Variational Stokes with Polynomial Reduced Fluid Model

by

Jonathan Panuelos

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

Standard fluid simulators often apply operator splitting to independently solve for pressure and viscous stresses. This decoupling, however, induces incorrect free surface boundary conditions. Such methods are unable to simulate various fluid phenomena reliant on the balance of pressure and viscous stresses, such as the liquid rope coil instability exhibited by honey. Unsteady Stokes solvers, when used as a sub-component of Navier-Stokes, retain coupling between pressure and viscosity, and are thus able to resolve these behaviours. The simultaneous application of stress and pressure terms, however, creates much larger, and thus more computationally expensive, systems than the standard decoupled approach.

To accelerate solving the unsteady Stokes problem, we propose a reduced fluid model wherein interior regions are represented with incompressible polynomial vector fields. Sets of standard grid cells are consolidated into super-cells, each of which are modelled using only 26 degrees of freedom. We demonstrate that the reduced field must necessarily be at least quadratic, with the affine model being unable to capture viscous forces. We reproduce the liquid rope coiling instability, as well as other simulated examples, to show that our reduced model provides qualitatively similar results to the full Stokes system for a smaller computational cost.

Acknowledgements

I want to express my gratitude to everyone whose support made this work possible.

To Christopher Batty, thank you for taking me on as your student and successfully guiding me through this degree. I will keep your insights into academia with me as I continue through the next steps of my journey.

To my committee members, Toshiya Hachisuka and Justin Wan, thank you for taking the time to review this work and ensure it is polished to completion.

To my colleagues in the Computational Motion Group: Henry, JC, Mengfei, Michael, Nathan, Ryan, Sina, Tumay, and Yu; many thanks for the hours of insightful discussions, both formal and informal, within the lab and elsewhere. I am also thankful for everyone's effort in keeping the lab lively, even in these extraordinary times.

Finally, I wish to pay special regards to my family. Without their love and support, I would not be the person I am today. To my parents, Alejandro and Rocina, thank you for raising me with a love of learning and providing me with the opportunities to pursue it. To my brother, Timothy, thank you for being a beacon of lightheartedness in the longest days and nights. For you three, I am forever grateful.

Dedication

To my brother, Timothy.

Table of Contents

List of Tables

List of Figures

Abbreviations

APIC affine particle-in-cell 12–14, 34, 52

BiCGSTAB biconjugate gradient stabilized 38, 46, 47, 49

FEM finite element method 22

FFT fast Fourier transform 3

FLIP fluid-implicit-particle 4, 13

FVM finite volume method 22

G2P grid-to-particle 4, 12, 13

MAC marker-and-cell 3

MPM material point method 4

P2G particle-to-grid 12, 13

PIC particle-in-cell 4, 12, 13

PolyPIC polynomial particle-in-cell 34

SDF signed distance field 9

SPD symmetric positive definite 39, 41, 51, 65

SPH smoothed particle hydrodynamics 13

Chapter 1

Introduction

Most computer graphics research has historically been focused on inviscid, incompressible liquids. The most abundant liquid we come in contact with daily—water—is of a relatively low viscosity. This, however, does not preclude the presence of interesting phenomena exhibited by more viscous materials. Indeed, dropping a thin stream of honey in just the right manner will form a rope-like coil [Ribe et al., 2012]. While not as ubiquitous as water, these viscous liquids are nonetheless still common in daily life with foods and oils, as well as industrial applications with greases and other lubricants, and academic studies such as the geology of the Earth's mantle. It is on the simulation of these highly viscous fluids that we focus our attention.

The physical behaviour of fluids is described by the Navier-Stokes equations which derive from conservation laws assumed to hold throughout the fluid. Operator splitting is standard practice in computer graphics for reducing the Navier-Stokes equations into smaller problems solved in a stepwise manner [Bridson, 2015]. We set aside advective and body force terms, which are solved on their own steps. This leaves just the coupled pressure and viscous shear stress terms, forming the unsteady Stokes equations which are the subject of our work. The coupling between these two terms has been found to be essential in simulating the aforementioned coiling behaviour [Larionov et al., 2017]. Typical methods once again apply operator splitting to independently solve for viscosity and pressure, which breaks the coupling required for accurate free surface treatment. Rather than reproducing the cylindrical rope coiling instability, decoupled methods result in random buckling of the falling liquid stream [Batty and Bridson, 2008]. Larionov et al. [2017] presented a variational method for solving the unsteady Stokes problem. While this unified solver is effective, it creates much larger systems than the decoupled methods, leading to higher computational costs.

Spatial adaptivity methods have been developed to focus computational cost in regions of interest by reducing resolution elsewhere in the fluid [Losasso et al., 2004]. In a similar vein, model reduction methods have been proposed to capture the most relevant characteristics of the flow in as small a dimensionality as possible [De Witt et al., 2012]. Integrating both concepts, Goldade et al. [2020] proposed a pressure projection method that divides the fluid into a uniform grid domain and a set of interior domains whose velocities are defined as incompressible affine vector fields. Their method accomplishes spatial adaptivity without using complicated stencils and data structures, and model reduction that retains compatibility with existing uniform grid methods, thus resulting in much simpler implementation.

Building on these ideas, we propose a unified Stokes solver featuring interior regions described using an incompressible *polynomial* vector field, primarily of degree 2. We show that an affine model is insufficient for the Stokes problem, and that the quadratic model is required for proper resolution of viscous forces while being small enough to provide increased computational efficiency. We reproduce the liquid rope coil instability, as well as other simulated examples, demonstrating that the quadratic model is able to resolve qualitatively similar results to a fully uniform method with reduced computational cost.

Chapter 2

Previous Work

We outline prior work in computer graphics that relates to, or approaches the same problems as, our work in improving the computational efficiency of the unified Stokes step.

Standard Fluid Solvers. Standard methods for solving incompressible flow in computer graphics use staggered pressure and velocity samples, originally developed for the marker-and-cell (MAC) method of Harlow and Welch [1965]. This was introduced into computer graphics by Foster and Metaxas [1996] who solved the pressure and velocity updates explicitly, and was limited to voxelized solid boundaries. In response to the timestep limitations imposed by explicit solvers, Stam [1999] developed a, now classical, unconditionally stable method using semi-Lagrangian advection and decoupled implicit solvers for pressure and viscosity. With regards to pressure, this work introduced to computer graphics the language of Helmholtz-Hodge decomposition and projection to refer to the operator splitting approach initially developed by Chorin [1967].

Viscous Liquids. Carlson et al. [2002] focused on solving highly viscous liquids, adopting an implicit Laplacian-based smoothing of velocity in place of Stam's fast Fourier transform (FFT)-based viscosity. Their treatment of free surface boundaries was found to produce incorrect translational motion, later corrected by Fält and Roble [2003]. Batty and Bridson [2008] adopted a variational approach for enforcing boundary conditions, and used a zero traction free surface boundary to correctly handle viscous shear stresses and recover rotational motion needed for buckling fluids. Larionov et al. [2017] extended the variational approach for solving unsteady Stokes flow, coupling the pressure and viscosity equations together. This method is able to recover cylindrical coiling behaviour in place of random

buckling of prior decoupled methods, and forms the base uniform grid solver that we build our reduced method on top of. Outside of Eulerian volume approaches, Bergou et al. [2010] were able to successfully recreate meandering and coiling patterns of viscous threads on a conveyor using a discrete rod-based model.

Particle-in-cell Methods. Intended to take advantage of desirable properties of both Eulerian and Lagrangian methods, particle-in-cell (PIC) and related hybrid methods have been developed. The first formulation of PIC was created by Harlow [1962], which suffered from excessive dissipation. Brackbill and Ruppel [1986] subsequently developed fluid-implicit-particle (FLIP) as a remedy. Both methods were introduced into computer graphics by Zhu and Bridson [2005], who suggest taking a weighted average of the two methods. The material point method (MPM) uses PIC/FLIP transfers along with constitutive models to solve elasto-plastic problems [Sulsky et al., 1995], being introduced into computer graphics by Stomakhin et al. [2013] for simulating snow. A locally affine variation of the PIC interpolation method was constructed by Jiang et al. [2015] in order to preserve angular momentum previously lost with each grid-to-particle (G2P) interpolation, and was later extended to be locally polynomial by Fu et al. [2017]. We use an APIC framework to perform the advection step of our implementation, though we note that the reduced Stokes solver we develop is agnostic to this choice.

Solid-Fluid Coupling. The coupling method between the reduced model and the uniform grid that we adopt from Goldade et al. [2020] is largely similar to the strong two-way rigid body-fluid coupling presented by Batty et al. [2007]. This work introduced an operator that converted boundary fluid pressures into generalized forces that act on the solid. The solid's intended behaviour determines the constraints that define the degrees of freedom of its generalized velocity. Goldade et al. [2020] replace the rigid field used for rigid bodies with an incompressible affine field intended to represent a fluid. They draw the same comparison as the rigid and affine variants of PIC developed by Jiang et al. [2015]. We extend this analogy, with our polynomial model being akin to the polynomial extension of Fu et al. [2017].

Spatial Adaptivity. With less of an emphasis on quantitative accuracy compared to other application domains in science and engineering, computer graphics often more aggressively trades accuracy for computational cost. Losasso et al. [2004] introduced the use of an octree data structure as a method for focusing computation on areas of interest by reducing grid resolution elsewhere, using finite-volume techniques to translate the uniform

grid-based stable fluids method into stencils suited to irregular octree grids. Tetrahedral meshes have also been explored, first with precomputed static meshes by Feldman et al. [2005], with dynamic meshes by Klingner et al. [2006], and later with embedded free surface and solid boundary conditions by Batty et al. [2010]. Chentanez et al. [2007] used tetrahedral meshes with an algebraic multigrid solver for pressure. Similarly, methods based on Voronoi diagrams rather than tetrahedral meshes have also been developed [de Goes et al., 2015], and later adapted to improve octree t-junctions [Aanjaneya et al., 2017]. Goldade et al. [2020] presented a reduced model pressure solver as a method for spatial adaptivity with uniform cells near the boundary and coarsened reduced model cells on the interior, which we directly adopt for our method. Approaching the same computational efficiency goals, Edwards and Bridson [2014] used variable polynomial degrees (p-adaptivity) on a discontinuous Galerkin method to focus computation near the fluid surface. Compared to our use of polynomials to allow coarsening on interior cells, they apply higher-order polynomials on the fluid surface to resolve finer detail out of a globally coarse grid.

Reduced Fluid Models. Rather than spatial adaptivity, some authors have proposed solving for a reduced set of variables to decrease computational cost. Treuille et al. [2006] used principal component analysis to find the reduced basis of target dimension that minimizes reconstruction error. In place of a global basis, Wicke et al. [2009] construct large simulation primitives called 'tiles' with local bases; coupling along faces is handled by shared boundary bases. De Witt et al. [2012] used Laplacian eigenfunctions as a divergence-free basis. This was later extended by Cui et al. [2018], who reduced memory cost with discrete sine and cosine transforms as well as introduce eigenfunctions for supporting Neumann boundary conditions. Ando et al. [2015] solved for pressure on a reduced-dimension grid and constructed an upsampler to compute corrected velocities that respect the free surface boundary condition. Da et al. [2016] presented a boundary-only method for high surface tension fluids, avoiding computation of internal volumetric degrees of freedom. The reduced solver of Goldade et al. [2020] uses an affine basis to reduce degrees of freedom allocated to interior regions of the fluid. We extend this to higher-order polynomial bases for simulating the decoupled viscosity step and the coupled Stokes problem.

Chapter 3

Background Theory

There is a wide assortment of fluids in nature; water, air, shampoo, pitch, and whipped cream all have different properties and behaviours. We focus on the animation of viscous, incompressible fluids. This does not include all possible fluids, but does represent a good majority of visually interesting ones from daily life.

3.1 Governing Equations

Our fluids of interest are governed by a set of partial differential equations known as the incompressible Navier-Stokes equations, typically given in the form,

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + \mathbf{g} \tag{3.1}$$

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

where $\mathbf{u}$ is the velocity, ρ is the density, p is the pressure, ν is the kinematic viscosity coefficient, and g are external (body) forces which typically contains gravity [Bridson, 2015].

The first equation is simply a conservation law for momentum, which is elucidated as Newton's second law, $\mathbf{f} = m\mathbf{a}$, with a simple rearrangement:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + \mathbf{g} \tag{3.3}$$

Note that the bracketed terms $\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}$ is simply the material derivative of velocity. That is, it is the rate of change of velocity within a coordinate system moving along with the velocity. Equation 3.2 enforces incompressibility as a divergence-free condition.

The above form of Navier-Stokes, however, does not enforce the correct free surface boundary condition, which requires a balance between pressure and viscous stresses (to be discussed in Section 3.5). To allow proper free surface behaviour, the following generalized form of the Navier-Stokes equations must be used,

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \cdot \tau + \mathbf{g} \tag{3.4}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.5}$$

$$\tau = \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\intercal} \right) \tag{3.6}$$

where τ is the symmetrized deviatoric stress tensor and $\mu = \nu \rho$ is the dynamic viscosity coefficient.

3.2 Time Discretization

To discretize the system in time, we can apply the usual Euler approximation for the time derivative,

$$\frac{\partial \mathbf{u}}{\partial t} \approx \frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{\Delta t} \tag{3.7}$$

where Δt is the timestep size and the subscripts indicate evaluation of a variable at that timestep. In other words, $\mathbf{u}_n$ is the value of the velocity at timestep n.

Crucially, it is difficult to solve the entire system in one step. The typical approach in computer graphics is to use operator splitting to solve the problem stepwise. We first integrate the velocity to an intermediate state, $\mathbf{u}^*$, subject to advection and body forces:

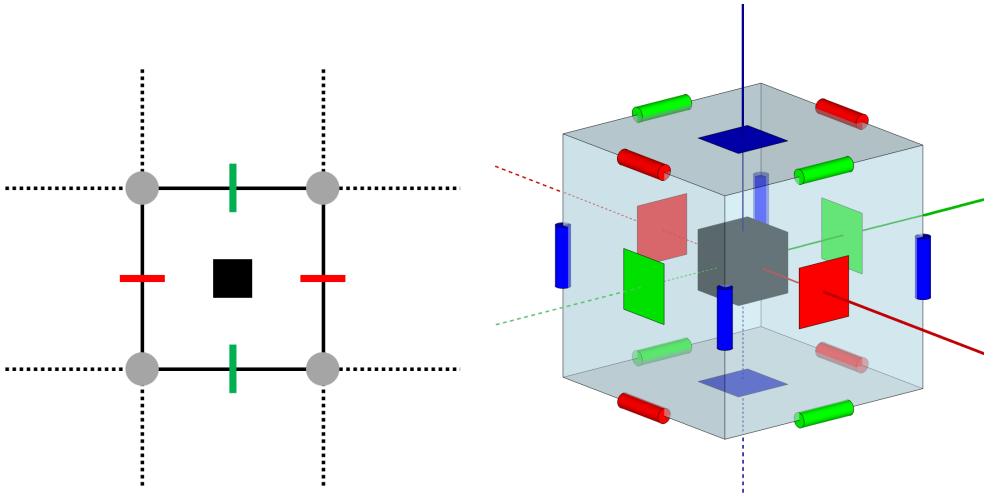
$$\frac{\mathbf{u}^* - \mathbf{u}_n}{\Delta t} = -\mathbf{u}_n \cdot \nabla \mathbf{u}_n + \mathbf{g} \tag{3.8}$$

This leaves the pressure and viscosity update for the second step:

$$\frac{\mathbf{u}_{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \left(\nabla p - \nabla \cdot \tau \right) \tag{3.9}$$

$$\nabla \cdot \mathbf{u}_{n+1} = 0 \tag{3.10}$$

$$\tau = \mu \left(\nabla \mathbf{u}_{n+1} + (\nabla \mathbf{u}_{n+1})^{\intercal} \right) \tag{3.11}$$



(a) (b)

Figure 3.1: Staggered grid sampling points on a standard (a) 2D and (b) 3D cell. (a) The filled black square indicates the cell-centered sampling point for pressure, p, and diagonal stresses, τ_{xx} and τ_{yy}; the grey disks indicate the node-centered off-diagonal stresses, τ_{xy}; the red and green lines indicate face-centered velocities, u and v respectively. (b) x, y, and z axes are shown in red, green, and blue lines respectively. The black cube indicates the cell-centered sampling point for pressure, p, and diagonal stresses, τ_{xx}, τ_{yy}, τ_{zz}; the red, green, and blue cylinders indicate the edge-centered off-diagonal stresses, τ_{yz}, τ_{xz}, τ_{xy}, respectively; the red, green, and blue squares indicate the face-centered velocity samples, u, v, w respectively [Larionov et al., 2017].

Our contribution is primarily concerned with solving the *unsteady Stokes flow* defined by Equations 3.9-3.11 as an implicit system in velocity [Larionov et al., 2017].

3.3 Spatial Discretization

3.3.1 Staggered Grids

For our spatial discretization, we use the staggered grid standard in use in computer graphics [Harlow and Welch, 1965]. Rather than variables being collocated, different variables are sampled on different points on the standard cell, shown on Figure 3.1.

These sampling points are chosen such that numerical derivative approximations re-

quired for a variable's update equation fall collocated with that variable's location. For example, p must be collocated with $\nabla \cdot \mathbf{u}$, which is satisfied by choosing pressures on cell centers and velocities on cell faces, as shown on Figure 3.1a. These faces form the numerical stencil for divergence—the geometric arrangement of variables required for the numerical evaluation of a given operator.

This method of choosing sampling points based on the construction of a friendly stencil largely derives from finite difference methods, where a centered difference is ideal as it has second-order accuracy compared to forward and backwards differences. Using centered differences with a collocated grid results in an update equation that ignores the value being updated, resulting in a non-trivial null-space. In particular, this means that high-frequency noise—with period smaller than two cells wide—is not automatically filtered out; a naïve centered difference with a collocated setup, lacking any extra filtering, is actually blind to such checkerboarding error [Bridson, 2015].

