau^T M^{-1}W_F^\tau\tau \tag{4.38}$$

The linear system for the solid wall Stokes problem is:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_F^u & W_F^u D^T & W_F^u G \\ DW_F^u & -\frac{1}{2}M^{-1}W_F^\tau & 0 \\ G^T W_F^u & 0 & 0 \end{pmatrix}\begin{pmatrix} u \\ \tau \\ p \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_F^u u^* \\ 0 \\ 0 \end{pmatrix} \tag{4.39}$$

The combined linear system which handles both free surfaces and solid boundaries is:

$$\begin{pmatrix} \frac{1}{\Delta t}PW_F^u W_L^u & W_F^u D^T W_L^\tau & W_F^u GW_L^p \\ W_L^\tau DW_F^u & -\frac{1}{2}M^{-1}W_F^\tau W_L^\tau & 0 \\ W_L^p G^T W_F^u & 0 & 0 \end{pmatrix}\begin{pmatrix} u \\ \tau \\ p \end{pmatrix}=\begin{pmatrix} \frac{1}{\Delta t}PW_F^u W_L^u u^* \\ 0 \\ 0 \end{pmatrix} \tag{4.40}$$

For these Stokes systems, eliminating stress using a Schur complement on the centre block yields the symmetric indefinite system typically associated with the

Stokes problem, which requires solving for pressure and velocity. We can instead eliminate velocity, which results in a sparse symmetric *positive-definite* system for pressure and stress that is more amenable to fast solvers such as preconditioned conjugate gradient methods, domain decomposition, etc.

The symmetric positive-definite form is:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} \begin{pmatrix} \tau \\ p \end{pmatrix} = \begin{pmatrix} W_L^\tau D W_F^u u^* \\ W_L^p G^T W_F^u u^* \end{pmatrix} \tag{4.41}$$

where the blocks of the matrix are

$$\begin{aligned} A_{11} &= \frac{1}{2} M^{-1} W_L^\tau W_F^\tau + \Delta t W_L^\tau D P^{-1} W_L^{u-1} W_F^u D^T W_L^\tau \\ A_{12} &= \Delta t W_L^\tau D P^{-1} W_L^{u-1} W_F^u G W_L^p \\ A_{22} &= \Delta t W_L^p G^T P^{-1} W_L^{u-1} W_F^u G W_L^p \end{aligned}$$

Our results for the Stokes problem consistently indicate first order convergence for all variables in L^1, and first order convergence for velocity in L^∞. Velocity apparently fails to converge in L^∞.

4.6 Non-Homogeneous Boundary Conditions

The preceding formulations for the pressure, viscosity, and Stokes problems exploit natural homogeneous boundary conditions to simplify handling of irregular domains on Cartesian grids. However, many practical situations will call for *non-homogeneous* boundary conditions where the boundary values are non-zero. For example, moving solid boundaries will require non-zero boundary velocities to be enforced, as will inflow and outflow boundaries. Similarly, non-zero pressure boundary conditions have been used to support surface tension (eg. [ENGF03]),

and there may also be situations where a boundary with specified traction is useful. To incorporate non-homogeneous boundary values into our framework in a consistent manner, we introduce additional energy terms to account for the work done by the boundary itself. We will discuss the Stokes problem in particular, but the same general approach can be used for the viscosity and pressure problems as well.

4.6.1 Prescribed Pressure and Stress Boundaries

To add a prescribed traction boundary, we need to account for the work done by the traction at the surface. We do this by adding the following boundary term to (4.34):

$$\iint_{\partial\Omega_A} \Delta t \vec{u}(p_{BC}I - \tau_{BC})\vec{n} \tag{4.42}$$

where p_{BC} and τ_{BC} are the prescribed pressure and deviatoric stress, respectively, and $\vec{n}$ in this case is the outward normal with respect to the air (non-liquid) region, Ω_A. (However, only the resulting normal component, ie. surface traction, will be enforced.) This can be converted to a volume integral using integration by parts, to arrive at:

$$\Delta t \iiint_{\Omega_A} p_{BC}\nabla\cdot\vec{u} - \tau_{BC} : \left(\frac{\nabla\vec{u} + (\nabla\vec{u})^T}{2}\right) + \vec{u}\cdot(\nabla p_{BC} - \nabla\cdot\tau_{BC}) \tag{4.43}$$

This ensures that all multiplied quantities are located at matching grid locations, so that the integrals can be estimated without interpolation in a manner consistent with the previously considered functionals. Using W_A terms to indicate the air fraction of a particular control volume, the discretized form is:

$$\Delta t \left(-p_{BC}^T W_A^p G^T u - \tau_{BC}^T W_A^\tau D u + u^T W_A^u (G p_{BC} + D^T \tau_{BC})\right) \tag{4.44}$$

The new terms modify the right hand side of the linear system (4.37), to be-

come:

$$\begin{pmatrix} \frac{1}{\Delta t} P W_L^u u^* + G W_A^p p_{BC} - W_A^u G p_{BC} + D^T W_A^\tau \tau_{BC} - W_A^u D^T \tau_{BC} \\ 0 \\ 0 \end{pmatrix} \tag{4.45}$$

Although the air region may extend far from the actual liquid surface, we only apply modifications to the right hand side for rows of the system in which the matrix has non-zero entries, indicating that there is liquid present. This means that in practice only the interface between the air and liquid regions plays a role.

4.6.2 Prescribed Velocity Boundaries

In the common case of moving solid boundaries with prescribed velocities $\vec{u}_{BC}$, we account for the work done by the solid on the fluid by adding the following term to (4.35):

$$\iint_{\partial \Omega_S} \Delta t \vec{u}_{BC} (pI - \tau) \vec{n} \tag{4.46}$$

where $\vec{n}$ is the outward normal to the solid region, Ω_S. In volume integral form we have:

$$\Delta t \iiint_{\Omega_S} p \nabla \cdot \vec{u}_{BC} - \tau : \left(\frac{\nabla \vec{u}_{BC} + (\nabla \vec{u}_{BC})^T}{2} \right) + \vec{u}_{BC} \cdot (\nabla p - \nabla \cdot \tau) \tag{4.47}$$

Labelling solid fractions W_S and discretizing, we arrive at the following energy term:

$$\Delta t \left(-p^T W_S^p G^T u_{BC} - \tau^T W_S^\tau D u_{BC} + u_{BC}^T W_S^u (Gp + D^T \tau) \right) \tag{4.48}$$

This results in a modification to the right hand side of the linear system (4.39), to become:

$$\begin{pmatrix} \frac{1}{\Delta t} P W_F^u u^* \\ -D W_S^u u_{BC} + W_S^\tau D u_{BC} \\ -G^T W_S^u u_{BC} + W_S^p G^T u_{BC} \end{pmatrix} \tag{4.49}$$

These right hand side modifications are also only applied to rows in which the matrix has valid, non-zero entries.

4.7 Null Space Elimination

An issue that arises with our approach is the presence of null spaces because of the use of overlapping volume weights assigned to different terms of the variational problem. For example, in the pressure problem with free surface boundary conditions, there are volume weights associated with both velocities (face centres) and pressures (cell centres). Cases frequently arise in the discretized system in which a pressure with a non-zero volume weight enforces a divergence constraint on at least one velocity face with zero associated volume. This velocity sample will appear in no other equations because of its zero volume weight, and therefore it can take on an arbitrary value as long as it satisfies the constraint. An example of such a null space scenario is shown in Figure 4.5.

In the final result, only samples with a positive associated volume weight are considered, so the physical solution is not adversely affected. However, such large spurious null spaces can pose problems when applying standard solvers for sparse linear systems; we would therefore like to eliminate them.

To do so, we identify each variable that enforces a relationship on a sample with zero volume weights. For example, in the free surface pressure problem a non-zero weighted pressure is tagged as invalid if it enforces the divergence-free

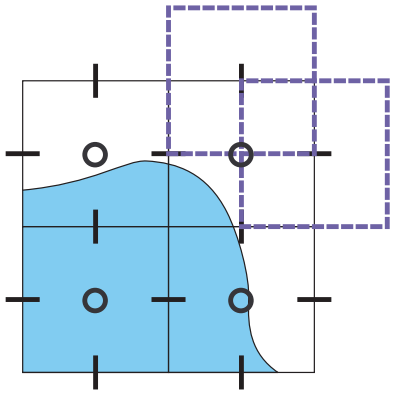


Figure 4.5: A case in which a null space arises in the free surface pressure projection problem. The top right cell contains some liquid, and therefore will have a positive volume weight associated with the divergence constraint on the cell. However, two of its associated velocity face control volumes (indicated by dashed blue squares) contain no liquid and will have zero volume weights. We identify this pressure sample as invalid, and replace it with the value of the boundary condition (eg. $p = 0$), thereby eliminating the null space.

condition on one or more velocity faces with zero weights. Likewise, in the solid pressure problem if a non-zero weighted velocity sample borders a zero-weighted pressure sample, that velocity sample is tagged as invalid. The same approach applies to all the problems we have considered. Once these invalid variables have been identified, they can be straightforwardly eliminated from the linear system, and replaced with the value of the boundary condition, resulting in a modification to the right-hand-side. This reduced set of equations retains symmetry and has the same solution, but no longer suffers from large null spaces.

There can be one additional null space in the pressure and Stokes problems in the case when the fluid domain is completely enclosed by solid walls. Because only the gradient of pressure affects the final velocities, pressure solutions that differ by a constant are effectively equivalent. This rank 1 null space doesn't pose substantial problems, so we have not bothered to eliminate it; if necessary, it could

be removed by arbitrarily fixing one pressure value in the domain.

4.8 Convergence Results

We have verified that the method computes the exact solution for linear problems even for irregular domains. This includes the case of hydrostatic fluid with solid (and possibly free surface) boundaries, as well as rigid translations and rotations of liquid bodies with free surface boundaries. We provide a range of examples below to illustrate the convergence orders achieved by our methods in more difficult scenarios. All of the examples make use of curved boundaries which do not align with the underlying Cartesian grid, and we consider a single time step of an unsteady (time-dependent) solution to the problem in question. The examples make use of our methods for pressure projection, implicit viscosity, and finally the full Stokes problems, in the presence of free surfaces and both static and moving boundaries. We compute both the L^∞ and L^1 error, where our discrete L^1 norm is computed as $\|u_h\|_1 = \sum_i |u_i| h^d$ for a uniform grid spacing $h = \Delta x$ in d spatial dimensions.

For each case, we transformed the linear system to the sparse symmetric positive-definite form as described above, and solved it with the conjugate gradient method preconditioned with overlapping multiplicative Schwarz domain decomposition. The preconditioner was determined purely algebraically, from a simple graph partition of the sparse matrix; we expect the positive-definite form would allow the use of many other black-box solvers as well.

4.8.1 Pressure Projection with Free Surface Boundaries

Our first pressure projection test problem is a disk of liquid centred at the origin, with radius $r = 1$ and density $\rho = 1$, bounded by a free surface. The true solution

Table 4.1: Convergence of pressure projection with free surface

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	4.1348E-002		2.3627E-002	
32^2	2.9268E-002	0.50	5.5243E-003	2.10
64^2	1.5932E-002	0.88	1.3897E-003	1.99
128^2	6.2977E-003	1.34	5.9423E-004	1.23
256^2	3.8238E-003	0.72	2.4308E-004	1.29
512^2	1.8285E-003	1.06	1.3022E-004	0.90
1024^2	9.7973E-004	0.90	6.7759E-005	0.94
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	3.5361E-002		1.5221E-002	
32^2	8.7156E-002	-1.30	1.2950E-002	0.23
64^2	1.1520E-001	-0.40	6.8599E-003	0.92
128^2	1.2895E-001	-0.16	3.8847E-003	0.82
256^2	1.4577E-001	-0.18	1.9743E-003	0.98
512^2	1.4454E-001	0.01	1.0496E-003	0.91
1024^2	1.4443E-001	0.00	4.9698E-004	1.08

computed over a time step of length $\Delta t = 1$ is:

$$p = x^2 + y^2 - 1 \tag{4.50}$$

$$\vec{u}_{final} = (2xy, -y^2)^T \tag{4.51}$$

$$\vec{u}_{input} = \vec{u}_{final} + \nabla p = (2xy - 2x, -y^2 - 2y)^T \tag{4.52}$$

The pressure satisfies $p = 0$ on the free surface (at $r = 1$). The convergence results are shown in Table 4.1 and Figure 4.6.

4.8.2 Pressure Projection with Solid Wall Boundaries

Our solid wall test is a disk of fluid centred at the origin, with radius $r = 1$ and density $\rho = 1$, bounded by a static solid wall. The true solution computed over a

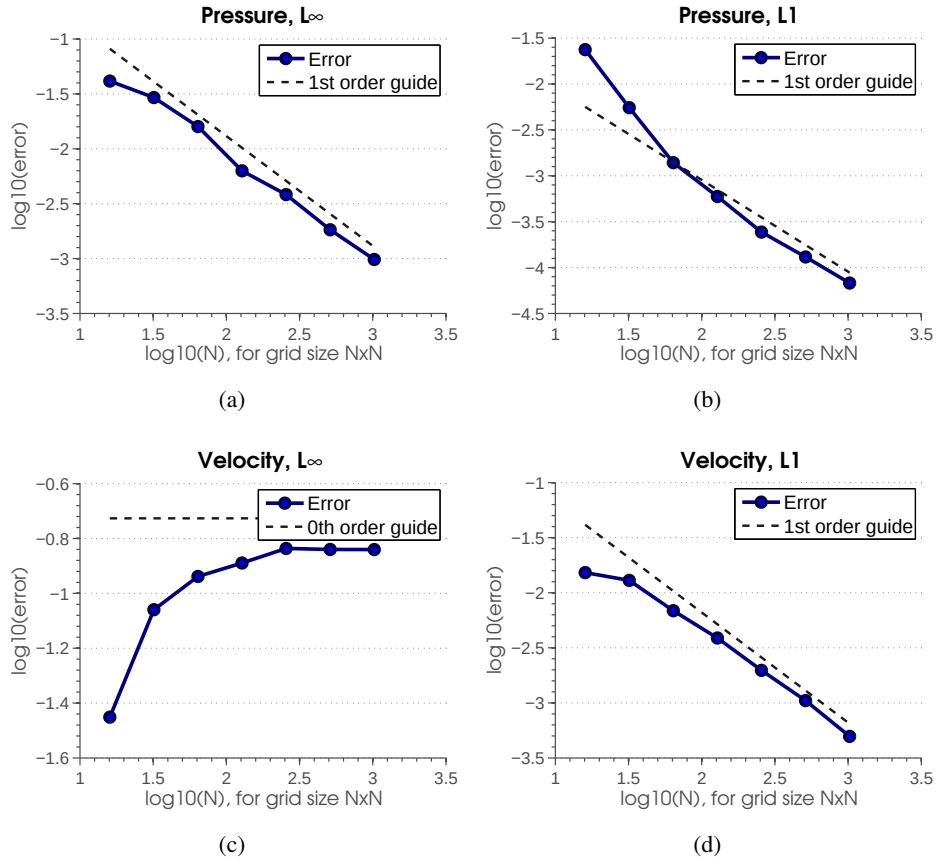


Figure 4.6: Convergence graphs for the pressure problem with free surface boundaries.

Table 4.2: Convergence of pressure projection with solid walls

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	4.7124E-003		2.6180E-003	
32^2	1.1997E-003	1.97	6.8267E-004	1.94
64^2	1.4278E-003	-0.25	2.9449E-004	1.21
128^2	6.4875E-004	1.14	9.4785E-005	1.64
256^2	2.8184E-004	1.20	3.7782E-005	1.33
512^2	4.2031E-004	-0.58	3.1794E-005	0.25
1024^2	1.5098E-004	1.48	1.2412E-005	1.36
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	8.7558E-002		2.5277E-002	
32^2	6.0128E-002	0.54	8.6143E-003	1.55
64^2	5.7966E-002	0.05	5.5291E-003	0.64
128^2	8.9706E-002	-0.63	2.8418E-003	0.96
256^2	1.0821E-001	-0.27	1.4015E-003	1.02
512^2	1.1635E-001	-0.10	8.5087E-004	0.72
1024^2	1.0786E-001	0.11	3.9986E-004	1.09

time step of length $\Delta t = 1$ is:

$$p = xy^3 \tag{4.53}$$

$$\vec{u}_{final} = (y,-x)^T \tag{4.54}$$

$$\vec{u}_{input} = \vec{u}_{final} + \nabla p = (y+y^3, -x+3xy^2)^T \tag{4.55}$$

The final velocity satisfies $\vec{u}\cdot\vec{n} = 0$ on the surface, since it is a rigid rotation. The convergence results are shown in Table 4.2 and Figure 4.7.

4.8.3 Implicit Viscosity with Free Surface Boundaries

Our viscous free surface test case consists of an annulus of fluid centred at the origin with inner radius $r = 0.5$, outer radius $r = 1$, density $\rho = 1$, and viscosity $\mu = 0.1$, bounded by inner and outer free surfaces. The final velocity after a timestep

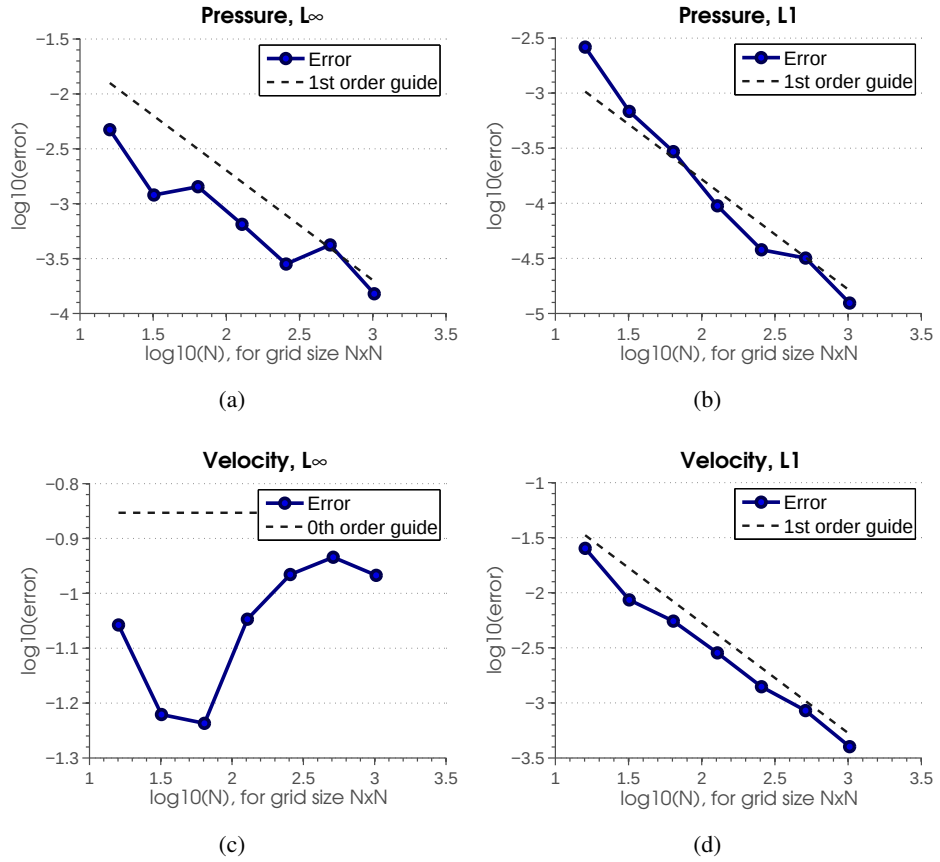


Figure 4.7: Convergence graphs for the pressure problem with solid wall boundaries.

Table 4.3: Convergence of viscosity with free surface

Grid	$\|\tau_{xx} - \tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx} - \tau_{xx}^h\|_1$	Order
16^2	1.2916E-003		8.7280E-004	
32^2	1.0563E-003	0.29	2.0309E-004	2.10
64^2	4.7442E-004	1.15	6.3926E-005	1.67
128^2	1.7365E-004	1.45	1.3454E-005	2.25
256^2	9.2763E-005	0.90	3.2821E-006	2.04
512^2	4.6659E-005	0.99	7.2463E-007	2.18
1024^2	2.4297E-005	0.94	2.1243E-007	1.77
Grid	$\|\tau_{xy} - \tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy} - \tau_{xy}^h\|_1$	Order
16^2	1.6220E-003		9.9892E-004	
32^2	6.6232E-004	1.29	2.5265E-004	1.98
64^2	2.8901E-004	1.20	6.4299E-005	1.97
128^2	1.3655E-004	1.08	1.5720E-005	2.03
256^2	6.4511E-005	1.08	3.4266E-006	2.20
512^2	3.3822E-005	0.93	7.7302E-007	2.15
1024^2	1.7097E-005	0.98	2.0550E-007	1.91
Grid	$\|u - u^h\|_\infty$	Order	$\|u - u^h\|_1$	Order
16^2	5.9857E-003		3.7023E-003	
32^2	5.4131E-003	0.15	1.3220E-003	1.49
64^2	3.5381E-003	0.61	5.1563E-004	1.36
128^2	3.9775E-003	-0.17	2.1451E-004	1.27
256^2	3.9989E-003	-0.01	1.0049E-004	1.09
512^2	3.8607E-003	0.05	4.6637E-005	1.11
1024^2	3.7256E-003	0.05	2.0294E-005	1.20

of length $\Delta t = 1$ is:

$$\vec{u}_{final} = (\frac{r^3}{3} - \frac{3r^2}{4} + \frac{r}{2})(-y,x)^T \tag{4.56}$$

This velocity field satisfies the zero traction boundary condition on the inner and outer surfaces. The stresses and input velocities can be derived straightforwardly from equations (4.20) and (4.21). The convergence results are shown in Table 4.3 and Figure 4.8.

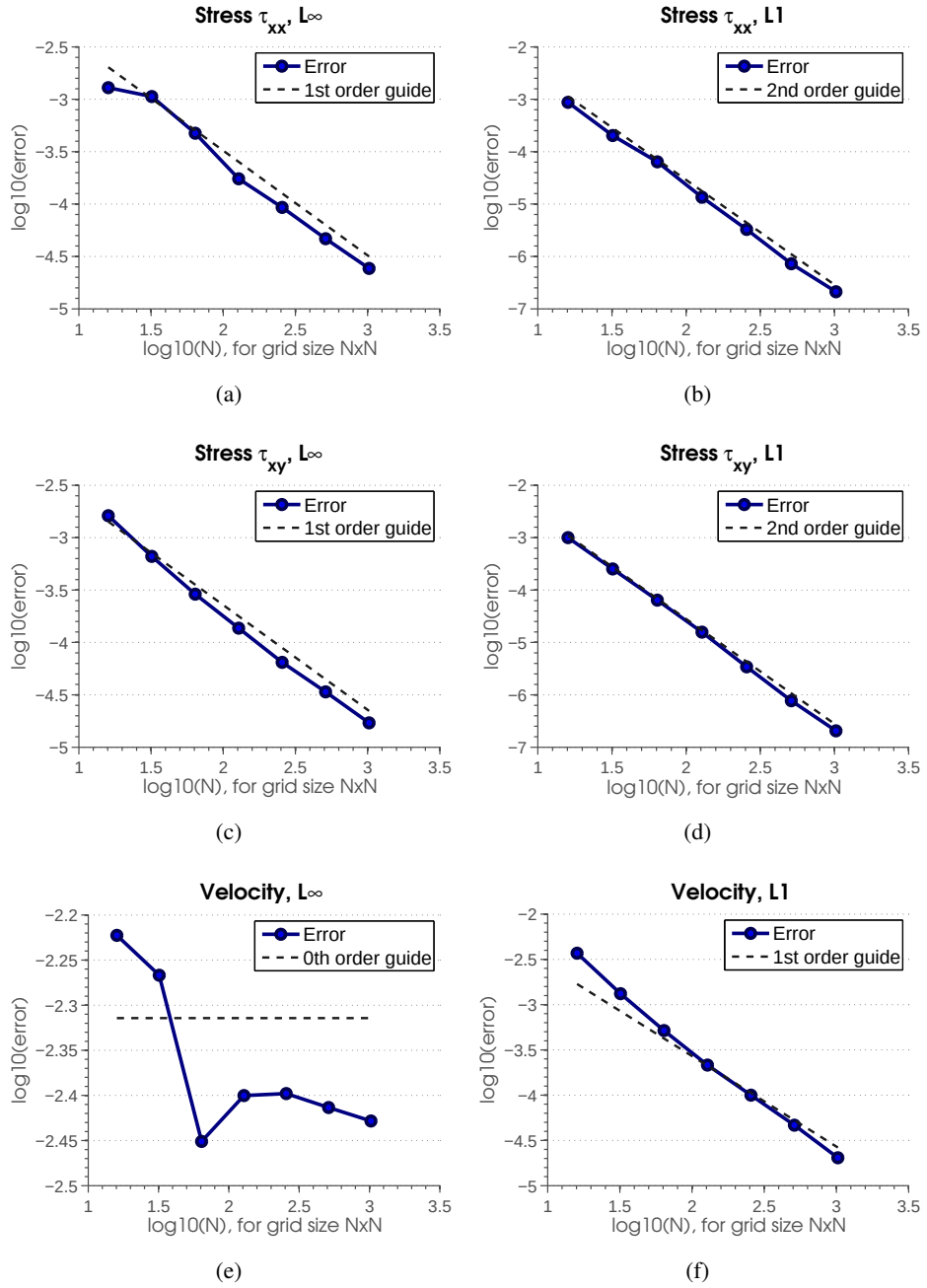


Figure 4.8: Convergence graphs for the viscosity problem with free surface boundaries.

4.8.4 Implicit Viscosity with Solid Wall Boundaries

Our viscous solid wall test case consists of an annulus of fluid centred at the origin, with inner radius $r = 0.5$, outer radius $r = 1$, density $\rho = 1$, and viscosity $\mu = 0.1$, bounded by inner and outer static solid walls. The final velocity after a timestep of length $\Delta t = 1$ is:

$$\vec{u}_{final} = \frac{(r-1)(r-0.5)}{r}(-y,x)^T \tag{4.57}$$

Clearly $\vec{u} = 0$ is satisfied at the solid boundaries. The stresses and input velocities can be derived from equations (4.20) and (4.21). The convergence results are shown in Table 4.4 and Figure 4.9.

4.8.5 Stokes Flow with Free Surface Boundaries

Our free surface Stokes test case is a fluid disk of radius $r = 0.75$ centred at the origin, with density $\rho = 1$ and viscosity $\mu = 0.1$, computed over a timestep $\Delta t = 1$. For simplicity of presentation, we describe the final velocity field in terms of a streamfunction, ψ, where the velocity field can be derived as $\vec{u}_{final} = \nabla \times \psi$. This also guarantees that the velocity field is divergence free. The streamfunction is:

$$\psi = \frac{128}{81} r^4 \cos(2\theta)\cos(\sqrt{3}\ln r)\left(15 - 30r + 16r^2\right) \tag{4.58}$$

This is a non-trivial velocity field designed to fulfill the free surface zero traction condition at $r = 1$, smoothly blended into a zero velocity at the origin ($r = 0$). The zero traction condition (4.5) enforces a relationship between the surface pressure and the viscous stress resulting from this velocity field. To satisfy this condition,

Table 4.4: Convergence of viscosity with solid walls

Grid	$\|\tau_{xx} - \tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx} - \tau_{xx}^h\|_1$	Order
16^2	2.9011E-003		2.5028E-003	
32^2	2.5539E-003	0.18	1.2913E-003	0.95
64^2	2.2107E-003	0.21	4.8423E-004	1.42
128^2	2.5849E-003	-0.23	2.2124E-004	1.13
256^2	2.2789E-003	0.18	1.0577E-004	1.06
512^2	2.3564E-003	-0.05	4.9623E-005	1.09
1024^2	2.3279E-003	0.02	2.5335E-005	0.97
Grid	$\|\tau_{xy} - \tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy} - \tau_{xy}^h\|_1$	Order
16^2	2.5159E-003		2.2478E-003	
32^2	3.1511E-003	-0.32	1.3944E-003	0.69
64^2	2.8531E-003	0.14	5.0453E-004	1.47
128^2	2.4778E-003	0.20	2.3010E-004	1.13
256^2	3.8106E-003	-0.62	1.0868E-004	1.08
512^2	4.0953E-003	-0.10	5.5941E-005	0.96
1024^2	4.0873E-003	0.00	2.7558E-005	1.02
Grid	$\|u - u^h\|_\infty$	Order	$\|u - u^h\|_1$	Order
16^2	1.5236E-002		9.7798E-003	
32^2	6.8872E-003	1.15	2.9812E-003	1.71
64^2	2.8205E-003	1.29	6.0274E-004	2.31
128^2	1.4400E-003	0.97	1.7162E-004	1.81
256^2	7.3699E-004	0.97	3.8508E-005	2.16
512^2	3.4658E-004	1.09	8.4925E-006	2.18
1024^2	1.8391E-004	0.91	2.6623E-006	1.67

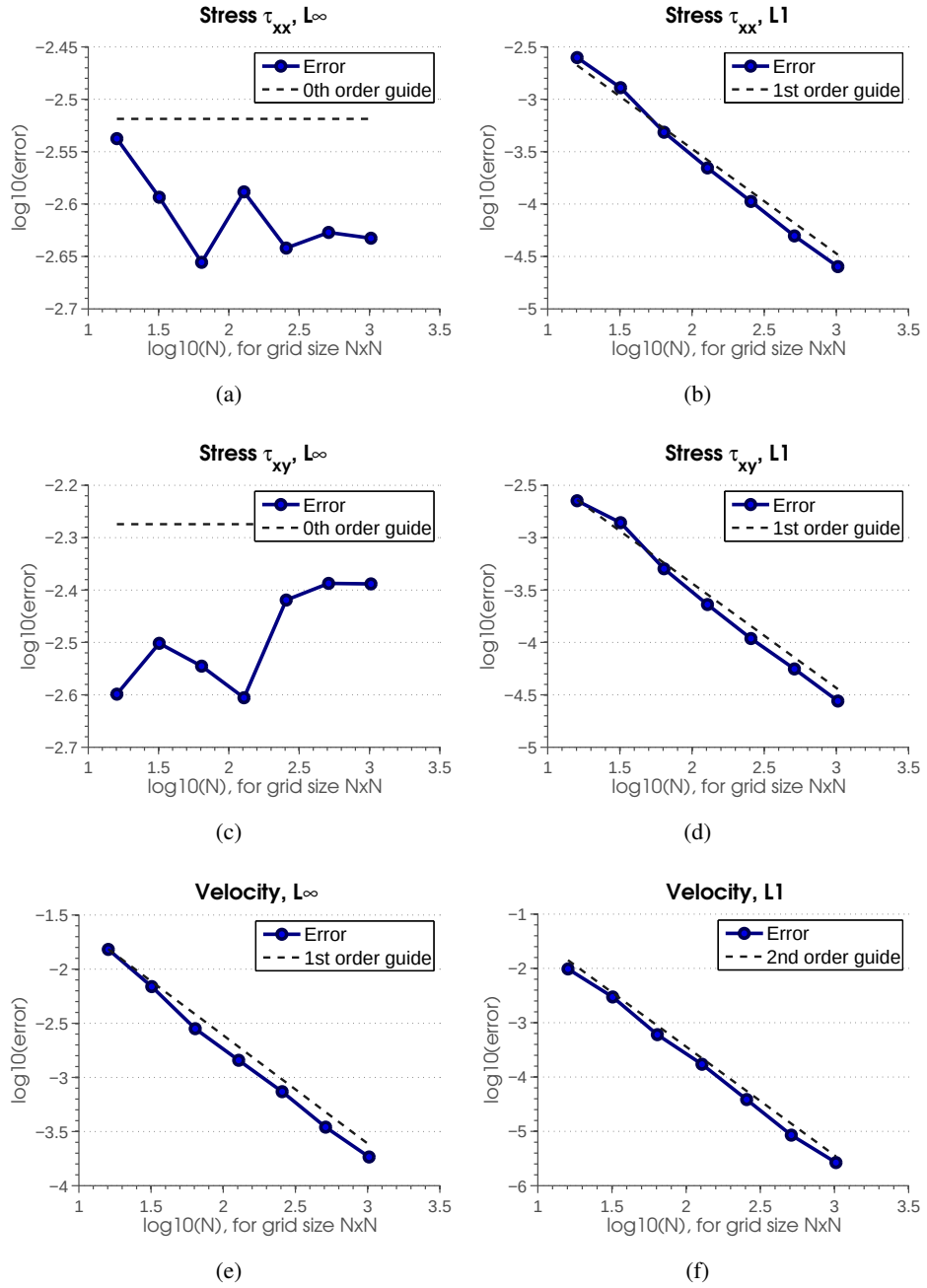


Figure 4.9: Convergence graphs for the viscosity problem with solid wall boundaries.

we use the following expression for pressure:

$$p = \frac{512\sqrt{3}}{81} r^2 \mu \sin(2\theta) \sin(\sqrt{3}\ln r)(15 - 30r + 16r^2) \tag{4.59}$$

The pressure in this expression will be non-zero at the interface; any method to solve this problem will need to correctly handle the coupling between pressure and viscous stresses. From this information, the expressions for the input velocity and final stresses can be derived using equations (4.31)-(4.33). We used a computer algebra system for this purpose. The convergence results are shown in Table 4.5 and Figure 4.10.

4.8.6 Stokes Flow with Solid Wall Boundaries

Our Stokes solid boundary test case is an annulus centred at the origin with inner radius $r = 0.5$, outer radius $r = 1$, density $\rho = 1$, and viscosity $\mu = 0.1$, computed over a timestep $\Delta t = 1$. Inner and outer boundaries are static solids. We will again use a streamfunction ψ to dictate our velocity field and ensure it is divergence free:

$$\psi = 256r^4 - 768r^3 + 832r^2 - 384r + 64 \tag{4.60}$$

For pressure, we use:

$$p = r^2 \cos(\theta) \sin(\theta) \tag{4.61}$$

Equations (4.31)-(4.33) can be used to derive the input velocities and final stresses. The convergence results are shown in Table 4.6 and Figure 4.11.

4.8.7 Stokes Flow with Both Solid Wall and Free Surface Boundaries

To test a scenario featuring both solid and free surface boundaries, we solve for fluid motion in an annulus centred at the origin with inner radius $r = 0.1$, outer

Table 4.5: Convergence of Stokes with free surface

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	1.2428E-001		1.4886E-001	
32^2	1.9439E-001	-0.65	5.3215E-002	1.48
64^2	1.6330E-001	0.25	2.2135E-002	1.27
128^2	1.3291E-001	0.30	8.1145E-003	1.45
256^2	1.7242E-001	-0.38	5.3320E-003	0.61
512^2	1.7261E-001	-0.00	2.1992E-003	1.28
1024^2	1.6894E-001	0.03	1.1239E-003	0.97
Grid	$\|\tau_{xx}-\tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx}-\tau_{xx}^h\|_1$	Order
16^2	2.7154E-001		1.7116E-001	
32^2	1.4558E-001	0.90	3.8890E-002	2.14
64^2	7.8877E-002	0.88	1.4642E-002	1.41
128^2	5.5237E-002	0.51	5.6140E-003	1.38
256^2	7.0687E-002	-0.36	3.2991E-003	0.77
512^2	7.1097E-002	-0.01	1.3680E-003	1.27
1024^2	7.4305E-002	-0.06	7.1219E-004	0.94
Grid	$\|\tau_{xy}-\tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy}-\tau_{xy}^h\|_1$	Order
16^2	1.6864E-001		8.7332E-002	
32^2	1.0266E-001	0.72	3.9547E-002	1.14
64^2	2.1950E-001	-1.10	1.8694E-002	1.08
128^2	2.0094E-001	0.13	6.9612E-003	1.43
256^2	2.6596E-001	-0.40	4.0559E-003	0.78
512^2	2.0961E-001	0.34	1.6447E-003	1.30
1024^2	2.5255E-001	-0.27	8.8242E-004	0.90
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	3.4171E-001		1.9727E-001	
32^2	7.4246E-002	2.20	4.3033E-002	2.20
64^2	2.6593E-002	1.48	1.2321E-002	1.80
128^2	9.2292E-003	1.53	3.3497E-003	1.88
256^2	6.7182E-003	0.46	1.5327E-003	1.13
512^2	3.0843E-003	1.12	5.9129E-004	1.37
1024^2	1.7877E-003	0.79	2.7579E-004	1.10

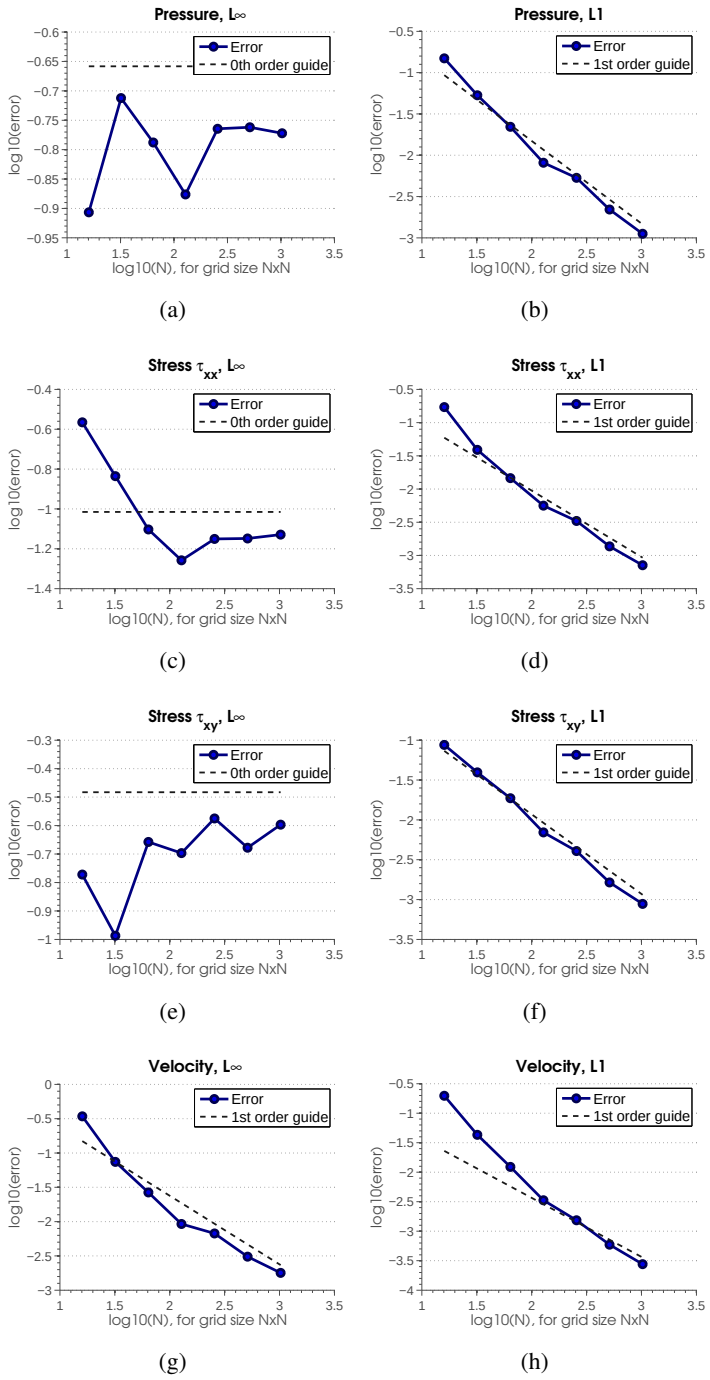


Figure 4.10: Convergence graphs for the Stokes problem with free surface boundaries.

Table 4.6: Convergence of Stokes with solid walls

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	3.4118E+000		2.1013E+000	
32^2	3.6933E+000	-0.11	9.0009E-001	1.22
64^2	3.0338E+000	0.28	4.5162E-001	0.99
128^2	2.5942E+000	0.23	1.5373E-001	1.55
256^2	2.4222E+000	0.10	8.5695E-002	0.84
512^2	2.3573E+000	0.04	3.9879E-002	1.10
1024^2	2.3381E+000	0.01	2.0266E-002	0.98
Grid	$\|\tau_{xx}-\tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx}-\tau_{xx}^h\|_1$	Order
16^2	1.0293E+000		1.0421E+000	
32^2	1.3935E+000	-0.44	7.4091E-001	0.49
64^2	1.2483E+000	0.16	3.1356E-001	1.24
128^2	9.5438E-001	0.39	1.2345E-001	1.34
256^2	1.3160E+000	-0.46	6.2448E-002	0.98
512^2	1.0903E+000	0.27	3.0554E-002	1.03
1024^2	1.0943E+000	-0.01	1.5407E-002	0.99
Grid	$\|\tau_{xy}-\tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy}-\tau_{xy}^h\|_1$	Order
16^2	2.5836E+000		1.8844E+000	
32^2	1.8510E+000	0.48	7.8641E-001	1.26
64^2	1.5152E+000	0.29	3.2160E-001	1.29
128^2	1.1931E+000	0.34	1.3456E-001	1.26
256^2	1.2560E+000	-0.07	6.8684E-002	0.97
512^2	1.2546E+000	0.00	3.4391E-002	1.00
1024^2	1.2466E+000	0.01	1.7487E-002	0.98
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	4.4449E+000		3.2953E+000	
32^2	2.8765E+000	0.63	1.0500E+000	1.65
64^2	8.4194E-001	1.77	1.8614E-001	2.50
128^2	4.1100E-001	1.03	5.3975E-002	1.79
256^2	2.2147E-001	0.89	1.4992E-002	1.85
512^2	9.8967E-002	1.16	7.0519E-003	1.09
1024^2	5.3807E-002	0.88	3.6056E-003	0.97

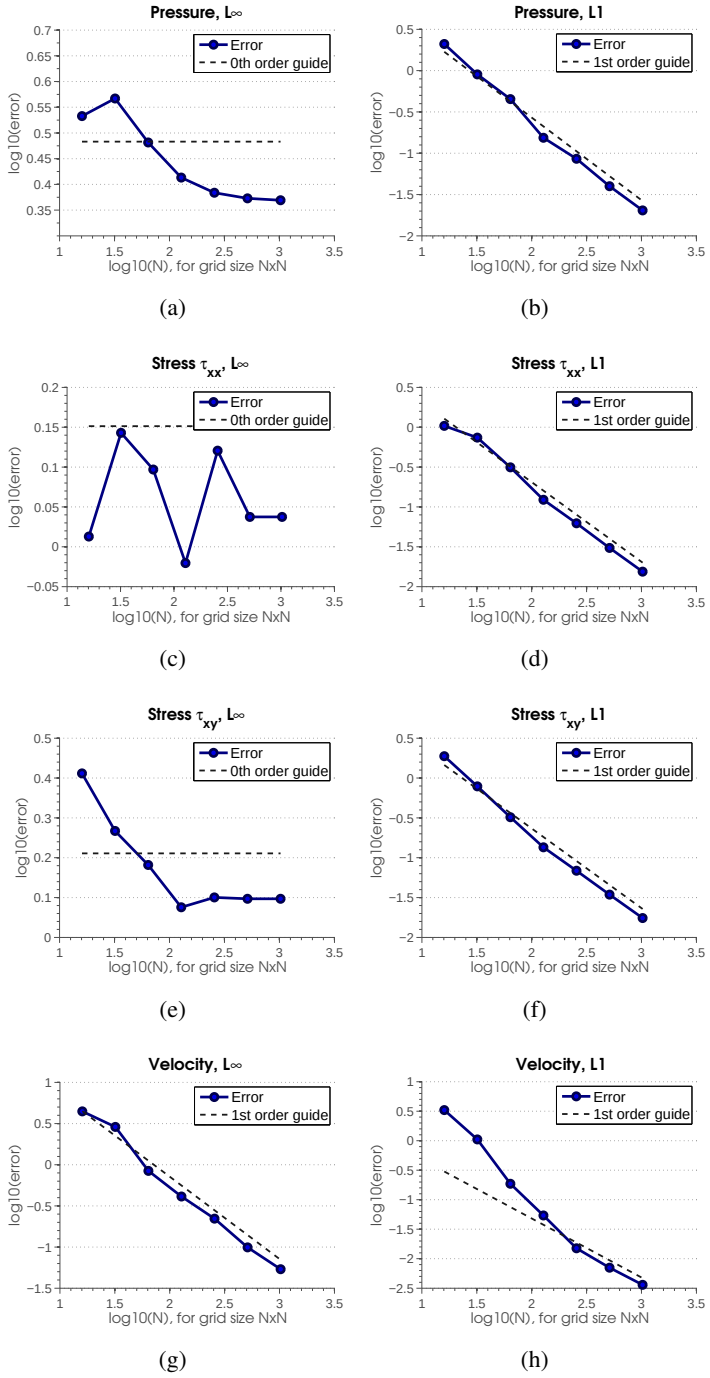


Figure 4.11: Convergence graphs for the Stokes problem with solid wall boundaries.

radius $r = 0.75$, density $\rho = 1$, and viscosity $\mu = 0.1$, over a timestep $\Delta t = 1$. The outer boundary is a free surface, and the inner boundary is a static solid. We will again use a streamfunction ψ to dictate our velocity field and ensure it is divergence free:

$$\psi = \frac{80000}{371293}\cos(2\theta)r^4\cos(\sqrt{3}\ln r)(169 - 390r + 240r^2) \tag{4.62}$$

For pressure, we use:

$$p = \frac{320000}{371293}\sqrt{3}\mu r^2\sin(2\theta)\sin(\sqrt{3}\ln r)(169 - 390r + 240r^2) \tag{4.63}$$

Equations (4.31)-(4.33) can be used to derive the input velocities and final stresses. The convergence results are shown in Table 4.7 and Figure 4.12.

4.8.8 Stokes Flow with Prescribed Velocity Solid Boundaries

To test solid boundaries with prescribed (non-zero) boundary velocities, we solve for fluid motion in an annulus centred at the origin, with inner radius $r = 0.5$, outer radius $r = 1$, density $\rho = 1$, and viscosity $\mu = 0.1$, over a timestep $\Delta t = 1$. The outer boundary is static, while the inner boundary rotates rigidly with a clockwise angular velocity $\omega = 2$. For the final velocity we use the streamfunction ψ:

$$\psi = r^4 - 3r^3 + \frac{9}{4}r^2 + \frac{1}{2}r + \frac{1}{4} \tag{4.64}$$

For pressure, we use:

$$p = r^2\cos(\theta)\sin(\theta) \tag{4.65}$$

As in the preceding examples, stresses and input velocities can be computed from equations (4.31)-(4.33). Convergence results are shown in Table 4.8 and Figure 4.13.

Table 4.7: Convergence of Stokes with solid walls and a free surface

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	2.2404E-001		2.1048E-001	
32^2	2.1216E-001	0.08	6.5787E-002	1.68
64^2	1.6222E-001	0.39	2.2932E-002	1.52
128^2	1.2874E-001	0.33	7.8514E-003	1.55
256^2	1.7168E-001	-0.42	5.1902E-003	0.60
512^2	1.7212E-001	-0.00	2.1723E-003	1.26
1024^2	1.6865E-001	0.03	1.1268E-003	0.95
Grid	$\|\tau_{xx}-\tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx}-\tau_{xx}^h\|_1$	Order
16^2	4.3812E-001		2.0762E-001	
32^2	1.8386E-001	1.25	5.7557E-002	1.85
64^2	9.5150E-002	0.95	1.6510E-002	1.80
128^2	4.9719E-002	0.94	5.9357E-003	1.48
256^2	6.7733E-002	-0.45	3.3888E-003	0.81
512^2	6.9916E-002	-0.05	1.3958E-003	1.28
1024^2	7.3807E-002	-0.08	7.2576E-004	0.94
Grid	$\|\tau_{xy}-\tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy}-\tau_{xy}^h\|_1$	Order
16^2	2.1174E-001		1.0047E-001	
32^2	1.2921E-001	0.71	4.8916E-002	1.04
64^2	2.6177E-001	-1.02	2.0869E-002	1.23
128^2	2.1197E-001	0.30	7.1786E-003	1.54
256^2	2.7070E-001	-0.35	3.9133E-003	0.88
512^2	2.1190E-001	0.35	1.5764E-003	1.31
1024^2	2.5361E-001	-0.26	8.5715E-004	0.88
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	4.2290E-001		2.3654E-001	
32^2	8.6887E-002	2.28	4.8516E-002	2.29
64^2	2.9712E-002	1.55	1.3486E-002	1.85
128^2	1.0943E-002	1.44	3.6100E-003	1.90
256^2	5.3552E-003	1.03	1.3797E-003	1.39
512^2	2.8869E-003	0.89	5.2047E-004	1.41
1024^2	1.6769E-003	0.78	2.4870E-004	1.07

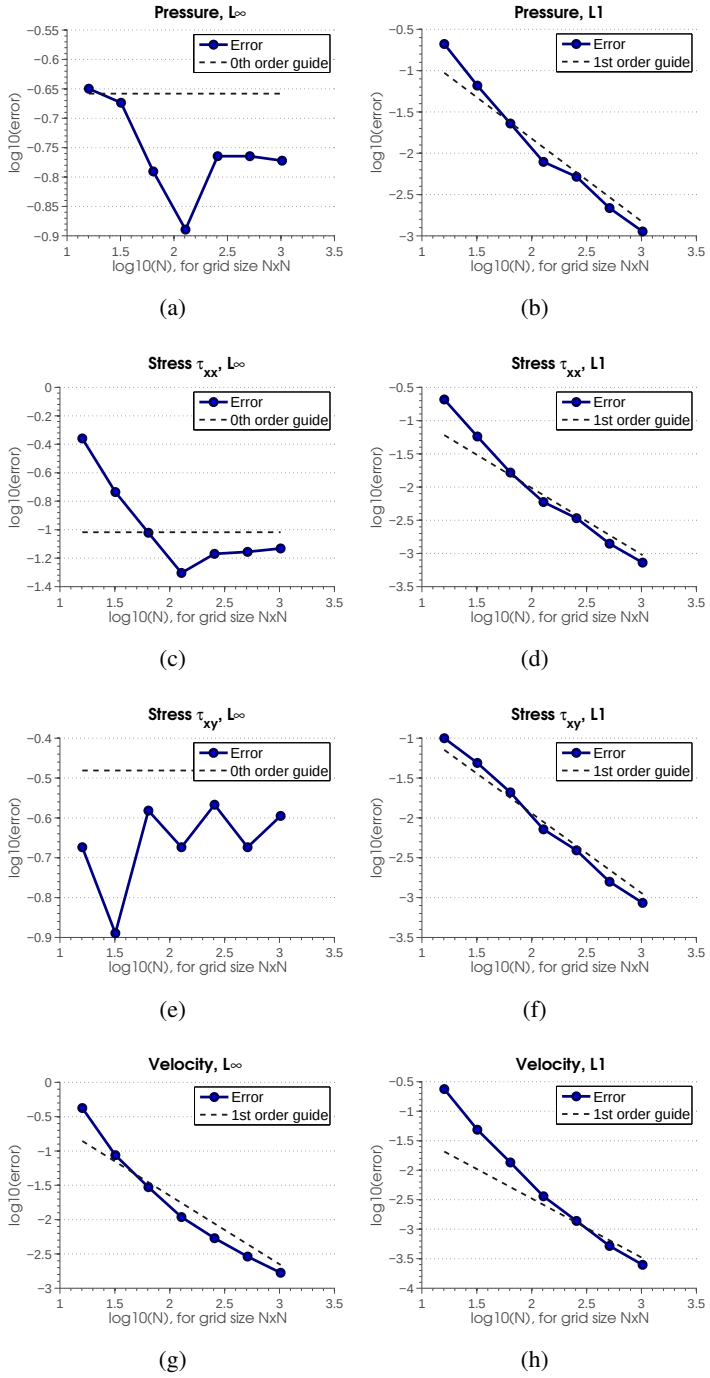


Figure 4.12: Convergence graphs for the Stokes problem with both a free surface and a solid wall boundary.