3.3.2 Level Set Geometry

In constructing our method, we require a numerical definition of various geometrical domains. We accomplish this using level sets, which define a surface implicitly as a signed distance field (SDF); that is, it does not define a mesh but rather uses a scalar function $\Phi(\mathbf{x})$ defined throughout the domain whose value dictates a point's distance to the surface, and whose sign dictates which side of the surface it lies on. Standard convention uses $\Phi(\mathbf{x}) < 0$ to indicate a region's interior and $\Phi(\mathbf{x}) > 0$ to indicate its exterior. Consequently, the boundary is implicitly defined as the zero isocontour, $\Phi(\mathbf{x}) = 0$ [Bridson, 2015]. In the discretized setting, level set values are typically stored in cell centers, with values being interpolated elsewhere when needed.

In the bulk interior of the fluid, the discretization stencils will simply be the regular grid stencils. Consequently, level sets are most important near the boundary of the fluid, being required for proper enforcement of the free surface and solid boundary conditions. There are a variety of ways to use the information contained in a level set for this purpose; the standard methods used for finite difference-based pressure projection are ghost fluid for the free surface (enforcing a Dirichlet boundary condition in pressure, $p = 0$), and cut cell for solid boundaries (enforcing a Neumann boundary condition in pressure, $\frac{\partial p}{\partial \mathbf{n}} = \nabla p \cdot \mathbf{n}$) [Bridson, 2015]. Both assume a linear interpolation of the level set values. The ghost fluid method uses a linear interpolation to find a grid point's distance to the surface, constructing a finite difference with a fictitious particle whose value is extrapolated based on the Dirichlet boundary condition [Gibou et al., 2002]. The cut cell method uses multilinear interpolation

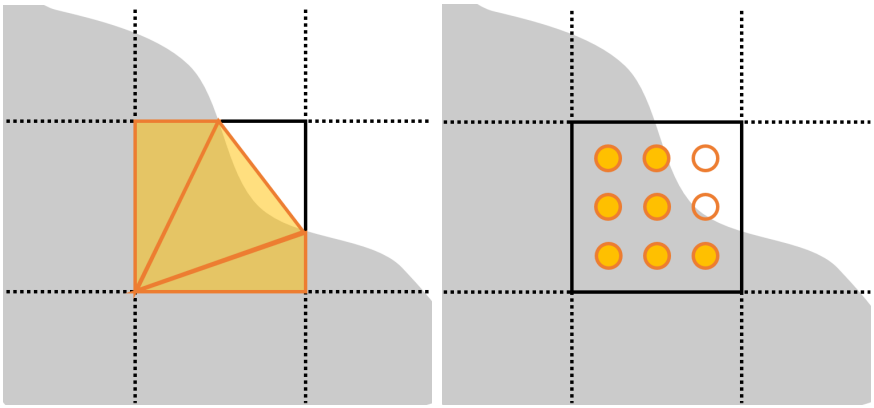


Figure 3.2: Two methods of computing volume fractions from level sets. The interior of an arbitrary level set is shown in light gray for both cases. (a) Computing volumes by triangulating a marching squares template. The interior of the computed volume is shown in light orange. (b) Computing volumes by sampling a level set with 3 points in each axis. Points within the level set are filled in orange and points outside are not filled in. The sampled points indicate a volume fraction of 7/9.

to compute a face fraction, applying it to modify the existing finite difference stencil in a manner similar to finite volume [Ng et al., 2009]. Note that since this computes face fractions, the cut cell method works best when level set values are defined on nodes rather than centers, though interpolation is always available to transfer center values to nodes.

We use neither ghost fluid nor cut cell methods, opting instead to use volume fractions. We expand on how to formulate the problem with volume fractions in Section 3.6. What remains is how to compute the volume fractions from the level set. One method is to adopt marching cubes templates [Wyvill et al., 1986; Lorensen and Cline, 1987], and use linear interpolants to exactly determine the intersection points along each edge. This creates fairly simple local meshes in each cell, whose volumes can be computed, shown on Figure 3.2a. We adopt a simpler approach of sampling the level set with n points along each axis volumetrically, and counting how many points fall inside the level sets, shown on Figure 3.2b. This sampling method comes with the advantage of less computation cost, assuming not too many sampling points are used, and trivial parallelism. Notice, however, that this is rather limited, considering that we have a granularity of 27 assuming 3 samples per axis in 3D. In our tests, we find this to be enough for satisfactory boundary treatment. Such low granularity actually has a rather nice consequence of automatically avoiding small

volume fractions, which require clamping otherwise. While this method is a somewhat aggressive approximation, the linear interpolation used in marching cubes is itself also just an approximation of the underlying fluid volume.

The volume shown on Figure 3.2 is sampled around a cell center. As we also have variables sampled on cell faces and edges, we require volume estimates for these points as well. These are taken by applying the above sampling method within the unit cell around each sampling point, simply shifting the sampling points by $(-0.5, 0, 0)\Delta x$ for an x-face, and $(0, -0.5, -0.5)\Delta x$ for a yz-edge for example, once again highlighting the simplicity of the method.

There is the additional matter of evolving the level set over time. Traditional grid methods typically are initialized with a level set sampled on the grid, and evolve these samples over time by advecting their values, using backwards semi-Lagrangian for example [Bridson, 2015]. This comes with the issue of the diffusion of level set values, causing smoothing of high frequency surface features such as sharp corners. Of course, many more sophisticated surface tracking methods have been developed, such as particle level set which seeds particles in a narrow band along the interface, which are subsequently advected [Enright et al., 2002].

The key detail here is that traditionally, the level set is the primary representation of the fluid geometry. The fluid is evolved by evolving the level set. Instead, we take a primarily particle representation of the fluid, to be described in Section 3.4.1. This means we solely construct a level set from the particles on a temporary basis, for computing volume fractions and for rendering. We need not evolve the level set since it is simply a representation of a fluid which is evolved independently of it.

3.4 Advection and Body Forces

To apply the advection and body forces in Equation 3.8, we further perform splitting to apply each in isolation,

$$\frac{\mathbf{u}^\star - \mathbf{u}_n}{\Delta t} = -\mathbf{u}_n \cdot \nabla \mathbf{u}_n \tag{3.12}$$

$$\frac{\mathbf{u}^* - \mathbf{u}^\star}{\Delta t} = \mathbf{g} \tag{3.13}$$

with $\mathbf{u}^\star$ being another intermediate velocity state between $\mathbf{u}_n$ and $\mathbf{u}^*$. Notice that Equation 3.12 is exactly the equation for zero material derivative, up to time discretization. This

makes sense as advection exactly follows streamlines, and highlights that advection is naturally a Lagrangian phenomenon.

3.4.1 APIC Advection

There is a wide range of methods for performing the advection step in Equation 3.12; due to the splitting used, any such advection method can be used in conjunction with our Stokes update. We choose to use the affine particle-in-cell (APIC) method of Jiang et al. [2015], which we describe here in brief.

APIC is an extension of PIC, a class of hybrid Lagrangian/Eulerian method that uses particles for advection, but computes and applies forces on a grid as usual. Using particles for advection takes advantage of the natural Langrangian structure of advection, while performing force computation on the grid allows for simpler discretization stencils and the use of existing implicit solvers.

Moving between Lagrangian and Eulerian settings is facilitated by interpolation schemes that transfer particle properties to the grid, and vice versa. Thus, a single PIC timestep consists of performing a particle-to-grid (P2G) transfer, applying all non-advection terms on the grid, performing a grid-to-particle (G2P) transfer, and finally applying advection to the particles [Zhu and Bridson, 2005; Harlow, 1962]. It is the P2G and G2P transfers that differentiate APIC from standard PIC.

Standard PIC simply uses a weighted average of nearby interpolants to perform both transfers [Zhu and Bridson, 2005]. The G2P transfer is,

$$\mathbf{u}_i^n = \frac{\sum_p w_{ip}^n \mathbf{u}_p^n}{\sum_p w_{ip}^n} \tag{3.14}$$

where i is a grid index, p iterates over all neighbouring points within i's kernel support, and $w_{ip} = N(\mathbf{x}_p - \mathbf{x}_i)$ is an interpolation kernel with compact support. Likewise, the P2G transfer is,

$$\mathbf{u}_p^{n+1} = \frac{\sum_i w_{ip}^{n+1} \mathbf{u}_i^{n+1}}{\sum_i w_{ip}^{n+1}} \tag{3.15}$$

now centered on some particle p with i iterating over neighbouring grid points within p's kernel support. Note that there is freedom in the choice of interpolating kernel; standard PIC as introduced by Zhu and Bridson [2005] used multilinear interpolation (i.e. a hat

kernel) with support of size $(2\Delta x)^3$, while later methods opted for smoother kernels adapted from smoothed particle hydrodynamics (SPH) [Batty and Bridson, 2008].

Standard PIC was found to be excessively diffusive, caused by the two interpolations performed at each timestep. This motivated the construction of the FLIP method by Brackbill and Ruppel [1986], which used the P2G transfer to *increment* particle properties rather than replace them outright. However, this direct path that retains prior particle values introduces a potential for accumulation of sub-grid scale noise between each timestep. This issue is largely mitigated by taking a linear combination of PIC and FLIP, typically on the order of above 90% FLIP, where a small amount of PIC acts as a mechanism to damp potential noise [Zhu and Bridson, 2005].

Despite this, the PIC/FLIP blend was still found to be dissipative, particularly with regards to angular momentum, which is not conserved in the G2P transfer. This insight lead to the construction of interpolants which preserve affine velocity fields for the APIC method. This requires a matrix $\mathbf{C}_p^n$ representing the linear coefficients of an affine velocity field $\mathbf{u}_p + \mathbf{C}_p(\mathbf{x}_i - \mathbf{x}_p)$. The APIC P2G transfer is,

$$\mathbf{u}_i^n = \frac{\sum_p w_{ip}^n \left(\mathbf{u}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1}(\mathbf{x}_i - \mathbf{x}_p^n)\right)}{\sum_p w_{ip}^n} \tag{3.16}$$

$$\mathbf{D}_p^n = \sum_i w_{ip}^n (\mathbf{x}_i - \mathbf{x}_p^n)(\mathbf{x}_i - \mathbf{x}_p^n)^\intercal \tag{3.17}$$

and the G2P transfer is,

$$\mathbf{u}_p^{n+1} = \frac{\sum_i w_{ip}^{n+1} \mathbf{u}_i^{n+1}}{\sum_i w_{ip}^{n+1}} \tag{3.18}$$

$$\mathbf{B}_p^{n+1} = \sum_i w_{ip}^{n+1} \mathbf{v}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^\intercal \tag{3.19}$$

where $\mathbf{C}_p^n = \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1}$, with $\mathbf{D}_p^n$ being analogous to an inertia tensor and $\mathbf{B}_p^n$ containing angular velocity information for each particle [Jiang et al., 2015]. Notice that in addition to linear velocities, particles now must keep track of their own $\mathbf{B}_p^n$ matrices; these effectively hold the angular momenta that was previously thrown out at each G2P transfer. This extra information also needs to be interpolated back into the grid, accomplished by the extra term in Equation 3.16. A minor quirk resulting from the need to invert $\mathbf{D}_p^n$ is that it becomes advisable to use quadratic or cubic interpolants rather than the prior hat functions, which become singular at grid edges [Jiang et al., 2015].

Summarizing this, we have the following algorithm for solving the incompressible Navier-Stokes equations (Equations 3.1-3.2).

Algorithm 1: One APIC timestep.

1: do APIC particle to grid transfer (Eq 3.16, 3.17);
2: apply body forces (Eq 3.13);
3: do pressure-viscosity solve;
4: do APIC grid to particle transfer (Eq 3.18, 3.19);
5: advect particles (Eq 3.12);

Note that we perform the advection last within a timestep simply to make the association that particles are "kicked" by newly computed forces, rather than by forces computed in the prior timestep—the order of operations here remain the same. As well, this ensures that advection in the first timestep uses the correct forces based on the input state, rather than requiring the user to define a correct initial input. For example, a user may input a non-incompressible field which is automatically handled before advection. Performing advection first in this case may push particles into an unintended, potentially pathological, state should the input field be not divergence-free.

We remind the reader that we simply use APIC as the framework to perform advection. The focus of our work is solely in the pressure-viscosity solve in the above algorithm, which is agnostic to the choice of advection scheme.

3.5 Decoupled Viscosity

While we seek to solve Equations 3.9-3.11 in a single step, we nonetheless describe the standard method for solving the pressure-viscosity update as a point of comparison. Here, operator splitting is once again applied to solve for viscosity alone,

$$\frac{\mathbf{u}^{\dagger} - \mathbf{u}^{*}}{\Delta t} = \frac{1}{\rho}\nabla \cdot \left(\mu \left(\nabla \mathbf{u}^{\dagger} + (\nabla \mathbf{u}^{\dagger})^{\intercal}\right)\right) \tag{3.20}$$

where $\mathbf{u}^{\dagger}$ is another intermediate velocity state between $\mathbf{u}^{*}$ and $\mathbf{u}_{n+1}$. A pressure projection step is used to enforce the divergence free condition,

$$\frac{\mathbf{u}_{n+1} - \mathbf{u}^{\dagger}}{\Delta t} = -\frac{1}{\rho}\nabla p \tag{3.21}$$

with pressure being given as a solution of the Poisson problem given by the divergence-free condition:

$$-\nabla \cdot \nabla p = -\frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^{\dagger} \tag{3.22}$$

Note, however, that the input $\mathbf{u}^{*}$ to Equation 3.20 is typically required to be divergence free, thus requiring two pressure projections (one before Equation 3.20 and one after) in one timestep [Batty and Bridson, 2008; Losasso et al., 2006].

Decoupling the pressure and viscosity in this way, while resulting in smaller systems to solve, introduces inaccuracies particularly with regards to the free surface boundary condition. The free surface boundary condition for the Stokes problem is,

$$\mathbf{t} = (-p\mathbf{I} + \tau)\mathbf{n} = \mathbf{t}_{BC} \tag{3.23}$$

where $\mathbf{t}$ is the traction and $\mathbf{t}_{BC}$ is the boundary condition for said traction. Assuming no surface tension, $\mathbf{t}_{BC} = 0$. Notice that this boundary condition requires a balance of both viscous (τ) and pressure stresses ($p\mathbf{I}$), which cannot be solved for in the decoupled case. Rather, for the decoupled case we simply independently apply a zero boundary condition to each stress [Batty and Bridson, 2008]:

$$p = 0 \tag{3.24}$$

$$\tau\mathbf{n} = 0 \tag{3.25}$$

While this does satisfy Equation 3.23, it does not find the true solution as in the coupled case. It is analogous to solving $a + b = 0$ by separately constraining $a = 0$ and $b = 0$.

3.6 Variational Stokes

Our method largely builds off the work of Larionov et al. [2017] and Batty and Bridson [2008], who construct a variational formulation for solving the unified Stokes problem and decoupled viscosity respectively. Here, we show a derivation for the objective function whose optimal value satisfies the unsteady Stokes equations, and the optimality conditions to find such a solution.

3.6.1 Free Surface Formulation

We start with the free surface case:

$$\rho\left(\mathbf{u}-\mathbf{u}^*\right)+\Delta t\nabla p-\Delta t\nabla\cdot\tau=0, \qquad \text{in } \Omega_L \tag{3.26}$$

$$-\Delta t\nabla\cdot\mathbf{u}=0, \qquad \text{in } \Omega_L \tag{3.27}$$

$$\frac{\Delta t}{2\mu}\tau-\Delta t\varepsilon(\mathbf{u})=0, \qquad \text{in } \Omega_L \tag{3.28}$$

$$\Delta t(-p\mathbf{I}+\tau)\hat{\mathbf{n}}=0, \qquad \text{in } \partial\Omega_L \tag{3.29}$$

where Equations 3.26-3.28 are straightforward rearrangements of the previously stated unsteady Stokes problem (Equations 3.9-3.11) and Equation 3.29 is the free surface boundary condition. $\varepsilon(\mathbf{u})=\frac{1}{2}(\nabla\mathbf{u}+(\nabla\mathbf{u})^\intercal)$ is the deformation rate tensor. These equations hold in the volume of the liquid, Ω_L, and its boundary, $\partial\Omega_L$.

We define variations $\mathbf{v}$ of $\mathbf{u}$, σ of τ, and q of p, and construct the *first variations*,

$$\begin{aligned}\langle\nabla J[u],\mathbf{v}\rangle=&\iiint_{\Omega_L}\left(\rho\mathbf{u}\cdot\mathbf{v}-\rho\mathbf{u}^*\cdot\mathbf{v}+\Delta t\nabla p\cdot\mathbf{v}-\Delta t\nabla\cdot\tau\cdot\mathbf{v}\right)dV\\&+\iint_{\partial\Omega_L}\left(-\Delta tp\mathbf{I}\hat{\mathbf{n}}\cdot\mathbf{v}+\Delta t\tau\hat{\mathbf{n}}\cdot\mathbf{v}\right)dA\end{aligned} \tag{3.30}$$

$$\langle\nabla J[p],q\rangle=\iiint_{\Omega_L}(-\Delta t\nabla\cdot\mathbf{u})q\;dV \tag{3.31}$$

$$\langle\nabla J[\tau],\sigma\rangle=\iiint_{\Omega_L}\frac{\Delta t}{2\mu}\tau:\sigma-\Delta t\varepsilon(\mathbf{u}):\sigma\;dV \tag{3.32}$$

by taking inner products of Equations 3.26 and 3.29 with $\mathbf{v}$, Equation 3.27 with q, and Equation 3.28 with σ respectively, using ':' to denote the tensor double dot product. Here, $J[a]$ is the objective function we are trying to find, given in terms of a. Using integration by parts and the divergence theorem, the following surface integrals can be converted into volume integrals:

$$\iint_{\partial\Omega_L}-\Delta tp\mathbf{I}\hat{\mathbf{n}}\cdot\mathbf{v}\;dA=\iiint_{\Omega_L}-\Delta t(p\nabla\cdot\mathbf{v}+\mathbf{v}\cdot\nabla p)\;dV \tag{3.33}$$

$$\iint_{\partial\Omega_L}\Delta t\tau\hat{\mathbf{n}}\cdot\mathbf{v}\;dA=\iiint_{\Omega_L}\Delta t\varepsilon(\mathbf{v}):\tau+\mathbf{v}\cdot\nabla\cdot\tau\;dV \tag{3.34}$$

Plugging into Equation 3.30 and cancelling out terms, we get:

$$\langle\nabla J[u],\mathbf{v}\rangle=\iiint_{\Omega_L}\rho\mathbf{u}\cdot\mathbf{v}-\rho\mathbf{u}^*\cdot\mathbf{v}-\Delta tp\nabla\cdot\mathbf{v}+\Delta t\varepsilon(\mathbf{v}):\tau\;dV \tag{3.35}$$

Remembering that we choose $\mathbf{u}$, p, and τ to be optimal, we constrain the first variations to be zero:

$$0 = \langle \nabla J[u], \mathbf{v} \rangle = \iiint_{\Omega_L} \rho \mathbf{u} \cdot \mathbf{v} - \rho \mathbf{u}^* \cdot \mathbf{v} - \Delta t p \nabla \cdot \mathbf{v} + \Delta t \varepsilon(\mathbf{v}) : \tau \; dV \tag{3.36}$$

$$0 = \langle \nabla J[p], q \rangle = \iiint_{\Omega_L} (-\Delta t \nabla \cdot \mathbf{u}) q \; dV \tag{3.37}$$

$$0 = \langle \nabla J[\tau], \sigma \rangle = \iiint_{\Omega_L} \frac{\Delta t}{2\mu} \tau : \sigma - \Delta t \varepsilon(\mathbf{u}) : \sigma \; dV \tag{3.38}$$

This is the *weak form* of the variational principle. Since $\mathbf{v}$, q, and σ are chosen arbitrarily, they also serve as our optimality conditions by dropping these test functions.

To arrive at a closed form for our functional, we note that if we can split a first variation $\langle \nabla J[a], b \rangle$ into symmetric bilinear, $B(a, b)$, and linear, $L(b)$, forms such that,

$$\langle \nabla J[a], b \rangle = B(a, b) + L(b) \tag{3.39}$$

then we can construct the corresponding functional with,

$$J[a] = \frac{1}{2} B(a, a) + L(a) + C \tag{3.40}$$

where C is a constant relative to the function of variation. Applying this to each of Equations 3.35, 3.31, and 3.32, we get the functionals:

$$J[\mathbf{u}] = \iiint_{\Omega_L} \frac{\rho}{2} \mathbf{u} \cdot \mathbf{u} - \rho \mathbf{u}^* \cdot \mathbf{u} - \Delta t p \nabla \cdot \mathbf{u} + \Delta t \varepsilon(\mathbf{u}) : \tau + C_{p,\tau} \; dV \tag{3.41}$$

$$J[p] = \iiint_{\Omega_L} -\Delta t p \nabla \cdot \mathbf{u} + C_{\mathbf{u},\tau} \; dV \tag{3.42}$$

$$J[\tau] = \iiint_{\Omega_L} \frac{\Delta t}{4\mu} \tau : \tau - \Delta t \varepsilon(\mathbf{u}) : \tau + C_{\mathbf{u},p} \; dV \tag{3.43}$$

Noting that we may choose to add any arbitrary constant, we add $\frac{\rho}{2} \mathbf{u}^* \cdot \mathbf{u}^*$ to Equation 3.41, and rearrange its first two terms with,

$$\frac{\rho}{2} \mathbf{u} \cdot \mathbf{u} - \frac{\rho}{2} \mathbf{u}^* \cdot \mathbf{u} - \frac{\rho}{2} \mathbf{u} \cdot \mathbf{u}^* + \frac{\rho}{2} \mathbf{u}^* \cdot \mathbf{u}^* = \frac{\rho}{2} \left\| \mathbf{u} - \mathbf{u}^* \right\|^2 \tag{3.44}$$

to get:

$$J[\mathbf{u}] = \iiint_{\Omega_L} \frac{\rho}{2} \left\| \mathbf{u} - \mathbf{u}^* \right\|^2 - \Delta t p \nabla \cdot \mathbf{u} + \Delta t \tau : \left(\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^\intercal}{2} \right) + C'_{p,\tau} \; dV \tag{3.45}$$

We also rearrange Equation 3.43 using the definition of ε:

$$J[\tau] = \iiint_{\Omega_L} \frac{\Delta t}{4\mu}\tau : \tau - \frac{\Delta t}{2\mu}\tau : \tau + C_{\mathbf{u},p} = -\frac{\Delta t}{4\mu}\left\|\tau\right\|_F^2 + C_{\mathbf{u},p}\, dV \tag{3.46}$$

where $\left\|\cdot\right\|_F$ is the Frobenius norm. By inspection, with the constants representing placeholders for missing terms, we can combine Equations 3.45, 3.42, and 3.46 to form the full objective function:

$$\boxed{J[\mathbf{u}, p, \tau] = \iiint_{\Omega_L} \frac{\rho}{2}\left\|\mathbf{u} - \mathbf{u}^*\right\|^2 - \Delta t p \nabla \cdot \mathbf{u} + \Delta t \tau : \left(\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^\intercal}{2}\right) - \frac{\Delta t}{4\mu}\left\|\tau\right\|_F^2 \; dV} \tag{3.47}$$

3.6.2 Free Surface Discretization

To discretize Equation 3.47, we need to construct an integral over the liquid domain, Ω_L. We do so by assuming that variables represent a constant field within a cell centered around the variable's sampling location, shown on Figure 3.3. This simply converts the integral into a sum of variables weighted by the fraction of their control volumes within the liquid domain. We represent this domain as a level set, $\Phi_L(\mathbf{x})$, whose fractional volume within each control volume can be found using the sampling procedure outlined in Section 3.3.2.

In 3D, the first term consists solely of velocities sampled on faces, and are accordingly assigned face-centered volume weights w_L^u (Figure 3.3e). The second term contains pressures and velocity divergences, both sampled on cell centers, and are assigned weights w_L^p (Figure 3.3d). Finally, the rest of the terms contain stresses sampled on edges and centers, and are assigned w_L^τ (Figure 3.3f, 3.3d). We can thus convert the integral, Equation 3.47, into the following semi-discrete summation,

$$\begin{aligned} J[\mathbf{u}, p, \tau] \approx \frac{1}{2}\sum_f w_{L,f}^u \left(\rho\left\|\mathbf{u} - \mathbf{u}^*\right\|^2\right)\Big|_f - \Delta t \sum_c w_{L,c}^p \left(p\nabla \cdot \mathbf{u}\right)\Big|_c \\ + \Delta t \sum_n w_{L,n}^\tau \left(\tau : \left(\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^\intercal}{2}\right) - \frac{1}{4\mu}\left\|\tau\right\|_F^2\right)\Big|_n \end{aligned} \tag{3.48}$$

where the vertical bars indicate evaluation of variables at faces, f, centers, c, and both edges and centers (or nodes in 2D), n.

Compiling the weights into diagonal matrices, W_L^u, W_L^p, and W_L^τ, as well as converting all derivative operators into their discrete counterparts, we construct the fully discrete

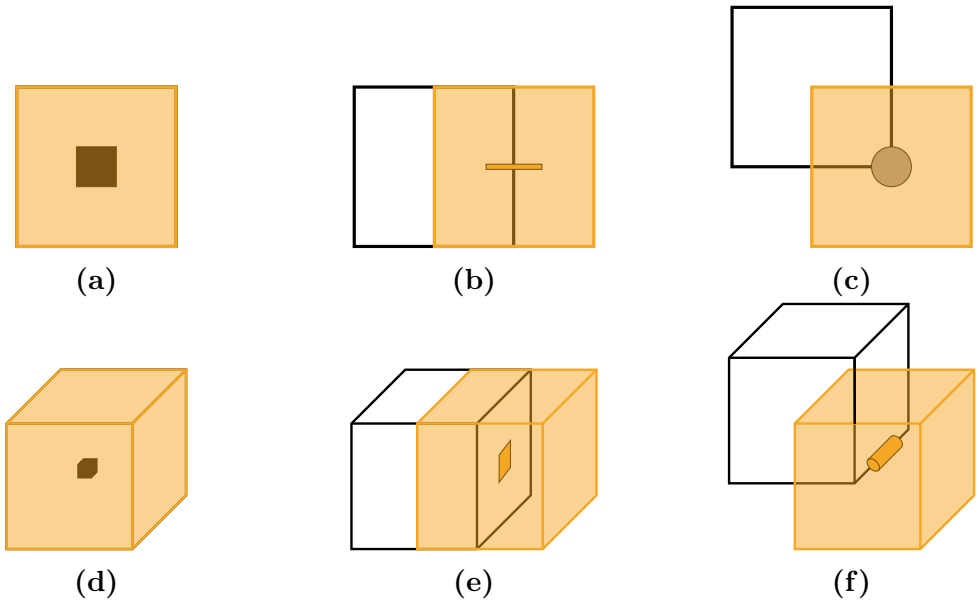


Figure 3.3: Control volumes (shown filled in transparent orange) for 2D (a,b,c) and 3D (d,e,f) geometry around the standard cell (shown in black outline). (a,d) show the relevant volumes around their respective cell-sampled variables, which are denoted by a black square and black cube respectively. (b,e) show the volumes around their respective face-sampled variables, which are denoted by an orange dash and an orange square respectively. (c) shows the volumes around a node-sampled variable denoted by a grey circle. (f) shows the analogous volume in 3D around an edge-sampled variable denoted by an orange cylinder.

matrix-vector expression,

$$\frac{1}{2}(\mathbf{u}-\mathbf{u}^*)^\intercal \mathbf{M} W_L^u (\mathbf{u}-\mathbf{u}^*) + \Delta t \mathbf{p}^\intercal W_L^p \mathbf{G}^\intercal \mathbf{u} + \Delta t \boldsymbol{\tau}^\intercal W_L^\tau \mathbf{D}\mathbf{u} - \frac{\Delta t}{4}\boldsymbol{\tau}^\intercal \boldsymbol{\mu}^{-1} W_L^\tau \boldsymbol{\tau} \tag{3.49}$$

where $\mathbf{u}$, $\mathbf{p}$, and $\boldsymbol{\tau}$ are now stacked vectors of discrete samples, $\mathbf{M}$ is a diagonal matrix of densities per velocity sample, $\boldsymbol{\mu}$ is the diagonal matrix of viscosity coefficients per stress sample, $\mathbf{G}$ is the discrete gradient operator, $\mathbf{D}$ is the discrete deformation rate operator with $\mathbf{D}\mathbf{u} = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal\right)$, $-\mathbf{G}^\intercal$ is the discrete vector divergence operator, and $-\mathbf{D}^\intercal$ is the discrete tensor divergence operator.

To build the system to be solved, we simply apply the same discretization process to our optimality conditions, Equations 3.36-3.38, dropping the arbitrary test functions. We thus get the matrix-vector system:

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} W_L^u & \mathbf{G} W_L^p & \mathbf{D}^\intercal W_L^\tau \\ W_L^p \mathbf{G}^\intercal & 0 & 0 \\ W_L^\tau \mathbf{D} & 0 & -\frac{1}{2}\boldsymbol{\mu}^{-1} W_L^\tau \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} W_L^u \mathbf{u}^* \\ 0 \\ 0 \end{bmatrix} \tag{3.50}$$

3.6.3 Solid Boundary Formulation and Discretization

To discretize the solid boundary case, we follow a similar process as the free surface problem,

$$\rho\left(\mathbf{u}-\mathbf{u}^*\right) + \Delta t \nabla p - \Delta t \nabla \cdot \tau = 0, \qquad \text{in } \Omega_F \tag{3.51}$$

$$-\Delta t \nabla \cdot \mathbf{u} = 0, \qquad \text{in } \Omega_F \tag{3.52}$$

$$\frac{\Delta t}{2\mu}\tau - \Delta t \varepsilon(\mathbf{u}) = 0, \qquad \text{in } \Omega_F \tag{3.53}$$

$$\mathbf{u} = 0, \qquad \text{in } \partial\Omega_F \tag{3.54}$$

where the last equation represents the solid boundary condition. This time, we perform the integrations within the volume of the fluid (i.e. non-solid) region, Ω_F. This gives the objective function:

$$\boxed{J[\mathbf{u}, p, \tau] = \iiint_{\Omega_F} \frac{\rho}{2}\left\|\mathbf{u}-\mathbf{u}^*\right\|^2 + \Delta t \mathbf{u}_{n+1} \cdot (\nabla p - \nabla \cdot \tau) - \frac{\Delta t}{4\mu}\left\|\tau\right\|_F^2 \, dV} \tag{3.55}$$

This discretizes into,

$$\frac{1}{2}(\mathbf{u}-\mathbf{u}^*)^\intercal \mathbf{M} W_F^u (\mathbf{u}-\mathbf{u}^*) + \Delta t \mathbf{u}^\intercal W_F^u (\mathbf{G}\mathbf{p} + \mathbf{D}^\intercal \boldsymbol{\tau}) - \frac{\Delta t}{4}\boldsymbol{\tau}^\intercal \boldsymbol{\mu}^{-1} W_F^\tau \boldsymbol{\tau} \tag{3.56}$$

with non-solid volume weights W_F^u, W_F^p, and W_F^τ. Discretizing the optimality conditions, we get:

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}W_F^u & W_F^u\mathbf{G} & W_F^u\mathbf{D}^\intercal \\ \mathbf{G}^\intercal W_F^u & 0 & 0 \\ \mathbf{D}W_F^u & 0 & -\frac{1}{2}\boldsymbol{\mu}^{-1}W_F^\tau \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}W_F^u\mathbf{u}^* \\ 0 \\ 0 \end{bmatrix} \tag{3.57}$$

3.6.4 Defining the Integration Domains

Notice that with two boundary problems, we need to keep track of two different domains: the liquid domain given by Ω_L and the non-solid domain given by Ω_F. We accomplish this by constructing two level sets, $\Phi_L(\mathbf{x})$ and $\Phi_F(\mathbf{x})$, whose interiors represent the domains Ω_L and Ω_F respectively, as shown on Figure 3.4. Using these two level sets, we can construct the corresponding weights W_L and W_F using the sampling method mentioned in Section 3.3.2.

We interpret the active fluid region as the intersection of the liquid and fluid domains, $\Omega_L \cap \Omega_F$. Note that while the liquid domain does extend into the solid, as shown in Figure 3.4c, this region, $\Omega_L \cap \Omega_S$, does not represent actual physical liquid, and we do not interpret it as such. Rather, it simply represents constraints on the liquid region to enforce the solid boundary condition.

3.6.5 Combined Free Surface and Solid Boundary Problem

With the two boundary value problems defined, with their respective integration domains, we can exploit the similar structure of the two systems to construct the combined free surface and solid boundary problem. In particular, notice that the optimality conditions for the free surface, Equation 3.50, and the solid boundary, Equation 3.57, differ only in volume weight diagonal matrices. The operators are all identical; without the volume weights, they reduce to exactly the finite difference operators for the Stokes problem on a regular grid.

We can thus simply apply both sets of volume weights simultaneously, to construct the combined system [Larionov et al., 2017]:

$$\boxed{\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}W_F^uW_L^u & W_F^u\mathbf{G}W_L^p & W_F^u\mathbf{D}^\intercal W_L^\tau \\ W_L^p\mathbf{G}^\intercal W_F^u & 0 & 0 \\ W_L^\tau\mathbf{D}W_F^u & 0 & -\frac{1}{2}\boldsymbol{\mu}^{-1}W_F^\tau W_L^\tau \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}W_F^uW_L^u\mathbf{u}^* \\ 0 \\ 0 \end{bmatrix}} \tag{3.58}$$

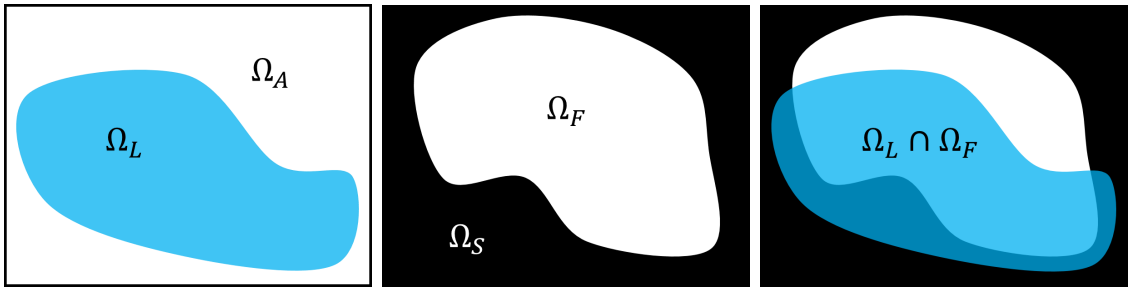


Figure 3.4: Integration domains for (a) the free surface case, (b) the solid boundary case, and (c) the combined case. For the free surface problem, the air domain is defined as the complement of the liquid domain, $\Omega_A = \bar{\Omega_L}$. Likewise, for the solid boundary problem, the solid domain is the complement of the fluid domain, $\Omega_S = \bar{\Omega_F}$. In the combined problem, the active fluid is defined as the intersection of liquid and fluid domains, $\Omega_L \cap \Omega_F$, as defined in the independent boundary cases. Figure adapted from Larionov et al. [2017].

Away from triple points, the system reduces to either Equation 3.50 or Equation 3.57 as the weight matrices of the complementary boundary condition simply become the identity matrix. This formulation was also found to produce natural behaviour at triple points [Larionov et al., 2017].

We additionally would like to note that the weight matrices act on the correct degrees of freedom. For example, the gradient operator maps pressure degrees of freedom into velocity degrees of freedom, $\mathbf{G} : \mathbf{p} \to \mathbf{u}$, therefore it is natural to apply the cell-centered weights corresponding to pressure before the transform, and the face-centered weights corresponding to velocities after the transform, producing the $W_F^u \mathbf{G} W_L^p$ term.