Table 4.8: Convergence of Stokes with prescribed velocity solid walls

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
16^2	4.7210E-002		2.2007E-002	
32^2	2.7144E-002	0.80	7.9999E-003	1.46
64^2	5.7009E-002	-1.07	5.0198E-003	0.67
128^2	4.5540E-002	0.32	2.2860E-003	1.13
256^2	5.5697E-002	-0.29	1.3316E-003	0.78
512^2	6.0562E-002	-0.12	6.4669E-004	1.04
1024^2	6.1643E-002	-0.03	3.3755E-004	0.94
Grid	$\|\tau_{xx}-\tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx}-\tau_{xx}^h\|_1$	Order
16^2	2.6730E-002		1.2788E-002	
32^2	1.1167E-002	1.26	6.2639E-003	1.03
64^2	2.2172E-002	-0.99	4.4126E-003	0.51
128^2	2.0462E-002	0.12	1.8498E-003	1.25
256^2	3.0896E-002	-0.59	1.0722E-003	0.79
512^2	2.7147E-002	0.19	5.2769E-004	1.02
1024^2	2.8528E-002	-0.07	2.7240E-004	0.95
Grid	$\|\tau_{xy}-\tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy}-\tau_{xy}^h\|_1$	Order
16^2	1.8177E-002		1.6688E-002	
32^2	1.9415E-002	-0.10	8.3726E-003	1.00
64^2	2.4280E-002	-0.32	4.4206E-003	0.92
128^2	2.6202E-002	-0.11	2.0984E-003	1.07
256^2	3.2232E-002	-0.30	1.1682E-003	0.85
512^2	3.3983E-002	-0.08	5.9705E-004	0.97
1024^2	3.3056E-002	0.04	3.0660E-004	0.96
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
16^2	6.6980E-002		2.9655E-002	
32^2	4.2451E-002	0.66	9.2806E-003	1.68
64^2	1.9403E-002	1.13	2.5526E-003	1.86
128^2	8.5696E-003	1.18	8.4643E-004	1.59
256^2	3.7606E-003	1.19	3.5600E-004	1.25
512^2	2.4402E-003	0.62	1.4966E-004	1.25
1024^2	1.4085E-003	0.79	7.1476E-005	1.07

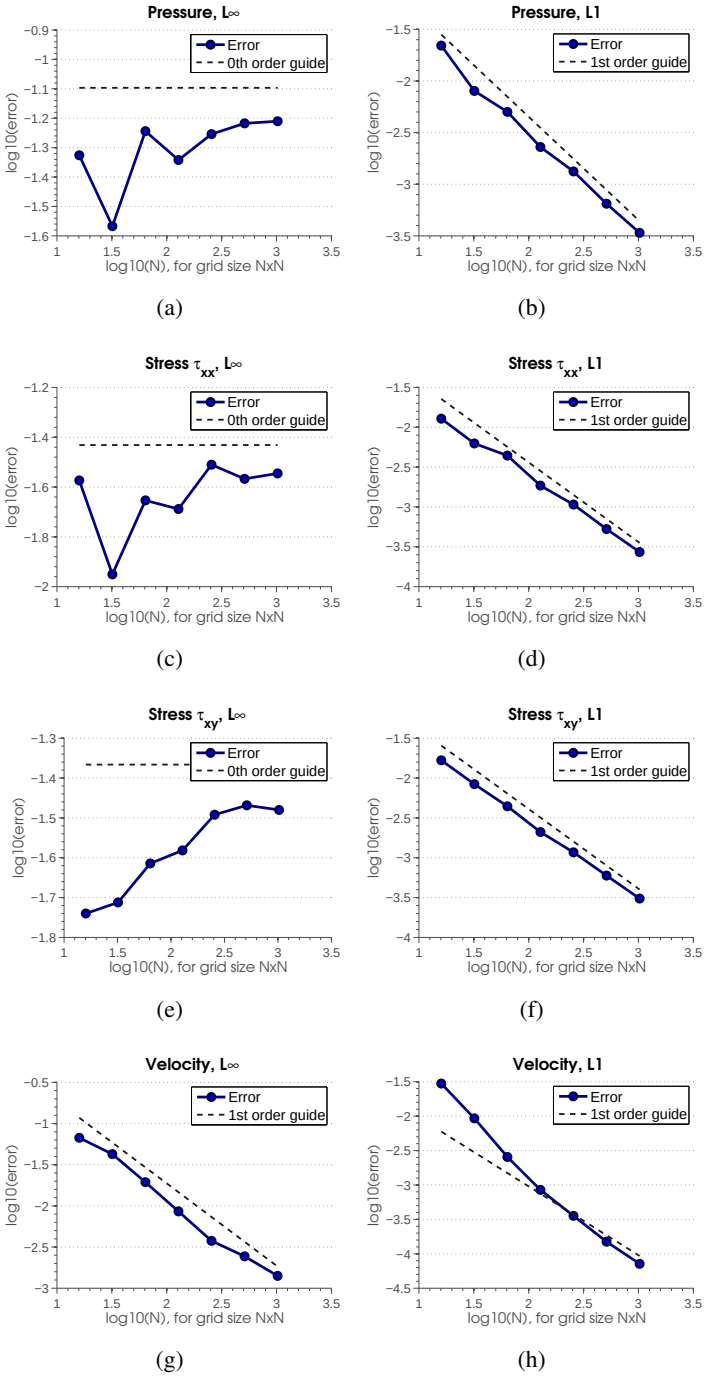


Figure 4.13: Convergence graphs for the Stokes problem with a moving solid wall boundary.

4.8.9 Three Dimensional Stokes Flow

To test our method in three dimensions where analytical solutions are substantially more difficult to derive, we created a numerical solution at resolution 155^3, and tested convergence towards this solution. The test case consists of a sphere of liquid centred at the origin with a free surface at $r = 1$, containing a nested static solid sphere of radius $r = 0.25$. Density was set to $\rho = 1$ and viscosity to $\mu = 0.1$. We compute a time step of length $\Delta t = 1$ starting from an input velocity field:

$$\vec{u}_{input} = (0.5 + x\sin(yz), \cos(0.25y) + 0.5xyz, x + 0.5yz^2 + \sqrt{2 + y^2}) \qquad (4.66)$$

Convergence results are shown in Table 4.9. Interestingly, this numerical experiment suggests first order convergence in L^∞ even for stress variables, an improvement over the results in 2D. We cannot yet explain why the move to 3D would bring greater accuracy, but several similar tests showed the same pattern.

4.8.10 Discussion

Table 4.10 summarizes the approximate orders of convergence suggested by our experiments in two dimensions. We have not yet developed a complete theoretical understanding with which to justify the results, however we can make some general comments.

In the absence of boundaries, our methods are equivalent to a straightforward discretization of the original PDE form with second order centred finite differences on an appropriately staggered grid, and can therefore expect to achieve uniform second order convergence. Near boundaries this clearly does not hold, and the effective quadrature is likely only first order (as for the method of Ng et al. [NMG09]); this is in line with our results in L^∞. Solution gradients can be expected

Table 4.9: Convergence of 3D Stokes with solid walls and a free surface

Grid	$\|p-p^h\|_\infty$	Order	$\|p-p^h\|_1$	Order
12^2	1.3912E-002		1.3312E-002	
24^2	7.5008E-003	0.89	6.3879E-003	1.06
48^2	4.9163E-003	0.61	3.5866E-003	0.83
Grid	$\|\tau_{xx}-\tau_{xx}^h\|_\infty$	Order	$\|\tau_{xx}-\tau_{xx}^h\|_1$	Order
12^2	1.7478E-002		2.0401E-002	
24^2	1.0965E-002	0.67	1.1213E-002	0.86
48^2	5.8661E-003	0.90	5.6263E-003	0.99
Grid	$\|\tau_{yy}-\tau_{yy}^h\|_\infty$	Order	$\|\tau_{yy}-\tau_{yy}^h\|_1$	Order
12^2	2.3604E-002		2.9434E-002	
24^2	1.3516E-002	0.80	1.5741E-002	0.90
48^2	6.8033E-003	0.99	7.7348E-003	1.03
Grid	$\|\tau_{xy}-\tau_{xy}^h\|_\infty$	Order	$\|\tau_{xy}-\tau_{xy}^h\|_1$	Order
12^2	6.4309E-003		7.9982E-003	
24^2	5.1895E-003	0.31	4.4602E-003	0.84
48^2	3.1232E-003	0.73	2.0025E-003	1.16
Grid	$\|\tau_{xz}-\tau_{xz}^h\|_\infty$	Order	$\|\tau_{xz}-\tau_{xz}^h\|_1$	Order
12^2	5.5459E-002		1.3162E-001	
24^2	3.0026E-002	0.89	7.6418E-002	0.78
48^2	1.3128E-002	1.19	3.6082E-002	1.08
Grid	$\|\tau_{yz}-\tau_{yz}^h\|_\infty$	Order	$\|\tau_{yz}-\tau_{yz}^h\|_1$	Order
12^2	1.6390E-002		2.8919E-002	
24^2	1.0325E-002	0.67	1.7670E-002	0.71
48^2	5.2896E-003	0.96	8.7857E-003	1.01
Grid	$\|u-u^h\|_\infty$	Order	$\|u-u^h\|_1$	Order
12^2	1.9163E-001		2.5741E-001	
24^2	1.2664E-001	0.60	1.4041E-001	0.87
48^2	7.1250E-002	0.83	6.3736E-002	1.14
Grid	$\|v-v^h\|_\infty$	Order	$\|v-v^h\|_1$	Order
12^2	6.2893E-002		5.4328E-002	
24^2	4.7289E-002	0.41	3.6334E-002	0.58
48^2	2.9547E-002	0.68	2.0160E-002	0.85
Grid	$\|w-w^h\|_\infty$	Order	$\|w-w^h\|_1$	Order
12^2	1.8633E-001		2.7837E-001	
24^2	1.2850E-001	0.54	1.5601E-001	0.84
48^2	7.4279E-002	0.79	7.1820E-002	1.12

to be one order less accurate than the solution itself, and this is also evident in the L^∞ results. The most intriguing issue, which we do not yet have a clear explanation for, is what determines which variable behaves as the gradient, and which as the primary solution variable. For example in the pressure projection problem, pressure is first order while pressure gradients (and therefore velocity) do not converge. For the Stokes case, this relationship is apparently reversed. For the viscosity problem, whether stress or velocity is the convergent variable (again in L^∞) hinges on which boundary condition is active. It is convenient that in the Stokes case velocity is convergent, since for physical scenarios this is often the most relevant quantity, however further study is needed to understand this phenomenon.

The closely related methods for the Poisson equation considered by Ng et al. [NMG09] and Gibou et al. [GFCK02] (which differ essentially only in the choices of weighting terms) achieve second order convergence in pressure and first order convergence in velocity — one order higher than achieved by our method. While this suggests the possibility of improving our results for the viscosity and Stokes problems by an order of accuracy through different weight choices, it so far seems difficult to introduce ghost fluid or finite volume type weights without breaking symmetry or introducing more complex stencils. In addition, as Ng et al. [NMG09] pointed out, the error in our method tends to be concentrated along the boundary, and exhibits substantial noise; preliminary tests indicate at least first order and possibly second order convergence in L^∞ for all variables away from the boundary. Exploring the connections between our method and ghost fluid/immersed interface methods, finite volume methods, and finite element methods may elucidate these issues.

In considering the pressure projection problem, the method of Bedrossian et al. [BvBZ+10] uses a variational formulation similar to ours, but makes use of piecewise bilinear Cartesian elements near the boundary to estimate the relevant

Table 4.10: Summary of Apparent Convergence Orders for 2D Problems

Pressure Projection	L^∞	L^1
Pressure	1	1
Velocity	0	1
Implicit Viscosity (free surface)	L^∞	L^1
Stress	1	2
Velocity	0	1
Implicit Viscosity (solid walls)	L^∞	L^1
Stress	0	1
Velocity	1	2
Time-Dependent Stokes	L^∞	L^1
Pressure	0	1
Stress	0	1
Velocity	1	1

integrals, at the cost of using denser 9-point stencils for boundary cells. Their results indicate second order convergence in pressure, which is consistent with the fact that our use of piecewise constant estimates yield first order convergence. This also suggests that the use of bilinear elements may be effective in raising the convergence orders for our method in the case of viscosity and Stokes flow, while maintaining the benefits of sparsity and positive-definiteness.

4.9 Application to the Navier-Stokes Equations

One particularly interesting phenomenon exhibited by highly viscous liquids is jet buckling. When a falling liquid column of sufficient viscosity impacts a solid surface, it will fold or coil over on itself rather than spreading out smoothly. Relatively few researchers have looked at simulating Newtonian viscous buckling, despite its prevalence in many common liquids. To the best of our knowledge, the GENSMAC code of Tomé, McKee and co-authors is the only prior finite difference scheme to do so in three dimensions [TM94, TM99, TGC$^+$04, OTCM08]. However, the

GENSMAC approach requires a case-by-case analysis of possible discrete surface orientations and the implicit formulation involves solving a non-symmetric linear system. Jet buckling has also been addressed in a finite element setting [BPL06] and with an SPH approach [RMH07].

With this problem in mind, we incorporate our Stokes solver into a simple pressure projection-based Navier-Stokes routine (similar to that presented in section 2 of the paper by Ng et al. [NMG09]). We break the Navier-Stokes equations up into two stages. First, starting with a velocity field $\vec{u}^n$ at time t^n, we compute advection and body forces to produce an intermediate velocity field $\vec{u}^*$:

$$\rho\left(\frac{\vec{u}^* - \vec{u}^n}{\Delta t} - \vec{u}^n \cdot (\nabla \vec{u}^n)\right) = \vec{F} \tag{4.67}$$

We solve for advective terms with a first order semi-Lagrangian scheme using bilinear interpolation of velocities. We then simply add external forces $\vec{F}$ (i.e. gravity in our examples). From this intermediate velocity, we then simultaneously incorporate viscous forces and project the velocity field to be divergence free using our Stokes solver, to arrive at time $t^{n+1} = t^n + \Delta t$:

$$\rho\frac{(\vec{u}^{n+1} - \vec{u}^*)}{\Delta t} = \nabla \cdot \tau^{n+1} - \nabla p^{n+1} \tag{4.68}$$

$$\nabla \cdot \vec{u} = 0 \tag{4.69}$$

$$\tau^{n+1} = \mu(\nabla \vec{u}^{n+1} + (\nabla \vec{u}^{n+1})^T) \tag{4.70}$$

with the appropriate free surface and solid boundary conditions applied. Tracking of the liquid surface position is performed using a simple semi-Lagrangian level set method [ELF05].

4.9.1 Two Dimensional Jet Buckling

Figure 4.14 presents the results of a two-dimensional simulation of planar viscous jet buckling. The simulation domain is a circle of radius $0.4[m]$ centred at $(0.5, 0.5)$. A horizontal ceiling is placed at $y = 0.8[m]$, featuring a liquid jet inflow centered at $x = 0.5[m]$ with a fixed vertical velocity of $U = -0.5[m/s]$ and a width of $D = 0.06[m]$. This configuration yields a drop height of $H = 0.7[m]$. The density of the liquid is $\rho = 1[kg/m^3]$ and the dynamic viscosity of the liquid is $\mu = 0.075[Pa \cdot s]$. Gravity is set at $-9.81[m/s^2]$. The simulation grid used a resolution of 150×150 cells.

Following Tomé and McKee [TM99], this yields a Reynolds number of $Re = \frac{\rho DU}{\mu} = 0.4$ and an aspect ratio for the liquid jet of $H/D = 0.7/0.06 = 11.667$. This falls within the guidelines for when planar buckling can be expected to occur ($Re < 0.5, H/D > 10$) according to Cruikshank and Munson[CM81, Cru88]; as shown our method reproduces the buckling phenomenon.

4.9.2 Three Dimensional Jet Buckling

Figures 4.15 and 4.16 present the results of a three-dimensional simulation of axisymmetric viscous jet buckling (ie. coiling). The simulation domain is a sphere of radius $0.4[m]$ centred at $(0.5, 0.5, 0.5)$. A circular inlet is centred at $(0.5, 0.8, 0.5)$, with a fixed vertical velocity of $U = -0.5[m/s]$ and a diameter $D = 0.08[m]$. This configuration yields a drop height of $H = 0.7[m]$. The density of the liquid is $\rho = 1[kg/m^3]$ and the dynamic viscosity of the liquid is $\mu = 0.3[Pa \cdot s]$. Gravity is set at $-9.81[m/s^2]$. The simulation grid used a resolution of $80 \times 80 \times 80$ cells.

This yields a Reynolds number of $Re = \frac{\rho DU}{\mu} \approx 0.133$ and an aspect ratio for the liquid jet of $H/D = 0.7/0.08 = 8.75$. This falls within the guidelines for when axisymmetric buckling typically arises ($Re < 1.2, H/D > 7$) according to Cruik-

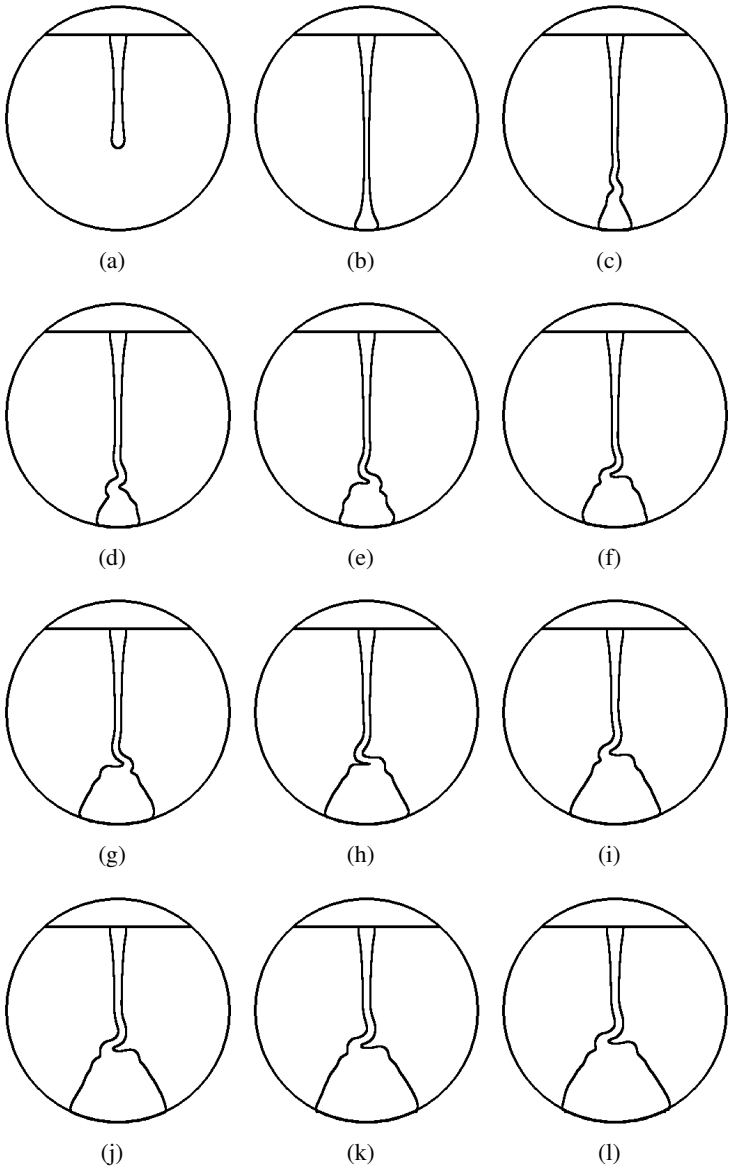


Figure 4.14: A two-dimensional example of viscous jet buckling performed using our simple Navier-Stokes routine. The first image occurs 0.5 seconds into the simulation, and subsequent frames occur at 0.2 second intervals.

shank and Munson[CM81, Cru88], and the result does indeed exhibit substantial buckling.

4.10 Conclusions and Future Work

We have shown that a relatively simple Cartesian grid finite difference method, derived from a variational principle, can correctly capture difficult irregular boundary conditions in Stokes flow problems, while providing unconditional stability and yielding a sparse, symmetric positive-definite linear system. Along the way, we have unified and extended recent work on embedded boundary methods for pressure projection and viscosity, which are useful for fractional step algorithms that segregate the Stokes equations in the name of efficiency. This raises a number of questions and directions for future work.

We observed a puzzling disparity in terms of better L^∞ convergence in 3D compared to 2D, and plan to investigate if this truly holds—perhaps beginning by deriving a full-fledged analytic test case as we have done in 2D.

The convergence test cases we presented only considered scenarios where the two different boundary conditions do not intersect. We suspect that this is an inherently more difficult problem to address, giving rise to issues analogous to those that often occur in the presence of sharp boundary features; our preliminary experiments support this conjecture. Nonetheless, such configurations occur frequently in the buckling examples we have included, illustrating that the method remains fully stable and provides reasonable results.

In terms of handling related phenomena, an obvious extension would be to consider fully dynamic solid-fluid interaction, as addressed by Batty et al. [BBB07] for the simpler Poisson problem, along with extending that method to support deformable objects. Extensions to support viscoelastic fluids would also be benefi-

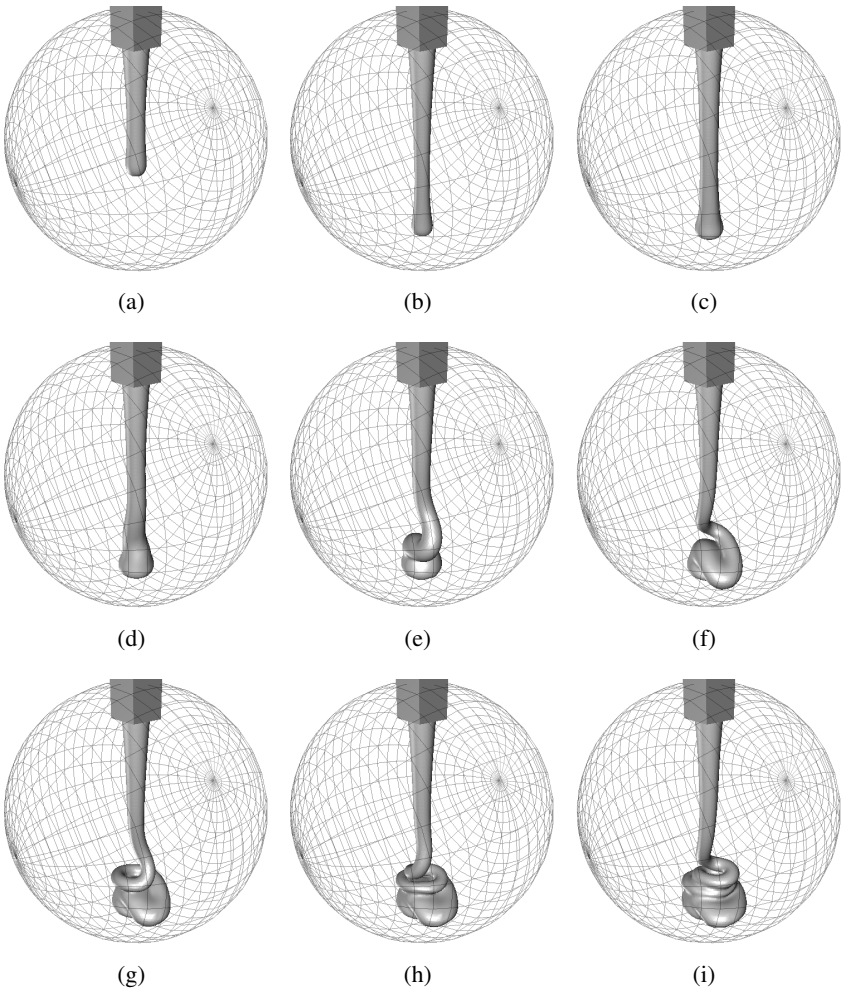


Figure 4.15: A three-dimensional example of viscous jet buckling performed using our simple Navier-Stokes routine. The first image occurs 0.6 seconds into the simulation, and subsequent frames occur at 0.3 second intervals. Additional images are shown in Figure 4.16.