While it is possible to use alternative discretizations of the continuous formulations (Equations 3.47, 3.55), particularly finite element methods (FEMs) and finite volume methods (FVMs), we point out key advantages of adopting the above variational finite difference approach. This discretization recovers the usual staggered finite differences on the interior of the fluid, with the weight matrices on Equation 3.58 reducing to identities. This consequently inherits the stability of the staggered grid against high-frequency noise, previously noted in Section 3.3.1. We also note the guaranteed symmetry of Equation 3.58, and the simplicity of boundary enforcement. Boundary conditions are handled solely by changes in weights around the fluid surface, thus bypassing construction of boundary meshes required for other methods [Larionov et al., 2017; Batty and Bridson, 2008].

3.7 Reduced Fluid Model

We have thus described above a fully uniform viscous fluid solver as outlined by Larionov et al. [2017], along with the accompanying regular grid methods. The key limitation we seek to solve is the large computational cost of the Stokes linear solve given by Equation 3.58. This stems from the large number of degrees of freedom being solved for, motivating a spatial adaptivity approach to focus computation only near the fluid surface.

We accomplish this by building off the work of Goldade et al. [2020], who constructed an affine fluid model for the pressure projection step given by Equations 3.21 and 3.22. This work retains the fully uniform grid on the surface while simplifying large interior regions into *super-cells*, with each region having a reduced set of variables which describe a local affine velocity field. They couple the fully uniform surface representation to the interior affine fields by adopting a similar two-way coupling to the work of Batty et al. [2007], which demonstrated coupling between rigid bodies and fluids. Here, we describe Goldade et al.'s affine pressure projection approach, before building on it for solving the viscosity and Stokes equations in the next chapter.

3.7.1 Affine Field

We define an affine description of a fluid field as follows:

$$\mathbf{u}_R(\mathbf{x}) = \mathbf{u}_{const} + \mathcal{G}(\mathbf{x} - \mathbf{x}_{COM}) \tag{3.59}$$

where $\mathcal{G} = \nabla \mathbf{u}_R$ is the gradient 2-tensor and $\mathbf{x}_{COM}$ is the affine region's center of mass. Since the 3D gradient matrix has the structure,

$$\mathcal{G}_{3D} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix} \tag{3.60}$$

it follows that its trace gives the divergence, $\mathrm{Tr}(\mathcal{G}_{3D}) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \nabla \cdot \mathbf{u}$, and similarly for the 2D case. This implies that enforcing the usual incompressibility constraint can be done by simply enforcing zero trace, $\nabla \cdot \mathbf{u}_R = \mathrm{Tr}(\mathcal{G}) = 0$. This reduces the required degrees

of freedom for representing $\mathcal{G}$ by 1, as shown in the following forms:

$$\mathcal{G}_{2D} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & -a_{11} \end{bmatrix}, \qquad \mathcal{G}_{3D} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & -(a_{11}+a_{22}) \end{bmatrix} \tag{3.61}$$

The velocity within some interior region Ω_R can thus be represented by a generalized velocity vector $\mathbf{v}_R$ of 5 elements in 2D (2 constant and 3 linear) and 11 elements in 3D (3 constant and 8 linear). We define a matrix $\mathbf{C}$ that gives the Euclidean velocity at any point $\mathbf{x}$. We elucidate the 2D case with the 3D case following straightforwardly,

$$\mathbf{u}_R = \mathbf{C}(\mathbf{x})\mathbf{v}_R \tag{3.62}$$

$$= \begin{bmatrix} 1 & 0 & \tilde{x} & \tilde{y} & \tilde{x} \\ 0 & 1 & -\tilde{y} & 0 & \tilde{x} \end{bmatrix} \mathbf{v}_R \tag{3.63}$$

with $\tilde{x} = x - x_{COM}$ and similarly for $\tilde{y}$.

Using this transformation, we can construct a generalized mass matrix using the fluid's kinetic energy as follows,

$$\iiint_{\Omega_R} \frac{\rho_R}{2} \left\|\mathbf{u}_R\right\|^2 \, dV = \iiint \frac{\rho_R}{2} \left\|\mathbf{C}\mathbf{v}_\mathbf{R}\right\|^2 \, dV \tag{3.64}$$

$$= \frac{1}{2}\mathbf{v}_R^\intercal \left(\iiint_{\Omega_R} \rho_R \mathbf{C}^\intercal\mathbf{C} \, dV \right) \mathbf{v}_R \tag{3.65}$$

$$= \frac{1}{2}\mathbf{v}_R^\intercal \mathbf{M}_R \mathbf{v}_R \tag{3.66}$$

with $\mathbf{M}_R = \iiint_{\Omega_R} \rho_R \mathbf{C}^\intercal\mathbf{C} \, dV$ being exactly the required generalized mass matrix.

3.7.2 Coupling to Regular Grid

We require a coupling between the affine and the regular grid representations of the fluid, with the intention of using the affine model in the interior and the regular grid in a narrow band on the fluid boundary, as shown on Figure 3.5. For simplicity of the coupling stencils between the two representations, we assume a voxelized affine interior region that is sufficiently far away from the physical fluid boundary.

To construct our coupling, we define a transfer matrix, $\mathbf{J} : \mathbf{u}_R \to \mathbf{v}_R$ which accumulates forces from the regular grid and transforms them into generalized forces applicable to the

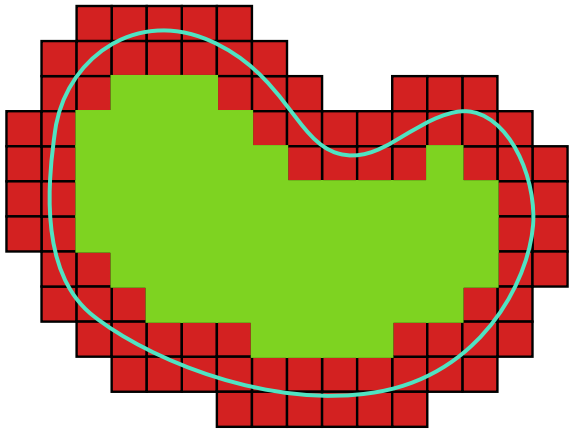


Figure 3.5: Simple fluid domain setup with the fluid to be discretized shown as a blue line, uniform grid cells shown in red, and a single reduced interior region shown in green. Black lines show velocity degrees of freedom on the uniform grid faces. Note that the faces on the boundary of the reduced region are described by the reduced field in green.

interior regions,

$$\mathbf{J}\mathbf{f} = \sum_{a,j} \mathbf{C}_a(\mathbf{x}_j)^\intercal \mathbf{f}_j \tag{3.67}$$

where a iterates over each axis, j iterates over boundary faces, $\mathbf{x}_j$ is the center of face j, and $\mathbf{f}_j$ is a force in Cartesian form. By construction, $\mathbf{J}$ couples all input forces around an interior region to the generalized degrees of freedom. The transpose operation, $\mathbf{J}^\intercal : \mathbf{v}_R \to \mathbf{u}_R$, distributes generalized forces from the affine representation into the surrounding regular grid representation.

Using $\mathbf{J}$ and $\mathbf{M}_R$, we can construct the generalized velocity form of the pressure-velocity update:

$$\mathbf{M}_R \frac{\partial \mathbf{v}_R}{\partial t} = \mathbf{J}(-\nabla p) \tag{3.68}$$

Note that this differs slightly from the construction given in Goldade et al. [2020], which combines both the gradient operation and the Cartesian-to-generalized coordinate transformation into $\mathbf{J}$. We choose to define $\mathbf{J}$ as solely the coordinate transformation so as to be able to use it with any force. This will be relevant when we need to transform both pressure and stresses for the Stokes problem.

We can thus construct the coupled pressure projection system as,

$$\rho \frac{\partial \mathbf{u}_F}{\partial t} = -\nabla p \tag{3.69}$$

$$\nabla \cdot \mathbf{u}_F = 0 \tag{3.70}$$

$$\mathbf{M}_R \frac{\partial \mathbf{v}_R}{\partial t} = \mathbf{J}(-\nabla p) \tag{3.71}$$

$$\mathbf{u}_C = \mathbf{v}_R, \text{on } \partial\Omega_C = \partial\Omega_R \tag{3.72}$$

where Ω_R is solely the parts of the fluid whose velocity is defined by the generalized velocities, $\mathbf{v}_R$, and $\Omega_C = \bar{\Omega}_R$ is solely parts whose velocity is defined with the usual Cartesian grid velocities, $\mathbf{u}_C$. Because we already assume incompressibility in our definition of the affine field, we need not have a divergence-free condition for the generalized velocities. This additionally means that pressures are only defined in Ω_C.

This discretizes into:

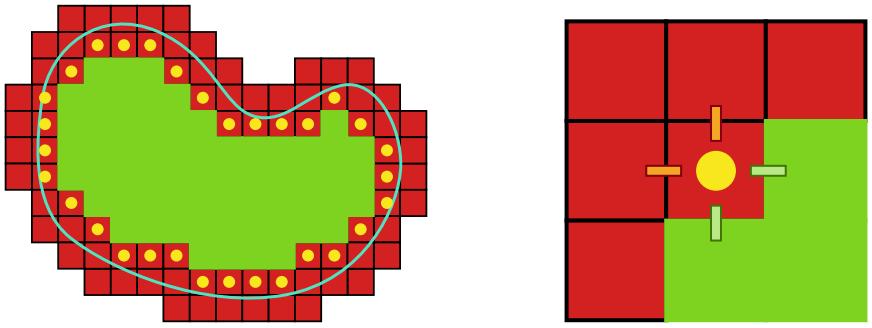
$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C & \mathbf{G} & 0 \\ \mathbf{G}^\intercal & 0 & \mathbf{G}^\intercal\mathbf{J}^\intercal \\ 0 & \mathbf{J}\mathbf{G} & \frac{1}{\Delta t}\mathbf{M}_R \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{p} \\ \mathbf{v}_R \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_F\mathbf{u}_C^* \\ \mathbf{0} \\ \frac{1}{\Delta t}\mathbf{M}_B\mathbf{v}_R^* \end{bmatrix} \tag{3.73}$$

where $\mathbf{M}_C$ is the diagonal matrix of face-centered density samples in the uniform grid, $\mathbf{M}_R = \sum_{a,j} \rho_R \mathbf{C}_a(\mathbf{x}_j)^\intercal \mathbf{C}_a(\mathbf{x}_j)\, dV$ is the discrete generalized mass matrix, and $\mathbf{u}_C^*$ and $\mathbf{v}_R^*$ are the intermediate input velocities in uniform and generalized coordinates respectively. $\mathbf{G}$ is the discrete gradient operator and $-\mathbf{G}^\intercal$ is the discrete vector divergence operator, both as previously defined in Section 3.6.2.

Separating out $\mathbf{J}$ highlights the structure of the system—we effectively solve the same pressure-velocity update separately for the Cartesian velocities (first row) and the generalized velocities (last row). The pressure stencil for this reduced region is shown on Figure 3.6a.

Coupling is handled across the shared boundary between the Cartesian and reduced regions using the equality constraint, Equation 3.72, as applied to the divergence free condition, Equation 3.70, resulting in the second row of the system. The $\mathbf{G}^\intercal$ operator is used for both sets of velocities, meaning that implementation-wise, we can adopt the usual gradient stencils, and simply use $\mathbf{J}^\intercal$ to transform any generalized velocities touched by the stencil into Cartesian velocities, as shown on Figure 3.6b.

With regards to boundary conditions, Goldade et al. [2020] used the aforementioned ghost fluid and cut cell methods to enforce free surface and solid boundaries respectively.



(a) (b)

Figure 3.6: Coupling between the uniform grid (red) and reduced regions (green). (a) shows the same fluid domain as Figure 3.5, with cell-centered pressures coupled to the reduced region shown in yellow, as subject to the last row of Equation 3.73. (b) shows the finite difference stencil for a single divergence that couples together uniform grid velocities (orange dashes) and affine velocities (light green dashes), as dictated by the second row of Equation 3.73. This discrete divergence uses an identical stencil to the usual finite difference method, but the light green velocity samples here are simply given by the reduced model.

We take the previously defined volume weights approach from Larionov et al. [2017]. Application of the volume weight terms to Equation 3.73 straightforwardly follows from the system given in Equation 3.58, using $W^u_{L/F}$ and $W^p_{L/F}$ as given in the upper left 2×2 block. This will be discussed further when we build our Stokes system. In particular, since we seek to immerse the affine domain, Ω_R, within the physical liquid, the volume weights of any entry touching the affine terms are identically 1.

3.7.3 Tiled Regions

Of course, we cannot expect the affine model to be as accurate a representation of the fluid as the uniform grid. Its role as a reduced model is to be able to represent the most important modes of the flow—which we have chosen to be the constant and linear modes—with as few degrees of freedom as possible. It thus becomes insufficient once higher-order modes become significant.

In particular, this limits the size of potential affine regions, as larger fluid bodies have more freedom to exhibit higher-order modes. To alleviate this issue, Goldade et al. [2020] proposed tiling the fluid interior, dividing it into smaller blocks of predefined size, as

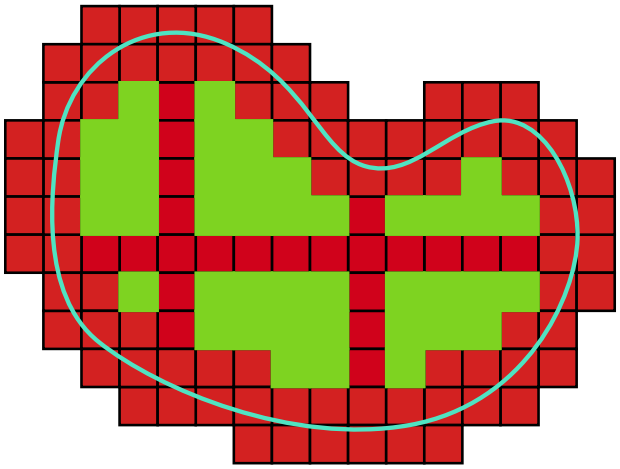


Figure 3.7: The same fluid domain as in Figure 3.5, but with a basic tiling scheme. Tiles are constructed to be at most 4-cells wide with 1-cell padding between.

shown on Figure 3.7. These blocks are separated by a layer of uniform grid cells, such that no two affine regions are immediately adjacent. This avoids the construction of a new operator acting between affine regions, allowing for straightforward implementation. The only change to the system is that the $\mathbf{M}_R$ and $\mathbf{J}$ matrices become block diagonal, with each block representing a local matrix for one interior region.

We clarify some terminology we use regarding these tiled interior regions on Figure 3.8. We call the maximum size of the interior region itself as the *interior region size*. The number of cells that separate neighbouring interior regions is called the *padding size*. Finally, the sum of the two, which represents one repeating unit that tiles the interior of the fluid, is called the *tile size*.

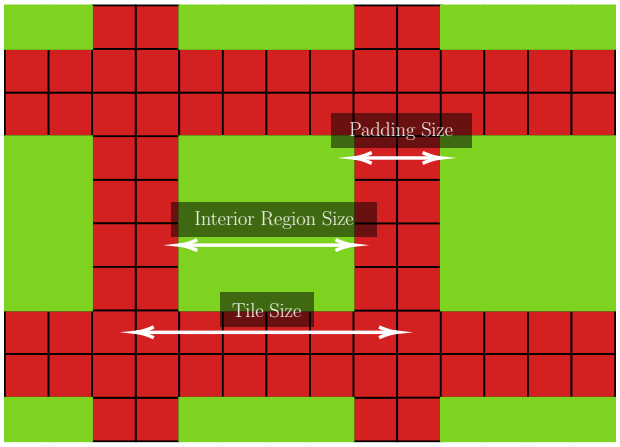


Figure 3.8: Terminology used for describing the tiling setup for reduced regions. For this example, the interior region size is 4 cells, the padding size is 2 cells, and the tile size is the sum which is 6 cells.

Chapter 4

Reduced Fluid Methods for Viscosity and Stokes

4.1 Affine Field on Viscosity

We begin constructing our reduced model solver by first looking at the decoupled viscosity formulation given in Section 3.5. Using the variational method of Batty and Bridson [2008] for solving the decoupled viscosity step, Equation 3.20, we have the following objective function:

$$J[\mathbf{u}] = \iiint_{\Omega_L} \frac{\rho}{2} \|\mathbf{u} - \mathbf{u}^*\|^2 + \Delta t \mu \left\| \frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^\intercal}{2} \right\|_F^2 dV \tag{4.1}$$

Notice that this is simply Equation 3.47 without the pressure terms. We separate this integration into the uniform Cartesian regions, Ω_C, and the reduced fluid regions, Ω_R, such that $\Omega_C \cap \Omega_R = \partial\Omega_R \subseteq \partial\Omega_C$ and $\Omega_C \cup \Omega_R = \Omega_L$. These domains and their boundaries are shown on Figure 4.1. The boundary region subset is important here so as to distinguish between the boundary separating the uniform Cartesian region and the reduced fluid region, $\partial\Omega_R$, from the general boundary of the Cartesian region, $\partial\Omega_C$, which includes the true liquid boundary, $\partial\Omega_L$. This subset relation holds assuming we immerse the reduced fluid region within the liquid.

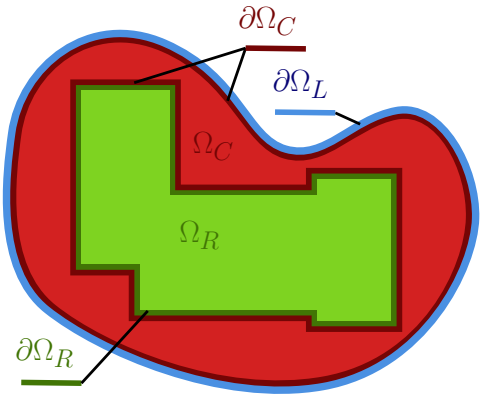


Figure 4.1: Schematic of various domains used in our reduced solver. The Cartesian domain, Ω_C, and reduced domain, Ω_R, form a decomposition of the full liquid domain, $\Omega_L = \Omega_C \cup \Omega_R$, such that they share only a boundary, $\Omega_C \cap \Omega_R = \partial\Omega_R \subseteq \partial\Omega_C$.

Taking $\mathbf{u} = \mathbf{u}_C$ in Ω_C and $\mathbf{u} = \mathbf{u}_R$ in Ω_R, we get the following objective function,

$$\begin{aligned} J[\mathbf{u}_C, \mathbf{u}_R] = \iiint_{\Omega_L} & \frac{\rho}{2}\left\|\mathbf{u}_C - \mathbf{u}_C^*\right\|^2 + \frac{\rho}{2}\left\|\mathbf{u}_R - \mathbf{u}_R^*\right\|^2 + \rho(\mathbf{u}_C - \mathbf{u}_C^*)\cdot(\mathbf{u}_R - \mathbf{u}_R^*) \\ & + \Delta t\mu \left\| \frac{\nabla\mathbf{u}_C + (\nabla\mathbf{u}_C)^\intercal}{2} \right\|_F^2 + \Delta t\mu \left\| \frac{\nabla\mathbf{u}_R + (\nabla\mathbf{u}_R)^\intercal}{2} \right\|_F^2 \\ & + 2\Delta t\mu \left\langle \frac{\nabla\mathbf{u}_C + (\nabla\mathbf{u}_C)^\intercal}{2}, \frac{\nabla\mathbf{u}_R + (\nabla\mathbf{u}_R)^\intercal}{2} \right\rangle_F dV \end{aligned} \tag{4.2}$$

with $\langle\cdot,\cdot\rangle_F$ as the Frobenius inner product. As shown in Appendix A, minimizing this objective function in $\mathbf{u}_{C/R}$ solves the following system:

$$\frac{\mathbf{u}_C - \mathbf{u}_C^*}{\Delta t} = \frac{1}{\rho}\mu\nabla\cdot(\nabla\mathbf{u}_C + (\nabla\mathbf{u}_C)^\intercal), \qquad \text{in } \Omega_C \tag{4.3}$$

$$\frac{\mathbf{u}_R - \mathbf{u}_R^*}{\Delta t} = \frac{1}{\rho}\mu\nabla\cdot(\nabla\mathbf{u}_R + (\nabla\mathbf{u}_R)^\intercal), \qquad \text{in } \Omega_R \tag{4.4}$$

$$\mathbf{u}_C = \mathbf{u}_R, \qquad \text{on } \partial\Omega_R \tag{4.5}$$

Notice that this is the viscosity system, Equation 3.20, solved separately for regions Ω_C and Ω_R, with two-way coupling being handled by the matching condition on their mutual boundary. We make the assumption of constant velocity (note the μ outside of the divergence, as compared to the original viscosity update in Equation 3.20), as a limitation of

the reduced model. Because the reduced model does not use individual stress degrees of freedom on the interior, there is no definition of a spatially-varying viscosity field, and we thus necessarily fall back on a constant viscosity value.

Using the mapping $\mathbf{J}^\intercal : \mathbf{v}_R \rightarrow \mathbf{u}_R$, we can construct a discretization of the corresponding first-order conditions in terms of uniform Cartesian velocities, $\mathbf{u}_C$, and generalized velocities, $\mathbf{v}_R$, for the reduced model:

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u + 2W_F^u \mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D} W_F^u & 2W_F^u \mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D} W_F^u \mathbf{J}^\intercal \\ 2\mathbf{J} W_F^u \mathbf{D}^\intercal W_L^\tau \boldsymbol{\mu} (W_F^\tau)^{-1} \mathbf{D} W_F^u & \frac{1}{\Delta t}\mathbf{M}_R + 2\mathbf{J} W_F^u \mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D} W_F^u \mathbf{J}^\intercal \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{v}_R \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u \mathbf{u}^* \\ \frac{1}{\Delta t}\mathbf{M}_R \mathbf{v}_R^* \end{bmatrix} \tag{4.6}$$

Assuming that all $\mathbf{v}_R$ samples are immersed within the active fluid domain, $\Omega_R \subset \Omega_{L \cap F}$, such that its boundary is sufficiently far from both boundary conditions, $dist(\partial\Omega_R, \partial\Omega_R \cup \partial\Omega_F) > \Delta x$ where $dist(A, B) = \inf\{\|x - y\| \mid x \in A, y \in B\}$ is a distance metric, then all weights relevant to $\mathbf{v}_R$ are unity and we can simplify the matrix system to:

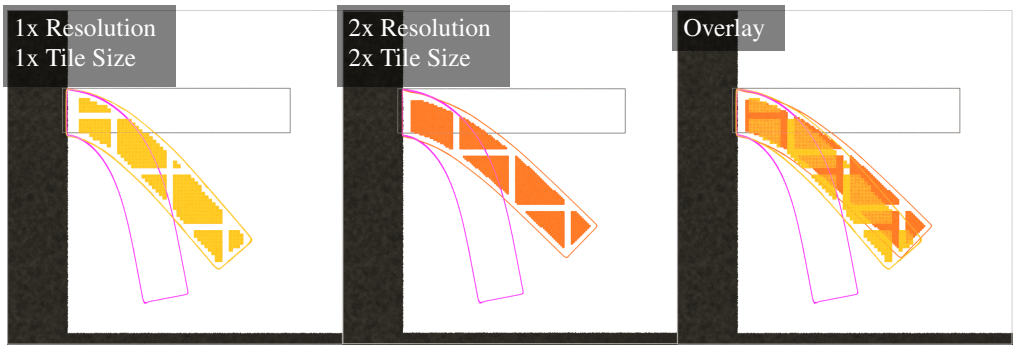
$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u + 2W_F^u \mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D} W_F^u & 2W_F^u \mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} \mathbf{J}^\intercal \\ 2\mathbf{J}\mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} W_F^u & \frac{1}{\Delta t}\mathbf{M}_R + 2\mathbf{J}\mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} \mathbf{J}^\intercal \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{v}_R \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u \mathbf{u}^* \\ \frac{1}{\Delta t}\mathbf{M}_R \mathbf{v}_R^* \end{bmatrix} \tag{4.7}$$

This already hints at an upcoming issue with using an affine representation for the viscosity equation. Remembering that by definition of the affine system, $\mathcal{G} = \nabla \mathbf{u}_R$. In the discrete setting, we have $\mathcal{G} = \mathbf{D}\mathbf{J}^\intercal \mathbf{v}_R$. Since $\mathcal{G}$ is constant in an affine field, it thus follows that the $\mathbf{J}\mathbf{D}^\intercal\mathbf{D}\mathbf{J}^\intercal$ term in Equation 4.7 becomes zero:

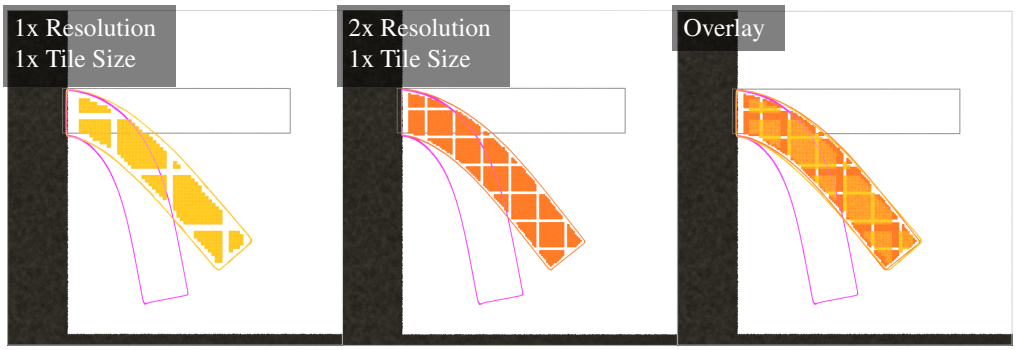
$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u + 2W_F^u \mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D} W_F^u & 2W_F^u \mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} \mathbf{J}^\intercal \\ 2\mathbf{J}\mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} W_F^u & \frac{1}{\Delta t}\mathbf{M}_R \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{v}_R \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u \mathbf{u}^* \\ \frac{1}{\Delta t}\mathbf{M}_R \mathbf{v}_R^* \end{bmatrix} \tag{4.8}$$

This means that we no longer solve for viscosity within the reduced region. The second equation is now strictly a velocity matching constraint on the boundary between the reduced model and the rest of the fluid, with no interior dynamics being performed.

The issue is thus not in the representative power of an affine field with respect to velocity, but rather of the viscous stresses. Because an affine field lacks a second derivative, it lacks any information for being able to evolve itself, and consequently enforce any stresses back onto the uniform grid. An alternative way to view this would be with the function of viscosity as a Laplacian smoothing operator, converting high frequency velocity modes



(a)



(b)

Figure 4.2: Falling viscous beam solved using a uniform grid (pink) and affine reduced fluid (yellow, orange) for the viscosity step in a decoupled pressure-viscosity solve. (a) provides a comparison between a base resolution simulation (yellow) and a simulation with both double spatial resolution and tile size (orange), such that the tiles have physically the same size. (b) similarly provides a comparison between a base resolution (yellow) and a double resolution but keeping the base tile size (orange). All images show a 2D cutaway view of a 3D simulation, with lines representing fluid boundaries for each method, reduced model interior regions shown filled in with their respective colours, and solid boundaries shown in black. Uniform grid cells are not shown for clarity, but take up all gaps between interior regions and a narrow band on the fluid surface. Initial condition shown in grey, and consists of a homogeneous fluid with $\rho = 1$ and $\mu = 100$. Result shown for $t = 1.6$. The affine results sharply disagree with the uniform reference, even under spatial refinement.

into low frequency. Since the affine field lacks any high frequency modes, this results in no net change.

We demonstrate a practical result of this issue with a simple collapsing viscous beam example, shown on Figure 4.2. Here, the affine method is considerably stiffer than the reference solution using a uniform grid. Notice that this is not resolved even under spatial refinement. Figure 4.2a shows that increasing resolution such that the interior regions represent the same physical space as the base resolution produces a worse result. This is in spite of the increased cell distance between interior regions, which would introduce more degrees of freedom by which to resolve boundary forces. Figure 4.2b shows that increasing resolution while keeping the same interior region size, and consequently producing smaller physical regions, does not improve the result.

4.2 Polynomial Field

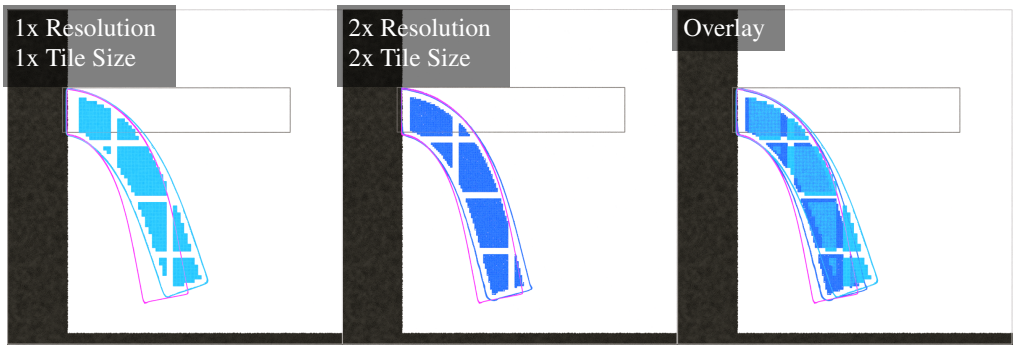
The lack of information for evolving viscosity in the reduced region motivates the construction of a higher-order *polynomial* representation of the fluid. In place of the affine description, Equation 3.59, we construct the following quadratic model:

$$\mathbf{u}_R(\mathbf{x}) = \mathbf{u}_{const} + \mathcal{G}(\mathbf{x} - \mathbf{x}_{COM}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_{COM})^\intercal \mathcal{H}(\mathbf{x} - \mathbf{x}_{COM}) \tag{4.9}$$

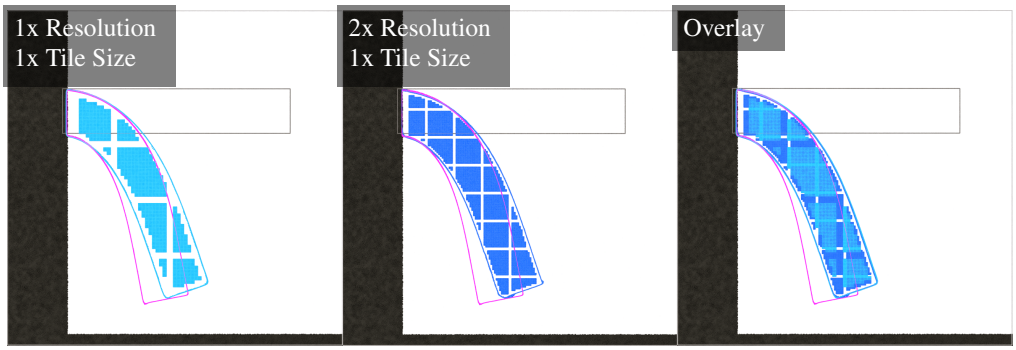
where $\mathcal{H}$ is the Hessian 3-tensor with $\mathcal{H}_{i,j,k} = \frac{\partial^2 u_i}{\partial x_j \partial x_k}$. Each page i of $\mathcal{H}$ is symmetric, thus requiring only 18 degrees of freedom rather than the full 27. We also enforce incompressibility by applying the usual $\nabla \cdot \mathbf{u}_R = 0$ constraint, producing a total of 26 degrees of freedom. That is, 3 for the constant term, 8 for the linear term, and 15 for the quadratic term. For clarity, we give the definition of $\mathbf{C}$ for the quadratic model in 3D in Appendix B.

Using this new transformation matrix, $\mathbf{C}$, we can use the same definitions of the generalized mass matrix, $\mathbf{M}_R$, and transfer matrix, $\mathbf{J}$, as in the affine case. Applying this to the viscosity system, we thus retain the $\mathbf{J}\mathbf{D}^\intercal\mathbf{D}\mathbf{J}^\intercal$ term that canceled away for affine fields, and consequently are able to resolve the missing dynamics within the interior regions. This extension of the affine model into a quadratic model is similar in spirit to the generalization of APIC into polynomial particle-in-cell (PolyPIC), which seeks to retain higher-order polynomial modes during each interpolation [Fu et al., 2017; Jiang et al., 2015].

Applying this quadratic field to the above viscous beam problem, we get the results shown on Figure 4.3. Once again, the first subfigure shows the result for increasing the



(a)



(b)

Figure 4.3: Falling viscous beam solved using a uniform grid (pink) and affine reduced fluid (sky blue, dark blue) for the viscosity step in a decoupled pressure-viscosity solve. (a) provides a comparison between a base resolution simulation (sky blue) and a simulation with both double spatial resolution and tile size (dark blue), such that the tiles have physically the same size. (b) similarly provides a comparison between a base resolution (sky blue) and a double resolution but keeping the base tile size (dark blue). All images show a 2D cutaway view of a 3D simulation, with lines representing fluid boundaries for each method, reduced model interior regions shown filled in with their respective colours, and solid boundaries shown in black. Uniform grid cells are not shown for clarity, but take up all gaps between interior regions and a narrow band on the fluid surface. Initial condition shown in grey, and consists of a homogeneous fluid with $\rho = 1$ and $\mu = 100$. Result shown for $t = 1.6$. The quadratic model's results show much closer agreement with the reference result than the affine model given in Figure 4.2.

resolution and interior region size such that they physically represent the same space, while the second subfigure shows the result of increasing just the spatial resolution without increasing the interior region size. We note that in all cases, the quadratic model performs better than the affine model, achieving much closer results to the reference. Interestingly, the increased resolution with larger interior regions shown on Figure 4.3a shows improved results, while maintaining the same interior region size as shown on Figure 4.3b does not. This indicates that convergence error is highly influenced by the padding size between interior regions, as we use the same error tolerance in all the viscous beam tests, and that keeping a padding of only two cells greatly limits accuracy.

4.3 Reduced Model on Stokes

Having reviewed the reduced pressure model in Section 3.7, and developed a reduced viscosity model in Section 4.1, we now combine them to construct our full reduced Stokes system. We begin with the variational Stokes objective function, Equation 3.47, and apply the same separation of Ω_C and Ω_R domains as used in Section 4.1. That is, we define a uniform Cartesian region, Ω_C, and an internal reduced fluid region, Ω_R, that combined span the entire liquid, $\Omega_C \cup \Omega_R = \Omega_L$, with a shared boundary, $\Omega_C \cap \Omega_R = \partial\Omega_R \subseteq \partial\Omega_C$. This shared boundary must be sufficiently far from both boundary conditions, such that $dist(\partial\Omega_R, \partial\Omega_R \cup \partial\Omega_F) > \Delta x$. Due to this immersion, it follows that $\partial\Omega_L = \partial\Omega_C \setminus \partial\Omega_R$

Applying these two domains, we get,

$$\begin{aligned} J[\mathbf{u}_C, \mathbf{u}_R, p, \tau_L] = &\iiint_{\Omega_C} \frac{\rho}{2\Delta t}\|\mathbf{u}_C - \mathbf{u}_C^*\|^2 + \rho(\mathbf{u}_C - \mathbf{u}_C^*)\cdot(\mathbf{u}_R - \mathbf{u}_R^*) \\ &\qquad - p\nabla\cdot\mathbf{u}_C + \tau_L : \varepsilon(\mathbf{u}_C) - \frac{1}{4\mu}\|\tau_L\|_F^2 \; dV \\ &+ \iiint_{\Omega_R} \frac{\rho}{2\Delta t}\|\mathbf{u}_R - \mathbf{u}_R^*\|^2 + \rho(\mathbf{u}_C - \mathbf{u}_C^*)\cdot(\mathbf{u}_R - \mathbf{u}_R^*) \\ &\qquad - p\nabla\cdot\mathbf{u}_R + \tau_L : \varepsilon(\mathbf{u}_R) - \frac{1}{4\mu}\|\tau_L\|_F^2 \; dV \end{aligned} \tag{4.10}$$

where $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal)$ is again the deformation rate tensor. τ_L here is defined throughout the liquid, but for the purpose of our discretization, we choose to define actual degrees of freedom $\tau_L|_{\Omega_C} = \tau$ only on Ω_C, using the relation $\tau_L|_{\Omega_R \setminus \partial\Omega_R} = 2\mu\varepsilon(\mathbf{u}_R)$ for the

complementary set. Plugging this into the above objective function, we get:

$$\begin{aligned} J[\mathbf{u}_C, \mathbf{u}_R, p, \tau_L] = \iiint_{\Omega_C} &\frac{\rho}{2\Delta t}\|\mathbf{u}_C - \mathbf{u}_C^*\|^2 + \rho(\mathbf{u}_C - \mathbf{u}_C^*)\cdot(\mathbf{u}_R - \mathbf{u}_R^*) \\ &- p\nabla\cdot\mathbf{u}_C + \tau : \varepsilon(\mathbf{u}_C) - \frac{1}{4\mu}\|\tau\|_F^2 \; dV \\ + \iiint_{\Omega_R} &\frac{\rho}{2\Delta t}\|\mathbf{u}_R - \mathbf{u}_R^*\|^2 + \rho(\mathbf{u}_C - \mathbf{u}_C^*)\cdot(\mathbf{u}_R - \mathbf{u}_R^*) \\ &- p\nabla\cdot\mathbf{u}_R + \tau : \varepsilon(\mathbf{u}_R) + \mu\|\varepsilon(\mathbf{u}_R)\|^2 \; dV \end{aligned} \tag{4.11}$$

Likewise, we define p solely on Ω_C, considering that incompressibility within the reduced fluid region is enforced by the definition of the reduced model.