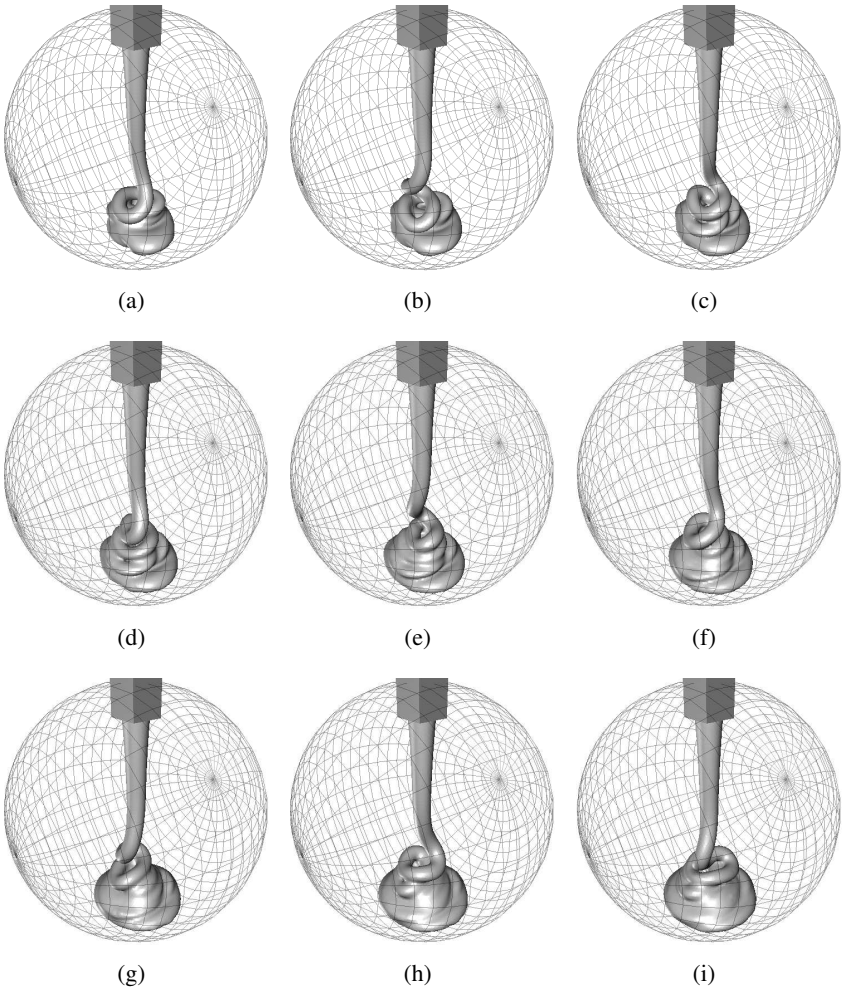


Figure 4.16: Additional images of viscous jet buckling, continued from Figure 4.15.

cial.

While our current viscous jet buckling example provides a tangible, practical validation of our method's boundary condition enforcement, the current underlying Navier-Stokes simulator is fairly simplistic. A thorough study of this phenomenon would likely need to consider improved advection and time-splitting methods in place of the first order approaches applied here.

Finally, it would be interesting to consider whether exploiting variational principles in this manner might be useful in discretizing irregular boundaries for other problems on staggered Cartesian grids. Some potential examples include vorticity-based formulations of fluid flow, Maxwell's equations of electromagnetism, diffusion problems, or porous flow.

Bibliography

[BB08] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*, pages 219–228, 2008.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):100, 2007.

[BPL06] Andrea Bonito, Marco Picasso, and Manuel Laso. Numerical simulation of 3D viscoelastic flows with free surfaces. *J. Comp. Phys.*, 215(2):691–716, 2006.

[BvBZ+10] Jacob Bedrossian, James H. von Brecht, Siwei Zhu, Eftychios Sifakis, and Joseph Teran. A second order virtual node method for Poisson interface problems on irregular domains. *J. Comp. Phys.*, (in press), 2010.

[Cho68] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[CM81] J. O. Cruikshank and B. R. Munson. Viscous-fluid buckling of plane and axisymmetric jets. *Journal of Fluid Mechanics*, 113:221–239, 1981.

[Cru88] J. O. Cruikshank. Low-Reynolds-number instabilities in stagnating jet flows. *Journal of Fluid Mechanics*, 193:111–127, 1988.

[CS70] Robert K.-C. Chan and Robert L Street. A computer study of finite amplitude water waves. *J. Comp. Phys.*, 6:68–94, 1970.

[DWB92] M. S. Darwish, J. R. Whiteman, and M. J. Bevis. Numerical modelling of viscoelastic liquids using a finite-volume method. *Journal of Non-Newtonian Fluid Mechanics*, 45(3):311–337, 1992.

[ELF05] Doug Enright, Frank Losasso, and Ron Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83(6-7):479–490, 2005.

[ENGF03] Doug Enright, Duc Nguyen, Frédéric Gibou, and Ron Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, 2003.

[GFCK02] Frédéric Gibou, Ron Fedkiw, L.-T. Cheng, and Myungjoo Kang. A second order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comp. Phys.*, 176(1):205–227, 2002.

[HS68] C. W. Hirt and J. P. Shannon. Free surface stress conditions for incompressible-flow calculations. *J. Comp. Phys.*, 2(4):403–411, 1968.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[JC98] Hans Johansen and Phillip Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comp. Phys.*, 147(1):60–85, November 1998.

[LIRO07] A. Limache, S. R. Idelsohn, R. Rossi, and E. Onate. The violation of objectivity in Laplace formulations of the Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 54(6-8):639–664, 2007.

[LL97] Randall J. LeVeque and Zhilin Li. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18(3):709–735, 1997.

[LSDI08] A. Limache, P. J. Sanchez, L. D. Dalcin, and S. R. Idelsohn. Objectivity tests for Navier-Stokes simulations: The revealing of non-physical solutions produced by Laplace formulations. *Computer Methods in Applied Mechanics and Mechanical Engineering*, 197(49–50):4180–4192, 2008.

[MD97] G. Mompean and M. Deville. Unsteady finite volume simulation of Oldroyd-B fluid through a three-dimensional planar contraction. *Journal of Non-Newtonian Fluid Mechanics*, 72(2-3):253–279, October 1997.

[MG07] Chohong Min and Frédéric Gibou. Geometric integration over irregular domains with application to level-set methods. *J. Comp. Phys.*, 226(2):1432–1443, October 2007.

[NH71] B. D. Nichols and C. W. Hirt. Improved free surface boundary condi-

tions for numerical incompressible-flow calculations. *J. Comp. Phys.*, 8(3):434–448, 1971.

[NMG09] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comp. Phys.*, 228(23):8807–8829, 2009.

[OTCM08] Cassio M. Oishi, Murilo F. Tomé, José A. Cuminato, and Sean McKee. An implicit technique for solving 3D low Reynolds number moving free surface flows. *J. Comp. Phys.*, 227(16):7446–7468, 2008.

[PB79] James W. Purvis and John E. Burkhalter. Prediction of critical Mach number for store configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[Pes02] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[Pra71] William E. Pracht. A numerical method for calculating transient creep flows. *J. Comp. Phys.*, 7(1):46–60, 1971.

[RbZF05] Doug Roble, Nafees bin Zafar, and Henrik Falt. Cartesian grid fluid simulation with irregular boundary voxels. In *SIGGRAPH Sketches*, 2005.

[RMH07] A. Rafiee, M. T. Manzari, and M. Hosseini. An incompressible SPH method for simulation of unsteady viscoelastic free surface flows. *International Journal of Non-Linear Mechanics*, 42(10):1210–1223, 2007.

[TGC+04] Murilo F. Tomé, L. Grossi, Antonio Castelo, José A. Cuminato, Norberto Mangiavacchi, Valdemir G. Ferreira, F. S. de Sousa, and Sean

McKee. A numerical method for solving three-dimensional generalized Newtonian free surface flows. *Journal of Non-Newtonian Fluid Mechanics*, 23(2-3):85–103, 2004.

[TM94] Murilo F. Tomé and Sean McKee. GENSMAC: A computational marker and cell method for free surface flows in general domains. *J. Comp. Phys.*, 110(1):171–186, 1994.

[TM99] Murilo F. Tomé and Sean McKee. Numerical simulation of viscous flow: Buckling of planar jets. *International Journal for Numerical Methods in Fluids*, 29(6):705–718, 1999.

[ZZFW05] Y. C. Zhou, Shan Zhao, Michael Feig, and G. W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *J. Comp. Phys.*, 213(1):1–30, 2005.

Chapter 5

Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids

5.1 Introduction

Adaptivity is a key feature for the efficient animation of fluids because it can focus computational resources on visually significant details. Examples include regions of greater vorticity, inside the viewing frustum, and flow near free surfaces and solid objects [LGF04, KFCO06, KIC06]. Tetrahedral meshes, octrees, elongated Cartesian cells [IGLF06], and general nested Cartesian (AMR) grids (eg. [BO84]) have all been used for this purpose. These methods' primary drawback is that domain boundaries must align with the underlying voxels or tetrahedra, which either limits the variety of boundaries that can be simulated or greatly increases the expense and difficulty of high quality mesh generation.

Embedded boundary methods that account for sub-grid geometry of free sur-

A version of this chapter has been published. Batty, C., Xenos, S., and Houston, B. (2010) Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids, Computer Graphics Forum (Proc. Eurographics) 29(2).

faces and solid objects on Cartesian grids have recently gained acceptance in graphics [ENGF03, BBB07], by offering improved resolution of irregular shapes with only minor modifications to existing solvers. These methods yield equivalent or better results than conforming tetrahedral meshes, because the regularity of the grid structure affords more accurate finite difference operators.

In terms of adaptivity however, tetrahedra offer substantial advantages over Cartesian grids. Octrees and nested grids provide only discrete jumps in grid size, which artificially prevent smooth grading between resolutions. At these "T-junction" faces where jumps in resolution occur, a single lower resolution face is shared by four or more higher resolution faces. Besides added implementation complexity, it is so far unclear how to apply embedded methods across such faces; at present grading must be disallowed along air and solid boundaries. T-junctions also require care to avoid losing accuracy and causing simulation artifacts [LGF04, CFL$^+$07, LFO05]. In contrast, tetrahedral meshes need no T-junctions and allow elements of arbitrary size, thus providing greater flexibility.

Our work seeks to hybridize tetrahedral methods with embedded boundary techniques. In this manner, we achieve the high quality results associated with embedded boundary Cartesian grid methods, while simultaneously providing the best combination of speed, flexibility, and adaptivity of state-of-the-art tetrahedral schemes. Our specific contributions are the following:

Embedded Free Surfaces: We extend the ghost fluid sub-grid free surface pressure projection [ENGF03] to tetrahedra, which improves the accuracy of free surfaces and removes their boundary alignment restriction.

Embedded Solid Boundaries: We adapt the sub-grid solid boundary pressure projection [BBB07] to tetrahedra, to provide accurate support for non-mesh-aligned solids.

The elimination of boundary alignment constraints provides several vital benefits:

Fast High Quality Delaunay Meshing: We can exploit high quality *non-conforming* Delaunay meshes with circumcentric pressures. Such meshes guarantee consistent finite difference estimates of at least first order, and are faster and easier to generate, particularly for complex domains.

Improved Surface Motion: Enforcing the free surface boundary condition precisely at the air-liquid interface improves the resulting fluid motion, especially for slow or still fluid.

Reduced Remeshing Frequency: Since adaptivity requirements typically exhibit high temporal coherence, we can reuse entire meshes over several timesteps.

Increased Flexibility: It becomes possible to easily grade along air or solid boundaries without simulation artifacts or more complex mesh generation, allowing effectively arbitrary mesh adaptivity.

5.2 Related Work

5.2.1 Adaptive Fluids and Tetrahedral Meshes

Tetrahedral meshes have become popular within the computer graphics fluid simulation community because they provide straightforward adaptivity and until recently were the only simple method for accurately incorporating non-axis-aligned boundary geometry into Eulerian schemes. Feldman et al. [FOK05] first mapped the basic Stable Fluids method [Sta99] to tetrahedra, using finite volume techniques (as in the octree work of Losasso et al. [LGF04]). This was extended to mildly deforming meshes [FOKG05], and then to rigid and deformable body coupling [KFCO06, CGFO06], by remeshing on every timestep. Klingner et al. [KFCO06] also specifically highlighted the adaptivity benefits of tetrahedra, where previous work had focused solely on matching irregular boundaries. Wendt et al. [WBOL07]

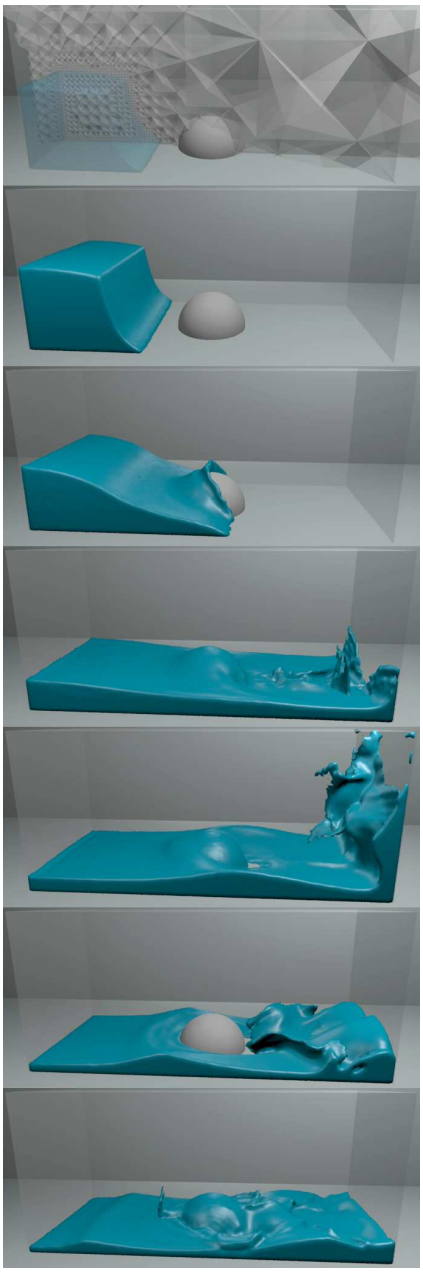


Figure 5.1: Our method yields quality results on a dam break example without matching air or solid boundaries. The top frame shows a cutaway of the mesh.

used a slightly different discretization and a level set method to include viscosity and non-conforming free surfaces. Chentanez et al. [CFL+07] presented an efficient algebraic multigrid method, along with conforming free surfaces through the use of isosurface stuffing for faster remeshing [LS07]. This method provides fast adaptive meshing with guaranteed angle bounds, at the cost of the Delaunay property. An alternate circulation-based approach was advocated by Elcott et al. [ETK+07], and Mullen et al. [MCP+09] demonstrated energy-preservation, using an Eulerian advection scheme combined with a unified, fully non-linear solver. Sin et al. [SBH09] recently presented a dual approach, solving the pressure projection on unstructured Voronoi meshes clipped against boundaries.

Tetrahedral methods achieve their best results when the meshes used possess the Delaunay property and pressures are stored at circumcentres; this ensures convex dual elements and consistent first order accurate finite difference approximations (see section 5.4). If the tetrahedralization aligns with a particular domain boundary, it is referred to as a conforming Delaunay tetrahedralization [CSCY04]. Such meshes are generally difficult to compute, although allowing flexibility in the surface by adding Steiner points or approximating the boundary simplifies matters [CSCY04, ACSYD05]. Nonetheless, it remains substantially slower and more difficult than either *non-conforming* Delaunay meshing [WT08] or conforming *non-Delaunay* meshing [LS07]. For example, Alliez et al. [ACSYD05] performed Delaunay meshing up to 50 times during their iterative variational scheme, while requiring heuristic vertex jittering to discourage slivers near boundaries. Klingner et al. [KFCO06] found that the same method required 5 minutes per frame for 500K tetrahedra. Recently Tournois et al. [TWAD09] interleaved Delaunay refinement with optimization and improved the boundary treatment to produce even higher quality meshes, but had meshing times in the tens of minutes for 120K tetrahedra and still provided

no guarantees against slivers.

Because guaranteed quality conforming Delaunay meshing remains challenging, research in computational physics has modified the finite volume method in hopes of achieving good accuracy on more general meshes. Perot and Subramanian [PS07] used an exact calculus discretization with improved interpolation to handle non-Delaunay meshes. Similarly, deferred correction approaches use geometric arguments to correct directional errors in gradients [TAL09]. Though effective, such methods are more expensive and complex, and it remains unclear how to enforce air and solid boundaries simultaneously. In effect, these approaches seek to relax the mesh quality requirements, whereas we will relax the boundary alignment requirement.

5.2.2 Embedded Boundaries

Two techniques have allowed embedded (ie. non-mesh-aligned) boundaries to be supported on Cartesian grids with relative ease. The first is the second order accurate ghost fluid free surface pressure condition of Enright et al. [ENGF03], which greatly improves the behaviour of liquid surfaces by modifying the finite difference gradient stencil. The second is the sub-grid solid boundaries approach which achieves high quality solid interaction on non-conforming grids by adding face weights to the divergence stencil. Roble et al. [RbZF05] first derived the latter idea in a finite volume manner, suggesting the use of face *area* weights. Batty et al. [BBB07] developed a related variational method to handle moving boundaries and full rigid body coupling. For the case of static objects this yields essentially the same discretization, except for the use of face *volume* weights. However, a recent paper by Ng et al. [NMG09] has shown that face area weights should be preferred, as this provides second order accuracy in pressure. This solid boundary discretiza-

tion can also be combined with the free surface condition above, as demonstrated by Batty & Bridson [BB08]. With the addition of a few simple weights to the standard finite difference stencils, these two embedded boundary approaches naturally extend the MAC scheme to irregular domains.

There are also numerous embedded boundary methods in the computational physics literature. These include immersed boundary methods [Pes02], cut-cell methods (eg. [MKLU05, KLMU05, SBCL06]), ghost fluid variants (eg. [BF08, MDB$^+$08]) and many more. Our focus on the works of Enright et al. [ENGF03] and Batty et al. [BBB07] is motivated by these methods' simplicity and effectiveness. Both also have longstanding roots in computational fluid dynamics. Enforcing the Dirichlet pressure condition at the sub-grid free surface position was first suggested by Chan and Street [CS70], albeit in rudimentary form. Similarly, the basic face area-weighting scheme for static embedded solids can be traced to work by Purvis and Burkhalter [PB79].

While the majority of embedded boundary schemes use Cartesian grids, a few have recently highlighted the benefits of an underlying simplex mesh, as we do here (eg. [LCC$^+$08, FD07]). However, they employ fundamentally different and more complex cut-cell schemes than those we propose.

Another loosely related area of research is embedding methods for simulating deformable objects. These approaches embed a more detailed surface mesh into a lower resolution simulation mesh to reduce computational costs [FPT97, CGC$^+$02]. This family was extended to handle extreme plastic deformations by Wojtan & Turk [WT08], with a simple but effective remeshing strategy that generates high quality non-conforming Delaunay meshes from an adaptive BCC lattice. Because the simulation mesh need not conform to the object's surface geometry, they can guarantee very high quality elements while apparently remeshing one to two orders of magnitude faster than state-of-the-art conforming, non-Delaunay ap-

proaches. Our approach applies essentially the same premise to fluid simulation.

5.3 System Overview

Our approach builds most directly on the work of Chentanez et al. [CFL+07], to which we refer the reader for implementation details and pointers to prior work. We will simply highlight the differences unique to our approach. The specific steps in our algorithm are:

1. Advect the liquid surface, with a standard surface tracker.
2. *Optional:* Remesh to generate a new tetrahedral mesh enveloping the liquid domain (section 5.4). We use a quality non-conforming adaptive Delaunay BCC lattice mesh generator in place of isosurface stuffing.
3. Apply semi-Lagrangian advection to mesh velocities. If remeshing occurred, this transfers velocities to the new mesh as usual (without extra smoothing [FOKG05]).
4. Add external forces.
5. Apply our tetrahedral embedded boundary pressure projection (sections 5.5 & 5.6). This replaces the standard conforming tetrahedral pressure projection [FOK05].

We have not applied any volume preservation strategies in our system. We solve the pressure projection with the conjugate gradient method, but expect that the algebraic multigrid (AMG) method proposed by Chentanez et al. [CFL+07] would provide a useful speed-up. A final key difference is that we store pressures at tetrahedra circumcentres, and velocities at face circumcentres, as in most earlier work (eg. [KFCO06]).

	Static test	Graded Boundaries	Remesh frequency	Remesh speed	Mesh quality
Embedded regular grids	***Pass***	No	N/A	N/A	***High***
Embedded octrees	***Pass***	No	***Low***	***Fast***	***High***
Conforming Delaunay	Fail	***Yes***	High	Slow	Low/ Moderate
Chentanez et al. [CFL$^+$07]	Fail	No	High	***Fast***	Low
Embedded Delaunay (ours)	***Pass***	***Yes***	***Low***	***Fast***	***High***

Figure 5.2: A qualitative comparison of different methods and simulation meshes. "Mesh quality" encompasses orthogonality, the Delaunay property, and angle bounds. "Static test" refers to still fluid where pressure should precisely balance gravity. Our embedded Delaunay method (using an underlying adaptive BCC lattice) provides a good combination of accuracy, speed, flexibility, and adaptivity.

5.4 High Quality Meshes

Basic staggered mesh methods for tetrahedra require a Delaunay mesh with pressures stored at tetrahedra circumcentres [KFCO06]. These "covolume" meshes are a natural generalization of classic staggered grid (MAC) schemes [Nic92, NW97, ZSP02]. Connecting neighbouring circumcentres on a primal Delaunay mesh yields its circumcentric or Voronoi dual, a valid Voronoi tesselation possessing two very useful properties, as discussed by Perot and Subramaniam [PS07]. First, local orthogonality refers to the fact that the line through neighbouring pressure locations (circumcentres) is perpendicular to the shared face of their tetrahedra, and thus parallel to the velocity sample stored at the face (Figure 5.3, left). When we apply the standard fluid velocity update ($\vec{u} = \vec{u^*} - \frac{\Delta t}{\rho}\nabla p$) the pressure gradient corrects for divergence in the appropriate velocity component, ensuring that the

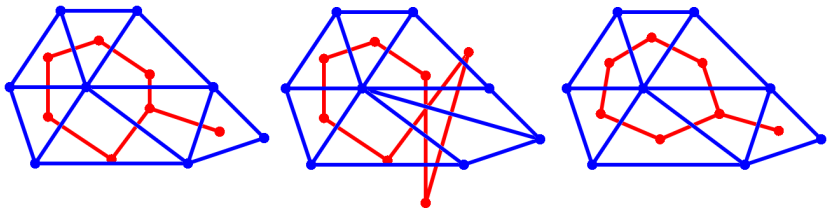


Figure 5.3: Different choices of triangulations (blue) and dual meshes (red) in 2D. From left to right: 1) Delaunay triangulation with circumcentric dual. 2) Non-Delaunay triangulation with circumcentric dual. The dual mesh is self-intersecting. 3) Delaunay triangulation with barycentric dual. Primal/dual edge pairs lack orthogonality. Based on figures by Perot & Subramaniam.

pressure gradient estimate is at least first order accurate. Perot and Subramaniam further note that while strict second order accuracy requires that dual edge midpoints between neighbouring circumcentres coincide exactly with primal face circumcentres (as for uniform grids), second order accurate convergence is frequently observed in practice for fairly well-behaved meshes. The second useful property of these meshes is the convexity and non-self-intersection of the dual mesh, which avoids dual elements with conceptually negative volumes and allows generalized barycentric interpolation of velocities [WSHD07, ETK+07, KFCO06]. Klingner et al. [KFCO06] also used properties of Delaunay meshes to simplify these interpolations.

Relaxing these two constraints would simplify mesh generation, but unfortunately this causes problems for simulation. The dual mesh generated by connecting circumcentres of non-Delaunay meshes is self-intersecting, though it retains orthogonality (Figure 5.3, centre). Conversely, the barycentric or median dual generated by connecting mesh barycentres is comprised of valid convex elements, but sacrifices the crucial orthogonality property (Figure 5.3, right). This latter situation gives rise to the linear inconsistency mentioned by Chentanez et al. [CFL+07] and partly accounts for the artifacts they observed in slow-moving fluids.

As an aside, we note that in the original octree method of Losasso et al. [LGF04] pressure gradients across T-junctions lose accuracy because they too give up orthogonality. This method places velocity samples on each small face incident on a T-junction, and estimates a non-orthogonal pressure gradient between the associated large and small cell pressures. However, Losasso et al. [LFO05] later corrected this to achieve second order accuracy with a slight modification. The solution was to use just a single velocity sample on the entire T-junction face and construct a fully orthogonal pressure gradient as an area-weighted combination of the small face pressure gradients. This further illustrates the importance of retaining orthogonality.

Instead of sacrificing mesh orthogonality or convexity, we will use embedded boundary methods to eliminate the restriction that meshes align with domain geometry. This has several practical consequences with respect to the meshes used for simulation. First, it allows us to more aggressively exploit temporal coherence. For example, in the case of a moving object the user might desire high resolution elements around the object to capture small flow details. With conforming methods, accommodating even slight motions of an object can require substantial changes to the mesh to maintain high quality. This holds true even if the mesher is warm-started with the previous mesh, as done by Klingner et al. [KFCO06]. With non-conforming methods, we remesh only when our mesh adaptivity criterion ceases to be satisfied (eg. in Figure 5.4 we might remesh when the liquid surface leaves the surrounding region of highest refinement). By tailoring such criteria appropriately, we can balance the benefits of frequent adaptivity updates against the costs of remeshing.