We demonstrate in Appendix C that minimizing the above objective function solves the following system,

$$\frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*) + \nabla p + \nabla\cdot\tau = 0, \qquad \text{in } \Omega_C \tag{4.12}$$

$$\frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*) + \nabla p + \nabla\cdot\tau + \mu\nabla\cdot\varepsilon(\mathbf{u}_R) = 0, \qquad \text{in } \Omega_R \tag{4.13}$$

$$(-p\mathbf{I} + \tau)\hat{\mathbf{n}} = 0, \qquad \text{on } \partial\Omega_L \tag{4.14}$$

$$\nabla\cdot\mathbf{u}_C + \nabla\cdot\mathbf{u}_R = 0, \qquad \text{on } \Omega_C \tag{4.15}$$

$$\varepsilon(\mathbf{u}_C) - \frac{1}{2\mu}\tau + \varepsilon(\mathbf{u}_R) = 0, \qquad \text{on } \Omega_C \tag{4.16}$$

noting that $\mathbf{u}_C$, p, and τ are defined only on Ω_C; and $\mathbf{u}_R$ is defined only on Ω_R. This thus solves the free surface boundary condition. We point out that the free surface boundary condition is unaffected by our new fluid representation, due to the boundary between representations being far away from the physical boundary. Consequently, the solid boundary condition follows straightforwardly and simply requires the same methods for enforcing the boundary as before.

Discretizing the optimality conditions for our objective function, and its solid boundary condition analogue, results in the matrix-vector equation,

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u & 0 & W_F^u\mathbf{G}W_L^p & W_F^u\mathbf{D}^\intercal W_L^\tau \\ 0 & \frac{1}{\Delta t}\mathbf{M}_R + 2\mathbf{J}\mathbf{D}^\intercal\boldsymbol{\mu}\mathbf{D}\mathbf{J}^\intercal & \mathbf{J}\mathbf{G}W_L^p & \mathbf{J}\mathbf{D}^\intercal W_L^\tau \\ W_L^p\mathbf{G}^\intercal W_F^u & W_L^p\mathbf{G}^\intercal\mathbf{J}^\intercal & 0 & 0 \\ W_L^\tau\mathbf{D}W_F^u & W_L^\tau\mathbf{D}\mathbf{J}^\intercal & 0 & -\frac{1}{2}\boldsymbol{\mu}^{-1}W_F^\tau W_L^\tau \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{v}_R \\ \mathbf{p} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u \mathbf{u}^* \\ \frac{1}{\Delta t}\mathbf{M}_R\mathbf{v}_R^* \\ 0 \\ 0 \end{bmatrix} \tag{4.17}$$

where any weights applied on $\mathbf{v}_R$ simply become the identity matrix. This system can then be solved using a choice of linear algebra solver. We choose to use the iterative biconjugate gradient stabilized (BiCGSTAB) method suitable for use on general square matrices. In our implementation, however, we do not solve this full form, but rather a smaller pressure-velocity form to be described in the next section.

This linear system, in effect, solves the momentum balance equation separately for the regular grid cell (first row) and the reduced model (second row), with coupling being handled by the stress and pressure forces being applied on the boundary. An explicit incompressibility condition is not required within the reduced model since it is enforced by definition of the reduced model itself.

With regards to implementation, this method can easily be added to an existing variational Stokes solver due to the $\mathbf{G}$ and $\mathbf{D}$ operators remaining the same. We simply perform the usual iteration through all faces using the same stencils, with the only difference being the handling of faces that fall inside the interior regions. For these, the weight that would fall on a standard face is instead distributed into the interior region using the $\mathbf{J}$ transformation operator.

4.3.1 Alternative Matrix-Vector Forms

This system is amenable to transformation into more convenient forms via Schur complements. That is, for a matrix-vector equation,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} \tag{4.18}$$

the equation in y can be eliminated by solving $(A - BD^{-1}C)x = a - BD^{-1}b$ instead. Should values of y be needed, a second solve for $Cx + Dy = b$ can be done.

For our implementation, we eliminated stress to produce the pressure-velocity form:

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u + W_F^u \mathbf{V} W_F^u & W_F^u \mathbf{V}\mathbf{J}^\intercal & W_F^u \mathbf{G} W_L^p \\ \mathbf{J}\mathbf{V} W_F^u & \frac{1}{\Delta t}\mathbf{M}_R + \mathbf{J}\mathbf{V}\mathbf{J}^\intercal & \mathbf{J}\mathbf{G} W_L^p \\ W_L^p \mathbf{G}^\intercal W_F^u & W_L^p \mathbf{G}^\intercal \mathbf{J}^\intercal & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_C \\ \mathbf{v}_R \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{M}_C W_F^u W_L^u \mathbf{u}^* \\ \frac{1}{\Delta t}\mathbf{M}_R \mathbf{v}_R^* \\ 0 \end{bmatrix} \tag{4.19}$$

where $\mathbf{V} = 2\mathbf{D}^\intercal (W_F^\tau)^{-1} \boldsymbol{\mu} W_L^\tau \mathbf{D}$ is the discrete volume-weighted viscosity operator. Note that for brevity, we use $\mathbf{V}$ which includes explicit volume weights everywhere, even where

they can be reduced to the identity matrix such as in the $\mathbf{JVJ}^\intercal$ term. This pressure-velocity form is ideal for implementation simplicity, as it reduces the system's size without requiring any additional code. Since we do not actually need values of τ to evolve the system, by removing it we also remove the need to construct its edge-valued grid structure. Additionally, by explicitly solving for velocities, we avoid needing to perform another solve—new velocities can be read directly off the solution vector, with a simple conversion using $\mathbf{C}(\mathbf{x})$ in the case of $\mathbf{v}_R$.

On the converse, it is possible to eliminate the Cartesian velocities and subsequently the generalized velocities to produce a pressure-stress form,

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} W_L^p \mathbf{G}^\intercal W_F^u \mathbf{u}_C^* + \Delta t W_L^p \mathbf{G}^\intercal \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{M}_R \mathbf{v}_R^* \\ W_L^\tau \mathbf{D} W_F^u \mathbf{u}^* + \Delta t W_L^\tau \mathbf{D} \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{M}_R \mathbf{v}_R^* \end{bmatrix} \tag{4.20}$$

where:

$$\mathbf{A}_{11} = \Delta t W_L^p \mathbf{G}^\intercal W_F^u (W_L^u)^{-1} \mathbf{M}_C^{-1} + W_L^p \mathbf{G}^\intercal \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{J} \mathbf{G} W_L^p \tag{4.21}$$

$$\mathbf{A}_{12} = \Delta t W_L^p \mathbf{G}^\intercal W_F^u (W_L^u)^{-1} \mathbf{M}_C^{-1} \mathbf{D}^\intercal W_L^\tau + W_L^p \mathbf{G}^\intercal \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{J} \mathbf{D}^\intercal W_L^\tau \tag{4.22}$$

$$\mathbf{A}_{21} = \Delta t W_L^\tau \mathbf{D} W_F^u (W_L^u)^{-1} \mathbf{M}_C^{-1} \mathbf{G} W_L^p + W_L^\tau \mathbf{D} \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{J} \mathbf{G} W_L^p \tag{4.23}$$

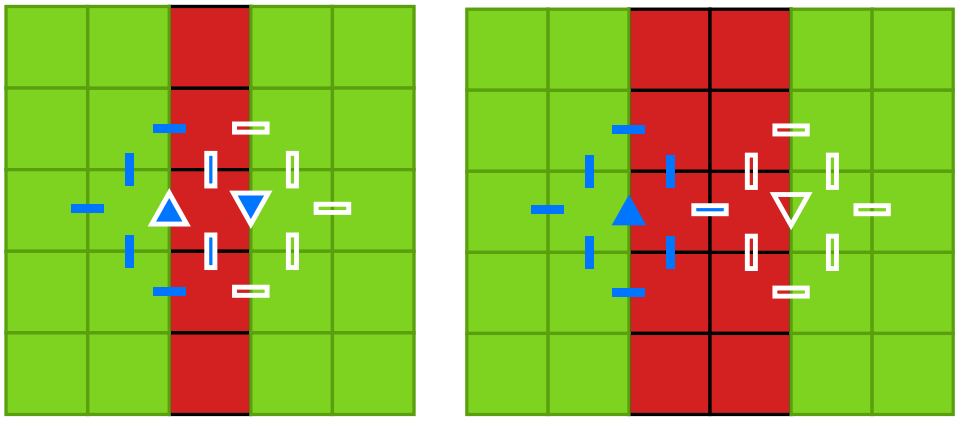
$$\mathbf{A}_{22} = \Delta t W_L^\tau \mathbf{D} W_F^u (W_L^u)^{-1} \mathbf{M}_C^{-1} \mathbf{D}^\intercal W_L^\tau + W_L^\tau \mathbf{D} \mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{J} \mathbf{D}^\intercal W_L^\tau + \frac{1}{2} \boldsymbol{\mu}^{-1} W_F^\tau W_L^\tau \tag{4.24}$$

$$\mathbf{B} = \frac{1}{\Delta t} \mathbf{M}_R + 2 \mathbf{J} \mathbf{D}^\intercal \boldsymbol{\mu} \mathbf{D} \mathbf{J}^\intercal \tag{4.25}$$

We reiterate, however, that in our implementation, we do not solve this form, nor the full form given on Equation 4.17; we solve solely the pressure-velocity form in Equation 4.19.

The pressure-stress formulation is appealing as it is guaranteed to be symmetric positive definite (SPD), and consequently allows for more efficient linear algebra techniques, notably preconditioned conjugate gradient. This system differs from the SPD system given in Larionov et al. [2017] only by the second term of each $\mathbf{A}$ block. The $\mathbf{B}$ matrix here has a block diagonal structure and mutually couples all pressure and viscous stress samples around each interior region's boundary. Assuming that viscous stencils of one interior region do not reach into a different interior region, i.e., the distance between any two regions is at least $2\Delta x$, then we can independently invert each 26×26 block, one for each interior region.

We once again point out that the same stencils used for a fully uniform case can be reused here. The second terms of each $\mathbf{A}$ block all have $\mathbf{J}^\intercal \mathbf{B}^{-1} \mathbf{J}$ in place of the $W_F^u (W_L^u)^{-1} \mathbf{M}_C^{-1}$ in the first term. This means that we can simply use the same iterations as before, replacing any faces that fall on the reduced regions with the scalar given



(a) (b)

Figure 4.4: Viscous stencil for two reduced region boundary faces from neighbouring regions with (a) one cell and (b) two cell padding. Stencil for the upright triangle is shown filled in blue, and stencil for the inverted triangle is shown in white outline. Regular grid cells are shown in red, and reduced regions in green. Reduced region faces are shown in green outline for clarity, but cells belonging to the same reduced region are represented by the same degrees of freedom.

by $\mathbf{J}^\intercal\mathbf{B}^{-1}\mathbf{J}$. Because we solve for pressure and stresses, however, we do end up needing to perform a subsequent solve using the known values of pressure and stresses to find the velocities. Since we have already computed $\mathbf{B}^{-1}$, this simplifies to basic matrix-vector arithmetic rather than a full linear solve.

4.4 Padding Sizes

We take a brief aside to discuss the size of padding required between neighbouring reduced regions. For their reduced model pressure projection, Goldade et al. constructed tiles with only one uniform cell padding in order to minimize the number of degrees of freedom constructed. This was sufficient for their application because the size of the divergence stencil (see Figure 3.6b) is small enough such that a single layer of pressure degrees of freedom is sufficient in preventing coupling between neighbouring reduced regions. Notice that the pressure system in Equation 3.73 has no differential operator acting between $\mathbf{v}_R$ and itself, which holds assuming these pressure degrees of freedom exist.

In comparison, the viscosity stencil representing the Laplace operator is significantly

larger, as shown on Figure 4.4. Notice that with only one-cell padding, stencils of reduced regions reach into neighbouring regions. This causes coupling between neighbouring regions, manifested as nonzero off-diagonal blocks in the $\mathbf{JVJ}^\intercal$ term of the linear system, Equation 4.19. This severely hampers performance, as it effectively locks the expressivity of neighbouring regions thus limiting their dynamics. This would also cause the SPD form, Equation 4.20, to be computationally unfriendly, as the $\mathbf{B}$ matrix is no longer block diagonal and consequently not easily invertible.

We have found that two-cell padding is sufficient for getting reasonable accuracy and spatial convergence, as shown with analytical tests in Sections 5.3 and 5.4. The shared velocity sample shown on Figure 4.4b is analogous to a shared pressure sample for reduced model pressure projection, thus making a choice of two-cell padding equivalent to Goldade et al.'s one-cell padding. We do, however, find that the iterative linear solve sometimes has difficulty converging to a result using two-cell padding for larger problems, suggesting that the shared velocity sample adds numerical stiffness. For these problems we find that three cells is generally sufficient.

4.5 Implementation Details

4.5.1 Constructing Reduced and Cartesian Regions

We construct the domains, separating the reduced region from the Cartesian region, strictly according to cell center labels. That is, we initially label all cell centers as either reduced or Cartesian, and assign faces as being reduced region faces if they are immediately adjacent to a reduced cell. We sketch out this labelling process on Figure 4.5.

Noting that we focus on applying only a narrow band of Cartesian regions only near the fluid boundaries, we employ a simple partial flood fill algorithm to construct a band of a given width. We perform this band construction separately for the free surface boundary and solid boundary to provide a separate user parameter for independently controlling the thickness of each boundary type.

As an initial setup, we first label the entire fluid domain to be solved as belonging to the reduced regions, shown on Figure 4.5a. This simply includes all cells adjacent to faces with nonzero volume weights. In a first pass, any cell adjacent to a cell that is not solved, i.e. it is entirely air (or later entirely solid), is added to a list of boundary cells. All these boundary cells are then labelled as Cartesian, shown on Figure 4.5b. In subsequent passes, any reduced cell adjacent to a Cartesian cell is added to the list and turned into Cartesian

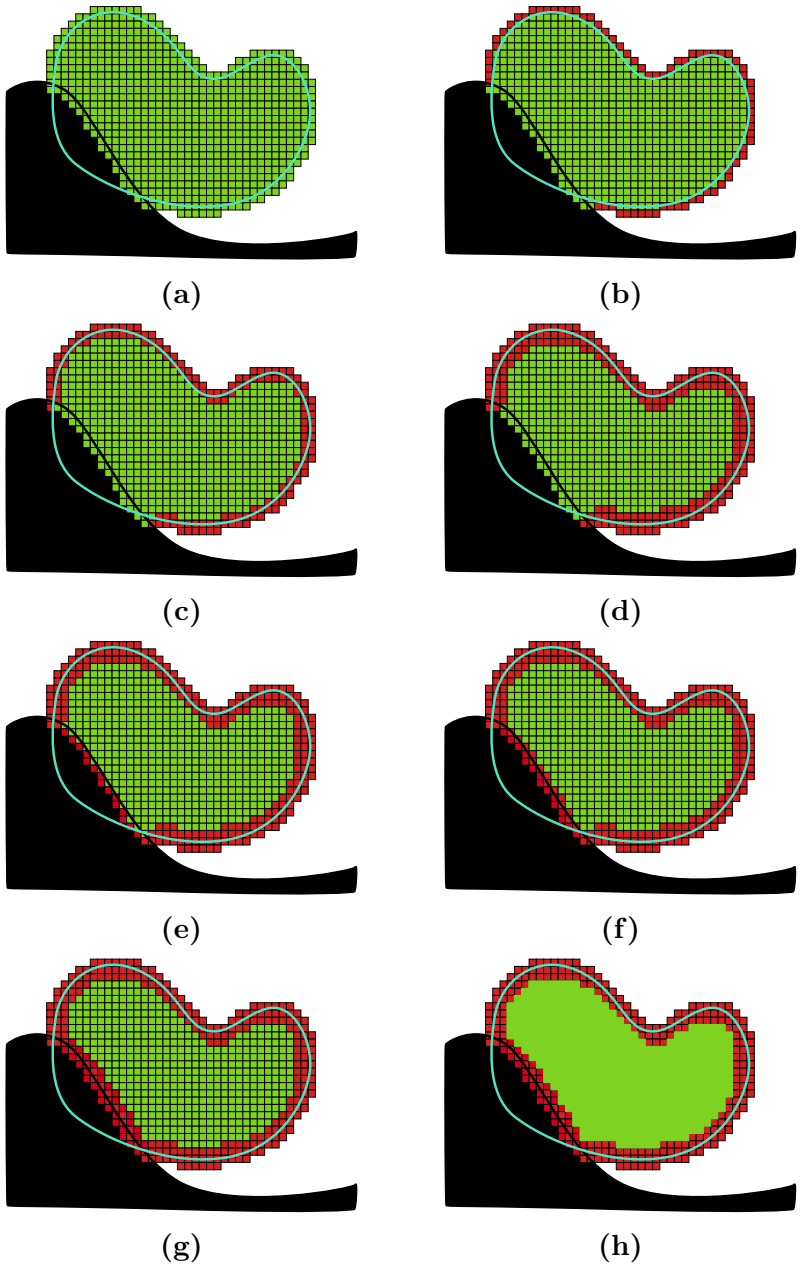
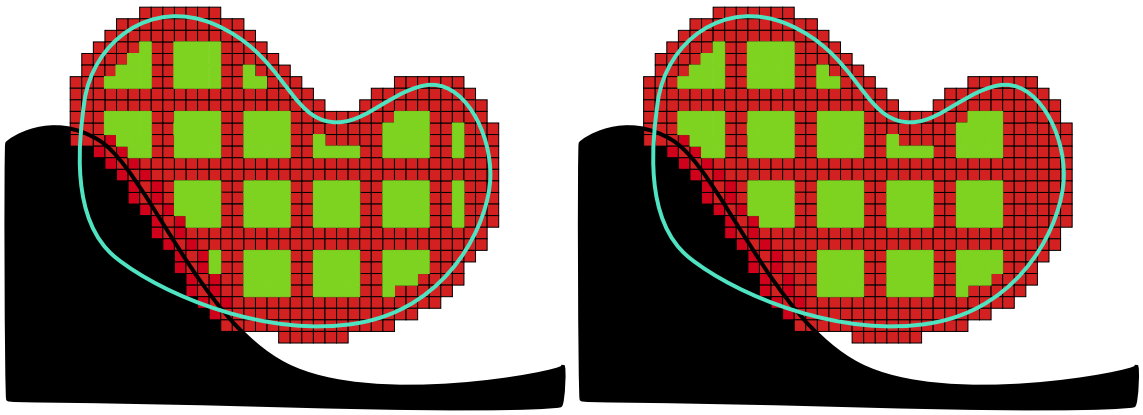


Figure 4.5: Process for constructing the uniform grid and reduced regions for a simple domain with both solid and free surface boundaries. Liquid domain is outlined in blue, solid domain is outlined and filled in black, and air domain is filled in white. Cells labelled for the uniform grid domain are shown in red, and those labelled for the reduced region are shown in green.