Second, because the true boundaries are allowed to cut through the mesh arbitrarily, mesh grading can occur even along free surfaces and solid boundaries. In contrast, the complexity of correctly handling octree T-junctions makes it unclear

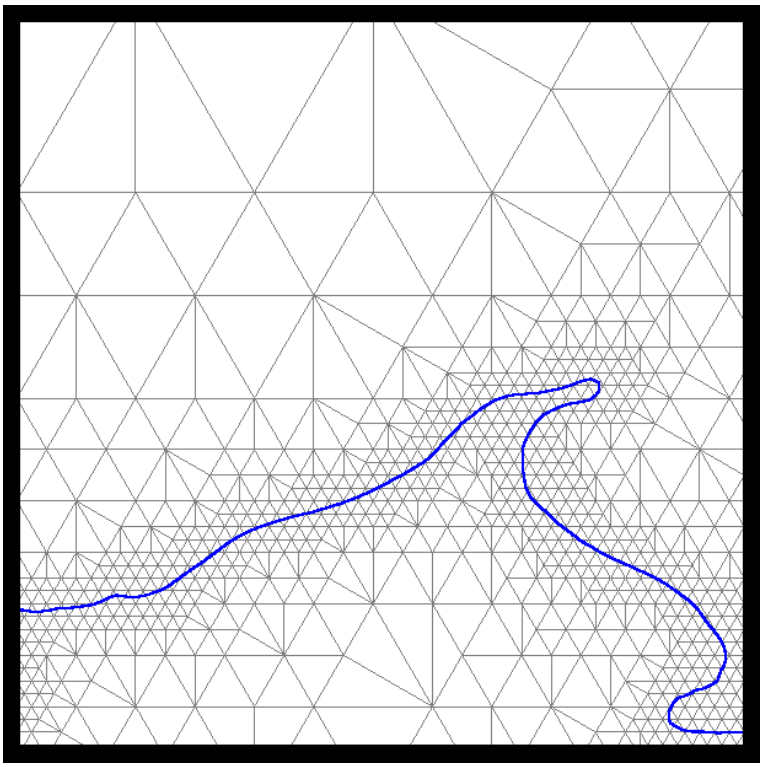


Figure 5.4: Embedded fluid simulation on a high quality adaptive non-conforming lattice mesh. Since we need not match boundaries, we can reuse consecutive meshes to save on meshing costs.

how to simultaneously combine them with embedded boundaries; a uniform resolution is thus required along all boundaries. Similarly, though isosurface stuffing is incredibly fast, it only provides effective angle guarantees if adaptivity is restricted to the interior of the mesh. Consider Figure 5.4, where high resolution is desired only near the liquid surface. If all boundary elements needed to be uniformly sized, many would be wasted resolving unnecessary details along the bottom wall.

Third and most vitally, using embedded boundaries accelerates and simplifies remeshing, and lets us easily guarantee that our simulator is provided with high quality Delaunay meshes with the properties necessary for maximum accu-

racy. This is particularly important because poor mesh quality and high meshing costs are two major drawbacks of tetrahedra as compared to grids. Any Delaunay mesh that fully envelops the domain may be used with our scheme. For maximum remeshing speed and regularity, we recommend the unmodified octree-graded BCC lattice as in the work of Wojtan and Turk [WT08]. As they pointed out, this results in high quality meshing that is effectively free compared to the remaining steps of the algorithm. Furthermore, its regularity may be exploited to accelerate point-location and save memory [CFL$^+$07]. Our single-threaded implementation generates over 500K tetrahedra/second for meshes up to three million elements.

5.5 Embedded Free Surfaces on Tetrahedra

The free surface (Dirichlet) pressure boundary condition presented by Enright et al. [ENGF03] allows free surfaces to lie between rather than strictly at grid cell centres. The discretization for a velocity update due to pressure in one dimension for a particular face at the boundary between liquid and air is:

$$u = u^* - \frac{\Delta t}{\rho} \cdot \frac{p_{fs} - p_i}{\theta \Delta x} \tag{5.1}$$

In this expression u is the final divergence-free face velocity, u^* is the velocity before projection, Δt is the time step, ρ is the liquid density, Δx is the grid cell size, p_i is the pressure variable in the liquid cell, p_{fs} is the specified boundary value for the free surface pressure (typically zero or standard atmosphere), and finally θ is the fractional distance from the last internal pressure sample to the sub-grid liquid surface. In situations where θ is at or near zero, it is perturbed to be slightly positive [GFCK02, Bri08]. θ is typically estimated from signed distance values stored at pressure samples, but more generally is extracted from

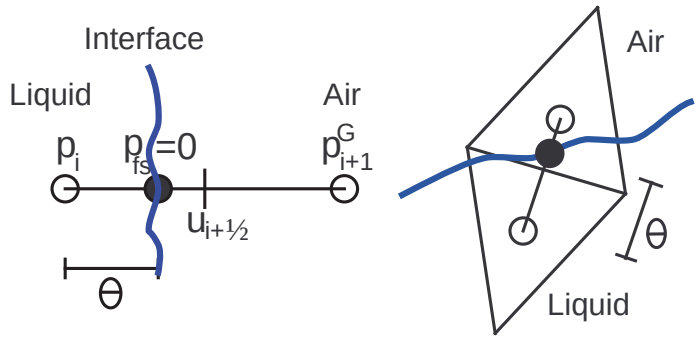


Figure 5.5: Left: A 1D example of the method of Enright et al. for capturing the free surface position between pressure samples. Right: The same idea applied to an unstructured triangle mesh.

the user's choice of surface tracker. Figure 5.5, left, illustrates this situation. This can most directly be understood as a shortened finite difference estimate of the pressure gradient from the last submerged pressure cell to the free surface position. Equivalently, it can be derived by placing a ghost pressure p^G_{i+1} in the adjacent air cell such that the linearly interpolated pressure value crosses p_{fs} precisely at the sub-grid interface location. In either case, this expression for the velocity update is plugged into the divergence constraint $\nabla \cdot \vec{u} = 0$, yielding a symmetric positive definite Poisson system and a second order accurate pressure solution for smooth boundaries. Enright et al. [ENGF03] showed that this drastically improves the behaviour of free surfaces on regular grids.

This method can be readily adapted to tetrahedral meshes with minimal modification. If the interface lies between two tetrahedra circumcentres (ie. where pressure is stored), we replace Δx with the distance between the circumcentres and modify θ to be an estimate of the fractional position of the interface along the line joining the two circumcentres (Figure 5.5, right). As in the Cartesian grid case this yields much improved small-scale behaviour, and eliminates the tetrahedral analog

of free surface stairstep artifacts (See Figure 5.6). (This is the "aliasing" noted by Wendt et al. [WBOL07].)

This discretization can beneficially be applied even to existing *conforming* tetrahedral methods. The current standard approach to enforcing the Dirichlet boundary condition is to use a mirrored ghost pressure set to p_{fs} on the outside of the appropriate face [FOK05, CFL$^+$07, SBH09]. Considering again Figure 5.5, right, this incorrectly sets the pressure value at the exterior ghost point, rather than precisely at the face where the liquid interface lies. This is likely the second source of error which prevented hydrostatic balance (where pressure precisely cancels gravity forces) from being achieved by Feldman [Fel07] and Chentanez et al. [CFL$^+$07]. Our method easily corrects this; the denominator in the pressure gradient calculation should be the perpendicular distance from the circumcentre to the surface face, rather than twice that value.

5.6 Embedded Solid Boundaries on Tetrahedra

The variational projection technique presented by Batty et al. [BBB07] allows for sub-grid resolution of solid (Neumann) boundaries, by modifying the divergence operator with weights to account for partial cells. Expressed in a finite volume-like form their divergence operator is the following:

$$\nabla \cdot \vec{u} \approx \sum_{i \in faces} w_i (\vec{u}_i \cdot \vec{n}_i) \tag{5.2}$$

where the set of faces are those of the original cell. $\vec{u}_i$ and $\vec{n}_i$ are the fluid velocity and normal at the face, respectively. A few weight choices are reasonable, but following Ng et al. [NMG09] we choose the weights w_i to be the partial non-solid area of the face. As noted by Roble et al. [RbZF05], this choice yields a slightly

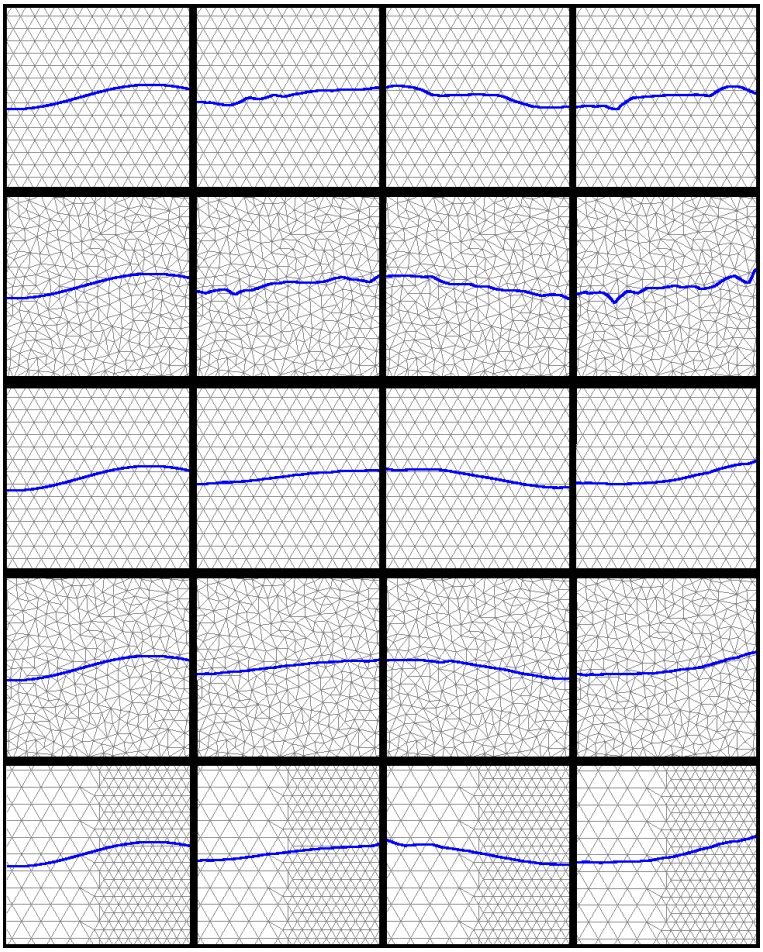


Figure 5.6: **Sloshing tank** Top row: The standard free surface approach with non-conforming meshes yields bumpy artifacts on the scale of one triangle. Second row: The same approach on an irregular mesh illustrates the mesh-dependency of these artifacts. Third row: Embedded free surfaces with a regular mesh yields smooth sloshing without artifacts. Fourth row: Embedded free surfaces with an irregular mesh yields behaviour consistent with the regular mesh case, demonstrating mesh independence. Bottom row: Grading across free surfaces introduces no errors.

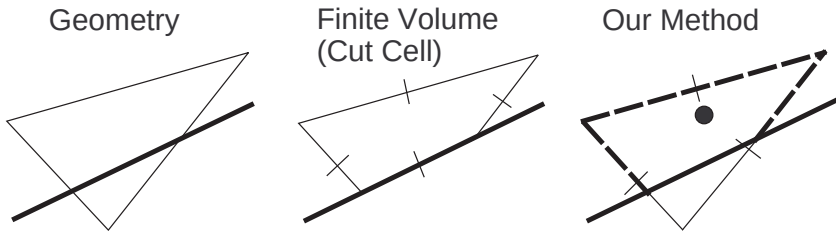


Figure 5.7: Left: A 2D example of a solid boundary (thick line) cutting through a triangular element. Centre: A standard finite volume discretization clips the triangle and relocates the velocity samples, requiring complex interpolation to accurately determine pressure gradients. Right: Our embedded boundary scheme uses finite volume face area weights (dashed lines) but leaves velocity positions unchanged, thereby retaining local orthogonality.

simplified cut-cell finite volume discretization, as illustrated in Figure 5.7. The simplification is that the standard cut-cell finite volume approach (eg. [SBCL06]) would interpolate velocity samples to the midpoints of the truncated faces and generate new faces along the boundary, arriving at a non-symmetric linear system. In contrast, our chosen approach weights the original faces by their partial areas, but creates no new faces and leaves the velocity samples at their original positions. This ensures that we retain both the accuracy provided by the local orthogonality property and the symmetric positive definiteness of the standard discretization, without requiring explicit clipping of geometry or complex interpolation schemes to compute orthogonal pressure gradients. This approach can be applied to tetrahedra by simply estimating partial tetrahedra face areas, and extensions to one- and two-way solid coupling are also straightforward, following Batty et al. [BBB07].

In our implementation we store signed distance values for solid boundaries on the vertices of the tetrahedra. This allows face area fractions to be estimated with simple 2D marching triangles cases, and eliminates the need for mesh-based geometric clipping. (In exchange, this gives up a slight amount of resolution, since flows through cracks below the mesh resolution are disallowed.)

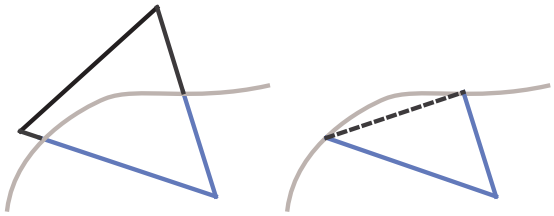


Figure 5.8: Left: A 2D example in which one triangle face is cut off by the solid boundary (curved line) and is assigned a zero area weight. Right: The approximated physical boundary (dashed) has a different average normal than the original triangle face.

For use during advection, a full velocity for each tetrahedron's circumcentre is typically reconstructed from the normal components on its faces using a least-squares fit [ETK+07, KFCO06, PVW06]. Given these circumcentre velocities, we perform interpolation and advection exactly following Chentanez et al. [CFL+07]. However, in our scheme when a face is clipped entirely it has zero associated area weight (Figure 5.8, left). It therefore does not participate in the pressure solve and is not assigned a valid velocity. Naïvely using a zero velocity (or more generally, the wall velocity) for this face frequently destroys free-slip in the reconstruction. This is because the missing face's normal doesn't necessarily match the solid boundary normal (Figure 5.8, right), so it may actually require a non-zero velocity component to be consistent with the fluid velocity.

We handle this by simply dropping the zero-area face's row from the least-squares solve. The use of nodal signed distances to compute weights ensures that if a quality tetrahedron is not entirely solid, it can have only one such zero-area face, leaving three valid velocity components with linearly independent normal vectors. Three independent components are always sufficient to reconstruct a three-dimensional velocity vector, so the linear system is never underdetermined. Furthermore, the divergence-free condition ensures that this velocity is in fact unique

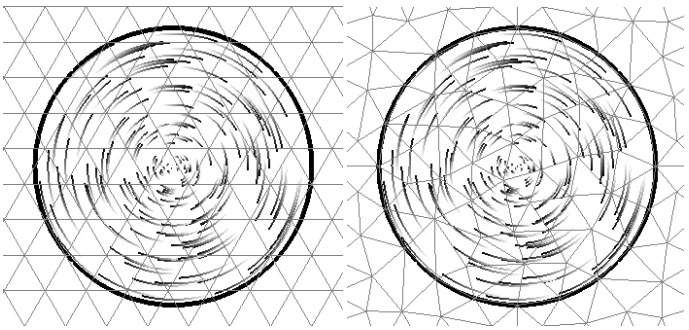


Figure 5.9: Our non-conforming embedded solid boundary method (left) compared against the standard conforming method (right), for a low resolution rotating disk of fluid visualized with streamlines seeded at the same points. The two are essentially indistinguishable, illustrating that our method accurately handles boundaries and reconstructs the free slip velocity even near zero-area faces.

[ETK+07], and hence no information is lost in dropping the face. This approach robustly reconstructs the free slip fluid velocity.

Conveniently, the free surface and solid embedded boundary methods above are entirely complementary, as illustrated by Batty & Bridson [BB08]. Perhaps the simplest interpretation is that the free surface condition modifies the gradient operator near air, while the variational pressure projection modifies the divergence operator near solids. By plugging the modified velocity update into the modified divergence operator, we get a method that straightforwardly handles both boundaries without special cases.

5.7 Results

We focus on two-dimensional examples to emphasize the relationship between the simulation and the underlying mesh, however we stress that all of our results extend straightforwardly to 3D. Furthermore, while we do not compare in detail the speed of our method to that of Chentanez et al. [CFL+07], we expect that optimized

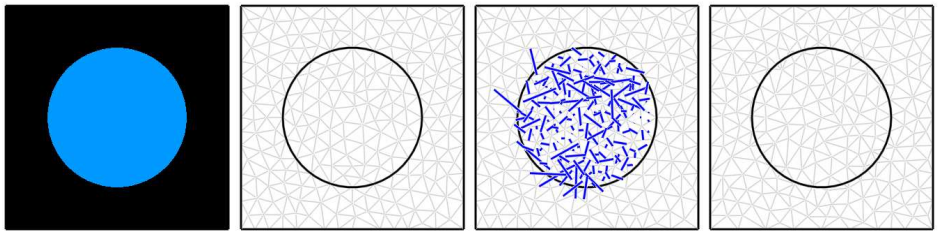


Figure 5.10: From left to right: 1) Input geometry for a closed hydrostatic scene under uniform gravity. 2) A conforming Delaunay mesh with circumcentric pressures finds the exact solution (no motion.) 3) The same mesh using barycentric pressures yields substantial spurious velocities. 4) Using our embedded solid boundary method, the exact solution is found on a non-conforming Delaunay mesh with circumcentric pressures.

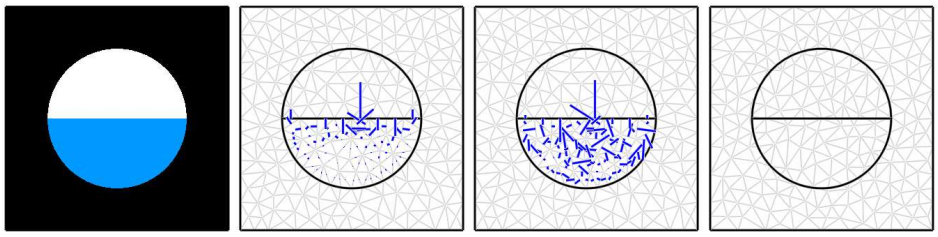


Figure 5.11: From left to right: 1) Input geometry for a free surface hydrostatic test under uniform gravity. 2) Standard free surface boundary conditions introduce spurious velocities near the surface, despite using a conforming circumcentric Delaunay mesh. 3) The use of barycentric pressures with the same mesh and boundary conditions worsens the errors. 4) A non-conforming Delaunay mesh with circumcentric pressures using our embedded solid and free surface boundary conditions finds the exact solution.

implementations will exhibit approximately the same speed, assuming we replace our CG routine with AMG and remesh continuously for both. Our reasoning is that both our proposed method and isosurface stuffing first build a (potentially adaptive) BCC lattice. Isosurface stuffing then queries the boundary isosurface in order to stuff the tetrahedra inside it, whereas our method instead uses these queries to determine the solid and free surface weights used by the pressure discretization. The methods can otherwise be made nearly identical, so we will aim to demonstrate the key accuracy and flexibility benefits of our method.

Figures 5.10 and 5.11 illustrate the ability of our projection method to correctly handle the hydrostatic scenario, consisting of a vertical gravity force that should be precisely cancelled by the resulting pressure gradient, for a completely enclosed case and a free surface case. We compare against both circumcentric and barycentric conforming Delaunay meshes, confirming that circumcentres are preferred. Note that only our scheme is able to achieve the correct cancellation in the difficult free surface scenario, and to our knowledge it is the first tetrahedral method in computer graphics able to do so.

To illustrate the improved motion provided by embedded free surfaces, Figure 5.6 compares several versions of a slow-moving sloshing scenario. The basic non-conforming method yields bumpy artifacts due to its inability to "see" waves below the scale of one triangle. These artifacts are also highly mesh-dependent; the irregular mesh produces different (and substantially worse) motion. The same example using embedded free surfaces yield smoother and more consistent results regardless of the underlying mesh, even when grading is performed across the free surface itself. The associated animations demonstrate that the embedded approach also exhibits less artificial damping.

An example in our accompanying video compares a 2D free surface flow simulated with no remeshing, continuous remeshing, and intermittent remeshing every

5 time steps, on a high quality non-conforming adaptive lattice, like that in Figure 5.4. The non-remeshed example is chosen to have approximately the same number of triangles as an early frame of the adaptive case. The low resolution of the non-remeshed example performs comparatively poorly, however both intermittent and continuous remeshing provide much better and qualitatively similar results despite the former expending one-fifth as much effort on remeshing. Real applications might use more elaborate adaptivity criteria, but this already illustrates the flexibility and power of combining adaptivity with our embedding scheme: it enables an explicit trade-off between accuracy and remeshing costs. This also suggests an interesting potential optimization: at the cost of slightly lagged adaptivity updates, remeshing could be performed in parallel such that the simulator proceeds with the current mesh until a new one becomes available. This holds out the possibility of entirely hiding the costs of remeshing.

To illustrate that our non-conforming solid boundary approach gives qualitatively identical results to a conforming scheme and fully reconstructs free slip velocities near walls, we compare frames from a simple rotation inside a disk-shaped 2D domain. Visualized with streamlines in Figure 5.9, it is clear that the two methods are almost perfectly consistent. The accompanying video makes a similar comparison against a naïve rasterized non-conforming approach, which exhibits erroneous damping and inaccurate motion near walls.

Our video also includes a complex 2D animation consisting of liquid drops splashing against curved and angled solid boundaries. This emphasizes that our two boundary conditions can be used together without artifacts. Lastly, Figure 5.1 shows a 3D breaking dam example similar to one by Chentanez et al. [CFL+07]. The simulation used adaptive BCC lattice meshes ranging between 400K and 1.1M non-conforming tetrahedra, yet achieves accurate and smooth liquid motion. Computation times averaged 31 seconds/frame (about 40% of which is our single-

threaded particle level set surface tracker) on a 4-core Intel i7 860. The simulator was parallelized using Intel's Threading Building Blocks library.

5.8 Conclusions and Future Work

We have demonstrated a few simple modifications to tetrahedral mesh fluid simulation that can improve its accuracy, flexibility, and speed. The use of non-conforming Delaunay meshes together with embedded boundary techniques improves the liquid behaviour in many scenarios while substantially reducing the frequency, complexity, and costs of high quality meshing. This has the potential to make tetrahedral schemes more competitive with regular grid methods, while retaining the vital advantage of adaptivity.

There are several directions for future work. Studying the accuracy and convergence of our tetrahedral embedded boundary techniques would be valuable. A potential drawback of circumcentric pressures is that they are not necessarily contained in their associated tetrahedra, and though orthogonality ensures first order accuracy, this might impact the magnitude of the approximation error on low quality meshes. Research into so-called well-centred meshes might prove useful in this respect [VHG08]. Relatedly, our method should adapt seamlessly onto unstructured Voronoi meshes (eg. [SBH09]), where Voronoi sites *are* guaranteed to be inside their associated cells. Extending unstructured meshes to support free surface viscosity, viscoelasticity, and surface tension are other interesting directions. Finally, given the ubiquity of Poisson problems in graphics research, these embedded boundary schemes could likely benefit applications outside of fluid animation.

Acknowledgements We thank Evan VanderZee and Dr. Anil Hirani for sharing their expertise on well-centred meshes, and Dr. Robert Bridson for his encouragement, advice, and public domain code. We also thank the anonymous reviewers whose comments lead

to substantial improvements of this paper.

Bibliography

[ACSYD05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):617–625, 2005.

[BB08] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*, pages 219–228, 2008.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):100, 2007.

[BF08] Petter Berthelsen and Odd Faltinsen. A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *J. Comp. Phys.*, 227(9):4354–5397, 2008.

[BO84] Marsha J. Berger and Joseph E. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53(3):484–512, 1984.

[Bri08] Robert Bridson. *Fluid Simulation for Computer Graphics*. 2008.

[CFL+07] Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan Richard Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Symposium on Computer Animation*, pages 219–228, 2007.

[CGC+02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A multiresolution framework for dynamic deformations. In *Symposium on Computer Animation*, pages 41–47, 2002.

[CGFO06] Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Symposium on Computer Animation*, pages 83–89, 2006.

[CS70] Robert K.-C. Chan and Robert L Street. A computer study of finite amplitude water waves. *J. Comp. Phys.*, 6:68–94, 1970.

[CSCY04] David Cohen-Steiner, Eric Colin De Verdiére, and Mariette Yvinec. Conforming Delaunay triangulations in 3D. *Computational Geometry*, 28:217–233, 2004.

[ENGF03] Doug Enright, Duc Nguyen, Frédéric Gibou, and Ron Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, 2003.

[ETK+07] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[FD07] Krzysztof J. Fidkowski and David L. Darmofal. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *J. Comp. Phys.*, 225(2):1653–1672, 2007.

[Fel07] Bryan E. Feldman. *Fluid animation from simulation on tetrahedral meshes*. PhD thesis, 2007.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):904–909, July 2005.

[FOKG05] Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. Fluids in deforming meshes. In *Symposium on Computer Animation*, pages 255–259, 2005.

[GFCK02] Frédéric Gibou, Ron Fedkiw, L.-T. Cheng, and Myungjoo Kang. A second order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comp. Phys.*, 176(1):205–227, 2002.