(a) **(b)**

Figure 4.6: Process for tiling the interior region for the same setup as Figure 4.5. These two steps directly continue after the labelling process in Section 4.5.1

cells. The number of these above passes determines the thickness of the surface band. Two such secondary passes are shown on Figures 4.5c and 4.5d. We repeat this above process for the solid boundary, doing an initial pass on Figure 4.5e and again two secondary passes on Figures 4.5f and 4.5g.

If we are satisfied with a single reduced region, we can assign individual indices to each cell in the uniform grid, and a single collective index to the reduced region, as shown on Figure 4.5h. Alternatively, we can construct a tile of interior regions for greater simulation flexibility, to be described in the next section.

4.5.2 Constructing Interior Tiles

To construct $n \times n$ sized interior tiles, we reassign every n^{th} interior cell in each axis back to being a Cartesian cell. To enlarge this padding to a required size m, we simply offset our counting by $[1, m]$ and repeat. In other words, we reassign any interior cell whose index satisfies,

$$\bigvee_{j=0}^{m-1} \left(\bigwedge_{a} a \equiv 0 \pmod{n+j} \right) \tag{4.26}$$

where a iterates through the axis indices (i, j, k). This tiling step is shown on Figure 4.6a. Recalling the tiling terminology outlined in Figure 3.8, this produces interior regions of

size $n - m$ and tile size n. We forewarn the reader that we use both terms in our results, generally defaulting to interior region size but using tile size whenever it is relevant to comparisons being made.

Once all tile boundaries are reassigned into Cartesian cells, we perform simple flood filling in each interior region to assign each region a unique index. Overlaying a grid in this manner may produce very small interior tiles near complicated surface geometry. We wish to avoid solving for these small interior regions as they are likely to be easier to compute using standard cells. To accomplish this, we construct the bounding box for each interior region and reassign back into Cartesian cells those regions whose bounding box size along each axis does not satisfy a minimum specified size. Figure 4.6b demonstrates such culling of interior regions using our default minimum size of two cells on each axis.

Upon completion of the interior tiles, all degrees of freedom for interior regions and Cartesian regions can thus be allocated. Each Cartesian face is given one degree of freedom, and each interior region is given the number of degrees of freedom corresponding to their reduced model.

4.5.3 Constructing the Polynomial Fit

Remembering that this reduced method is solely applied for the Stokes step, with all other components of the solver (mainly advection) being performed on the usual uniform grid, the above construction of reduced regions is necessarily performed at each timestep. This means that we initially have velocities defined on the uniform grid at the start of the Stokes step, which must be converted to generalized velocities wherever a reduced region is constructed. Likewise, we are required to convert generalized velocities back into uniform grid velocities at the end of the Stokes step. This conversion back is straightforwardly performed using the $\mathbf{C}(\mathbf{x})$ transformation matrix as stated by Equation 3.62. We thus turn our attention to the forward conversion into generalized velocities.

Because of the reduced degrees of freedom, we cannot exactly represent the fields being replaced. As such, we construct the generalized velocities for each reduced region using an ordinary least squares fit to the input uniform grid field. Since a generalized velocity $\mathbf{v}$ can be transformed into a Cartesian velocity $\mathbf{u}$ using the $\mathbf{C}(\mathbf{x})$ matrix via Equation 3.62, we aim to minimize the error between the actual velocity and its fitted velocity,

$$\arg\min_{\mathbf{v}_R^*} \sum_j \sum_a \left(u_{a,j} - \mathbf{C}_a(\mathbf{x}_j)\mathbf{v}_R^*\right)^2 \tag{4.27}$$

where a iterates through the three axes and j iterates through faces to be replaced. We thus solve,

$$\hat{\mathbf{v}}_R^* = \left(\sum_j \mathbf{C}_j^\intercal \mathbf{C}_j \right)^{-1} \left(\sum_j \mathbf{C}_j^\intercal \mathbf{u}_j \right) \tag{4.28}$$

where we use the shorthand $\mathbf{C}_j = \mathbf{C}(\mathbf{x}_j)$ and $\hat{\mathbf{v}}_R^*$ is our resulting best fit generalized velocities. This formulation differs slightly from the usual $(\mathbf{X}^\intercal \mathbf{X})^{-1}\mathbf{X}^\intercal$ least squares system where $\mathbf{X}$ contains a row for each sampling point, but nonetheless minimizes the required error as shown on Appendix D. This was done to save memory costs on requiring a row for each velocity sample, and consequently reduce the size of the matrix to be inverted. Equation 4.28 requires only a 26×26 matrix inverse for each interior region, which we perform via direct LU factorization.

The summations here collect transformations for velocities to be replaced by a given interior region. Note that although this looks similar to the definition of the matrix $\mathbf{J}$ given in Equation 3.67, j there iterates through all reduced boundary faces touched by the viscous stencil, while here we iterate through faces that we want to influence the fitted reduced model. This is because $\mathbf{J}$ accumulates forces from the regular grid and transforms them into generalized forces on the reduced regions—any faces whose stencils require these forces necessarily require an entry in the $\mathbf{J}$ matrix. For the least square system, we simply include any velocity that we want to fit as a data point.

That being said, there is a bit of freedom in defining the iterants for j here, and it is somewhat unclear what the correct choice is. One could either iterate through faces strictly along the the interior region boundaries, or iterate through all faces within the interior region domain. Taking the more mathematical perspective of the interior regions strictly acting as boundary conditions, the former would be the more sensible option, as the velocities occurring on the interior are irrelevant to the problem. On the other hand, we do end up translating all velocities back to a uniform grid after dynamics are solved for, which are then sent through to the next timestep. If the internal velocities are to be considered as being truly physical, then the second option is more valid. This second option is also more robust against potential degenerate cases, such as a velocity function that is zero exactly on the boundary but varies on the interior. Such a case would be captured as a constant velocity field if fitting only according to boundary velocities.

In our tests, we found no practical difference between the two, and settled on iterating solely the boundary faces. This does, however, bring up the notion that if interior regions are kept consistent between timesteps, save for the fluid's free surface and solid boundary

conditions cutting through them, then iterating through faces would be the clear option as we can safely throw away the interior data.

4.5.4 Warm Starting Pressure

The temporal inconsistency of the interior regions also brings us to question the benefits of warm starting pressure. Warm starting is the practice of using pressure values from the prior timestep as the initial guess when using an iterative solver. Because the reduced regions do not have valid pressure samples, as incompressibility is inherent to the reduced model itself, there is little reason to use warm starting in our method. This may actually cause harm as pressure values using our method differ completely from expected physical pressures—should the interior region boundaries move, this could result in an initial guess that is considerably far from the converged pressure values. Instead of warm starting, we simply opt to use an initial zero vector as the initial pressure guess.

4.5.5 Putting it All Together

This completes all necessary portions of our full reduced model Stokes algorithm:

Algorithm 2: One Stokes timestep, performed in place of Line 3 of Algorithm 1.

1: compute volume integration weights (Sec 3.3.2);
2: assign liquid cell labels (Sec 4.5.1);
3: build free surface boundary layers;
4: build solid boundary layers;
5: build padding layers (Sec 4.5.2);
6: build interior region connected labels;
7: remove small interior regions;
8: assign degrees of freedom;
9: compute interior region centers of mass;
10: build least squares polynomial fit (Sec 4.5.3);
11: build generalized mass matrix (Eq 3.66);
12: construct linear system (Eq 4.19);
13: solve linear system (BiCGSTAB);
14: update velocities from interior regions (Eq 3.62);

This algorithm is a drop-in replacement for the variational Stokes method of Larionov et al. [2017], and is performed as Line 3 in the overall fluid sim, Algorithm 1. This

is implemented in C++ as a plugin for Houdini 18.0.532 [Side Effects Software 2020]. Additionally, all the overhead in constructing the reduced and Cartesian domains (Lines 1-7 of Algorithm 2) is adopted from Goldade et al. [2020], with much of the code being directly ported from their implementation. The only significant additions to these steps are support for a user-specified padding thickness as given in Section 4.5.2 (the original implementation only constructed 1-cell padding), and separate uniform band construction for solid and free surface boundaries as given in Section 4.5.1. The Eigen library was used for matrix and vector structures, and to perform the BiCGSTAB linear solve [Guennebaud, Jacob, et al., 2010].

The above algorithm highlights the simplicity in implementation, being able to be injected into any uniform grid solver without worrying about new datatypes. This adaptability comes at the cost of requiring transfers back onto the uniform grid at each timestep. We later discuss potential improvements to this, at the cost of building a more monolithic solver.

Chapter 5

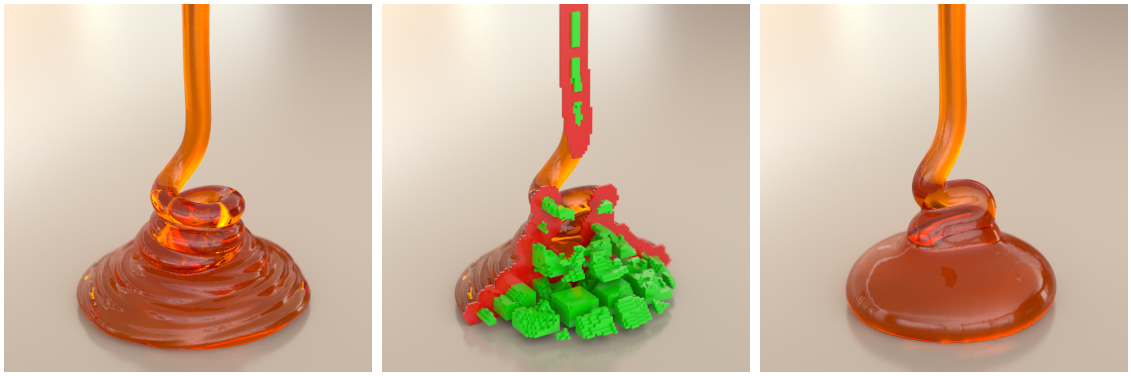
Results and Discussion

Here, we present results of using the new quadratic method for both unified Stokes and decoupled pressure-viscosity problems. We perform the 3D results given in Sections 5.1 and 5.2 on an 8-core R7 1700 CPU with 16GB of RAM. 2D analytical results given on Sections 5.3 and 5.4 are performed on a 4-core i7-7700HQ CPU with 8GB of RAM.

5.1 Honey Coil

We demonstrate that our quadratic reduced method retains the desirable free surface accuracy of a fully uniform Stokes approach by replicating the liquid rope coiling instability, as shown on Figure 5.1. We also present here the result of using our quadratic reduced method for solely the viscous step of the decoupled approach, shown on Figure 5.1c, which uses the system provided on Equation 4.7. On top of the desired cylindrical coil, the Stokes system is able to retain more surface detail relative to the decoupled method, which is consistent with results found by Larionov et al. [2017].

We compare the results of using affine regions to quadratic regions on Figure 5.2. The affine method forms a more erratic coil, especially at the beginning, before gradually settling to a stable coil. By comparison, the quadratic method forms a near perfect cylinder from the start of the simulation. The difference in behaviour is consistent with earlier observations in Section 4.1, which suggested that the affine model has convergence problems for the viscosity equation. It is reasonable to expect that such difficulty extends into the Stokes problem, which contains the same viscous term.



(a) (b) (c)

Figure 5.1: Liquid rope coiling instability as simulated using our (a,b) unified Stokes solver and (c) decoupled pressure-viscosity solver, both using quadratic interior regions. A cutaway view of the unified Stokes solver is shown in (b), with uniform grid cells in red and interior regions in green. Interior regions are capped at 12^3 with 4-cell padding between.

We point out the flexibility of the interior regions, having placed no restrictions on their shape. In particular, these interior regions need not be convex, or even simply connected; examples of such highly irregular regions are shown on Figure 5.3. We did not notice any degradation in our results from having seemingly pathological interior regions. Any irregularities in the shape of interior regions are naturally accounted for by the $\mathbf{C}$ mapping matrix, which is consolidated into the generalized mass matrix, $\mathbf{M}$, and transfer operator, $\mathbf{J}$.

Because of the thin feature size of the liquid coil, the size of reduced regions is greatly limited, resulting in long aspect ratio regions. Since these regions do end up requiring the regular Cartesian grid on the surface, this example only achieves minor runtime benefits: the BiCGSTAB matrix-vector solve takes 2h:29m:20.7s with the quadratic interior regions and 3h:01m:50.0s with a fully uniform grid, which corresponds to only a 17.9% reduction relative to the full solve. The extra steps required to construct the reduced regions and solve for the least squares fits decreases this runtime benefit. The full simulation (including all steps external to the Stokes solve such as advection) takes 3h:48m:38.5s for the quadratic model and 4h:07m:43.0s for the fully uniform grid, corresponding to only a 7.7% speedup.

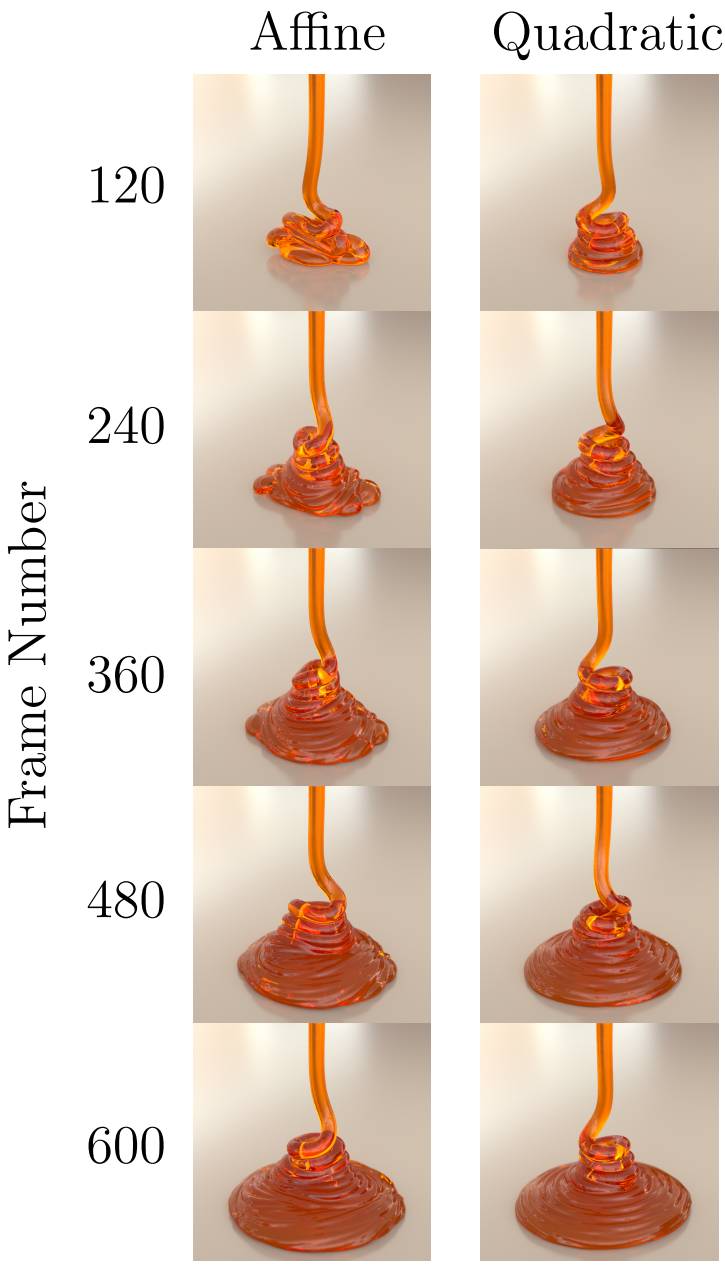
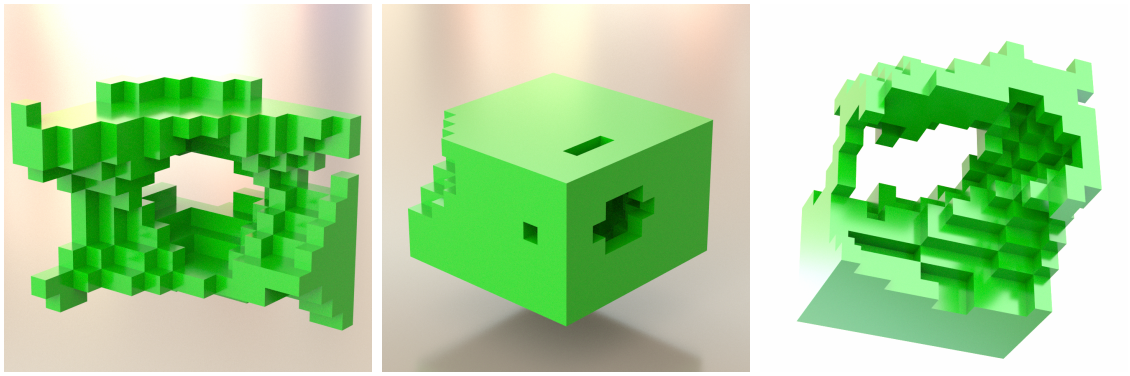


Figure 5.2: Sequence of frames showing the liquid coil instability simulated using (left) an affine reduced model and (right) a quadratic reduced model.