[IGLF06] Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):805–811, 2006.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):820–825, 2006.

[KIC06] Janghee Kim, Insung Ihm, and Deukhyun Cha. View dependent adaptive animation of liquids. *ETRI Journal*, 28(6):697–708, 2006.

[KLMU05] S. Krishnan, Hao Liu, S. Marella, and H. S. Udaykumar. Sharp interface Cartesian grid method II: A technique for simulating droplet interactions with surfaces of arbitrary shape. *J. Comp. Phys.*, 210(1):32–54, 2005.

[LCC+08] Rainald Löhner, Juan R. Cebral, Fernando F. Camelli, S. Appanaboyina, Joseph D. Baum, Eric L. Mestreau, and Orlando A. Soto. Adaptive embedded and immersed unstructured grid techniques. *Computer Methods in Applied Mechanics and Engineering*, 197(25-28):2173–2197, April 2008.

[LFO05] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995–1010, 2005.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):457–462, August 2004.

[LS07] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):57, 2007.

[MCP+09] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):38, 2009.

[MDB+08] Rajat Mittal, H. Dong, M. Bozkurttasa, F. M. Najjarb, A. Vargasa, and A. von Loebbeckea. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *J. Comp. Phys.*, 10(1):4825–4852, 2008.

[MKLU05] S. Marella, S. Krishnan, Hao Liu, and H. S. Udaykumar. Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations. *J. Comp. Phys.*, 210(1):1–31, 2005.

[Nic92] Roy A. Nicolaides. Direct discretization of planar div-curl problems. *SIAM J. Numer. Anal.*, 29(1):32–56, 1992.

[NMG09] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comp. Phys.*, 228(23):8807–8829, 2009.

[NW97] Roy A. Nicolaides and Xiaonan Wu. Covolume solutions of three-dimensional div-curl equations. *SIAM J. Numer. Anal.*, 34(6):2195–2203, 1997.

[PB79] James W. Purvis and John E. Burkhalter. Prediction of critical Mach number for store configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[Pes02] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[PS07] Blair Perot and V. Subramanian. Discrete calculus methods for diffusion. *J. Comp. Phys.*, 224(1):59–81, 2007.

[PVW06] Blair Perot, Dragan Vidovic, and Pieter Wesseling. Mimetic reconstruction of vectors. In Douglas N Arnold, Pavel B Bochev, Richard B Lehoucq, Roy A Nicolaides, and Mikhail Shashkov, editors, *Compatible Spatial Discretizations*, pages 173–188. 2006.

[RbZF05] Doug Roble, Nafees bin Zafar, and Henrik Falt. Cartesian grid fluid simulation with irregular boundary voxels. In *SIGGRAPH Sketches*, 2005.

[SBCL06] Peter Schwartz, Michael Barad, Phillip Colella, and Terry Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comp. Phys.*, 211(2):531–550, 2006.

[SBH09] Funshing Sin, Adam W. Bargteil, and Jessica K. Hodgins. A point-based method for animating incompressible flow. In *Symposium on Computer Animation*, pages 247–255, 2009.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999.

[TAL09] Philippe Traoré, Yves Marcel Ahipo, and Christophe Louste. A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries. *J. Comp. Phys.*, 228(14):5148–5159, 2009.

[TWAD09] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):75, 2009.

[VHG08] Evan VanderZee, Anil N. Hirani, and D. Guoy. Triangulation of simple 3D shapes with well-centered tetrahedra. In *Proceedings of the 17th International Meshing Roundtable*, 2008.

[WBOL07] Jeremy D. Wendt, William Baxter, Ipek Oguz, and Ming C. Lin. Finite volume flow simulations on arbitrary domains. *Graphical Models*, 69(1):19–32, 2007.

[WSHD07] Joe Warren, Scott Schaefer, Anil N. Hirani, and Mathieu Desbrun. Barycentric coordinates for convex sets. *Advances in computational mathematics*, 27(3):319–338, 2007.

[WT08] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):47, 2008.

[ZSP02] Xing Zhang, David Schmidt, and Blair Perot. Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics. *J. Comp. Phys.*, 175(2):764–791, 2002.

Chapter 6

Matching Fluid Simulation Elements to Surface Geometry and Topology

6.1 Introduction

One of the most visually compelling aspects of liquids is the variety of complex thin sheets and droplets that arise during splashing. However, these remain among the most difficult features to simulate plausibly and accurately with existing techniques. Such detailed behaviour is extremely computationally expensive to resolve because of the tremendous grid resolution required for both the fluid solver and the surface tracking mechanism.

Recent advances in explicit surface tracking with triangle meshes [WTGT09, BB09, MÖ9] have made feasible the geometric representation and manipulation of small features, without the loss of detail exhibited by implicit surface methods. However, when the surface is coupled to a standard Eulerian simulator, the liquid

A version of this chapter has been published. Brochu, T., Batty, C., and Bridson, R. (2010) Matching Fluid Simulation Elements to Surface Geometry and Topology, ACM Transactions on Graphics (Proc. SIGGRAPH) 29(3).

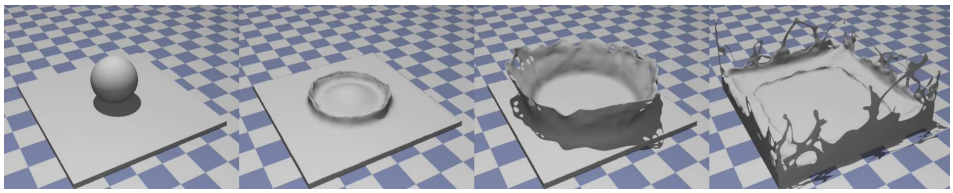


Figure 6.1: **Sphere Splash.** Coupling an explicit surface tracker to a Voronoi simulation mesh built from pressure points sampled in a geometry-aware fashion lets us capture very fine details in this sphere splash animation that uses only 314K tetrahedra.

volume must first be resampled onto the simulation mesh or grid to provide geometric information for boundary conditions. As this resampling process typically destroys small details, they are invisible to the fluid solver and cannot be advanced appropriately. This can lead to a variety of visible artifacts including lingering surface noise, liquid behaving as if it were connected when it is not (and vice versa), and thin features simply halting in mid-air because the simulator fails to see them [BOGS06, KSK09]. When combined with surface tension forces, noisy sub-mesh details can also severely hamper stability if they are not artificially smoothed out.

We will address these problems by constructing a simulator that "sees" every detail in the explicit liquid surface. We carefully generate pressure sample points near the liquid surface, build a Voronoi simulation mesh from these points and a background lattice, and apply a ghost fluid/finite volume pressure discretization which captures the precise position of the liquid interface. We couple this with a semi-Lagrangian advection scheme and a new approach to surface tension, arriving at a complete liquid simulator.

In summary, our key contribution is coupling an explicit surface tracker to a Voronoi-based liquid simulator with:

- a pressure sample placement strategy that captures the complete liquid surface geometry,

- an accurate surface tension model combining mesh-based curvature estimates and ghost fluid boundary conditions,
- embedded free surface and solid boundary conditions adapted to Voronoi cells, avoiding the need for more onerous conforming tetrahedral mesh generation,
- and a new velocity interpolant over unstructured meshes.

The practical benefits of such a system include:

- improved animation of detailed liquid features, including very thin sheets, tendrils and droplets,
- elimination of noise in explicit surface tracking without non-physical smoothing,
- more detailed and less damped surface tension effects,
- and faster semi-Lagrangian advection on unstructured meshes without increased dissipation.

6.2 Related Work

6.2.1 Unstructured Mesh Fluids

Unstructured and semi-structured meshes have a long history in computational fluid dynamics, and have gained traction in computer animation as well. An important reason for their popularity is that careful control of mesh geometry can simplify the discretization or improve accuracy. For example, conforming the simulation mesh to solid walls makes the no-flow boundary condition trivial, and adaptivity can be easily introduced by grading mesh elements as desired. Past work in

graphics has extensively explored finite volume methods for tetrahedral meshes [FOK05, FOKG05, KFCO06, CGFO06, ETK+07, WBOL07, CFL+07], and now many of the features of standard grid-based solvers are supported on tetrahedra, including free surfaces and implicit coupling to dynamic solids. Batty et al. [BXH10] augmented this approach with embedded boundaries [ENGF03, BBB07], improving free surface accuracy and reducing remeshing complexity. Our method extends these advantages to Voronoi meshes.

In a related approach, Sin et al. [SBH09] developed a particle method which solves a finite volume pressure projection on the Voronoi diagram of the liquid particles. An advantage of this approach is that the pressure degrees of freedom are directly tied to the number of particles, so there can never be a resolution mismatch between surface geometry and simulator. This idea motivates our work.

Franklin & Lee [FL10] subdivide polyhedra into tetrahedra for interpolation similar to our method, but our method is simpler due to use of the Voronoi diagram.

6.2.2 Surface Tracking

Implicit surfaces have long been used to capture liquid geometry in animation; this family of schemes includes level set (LS) methods [EFFM02], volume-of-fluid (VOF) [MUM+06, MMTD07], and semi-Lagrangian contouring (SLC) [BOGS06]. Implicit approaches naturally yield smooth surfaces and seamlessly handle topological change. However, the resolution of the underlying grid imposes a severe limit on the smallest representable feature, beyond which geometry either vanishes (LS, SLC) or artificially coalesces into grid-scale "flotsam and jetsam" (VOF). Ensuring temporal coherence and avoiding visual artifacts due to the use of regular grids can also be problematic.

The shortcomings of implicit schemes have spurred interest in explicit meth-

ods, i.e. "front tracking" [GGL+98]. Here the surface is represented explicitly as a triangle mesh, whose vertices are moved with the fluid velocity field. The greatest challenge is handling topological change, due to mesh tangling that may occur during merging and splitting. One solution is to determine problematic regions, switch to an implicit surface to repair the tangles there, then stitch back in a new consistent mesh patch [DFG+06, WTGT09]. Müller [MÖ9] takes a similar grid-based approach to untangling, rebuilding a consistent mesh using marching-cubes-like stencils. Unfortunately these methods still are subject, in complex regions, to a resolution limited by the voxel grid.

Another approach is to work strictly on the triangle mesh itself, using "mesh surgery" for repairs. While this is difficult in general, Brochu & Bridson [BB09] recently showed that the problem can be simplified using ideas from cloth animation, enforcing the invariant that the surface remain intersection-free. Topological operations are only allowed when safe, while robust collision processing is used as a last resort to avoid tangles, i.e. the surface is minimally perturbed to avoid problems. We use this method in the presented examples, though note that other front tracking methods could easily be used instead—for example, recent work by Campen & Kobbelt [CK10] suggests that the need for collision processing could be obviated with exact Boolean operations.

6.2.3 Surface Resolution vs. Simulation Resolution

A prime focus of our work is matching the surface mesh resolution to that of the liquid solver. Most level set-based solvers use one level set sample per pressure grid cell, conservatively avoiding resolution inconsistencies (e.g. [FF01, EMF02]). Goktekin et al. [GBO04] experimented with a double-resolution level set, trading better volume conservation for other artifacts. Bargteil et al. [BOGS06] similarly

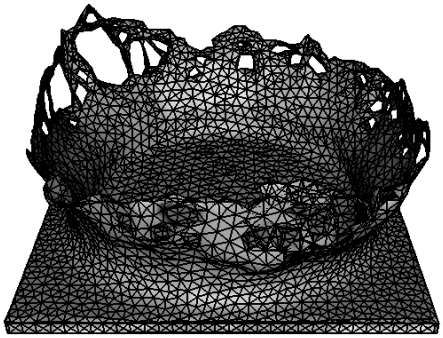


Figure 6.2: **Explicit Surface Tracking.** Our method exploits the *El Topo* explicit mesh tracking software to capture thin features.

coupled an octree contouring method to a uniform grid fluid solver and explicitly discussed potential artifacts due to resolution mismatch, such as erroneously preserving surface noise and the solver interpreting disconnected fluid regions as connected. Kim et al. [KSK09] coupled a high resolution particle level set to a low resolution ghost fluid-based liquid solver, but ensured that pressure projection captured all liquid geometry by resampling an inflated level set at the pressure grid resolution—however, this can exacerbate other artifacts, since liquid components behave as if half a cell-width larger than they appear. Kim et al. also introduced extra surface smoothing to prevent retention of small-scale noise.

Mismatched resolutions have been found useful for deformable solids, particularly as surface details are expected to generally persist, unlike in liquids. For example, Wojtan & Turk [WT08] used a surface mesh coupled to a lower resolution finite element solver; forcing the simulation mesh to have the same topology, if not resolution, as the embedded surface mesh may improve realism [TSB$^+$05, NKJF09].

6.2.4 Surface Tension Models

Approaches to surface tension generally fall into two categories: those which apply surface tension as a body force in a region around the interface via smeared delta functions [BKZ92, HK03, ZYP06, WTGT09], and those which apply surface tension *discontinuously* at the interface, typically as a boundary condition in the pressure projection step. The latter is exemplified by the ghost fluid method and related approaches [ENGF03, HK05, HSKF07], and has been shown to provide more realistic results.

Surface tension models can also be compared in terms of how the force itself is approximated. In level set schemes, finite differences are often used to estimate mean curvature, though this can be quite inaccurate without careful modification (e.g. [Shi07]) and cannot capture small details. If a surface mesh is available, a more accurate approach is either to use mesh-based curvature operators (e.g. [MDSB02]), or as proposed recently, to model a physical tension directly in the surface mesh geometry [PN03, Bro06, WT08].

We take the best of each, computing an accurate force from the surface mesh and incorporating it precisely at the surface with the ghost fluid method. We also remedy a shortcoming of existing mesh-based approaches: that surface details below the simulation resolution add energy but cannot be correctly evolved by the solver; without correct feedback from the physics this noise tends to worsen and destroy stability. Wojtan & Turk [WT08] handle this with Laplacian smoothing to eliminate small features: note, however, this non-physical operation is dissipative rather than conservative. By instead combining our surface tension model with a geometry-aware sampling, we ensure all relevant details are properly resolved. This yields accurate and comparatively stable surface tension effects without artificial smoothing.

6.3 Algorithm Outline

We simulate inviscid liquids with semi-Lagrangian advection and an embedded-boundary finite volume pressure projection. We generally follow the tetrahedral scheme of Batty et al. [BXH10] with modifications to use specially designed Voronoi meshes instead. Like Sin et al. [SBH09], we place pressure samples on the vertices of a Delaunay tetrahedral mesh, corresponding to the sites of the dual Voronoi diagram (figures 6.3(a) and 6.3(b)). Normal components of velocity lie on the faces of the Voronoi cells, so that the velocity sample is parallel to the line segment connecting the pressure samples in the Delaunay mesh. This configuration requires a slightly different velocity reconstruction compared to previous methods, but semi-Lagrangian advection is otherwise straightforward.

For front tracking, we used Brochu & Bridson's *El Topo* code [BB09], in particular using its triangle mesh surface to determine the location of pressure samples for our Voronoi simulation mesh.

Purely explicit front tracking algorithms generally use mesh refinement and coarsening to maintain a high quality discretization as the surface deforms. *El Topo* uses a sequence of edge subdivision, collapse and flipping operations, combined with null-space Laplacian smoothing. While these operations change mesh connectivity, they are designed to be geometry-preserving. For example, the smoothing moves vertices only in the null space of the local quadric metric tensor [GH97], as suggested by Jiao [Jia07]. If the vertex lies on a locally smooth patch it is moved in the plane tangent to the surface, but if on a ridge or corner it is moved only along this line. Therefore, sharp features are preserved, allowing the present paper's algorithm to handle them physically.

The solver runs through the following stages each time step:

1. Advect the explicit surface with *El Topo*.

2. Generate a new simulation mesh as the Voronoi diagram of a lattice with extra samples near the liquid surface (section 6.5).

3. Advect velocities onto the new mesh with semi-Lagrangian advection (section 6.6).

4. Add external forces—typically just gravity.

5. Solve for the embedded-boundary pressure projection on the Voronoi mesh, including surface tension forces (section 6.4).

6.4 Embedded Boundaries on Voronoi Meshes

We use finite volumes on a Voronoi mesh for the pressure projection step, similar to Sin et al. [SBH09]. However, rather than applying boundary conditions as they describe, we adapt the embedded boundary methods of Batty et al. [BXH10] to Voronoi meshes. Conveniently, the duality/orthogonality relationship between Voronoi and Delaunay meshes lets the accuracy benefits of the method carry over. Figure 6.3 illustrates our mesh configuration, and the computation of the required weights, as discussed below. We solve the resulting symmetric positive definite linear system using incomplete Cholesky-preconditioned conjugate gradients.

To enforce embedded solid boundary conditions, we need to estimate the partial unobstructed area of each element face (figure 6.3(d)). Batty et al. [BXH10] used marching triangles cases for computing tetrahedra face fractions from signed distance values on the vertices. However, in the Voronoi setting, the faces are arbitrary convex planar polygons rather than triangles. To handle this, we temporarily place an extra vertex at the face centroid, and use it to triangulate the face. We then use signed distance estimates at the vertices to compute each sub-triangle's partial area, and sum them to determine the partial area for the complete face.

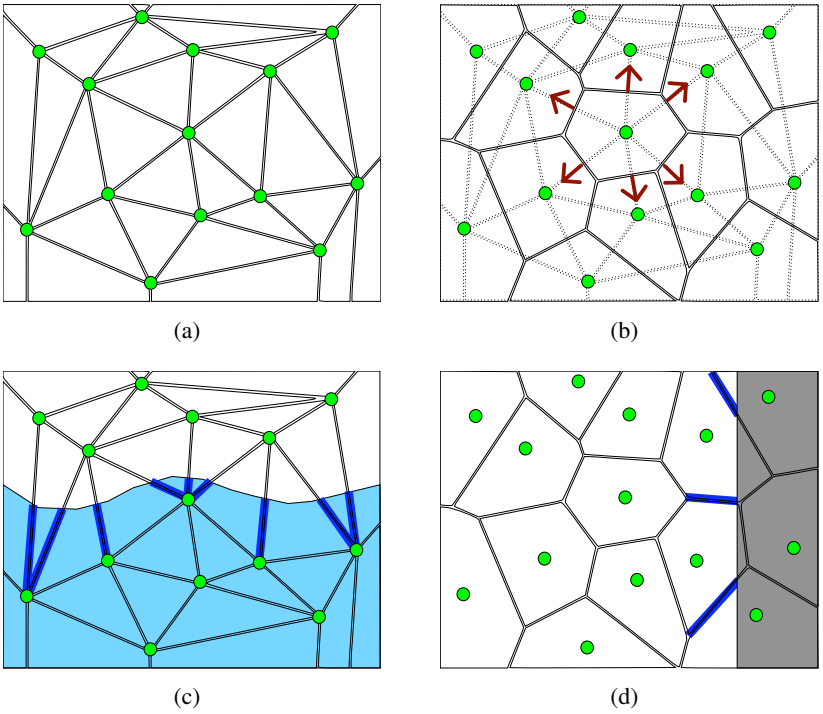


Figure 6.3: **Embedded boundaries on Voronoi/Delaunay meshes.** Pressure samples are shown as green circles. (a) Delaunay triangulation. (b) Voronoi diagram dual to the Delaunay triangulation (velocity components for the central cell are shown as red arrows). (c) Computation of ghost fluid weights on the edges of the triangulation. (d) Computation of non-solid weights on the faces of the Voronoi diagram. In 2D, Voronoi faces are simply line segments, so solid weights are just fractions of segment lengths. In 3D, Voronoi faces are convex polygons, so determining non-solid weights involves computing polygon areas.

The embedded (ghost fluid) free surface condition uses signed distance estimates at pressure samples to estimate the surface position; these are now located at Voronoi sites rather than tetrahedra circumcenters, but the method is otherwise unchanged (figure 6.3(c)). A slight improvement can be achieved by casting rays to find the exact position of the surface mesh between pressure samples. In some cases this is much more accurate than the estimate derived from signed distances, but in practice we found it made minimal visual difference. To actually compute the liquid signed distance field on the tetrahedral mesh, we compute exact geometric distance for a narrow band of tetrahedra near the surface, then use a graph-based propagation of closest triangle indices to roughly fill in the rest of the mesh. This family of redistancing schemes is described by Bridson [Bri08], and is easily adapted to tetrahedra.

6.4.1 Surface Tension

To incorporate surface tension, we follow Enright et al. [ENGF03] in setting the free surface pressure $p_{\text{fs}} = p_{\text{air}} + \gamma\kappa_{\text{fs}}$, where p_{air} is the constant air pressure, γ is the surface tension coefficient and κ_{fs} is the mean curvature of the surface.

Rather than using level set finite differences, we compute curvature directly from the surface mesh to accurately capture high-frequency features. We chose the operator of Meyer et al. [MDSB02] because it provides high quality estimates using just the one-ring of triangles surrounding each vertex, but others could work too.

Curvature is evaluated at the intersection point between the the triangle mesh surface and the line joining an interior pressure sample to an exterior one. Often this intersection point will coincide with a surface mesh vertex due to our choice of sampling scheme; where it does not, we use simple linear interpolation between

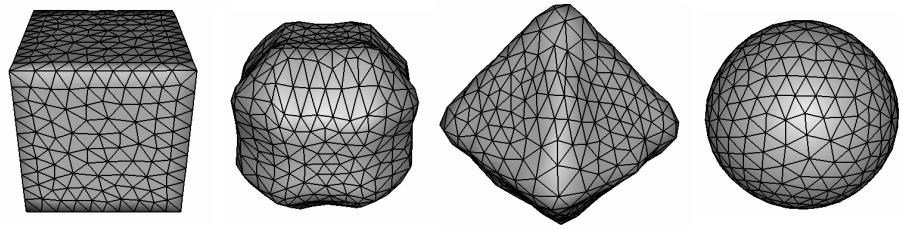


Figure 6.4: **Surface Tension.** Our accurate surface tension model captures capillary waves even on relatively low resolution meshes. From left to right: A cube in zero gravity begins to collapse due to surface tension, inverts to become an octahedron, and continues to oscillate rapidly before settling down to a sphere.

the vertices of the surface triangle mesh. This method appears highly accurate, and leads to much less damping than that of Wojtan et al. [WTGT09].

6.5 Mesh Generation

An advantage of a Voronoi-based discretization is the freedom to explicitly choose pressure sample locations, which is critical for accurate ghost fluid free surface conditions as the signed distance at these samples communicate the surface geometry to the solver. We can visualize the solver's "knowledge" by contouring this level set: figures 6.5 and 6.6 illustrate how uniform sampling may fail.

Careful pressure sample placement with respect to the surface helps in three important ways. First, we can inform the solver of all local geometric extrema, allowing the physics to act upon them correctly. This eliminates the accumulation of erroneous surface noise without requiring non-physical smoothing; this is especially vital for surface tension where spurious noise affects the curvature estimates and induces disastrously large yet futile compensating velocities that destabilize the simulation. Second, we can ensure that the solver sees the correct surface *topology* so that the physics responds to merging or splitting only when the surface mesh

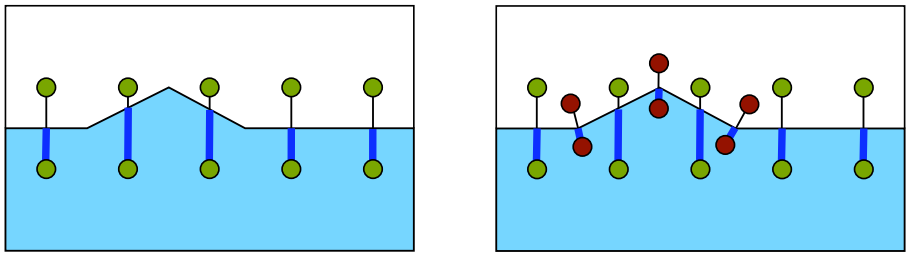


Figure 6.5: Left: Even with the ghost fluid method, regular sampling may miss surface details which do not align with the simulation mesh, such as this wave crest. Right: Adaptive samples (shown in red) placed on either side of each mesh vertex ensure that all geometric detail is resolved by the simulation.

itself merges or splits. Lastly, grid-scale features often disappear and reappear in regular grid sampling, from the perspective of the solver, as the surface translates through the grid. By specifically placing points inside such small features, we ensure they cannot be missed.

Comparison to Adaptive Lattices: The brute-force approach to these issues is to locally refine using octree grids or graded BCC lattice tetrahedra to capture smaller features. However, this scales poorly since many of the extra samples yield little benefit, while incurring memory and computational overhead. Furthermore, there remains no guarantee that features below the smallest grid cell size will be captured. By choosing sample points to precisely capture the geometry rather than naïvely increasing sample density, we can guarantee sampling of features which would require potentially orders of magnitude more samples with pure adaptive lattices.

Comparison to Conforming Tetrahedra: While the tetrahedral method of Chentanez et al. [CFL$^+$07] also builds a volumetric mesh that attempts to respect

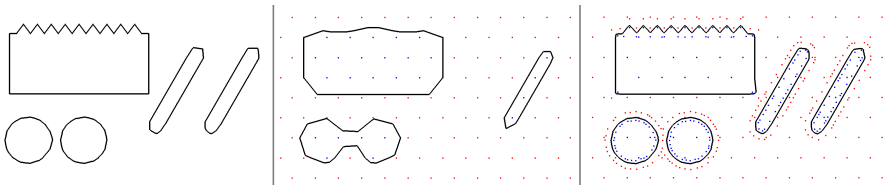


Figure 6.6: Left: The input surface geometry. Centre: The resulting surface after resampling onto a regular lattice simulation mesh. Note the spurious topology change, rounding of sharp features, aliasing of high frequency details, and the complete disappearance of one small fluid component due to poor placement relative to the mesh samples. Right: The resampled surface after adding geometry-aware sample points to the simulation mesh; the result is much more consistent with the input. (Mesh sample locations are indicated by points, coloured blue when inside, red when outside.)

the liquid surface, it matches boundary faces rather than positioning pressure samples. This is considerably more difficult than non-conforming Delaunay tetrahedralization, and generally requires more Steiner points, worse-shaped tetrahedra, and/or the loss of the Delaunay property. Since our method uses embedded boundary conditions, we do not require conforming elements. (Note that this advantage is shared by the method of Batty et al. [BXH10].) Moreover, the position of pressure samples plays a more important role in free surface conditions than the position of element faces. As accuracy requires that tetrahedral schemes store pressures at circumcenters [KFCO06, BXH10], and since circumcenters often lie outside their associated tetrahedra, even filling a thin feature with conforming tetrahedra provides no guarantee that its interior will be sampled at all.

6.5.1 Pressure Sample Placement Strategy

We begin by choosing a characteristic length scale for the simulation, Δx, and configure *El Topo* to try to maintain triangle edge lengths in the range $[\frac{1}{2}\Delta x, \frac{3}{2}\Delta x]$. To resolve all surface details with our volumetric mesh, we need to place pressure

samples so that they capture the surface's local geometric extrema, i.e. around surface mesh vertices. In particular, we try to ensure that one edge of the Delaunay triangulation passes through each surface vertex, with one sample inside and one outside. Therefore we take the inward and outward normal at each surface vertex (averaged from the incident surface triangles), and attempt to place a pressure sample a short distance along each. We placed outward samples at $\frac{1}{2}\Delta x$ and inner samples at $\frac{1}{4}\Delta x$, though other ratios would work as well. As a result, surface mesh normal directions will often align exactly with a velocity sample in the simulation mesh; this lends additional accuracy to the vertex's normal motion, and to the incorporation of the normal force due to surface tension calculated at the vertex.

This placement may miss very thin sheets or other fine structures: to robustly sample such features, we check line segments of length Δx from each surface vertex in both offset directions for intersection with the rest of the surface mesh. If we find any triangle closer than Δx, we store the distance d to the closest intersection, and use d in place of Δx in the offset distance calculations above (see figure 6.7). We further reject new pressure samples which are too close to an existing sample by some epsilon, which would cause a very short edge in the final mesh.

If the distance between the surface vertex and the first intersection is below some threshold (e.g. $\frac{1}{20}\Delta x$) at which we consider the two surfaces to have effectively collided, and the proposed sample is an air sample, we also discard it. This is necessary because the divergence constraint is not enforced on air cells, so they can act as liquid sinks [LSSF06] and destroy liquid volume until the geometry finally merges. Unfortunately, merging in this scenario can often take several time steps to resolve because the interpolated velocity in the air gap still averages to zero, thereby preventing surface geometry from actually intersecting and flagging a collision. By not placing a sample point in these very small gaps, our simulator treats the two liquid bodies as merged and prevents volume loss; the geometric

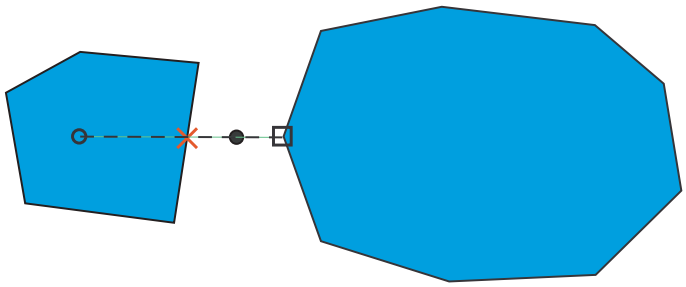


Figure 6.7: **Sampling Thin Features.** A pressure sample is seeded along the outward normal direction from a surface vertex (black square). The initial proposed pressure location (empty black circle) would land in the wrong component and potentially fail to resolve the intervening air gap. We instead place the final pressure sample (filled black circle) midway between the starting vertex and the first intersection point (red X).

merge is usually then processed within a few timesteps. (With regular sampling, merging will depend on where grid points happen to fall with respect to the surface; hence the physics can respond as if merged when the surfaces are still as much as Δx apart, as in figure 6.9. This generates non-physical air bubbles which linger for many timesteps before they self-collide and are eliminated.)

After placing the surface-adapted pressure samples, we complete the sampling of the domain by adding regularly-spaced points from a BCC lattice with cell size $2\Delta x$, again rejecting samples which fall too near existing samples—of course, a graded octree or any other strategy could also be used to fill the domain. All samples are then run through a Delaunay mesh generator such as *TetGen* [Si06]. Figure 6.8 illustrates in 2D how this sampling approach is able to capture thin features such as splashes. Further experimentation with relative mesh spacing parameters could yield improved results.

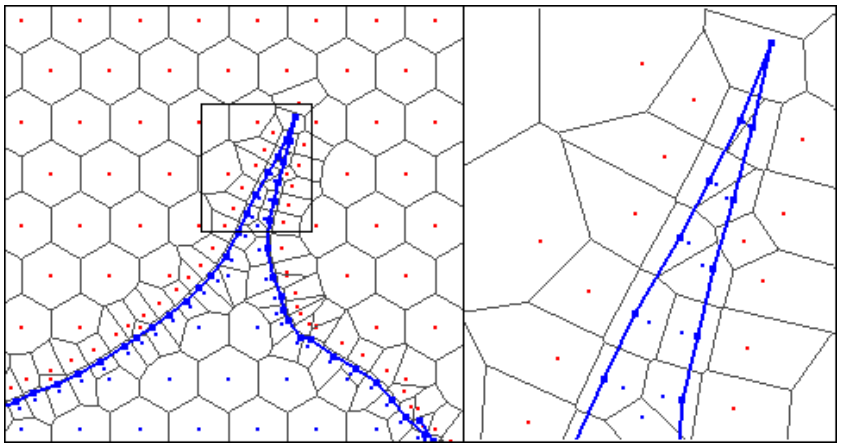


Figure 6.8: **Simulating Thin Features.** A 2D example of a thin feature simulated with our method. The zoom on the right illustrates the sample placement with respect to surface vertices, and the resulting Voronoi mesh. Notice that even the very sharp tip contains a pressure sample, as indicated by the surrounding Voronoi cell.

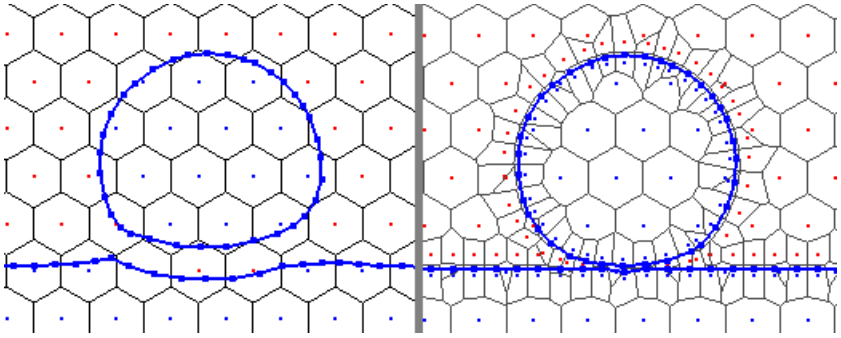


Figure 6.9: **Merging.** Left: Regular sampling erroneously identifies a topology change, causing a premature reaction in both liquid bodies. Right: Geometry-aware sampling responds correctly.

6.6 Interpolation and Advection

Velocity interpolation methods for unstructured meshes typically proceed in two steps [KFCO06, ETK+07, BXH10]. First, a full velocity vector is reconstructed at selected mesh locations using a least-squares fit to the nearby velocity components. Then barycentric or generalized barycentric interpolation between those locations interpolates velocity over the full domain. Given such an interpolant, advection of velocities and geometry is straightforward. We follow this general framework, with two modifications.

In previous work, face normal components on tetrahedra were used to reconstruct velocities at circumcenters (Voronoi vertices). In our configuration, velocity components instead lie along the tetrahedra edges (Voronoi faces) so we perform the least squares fit on this data instead. We could then apply the usual generalized barycentric interpolant over Voronoi cells, but this is expensive [CFL+07] and requires special case handling to avoid degeneracies [MBLD02]. A simple and fast alternative discussed by Klingner et al. and Chentanez et al. is to first interpolate velocities to Voronoi sites (tetrahedra vertices) and apply standard (and fast) barycentric interpolation over each tetrahedron. However, the interpolation onto tetrahedra vertices discards any local extrema at the Voronoi vertices, thereby severely over-smoothing the velocity field in practice, damping out interesting flow behavior.

Rather than discard extrema at Voronoi vertices, we use a slightly refined tetrahedral mesh that *includes* them. We conceptually tetrahedralize the Voronoi cells themselves by placing additional vertices at Voronoi face centroids and Voronoi sites (see figure 6.10). Velocities for each of these new points need to be computed; while previous work used the generalized barycentric interpolant for this transfer step, we found that simply averaging the velocities of the surrounding ring or cell

of Voronoi vertices is quicker and equally effective. For maximum fidelity at the face centroids, we also replace the normal component of the averaged full velocity with the exact normal component already stored at the face. Simple and efficient barycentric interpolations can then be applied on the resulting smaller tetrahedra. Because the sharper, more accurate velocities at the Voronoi vertices are retained and merely augmented with additional data, this is far less dissipative, yielding results that closely match generalized barycentric interpolation (see figure 6.11).

Lastly, note that reconstructions should only use face velocities which were assigned valid data by the pressure projection, and thus we can only reconstruct reasonable velocities inside the fluid. We therefore extrapolate velocities outwards from the fluid using a breadth-first graph propagation: each unknown point in a layer is set by averaging all adjacent known points from previous layers, repeating until we have a sufficiently large band of velocities surrounding the surface. This simple method, suggested in the context of cloth-fluid coupling by [GSLF05], sufficed for all our animations.

In summary, the steps of our interpolation scheme are:

1. Reconstruct full velocity vectors at Voronoi vertices using least squares.

2. Assign full velocity vectors to Voronoi sites and faces using simple averaging from neighboring vertices.

3. Subdivide the Voronoi cells into sub-tetrahedra using the sites and face centroids (see figure 6.10).

4. Apply a simple graph-based extrapolation of velocities to fill in velocities near the liquid.

5. To interpolate at a point, locate the sub-tetrahedron containing the point and apply basic barycentric interpolation from its four associated data points (i.e.

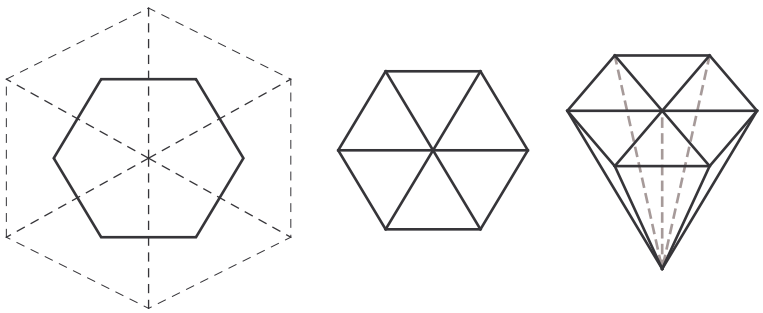
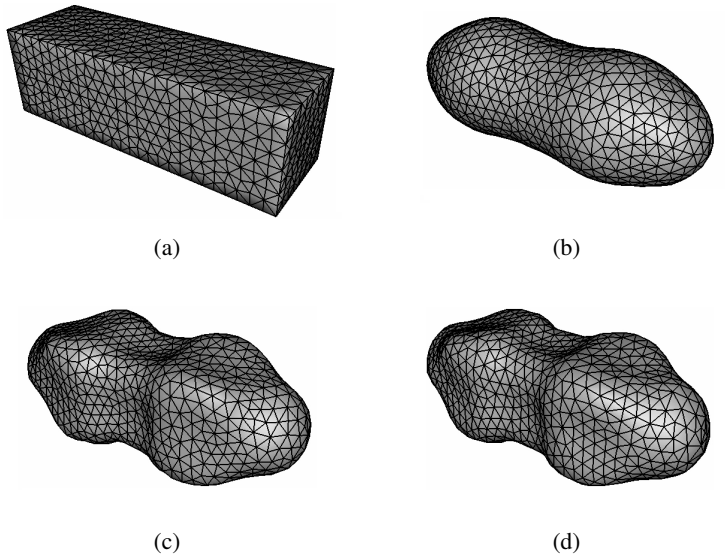


Figure 6.10: Rather than interpolating velocity over Voronoi regions directly, we tetrahedralize them and use simple barycentric interpolation. Left: A 2D Voronoi cell with standard dual Delaunay mesh overlaid. Centre: The same cell subdivided into smaller triangles that include the Voronoi vertices. Right: In 3D, each Voronoi face is triangulated using its centroid, and joined to its Voronoi site to build a tetrahedralization.

one site, one face centroid, and two Voronoi vertices).

One potential issue, not unique to our method, is that despite enforcing a lower bound on the distance between pressure samples, our unstructured sampling can cause sliver tetrahedra in the unmodified Delaunay tetrahedralization. While we found this posed little problem for the pressure projection, it can cause the least squares velocity reconstructions to be ill-conditioned due to nearly co-planar face normals. This can be readily resolved by requesting that the mesh generator add Steiner points to enforce fairly lax quality bounds; because our embedded pressure projection does not require the mesh generator to match boundaries, this is relatively inexpensive and effective. If mesh quality cannot be improved sufficiently, using additional nearby velocity samples in the reconstruction can ameliorate this at the cost of a smoother result.



(a) (b)

(c) (d)

Figure 6.11: a) Initial conditions for the collapse of a liquid block due to surface tension in zero gravity. (b) Naïve barycentric interpolation on tetrahedra generates very little detail. (c) Generalized barycentric interpolation over Voronoi cells retains interesting small scale structure. (d) Applying barycentric interpolation over our refined tetrahedra produces qualitatively consistent results.

6.7 Results

6.7.1 Sampling

The issues that arise from regular, non-geometry-aware pressure sampling are common and consistent across Cartesian grids, octrees, Voronoi meshes, and tetrahedral meshes. We will therefore use Voronoi meshes throughout, and simply compare our geometry-aware sampling against naïve regular sampling.

Surface Noise: As discussed above, regularly-spaced pressure samples can miss fine surface details, resulting in surface noise which is never physically smoothed out. Figure 6.12 illustrates that our sampling approach successfully resolves and corrects such small surface details. In contrast, regular samples cannot fully capture the initial surface perturbation, so it cannot be rectified. Though the ghost fluid method on regular samples does detect some differences in surface height, this actually exacerbates the problem because noisy sub-mesh details will appear to the simulator as rapid discontinuous changes in surface position over time, inducing noisy responses in the fluid velocity.

Topology Mismatch: Another visible artifact of using mismatched surface and simulation resolutions is topological inconsistencies. For example, a surface with two disjoint volumes of liquid may appear to the solver as one volume, resulting in a premature response. Figure 6.9 shows a liquid drop impacting a still surface. With regular sampling, the droplet begins to influence the static liquid before the surfaces are actually joined. Because our adaptively-placed samples match the topology of the surface tracker, they easily correct this spurious motion. Figure 6.1 also features such a topological merge, along with many splitting and tearing operations, with timings listed in table 6.1.

Thin Features: To illustrate our method's ability to animate thin features, figure 6.13 shows a scene in which we drop a small sphere of liquid onto the ground.

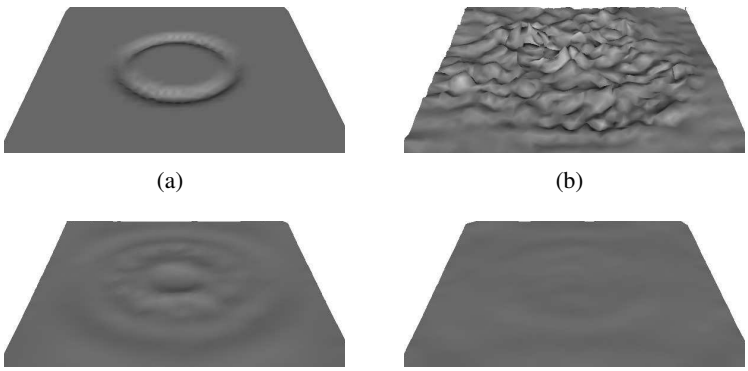


Figure 6.12: **Surface noise.** (a) A perturbation is introduced into a smooth surface. (b) On a regular tetrahedral mesh, the sub-mesh-resolution noise causes instability. (c, d) With adaptively-placed samples, the surface noise is accurately captured by the fluid solver and initially causes ripples before steadily settling.

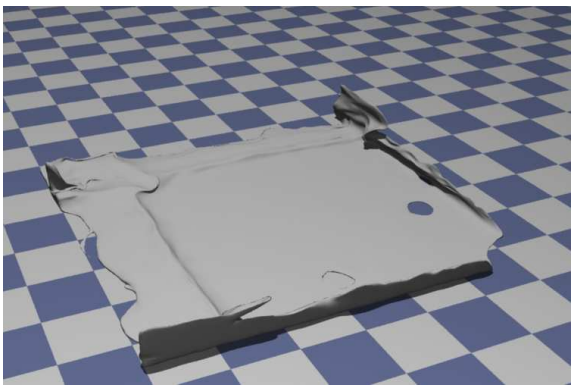


Figure 6.13: **Thin Sheet.** Seeding pressure samples directly inside the fluid volume allows us to capture almost arbitrarily thin sheets.

Thin sheets rapidly develop as the fluid spreads out across the floor. With regular pressure samples, sheets of this kind often end up between samples, effectively disappearing from the solver. Our sampling ensures that almost arbitrarily thin sheets of liquid remain visible to the solver, and as such, interesting rippling and splashing motion still occurs.

Our method also resolves thin sheets and small surface details generated by large splashes, as shown in figure 6.1. To counteract gradual volume drift, we do add a corrective motion-in-the-normal-direction [Bro06, MÖ9], which further aids in preserving thin sheets. Our video also includes an example of a column of liquid being released into a still pool. Although we are using only first-order semi-Lagrangian advection, the liquid motion remains lively and active throughout. We suspect that because our method retains sharp wave peaks and splashes rather than continually eroding them, their extra kinetic and gravitational potential energy is retained in the simulation, accounting for this reduced dissipation.

Table 6.1 gives timings for our 3D examples. All figures are averages per frame and all timings are in seconds. These simulations used no more than 320K tetrahedra each, whereas recent tetrahedra-based free surface methods used up to 4 times more tetrahedra to achieve a similar level of detail.

6.7.2 Surface Tension

Figure 6.4 illustrates the action of our surface tension model on a low resolution cube in zero gravity. Rather than quickly collapsing into a sphere, a cascade of detailed capillary waves propagate along the surface, causing it to oscillate rapidly. It initially inverts almost completely into an octahedron (the geometric dual of a cube), and continues to oscillate for many subsequent frames. To illustrate the benefits of our sampling approach in the context of surface tension, we launch an

Statistic	Thin sheet	Liquid column	Sphere Splash
# tetrahedra	141,701	197,911	313,587
Velocity reconstruction (s)	3	8	18
Surface tracking (s)	7	37	26
Remeshing (s)	15	39	69
Velocity advection (s)	7	18	15
Redistancing (s)	5	22	42
Pressure solve (s)	0.29	1.8	0.45
Total simulation time (s)	37	127	171

Table 6.1: Simulation statistics for 3D examples (all statistics are per-frame values, averaged over all frames).

identical simulation using the same time steps on a regular mesh. Because this mesh cannot respond and correct high frequency sub-mesh details present in the curvature estimates, the simulation becomes unstable almost immediately. Applying an excessively strict timestep restriction only brings the simulation to a halt as the surface noise introduces increasingly sharp features.

Inspired by an example from the work of Wojtan & Turk [WT08], we run another zero gravity simulation on a rectangular block (see figure 6.11). Because our simulation does not use diffusive Laplacian mesh smoothing and applies accurate mesh-based surface tension forces discontinuously at the interface, we retain substantially greater detail in the resulting capillary wave motion.

6.7.3 Interpolation

We revisit our surface tension block example to compare different interpolation schemes. As seen in figure 6.11, our barycentric method is substantially less damped than the naïve barycentric interpolation approach, and matches the more complex generalized barycentric interpolant.

6.8 Discussion and Limitations

Our implementation is not heavily optimized, and we defer various potential performance gains to future work. Obvious optimizations include: reducing the number of tetrahedra through smarter sampling, improving the broad phase algorithm for point-location queries, and streamlining the construction of mesh data structures. More fundamentally, our Voronoi simulator is in many ways dual to a tetrahedral scheme, and for a given mesh the number of velocity samples is identical; we believe that approximately comparable costs are therefore reasonable to expect.

The main contribution of this paper is the coupling of simulation elements to an existing explicit surface tracking method, and not the explicit surface tracking itself. Therefore, not all artifacts due to surface tracking are addressed. For example, *El Topo* delays handling some very difficult collisions for a few timesteps until the topological operations can be safely processed, which occasionally yields visible lingering surface noise. (Reducing the time step size can help by introducing fewer and simpler collisions, and more aggressive simplification can also be enabled by tuning the volume change tolerance that *El Topo* uses to decide whether to accept a given simplification.) Likewise, despite the use of feature-preserving mesh improvement, some popping artifacts due to on-the-fly remeshing are still visible in our animations. We chose *El Topo* because its resolution is not constrained to a regular grid and it is therefore able to showcase very thin features; nevertheless our method could adapt to any of the front tracking methods mentioned in section 6.2.2.

Surface tension was only used for examples in subsections 6.7.2 and 6.7.3. Our goal in many of the other examples was to highlight the ability to track thin sheets, whereas surface tension would break these sheets into droplets. Moreover, explicit surface tension schemes, such as the ghost-fluid-based method used in this

paper, suffer from a stringent $O(\Delta x^{\frac{3}{2}})$ time step restriction for stability, which is particularly costly when small scale capillary waves are not erroneously damped out. Pursuing a more efficient, fully implicit surface tension model is a promising future direction.

6.9 Conclusions and Future Work

We have shown that with careful placement of pressure samples, our Voronoi mesh-based fluid solver makes it possible for explicit surface tracking to achieve its full potential in capturing small scale liquid features. In addition, we adapted embedded boundary pressure projection techniques to Voronoi meshes, introduced a simple improvement to barycentric velocity interpolation for Voronoi/Delaunay meshes, and extended the ghost fluid surface tension model with mesh-based curvature in order to capture complex capillary waves with minimal damping.

Several directions for future work remain. For example, it may be possible to enhance our sampling scheme in various ways, perhaps by exploiting curvature adaptivity, topological information, or measures of vorticity and velocity variation. Likewise, improvements to front tracking would be welcome, such as curvature-driven adaptivity, or greater robustness and efficiency. Lastly, many common extensions to basic inviscid liquid simulation rely on regular grids, and would need to be adapted to accomodate our approach.

6.10 Acknowledgements

This work was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada. We would like to thank our anonymous reviewers for their helpful comments and suggestions.

Bibliography

[BB09] Tyson Brochu and Robert Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput.*, 31(4):2472–2493, 2009.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):100, 2007.

[BKZ92] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comp. Phys.*, 100(2):335–354, 1992.

[BOGS06] Adam W. Bargteil, James F. O'Brien, Tolga G. Goktekin, and John A. Strain. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1):19–38, January 2006.

[Bri08] Robert Bridson. *Fluid Simulation for Computer Graphics*. 2008.

[Bro06] Tyson Brochu. Fluid animation with explicit surface meshes and boundary-only dynamics. Master's thesis, University of British Columbia, 2006.

[BXH10] Christopher Batty, Stefan Xenos, and Ben Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics)*, 29(2):695–704, 2010.

[CFL+07] Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan Richard Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Symposium on Computer Animation*, pages 219–228, 2007.

[CGFO06] Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Symposium on Computer Animation*, pages 83–89, 2006.

[CK10] Marcel Campen and Leif Kobbelt. Exact and robust (self-)intersections for polygonal meshes. *Computer Graphics Forum (Eurographics)*, 29(2):397–406, June 2010.

[DFG+06] Jian Du, Brian Fix, James Glimm, Xicheng Jia, Xiaolin Li, Yuanhua Li, and Lingling Wu. A simple package for front tracking. *J. Comp. Phys.*, 213(2):613–628, 2006.

[EFFM02] Doug Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.*, 183(1):83–116, 2002.

[EMF02] Doug Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):736–744, 2002.

[ENGF03] Doug Enright, Duc Nguyen, Frédéric Gibou, and Ron Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, 2003.

[ETK+07] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH*, pages 23–30. ACM Press, 2001.

[FL10] Jeffrey D. Franklin and Joon Sang Lee. A high quality interpolation method for colocated polyhedral/polygonal control volume methods. *Computers & Fluids*, 39(6):1012–1021, June 2010.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):904–909, July 2005.

[FOKG05] Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. Fluids in deforming meshes. In *Symposium on Computer Animation*, pages 255–259, 2005.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):463–468, August 2004.

[GGL+98] James Glimm, John W. Grove, Xiaolin Li, Keh-ming Shyue, Yanni Zeng, and Qiang Zhang. Three-dimensional front tracking. *SIAM J. Sci. Comput.*, 19(3):703–727, 1998.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*, page 209, 1997.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw.

Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):973–981, 2005.

[HK03] Jeong-Mo Hong and Chang-Hun Kim. Animation of bubbles in liquid. *Computer Graphics Forum*, 22(3):253–262, 2003.

[HK05] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):915–920, July 2005.

[HSKF07] Jeong-Mo Hong, Tamar Shinar, Myungjoo Kang, and Ronald Fedkiw. On boundary condition capturing for multiphase interfaces. *SIAM J. Sci. Comput.*, 31(1-2):99–125, 2007.

[Jia07] Xiangmin Jiao. Face offsetting: A unified approach for explicit moving interfaces. *J. Comp. Phys.*, 220(2):612–625, 2007.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):820–825, 2006.

[KSK09] Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. Stretching and wiggling liquids. *ACM Trans. Graph. (SIGGRAPH Asia)*, 28(5):120, 2009.

[LSSF06] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):812–819, 2006.

[MÖ9] Matthias Müller. Fast and robust tracking of fluid surfaces. In *Symposium on Computer Animation*, pages 237–245, New York, NY, USA, 2009. ACM.

[MBLD02] Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *J. Graph. Tools*, 7(1):13–22, 2002.

[MDSB02] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, 2002.

[MMTD07] Patrick Mullen, Alexander McKenzie, Yiying Tong, and Mathieu Desbrun. A variational approach to Eulerian geometry processing. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):66, 2007.

[MUM+06] Viorel Mihalef, B. Unlusu, Dimitris Metaxas, Mark Sussman, and M. Y. Hussaini. Physics based boiling simulation. In *Symposium on Computer Animation*, pages 317–324, 2006.

[NKJF09] Matthieu Nesme, Paul G. Kry, Lenka Jevrábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):52, 2009.

[PN03] Blair Perot and Ramesh Nallapati. A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows. *J. Comp. Phys.*, 184(1):192–214, 2003.

[SBH09] Funshing Sin, Adam W. Bargteil, and Jessica K. Hodgins. A point-based method for animating incompressible flow. In *Symposium on Computer Animation*, pages 247–255, 2009.

[Shi07] Seungwon Shin. Computation of the curvature field in numerical simulation of multiphase flow. *J. Comp. Phys.*, 222(2):872–878, 2007.

[Si06] Hang Si. *TetGen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator*, 2006.

[TSB+05] Joseph Teran, Eftychios Sifakis, Silvia S. Blemker, Victor Ng-Thow-Hing, Cynthia Lau, and Ronald Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE TVCG*, 11(3):317–328, 2005.

[WBOL07] Jeremy D. Wendt, William Baxter, Ipek Oguz, and Ming C. Lin. Finite volume flow simulations on arbitrary domains. *Graphical Models*, 69(1):19–32, 2007.

[WT08] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):47, 2008.

[WTGT09] Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):76, 2009.

[ZYP06] Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. Simulation of bubbles. In *Symposium on Computer Animation*, pages 325–333, 2006.

Chapter 7

Conclusions

In the relatively short time that fluid animation has been studied in computer graphics, it has become an important element in the tool box of actual visual effects artists. In fact, in 2008 the Academy of Motion Picture Arts and Sciences presented technical Oscars to a number of pioneers in the development and use of fluid animation techniques for film. Academic research in this burgeoning area is likely to see continued prominence, particularly as hardware improves to the point that computer games and interactive applications like virtual surgery become viable. This thesis contributes meaningfully to the body of fluid simulation literature by introducing novel embedded boundary approaches for common fluid problems. Their effectiveness is validated both on numerical convergence tests and on a range of practical, graphics-oriented examples. Already, other researchers have adopted and extended some of these methods, both in graphics and computational physics, while developers at various visual effects studios have put them to use in film.

7.1 Recent and Concurrent Work

Since the publication of some of the thesis chapters, there has been relevant follow-up work by other researchers. Robinson-Mosher et al. have studied the solid-fluid coupling problem with a particular focus on handling deformable objects, including thin materials such as cloth and shells. First, they demonstrated an alternative coupling mechanism that handles these interactions through mass-lumping

[RMSG+08]. At the same time, they showed that the dense blocks that arise in our rigid-body coupling technique can be eliminated by retaining the solid velocity as a variable, and using a symmetric indefinite formulation similar to those we discuss in our method for Stokes flow. (This symmetric indefinite formulation for avoiding the dense blocks is also discussed by Bridson [Bri08].) While the accuracy of this mass-lumping approach is unclear, it is clear that it erroneously gives up tangential free-slip along solid boundaries, since the fluid and solid in each lumped cell are forced to have the same velocity, even for inviscid flows.

Robinson-Mosher et al. subsequently addressed this shortcoming by replacing mass-lumping with a more general constraint-based approach [RMEF09] that does not (necessarily) damp the tangential flow. Because the existing pressure samples already act as Lagrange multipliers, the new constraints simply introduce additional pressure-like variables, and the resulting system remains symmetric positive-definite (or indefinite in the presence of deformable objects). The downside of this more explicit constraint method, as compared to the use of the natural boundary conditions advocated in this thesis, are that it requires estimating potentially noisy solid normals and interpolating velocities; there is also evidence that it cannot handle perfect free slip, such as in the case of a thin disc translating tangentially through static smoke. More complex boundary constraints such as viscous free surfaces also seem difficult to handle effectively with this approach. However, the use of explicit constraints provides greater flexibility in that the range of constraints is not limited to those that can be expressed conveniently as natural boundaries.

A recent further extension of this fluid coupling approach for deformables is a change of variables in the discretization of solid damping terms, to solve for stress rather than velocity [RMSF10]. This results in a symmetric positive-definite system for fluid pressure and solid damping stresses, rather than a symmetric indefinite system for fluid pressure and solid velocity. This is loosely analogous

to the transformation we used to derive a symmetric positive-definite formulation for the Stokes problem in terms of pressures and stresses, rather than the usual pressure-velocity formulation; Robinson-Mosher et al. even suggested handling viscous forces with an SPD system as potential future work, and this is precisely what we have achieved in chapter 4. Moreover, Robinson-Mosher et al. achieve a positive-definite formulation primarily through a series of algebraic manipulations; our approach to Stokes flow suggests that there is almost certainly a variational interpretation that would guarantee positive-definiteness with somewhat less effort.

In terms of handling thin liquid sheets, Wojtan et al. [WTGT10] recently presented a new approach to handling topology changes in explicit surface tracking that retains thin sheets almost indefinitely, but performs mesh simplification whenever the surface topology is too detailed to be captured by a lower resolution simulation grid. Although the actual topological mesh operations are orthogonal to our work on Voronoi meshes, we share the common goal of addressing the issues that arise due to mismatched simulator resolution. However, the two approaches are dual in the sense that Wojtan et al. simplify the surface geometry to match the simulation grid, whereas we modify the simulation mesh to capture detailed geometric features. In effect they trade some over-smoothing and temporal artifacts in exchange for much greater speed, where we employ a more heavyweight approach to achieve better quality and reduce the need for non-physical simplifications.

Concurrent work on surface tension by Thuerey et al. [TWGT10] presents a more stable approximation of mesh-based surface tension that extends the method of Sussman et al. [SO09] to work with mesh-based surfaces. For each timestep, this approach essentially computes a solution to volume-conserving motion by mean curvature, and determines surface tension forces based on the distance to the nearest point on this evolved surface. Sussman et al. provide a theoretical justification for this method, and show that the resulting time step restriction is pro-

portional to Δx rather than $\Delta x^{\frac{3}{2}}$, making it much more stable than typical explicit schemes. However, this scheme on its own apparently has difficulties handling features close to the grid scale. An interesting experiment would be to apply surface tension forces computed in this way, but using our geometry-aware mesh to perform the simulation. Thuerey et al. instead augment their scheme with a secondary mesh-based wave equation solver to handle the small scale capillary waves that the grid cannot stably or correctly handle. In contrast, with our approach all features are captured by the Eulerian simulator, and thus this mechanism is unnecessary. Nonetheless, we could potentially subdivide our surface mesh and exploit this approach to add even smaller details if desired.

Considering the novel inequality constraint boundary condition that allows liquid to separate from walls, presented in Chapter 2, there has been little progress in this direction given the difficulty of efficiently solving the resulting large sparse LCP. Nonetheless, in the area of crowd simulation, a related approach was introduced by Narain et al. to handle what might be termed "one-sided" incompressibility in the sense that incompressibility is only enforced in a region if density is above a given threshold [NGCL09]. This allows crowds to thin out arbitrarily, but prevents them from compressing unrealistically beyond some reasonable limit. Crowd simulation scenarios are however inherently two dimensional, making it a slightly less difficult problem.

7.2 Discussion and Future Directions

7.2.1 Variational Principles for Irregular Domains

One interesting issue in these embedded methods for pressure projection is the choice between volume weights and area (finite volume) weights to handle irregu-

lar solid boundaries. Volume weights provide the ability to plausibly handle very small objects that may not even cross cell boundaries, because true volume fractions will still capture their presence. This opens the possibility of simulating subgrid phenomena such as sediment, dust, or hair. Moreover, volume weights fit more naturally with the variational interpretation, and provide a clear extension to viscosity. In fact, my attempts to derive improved free surface viscosity models using ghost fluid or finite volume ideas yielded non-symmetric systems that nonetheless failed to achieve better convergence. On the other hand, face area weights have been shown to provide better convergence on analytical tests, with much less noise occurring along domain boundaries [NMG09]. The same applies to the free surface: linear ghost fluid weights yield second order pressure convergence, but miss small droplets that are captured by volume weights.

One possible approach to seeking second and higher order spatial convergence is estimating the required integral forms more accurately, though this would come at the cost of denser, more complex stencils similar to finite element methods. For example, Bedrossian et al. used a variational formulation for the Poisson equation that yields the usual 5-point stencil in the interior and denser stencils along boundaries to achieve second order convergence [BvBZ+10]. It would also be worth investigating whether higher order time integration could be handled with our method by modifying the variational formulations to yield higher order backwards differentiation formulas (eg. BDF2).

We do not yet have a convergence theory for the variational finite difference methods we have presented. Researchers have considered convergence proofs and error estimates for the classic ghost fluid method and for the finite volume-like solid-boundary method of Ng et al. [LS03, JM05, NMG09]; studying connections with these methods, or traditional finite element theory, might provide useful insights in this direction.

Beyond fluid animation, Poisson and diffusion problems are ubiquitous throughout many other areas of computer graphics. Examples include image processing, rendering, shape deformation, and physical simulation [PGB03, HMBR05, JMD+07, KMB+09]. Since relatively few application scenarios actually include strictly voxelized boundaries or geometry, it is my hope that the embedded boundary approaches I have presented will spur applications to these other areas as well.

7.2.2 Solid-Fluid Coupling

As discussed earlier, the work of Robinson-Mosher et al. has considered interaction with deformable objects and thin shells, whereas the coupling model in Chapter 2 considered only rigid bodies. The methods presented in this thesis should be easily extensible to the deformable object case in a similar way, by adding appropriate energy terms for solid damping. Because damping, like viscosity, can be expressed as a minimization problem in terms of velocity and stress, this coupling should be relatively straightforward to derive. Incorporating thin shells poses additional difficulties. First, one would need to provide a mechanism for duplicating pressures and velocity samples, since these elements would be reused in cells cut by a thin shell, once for each side of the surface. This is simply a matter of careful bookkeeping, and was successfully carried out by Robinson-Mosher et al. [RMEF09]. Secondly, cells containing shell edges or endpoints pose potential issues, due to non-manifold simulation mesh connectivity. A given velocity face would need to be duplicated, yet both would be connected to only a single pressure cell; how can a meaningful volume weight then be assigned to the two duplicate faces? Finite volume weights would more easily handle this scenario, and this suggests that dividing the actual face volume between the two duplicates is the right approach, though how to split the volume remains unclear. In general this yields a pressure

projection on a more general graph structure as opposed to a simple grid; Day et al. considered this problem in two dimensions, though in the context of a non-symmetric cut cell method for full second order velocity convergence [DCL+98]. In effect, this method would need to find the intersection of the simulation mesh with the geometry, rather than use the simpler ray-based neighbour visibility approach taken by Robinson-Mosher et al. The advantages would be the ability to correctly handle multiple thin shells cutting through a single cell, as might be the case in cloth folding over on itself, and the proper elimination of errors in tangential motion of shells passing through still fluid. (A similar graph-based structure has also been suggested by Wojtan et al. [WTGT10] for simulating multiple disconnected free surface components within a single geometric cell, building on the use of non-manifold mesh structures in elasticity [TSB+05, NKJF09]. However, our unstructured Voronoi mesh simulator avoids the need for this extension.)

Another difficulty in rigid body coupling is that the fluid is tightly coupled only to basic rigid body dynamics, and not to forces deriving from collision or contact. This often means that one must choose between enforcing fluid incompressibility and preventing penetration between rigid bodies and walls. Guendelman et al. and Robinson-Mosher et al. addressed this through careful ordering of the time integration steps [GSLF05, RMSG+08]. They first perform a tentative coupled solve to determine fluid forces on the solid, then advance just the solids in time while handling all collision and contact, and then finally do a secondary fluid-only solve from the start of the timestep, with the computed solid motion held fixed. This ensures both incompressibility and non-penetration; however, it also entails a weaker, less accurate coupling between contact forces and fluid.

Another simple extension of our rigid body coupling would be to incorporate it into our full Stokes solver, rather than coupling solely to pressure projection. Because sedimentation of granular particles is an important application area for

Stokes flow, this might be particularly useful for studies in this area, particularly since our scheme includes a degree of support for sub-grid coupling. This should be a simple matter of adding energy terms to account for the traction forces between fluid and solid.

7.2.3 Advection

Apart from minor modifications to velocity reconstruction and interpolation for embedded boundaries on unstructured meshes, this thesis has not addressed the advection stage of the Navier-Stokes equations. Nonetheless, our embedded boundary methods would benefit from improved advection in a number of ways. In the Cartesian grid case, bi-/tri-linear interpolation is guaranteed to fail in exactly respecting the embedded boundaries, because this simple interpolation has no knowledge of the sub-grid boundary position. In many situations smoke or liquid penetrating boundaries causes visually disturbing artifacts, suggesting that this improved interpolation is a worthwhile avenue to pursue. A potential solution would be to actually construct a polygonal/polyhedral representation of all truncated cells, reconstruct boundary-respecting velocities on the vertices, and employ a (generalized) barycentric interpolation within each cell. We have experimented with this approach in 2D, and it seems quite promising, though it is certainly more expensive than basic interpolation. This idea should also be applicable to Voronoi meshes with minimal modification. A related approach has already been presented by Rosatti et al. [RCB05] in two dimensions, though making use of radial basis functions rather than mesh-based interpolation.

Another general problem with advection is reducing the dissipation introduced by averaging in the standard first-order semi-Lagrangian approaches — our unstructured mesh scheme is no better than

Stam's original "Stable Fluids" method in this respect [Sta99]. There have been substantial improvements for the Cartesian case, such as the FLIP method [ZB05], semi-Lagrangian MacCormack/BFECC methods [SFK+08], higher order monotonic interpolation [FSJ01], and semi-Lagrangian WENO schemes [CFR05]. However, for unstructured meshes there has been minimal progress. There are a few WENO schemes for tetrahedral meshes in the computational physics literature (eg. [DK07]), but these are quite complex to implement. In theory, the semi-Lagrangian MacCormack methods should be directly extensible to unstructured meshes, since they use only simple semi-Lagrangian building blocks. However, re-meshing poses additional difficulties that make this a non-trivial extension. The most relevant work in graphics remains that of Sin et al. which presents a FLIP-like method on Voronoi meshes [SBH09]. This approach features exactly one particle per Voronoi cell, and would therefore seem an ideal approach to make FLIP adaptive and eliminate its problems with sub-grid noise, yet the results instead appear somewhat damped. It is my conjecture that the use of smooth MLS kernels resulted in overly smooth velocities, destroying the benefits of the FLIP scheme. Replacing MLS interpolation with our mesh-based interpolation might provide improved results since it uses a much smaller stencil of sample points. On a related note, FLIP schemes have typically used the fluid particles to represent surface geometry in addition to tracking velocities, which often produces noisy surfaces during rendering. Combining the FLIP scheme with mesh-based surface tracking is another direction that should therefore be explored.

A final approach to reducing dissipation in unstructured meshes is the use of fully-coupled Eulerian advection, as suggested by Mullen et al. [MCP+09]. This has been shown to preserve kinetic energy, ensure reversibility, and better retain vorticity, in part because it eliminates the time splitting errors that arise in standard pressure projection-based methods. However, the non-linearity of advection

implies the need for a fully non-linear solver with the associated numerical machinery and computational overhead. To date it is also unclear how to combine such conservative approaches with dynamic meshes. Nonetheless, this does point to an overall need to reduce time splitting as a source of error in computer graphics fluid simulations, as also suggested by Schechter & Bridson [SB08].

7.2.4 Unstructured Meshes

In light of the benefits provided by our Voronoi-based simulation scheme, and our earlier discussion of viscosity and Stokes flow, a natural extension would be to consider whether unstructured meshes can also support our variational viscosity formulation. So far this appears to be a more difficult question, because computing velocity gradients on such meshes is less natural than for Cartesian grids; this has been noted as a problem in the CFD context as well [Mav07]. A solution to this might be pursued in two different ways. First, full velocity vectors could be reconstructed either as part of the viscosity solver itself, or as a pre-process, which would make it simpler to construct full velocity gradients and stress tensors. These could then be used more directly in discretizing our variational formulation. As a second possibility, research in the field of discrete differential geometry and geometric mechanics has made progress in extending classical grid-based methods to unstructured meshes (eg. [MCP+09]). This may yet lead to more natural ways to understand and discretize viscous fluid stresses that avoid having to reconstruct full velocity vectors at all.

Another potential advantage of Voronoi meshes alluded to in Chapter 6 is their benefits for lattice-based adaptive approaches. Current octree pressure projection methods either lose accuracy due to lack of orthogonal pressure gradients at T-junction faces (where a large cell face meets multiple small cell faces), or they

require substantially greater care to ensure consistency [LGF04, LFO05, BXH10]. It's also not clear how to apply embedded boundary methods in the presence of T-junctions. By constructing pressure samples on an octree-type lattice for convenience, but using the discretization dictated by the samples' Voronoi regions, we can avoid the presence of T-junctions altogether, while simultaneously ensuring that linear pressure gradients can be represented properly. Furthermore, on regions that have uniform resolution the resulting Voronoi cells form an exact uniform grid, so that one can locally revert to simpler and faster algorithms. As with the tetrahedral method in Chapter 5, such an octree/Voronoi approach method would allow embedded boundaries and surfaces to lie anywhere in the domain, rather than strictly in regions of uniform resolution.

It would also be beneficial to consider improved mesh generation for the specific problem of geometry-aware sampling of liquid surfaces. As noted in Chapter 6, problematic sliver tetrahedra will occasionally be generated by our method, unless the mesh generator is directed to add Steiner points to prevent them. However, I suspect that our geometry-aware simulator is not highly sensitive to the precise positioning of sample points, so it may be possible to avoid Steiner points through small local perturbations to the sample positions, perhaps along the surface normal direction. It may also be useful to attempt to encourage our Voronoi mesh to be closer to a centroidal Voronoi tessellation (CVT), since these special Voronoi diagrams are known to minimize truncation error for certain finite difference schemes [DFG99]. In fact, although our embedded methods on general tetrahedral and Voronoi meshes may be only first order, there is reason to believe that they would achieve second order accuracy in pressure on meshes with this improved regularity, as they do on Cartesian grids. CVTs and the closely related optimal Delaunay triangulations (ODTs) have already been used for mesh generation where the triangle faces are constrained [ACSYD05], and it would likely also benefit our scenario in

which certain Voronoi sites are (loosely) constrained.

7.3 Summary

The goal of my work has been to improve simulation of several fundamental aspects of fluid behaviour, including interaction with irregular solid boundaries, dynamic rigid-body coupling, viscous flows with free surfaces, surface tension, and detailed thin splashes. A key challenge in all of these cases lies in determining the proper boundary treatment, and for this purpose I proposed the use of embedded boundary conditions that can accurately treat non-conforming physical geometry. The first half of the thesis demonstrated a variational technique to derive such methods for viscous incompressible flows interacting with solids. Within this topic, I considered the solid boundary conditions for pressure projection, then free surface conditions for viscosity, and finally a unified variational finite difference method for Stokes flow. In many ways this can be considered the simplest extension of the classic MAC method to handle irregular domains and fully implicit, spatially varying viscosity. In the second half of the thesis, I revisited the pressure projection problem and applied embedded boundary methods to unstructured meshes. This combination is a powerful one, which I showed simplifies meshing and improves the accuracy of existing adaptive methods for Delaunay tetrahedral meshes. I subsequently applied this same idea to Voronoi meshes with pressure samples chosen in a geometry-aware fashion, in order to discretize and simulate arbitrarily thin liquid features. This approach also made possible a novel surface tension discretization that exploits mesh-based curvatures to simulate detailed capillary waves with minimal dissipation. Considered together, these embedded boundary methods yield substantial practical benefits while requiring fairly simple modifications to existing simulation techniques. Looking forward, the variational interpretation

I introduced provides a useful conceptual framework with which to tackle future fluid animation problems, while our geometry-aware Voronoi-based fluid simulator makes a strong case for continued research into unstructured mesh fluid dynamics for computer graphics applications.

Bibliography

[ACSYD05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):617–625, 2005.

[BvBZ+10] Jacob Bedrossian, James H. von Brecht, Siwei Zhu, Eftychios Sifakis, and Joseph Teran. A second order virtual node method for Poisson interface problems on irregular domains. *J. Comp. Phys.*, (in press), 2010.

[BXH10] Christopher Batty, Stefan Xenos, and Ben Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics)*, 29(2):695–704, 2010.

[CFR05] E. Carlini, R. Ferretti, and G. Russo. A weighted essentially nonoscillatory, large time-step scheme for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 27(3):1071–1091, January 2005.

[DCL+98] Marcus Day, Phillip Colella, Michael Lijewski, Charles Rendleman, and Daniel Marcus. Embedded boundary algorithms for solving the Poisson equation on Complex Domains, 1998.

[DFG99] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi tessellations : Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[DK07] M. Dumbser and M. Kaser. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. *J. Comp. Phys.*, 221(2):693–723, February 2007.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH*, pages 15–22, 2001.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):973–981, 2005.

[HMBR05] Tom Haber, Tom Mertens, Philippe Bekaert, and Frank Van Reeth. A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects. In *Graphics Interface*, pages 79–86, 2005.

[JM05] Z. Jomaa and C. Macaskill. The embedded finite difference method for the Poisson equation in a domain with an irregular boundary and Dirichlet boundary conditions. *J. Comp. Phys.*, 202(2):488–506, January 2005.

[JMD+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):71, 2007.

[KMB+09] Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):50, 2009.

[LFO05] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995–1010, 2005.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):457–462, August 2004.

[LS03] Xu-Dong Liu and Thomas C. Sideris. Convergence of the ghost fluid method for elliptic equations with interfaces. *Mathematics of Computation*, 72(244):1731 – 1746, 2003.

[Mav07] Dmitri Mavriplis. Unstructured mesh discretizations and solvers for computational aerodynamics. In *AIAA Computational Fluid Dynamics Conference*, 2007.

[MCP+09] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):38, 2009.

[NGCL09] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph. (SIGGRAPH Asia)*, 28(5):122, 2009.

[NKJF09] Matthieu Nesme, Paul G. Kry, Lenka Jevrábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):52, 2009.

[NMG09] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comp. Phys.*, 228(23):8807–8829, 2009.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph. (SIGGRAPH)*, 22(3):313–318, 2003.

[RCB05] G. Rosatti, D. Cesari, and L. Bonaventura. Semi-implicit, semi-Lagrangian modelling for environmental problems on staggered Cartesian grids with cut cells. *J. Comp. Phys.*, 204(1):353–377, March 2005.

[RMEF09] Avi Robinson-Mosher, R. Elliot English, and Ronald Fedkiw. Accurate tangential velocities for solid fluid coupling. In *Symposium on Computer Animation*, pages 227–236, 2009.

[RMSF10] Avi Robinson-Mosher, Craig Schroeder, and Ron Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *In submission*, 2010.

[RMSG+08] Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):46, 2008.

[SB08] Hagit Schechter and Robert Bridson. Evolving sub-grid turbulence for smoke animation. In *Symposium on Computer Animation*, pages 1–7, 2008.

[SBH09] Funshing Sin, Adam W. Bargteil, and Jessica K. Hodgins. A point-based method for animating incompressible flow. In *Symposium on Computer Animation*, pages 247–255, 2009.

[SFK+08] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable MacCormack method. *SIAM J. Sci.Comput.*, 35(2-3):350–371, 2008.

[SO09] Mark Sussman and Mitsuhiro Ohta. A stable and efficient method for treating surface tension in incompressible two-phase flow. *SIAM J. Sci. Comput.*, 31(4):2447–2471, January 2009.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999.

[TSB+05] Joseph Teran, Eftychios Sifakis, Silvia S. Blemker, Victor Ng-Thow-Hing, Cynthia Lau, and Ronald Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE TVCG*, 11(3):317–328, 2005.

[TWGT10] Nils Thuerey, Chris Wojtan, Markus Gross, and Greg Turk. A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph. (SIGGRAPH)*, 29(3), 2010.

[WTGT10] Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. Physically-inspired topology changes for thin fluid features. *ACM Trans. Graph. (SIGGRAPH)*, 29(3), 2010.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):965–972, July 2005.