(a) (b) (c)

Figure 5.3: Sample non-simply connected reduced regions from the liquid rope coil simulation.

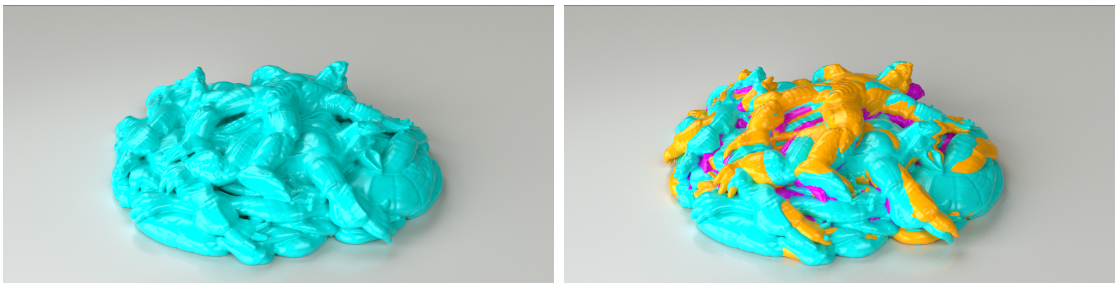
5.2 Armadillo Drop

For a computationally friendlier example with more bulk geometry, we use the piling armadillos test from Larionov et al. [2017], which drops nine copies of randomly oriented Stanford armadillos. Simulated results using the affine, quadratic, and fully uniform models are shown on Figure 5.4, and runtime measurements are shown on Figure 5.5.

Both affine and quadratic regions have qualitatively similar results to the fully uniform reference solution, particularly in retaining surface detail, but achieved this outcome with significantly lower cost. The affine method in particular is nearly twice as fast as the fully uniform method. While the linear solve time for the quadratic method is about two-thirds the cost relative to the fully uniform case, some of that runtime improvement is again offset by the overhead cost of constructing the quadratic system. This likely comes from the least-squares fit, which requires a 26×26 direct matrix solve for each interior region. The overhead for the affine method, which has a much smaller 11×11 system, is nearly identical to the fully uniform case.

A potential method for accelerating this least squares construction would be to precompute the matrix inverse for regular cube regions, noting that it depends only on the face positions relative to the center of mass and not the input velocity data, as shown in Equation 4.28. Prefactoring could similarly be applied to the SPD form of the linear solve. Here, the $\mathbf{B}$ block defined in Equation 4.25 likewise only depends on the geometry of the reduced region. We conjecture, however, that because most tiles in our tests are irregular, the performance benefits from this would be negligible. In cases where there is a large

(a) Fully Uniform **(b)** Affine



(c) Quadratic **(d)** Overlay

Figure 5.4: Nine randomly oriented viscous armadillos dropped in a pile. (a) uses a uniform solver, (b) uses tiled affine regions, and (c) uses tiled quadratic regions. Both (b) and (c) use a maximum interior region size of 28^3, with 4-cell padding. Armadillos are made of a homogeneous fluid with $\mu = 500$ and $\rho = 1000$.

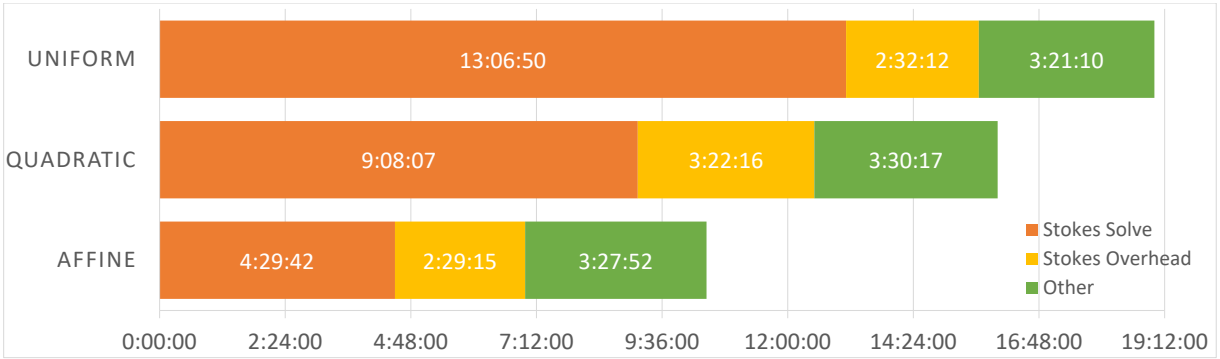


Figure 5.5: Wallclock runtime measurements for the armadillo drop example in Section 5.2. The orange region indicates solely the linear system solve, the yellow region indicates the overhead required in creating the Stokes system. The green region indicates everything outside of the Stokes solver, such as APIC advection.

number of regular tiles, it is likely that greater performance benefits can be achieved by simply using larger tiles. Alternatively, it may also be possible to build a small library of precomputed tiles, and attempt to find the best fit within a given irregular region, but we leave this to future exploration.

Because the velocity field is fairly simple in this example, consisting mostly of the free fall translational velocity, the affine and quadratic methods are very similar in result. Both, however, deviate slightly from the uniform method at the periphery of the armadillo models, i.e. the hands, feet, and snout. We note that the reduced region sizes that we used were fairly generous—about half the height of an armadillo—resulting in entire limbs being represented by single regions. This is problematic given that the bulk fluid is mostly static; it is easy to see how an interior region that ends up connecting the bulk fluid with a limb would quickly dissipate the limb's ongoing momentum. The errors arising from this are still reasonably small for this example, seeing as the affine and quadratic methods both look physically plausible, if only lacking some higher order velocity modes. These differences, however, are instructive in determining regions requiring more accuracy, such as for refining the tile sizes, and may also be the topic of future exploration.

5.3 Analytical Viscosity Test Problem

To complement the evaluations of computational cost and qualitative appearances given above, we now consider quantitative evaluations against exact analytical solutions in simple geometries.

We examine spatial convergence of a 2D analytical test case for solely the viscous step, Equation 3.20, using a fluid-filled $\pi \times \pi$ box with Dirichlet boundary conditions on velocity. The fluid is homogeneous with $\rho = 1$, $\mu = 1$, and an initial velocity of,

$$\mathbf{u}_{initial} = \begin{bmatrix} (1 + 3\Delta t)(\sin(x)\sin(y)) - \Delta t(\cos(x)\cos(y)) \\ (1 + 3\Delta t)(\sin(x)\sin(y)) - \Delta t(\cos(x)\cos(y)) \end{bmatrix} \tag{5.1}$$

which gives a final velocity field:

$$\mathbf{u}_{final} = \begin{bmatrix} \sin(x)\sin(y) \\ \sin(x)\sin(y) \end{bmatrix} \tag{5.2}$$

Taking $\Delta t = 1$, we solve this problem numerically using a single timestep of our reduced viscosity-only solver, as outlined in Section 4.1, with both affine and quadratic reduced models.

Table 5.1: L_1 errors for a viscosity-only step applied to the analytical test case given on Section 5.3. Both tables use tiled interior regions with (a) using the affine model and (b) using the quadratic model. Grey cells indicate tile and padding sizes too coarse for the given resolution.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	4.982E+00	2.307E+00	9.869E-01	3.309E-01	9.653E-02
32 / 4		5.162E+00	2.438E+00	1.058E+00	3.558E-01
64 / 8			5.241E+00	2.510E+00	1.098E+00
128 / 16				5.278E+00	2.548E+00

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	1.748E+00	3.799E-01	4.906E-02	5.839E-03	6.952E-04
32 / 4		1.975E+00	4.036E-01	5.171E-02	5.970E-03
64 / 8			2.084E+00	4.159E-01	5.169E-02
128 / 16				2.137E+00	4.217E-01

(b) Quadratic

We compute L_1 and L_∞ velocity errors according to,

$$L_1 = \sum_a \sum_j |u_{exact,a}(\mathbf{x}_{a,j}) - u_{numerical,a,j}| \, (\Delta x)^2 \tag{5.3}$$

$$L_\infty = \max_a \max_j |u_{exact,a}(\mathbf{x}_{a,j}) - u_{numerical,a,j}| \tag{5.4}$$

where Δx is the grid spacing, a iterates through the two axes, j iterates through all faces within an axis, $\mathbf{x}_{a,j}$ is the location of a staggered grid face, $u_{exact,a}$ is the a-axis of the true velocity field, and $u_{numerical,a,j}$ is the discrete velocity sample located at the a-axis j face.

L_1 errors are shown on Table 5.1, with L_∞ and reconstruction errors shown in Tables E.1 and E.2 respectively, both in Appendix E. The reconstruction error represents the error in the least squares fit for the affine and quadratic models relative to the initial condition, prior to any dynamics. This is slightly different from the backwards error since it is the difference between the exact input and an approximate input into the approximate discretized system, rather than the approximate input into the exact system.

We immediately see that the quadratic model is more accurate than the affine model in

Table 5.2: Ratio between result error and reconstruction error at the location of the L_∞ error for the analytical test case given on Section 5.3. This may be larger than simply dividing L_∞ values from Table E.1 by reconstruction errors taken from Table E.2 since the location of the L_∞ error may have a smaller reconstruction error.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	3.012E-01	5.382E-01	6.463E-01	7.722E-01	8.496E-01
32 / 4		2.978E-01	5.697E-01	6.840E-01	7.971E-01
64 / 8			2.959E-01	5.775E-01	6.899E-01
128 / 16				2.949E-01	5.818E-01

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	2.419E-01	2.746E-01	2.275E-01	2.530E-01	2.978E-01
32 / 4		2.429E-01	2.778E-01	2.272E-01	2.561E-01
64 / 8			2.432E-01	2.788E-01	2.360E-01
128 / 16				2.434E-01	2.803E-01

(b) Quadratic

all respects, typically having more than an order of magnitude less error. For both models, error is largely dependent on the physical size of the reduced regions; having smaller tiles means the reduced model is better able to capture the local field. Taken in the opposite perspective, having more tiles compensates for the lack of high-frequency modes in the affine and quadratic models.

The tile and padding sizes were chosen such that each diagonal represents exactly the same physical domain setup. The main diagonal has one full-size reduced region, the first upper diagonal has a 2×2 grid of reduced regions, the second has 4×4, and so on. These reduced regions and the padding in between are exactly the same physical size within each diagonal regardless of resolution. We note that along each diagonal, error is relatively constant. This makes sense as regardless of how much the interior is refined, a reduced region occupying the same physical space will still evolve in the same manner.

Perhaps more notable is how the error of the result compares to the reconstruction error. The ratio between the two is shown on Table 5.2. Interestingly, the quadratic model keeps a roughly constant ratio, while the affine model has a clear increase in the ratio as the physical tile size decreases. Exploring this further, we plot the results error against the

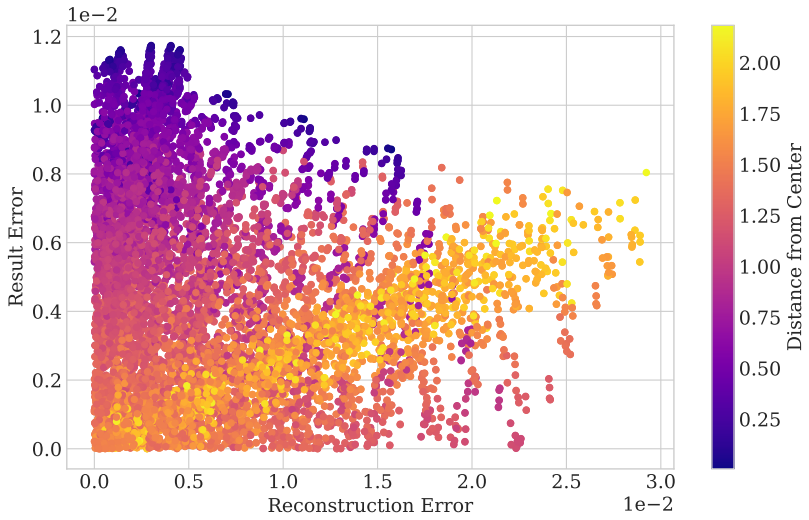
reconstruction error at each reduced boundary velocity sample, shown on Figure 5.6. Here, we see that error for the quadratic model generally follows the same linear trend throughout the domain, albeit with an increasing error offset towards the center of the domain. From this, it is clear that the largest errors will typically be the result of large reconstruction errors, reinforcing the prior observation of nearly constant error ratios independent of domain setup. Taken together, this means that the way the quadratic model performs dynamics is a good representation of the expected fluid behaviour.

In comparison, the affine model follows the same linear trend only near the edges of the domain. Elsewhere, the result error shows no apparent relation to the reconstruction error, instead being correlated to the distance from the center. We note that for this problem, the center has the greatest viscous forces. This plot shows that as viscous force increases, the affine model breaks down, once again pointing to the missing $\mathbf{D}^\intercal\mathbf{D}$ term mentioned in Section 4.1. Further, because the error is strongly dependent on the problem's viscous force, large errors are no longer a result of large reconstruction errors—the affine fit could be accurate, but the dynamics it imposes are not. This also means that even if the affine tiles are decreased in size, and consequently become more accurate fits of the input velocity field, the output velocities they produce cannot be trusted to improve in accuracy, resulting in the increase in output error to input error ratio shown in Table 5.2.

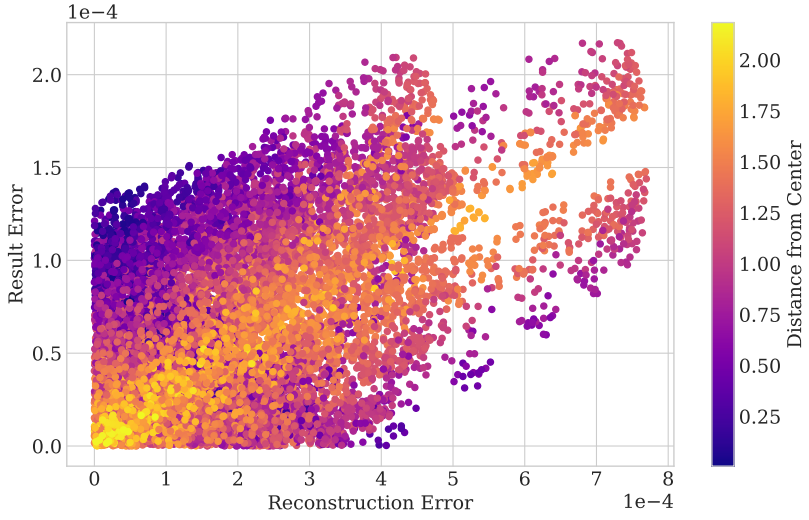
We repeat the above test using a constant padding size, with L_1 errors shown on Table 5.3, and L_∞ and reconstruction errors shown on Tables E.3 and E.4 respectively. These results show a clear increase in error as tile size increases, suggesting that the system is quite sensitive to the amount of padding. Remembering that the reduced regions couple all velocity samples along their boundary, increasing the number of coupled degrees of freedom while keeping the padding the same results in less ability to resolve conflicting boundaries. Cases where the quadratic fits of neighbouring tiles have errors with opposite signs are common, requiring considerable padding sizes to resolve.

Additionally, while padding sizes were *physically* constant in Table 5.1, the constant padding cell sizes of Table 5.3 means that their physical sizes shrink considerably for higher resolutions. The difference between the lowest and highest resolution results in a factor of eight decrease in physical size. While the error in fitted velocities does not change, this does result in an increase in the derivative's error, as the fitted velocities are effectively pushed closer together. Due to the Laplace operator, this derivative error carries through to influence the result.

Plotting convergence trends, as shown on Figure 5.7, gives further evidence of the affine model's insufficient dynamics. Because this problem has exact axis-aligned boundary conditions, the variational approach simplifies to second-order finite differences, with all



(a) Affine



(b) Quadratic

Figure 5.6: Result error vs reconstruction error of each reduced region boundary velocity sample for the analytical viscosity-only step. (a) uses the affine model and (b) uses the quadratic model. The color bar indicates the distance to the center of the problem, $(\pi/2, \pi/2)$. Both methods use a 256×256 resolution, with 16-cell tile sizes and 2-cell padding.

Table 5.3: L_1 errors for a viscosity-only step applied to the analytical test case given on Section 5.3 with constant padding cell size. Both tables use tiled interior regions with (a) using the affine model and (b) using the quadratic model. Grey cells indicate tile and padding sizes too coarse for the given resolution.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	4.982E+00	2.307E+00	9.869E-01	3.309E-01	9.653E-02
32 / 2		7.106E+00	3.731E+00	1.931E+00	7.215E-01
64 / 2			7.787E+00	5.041E+00	3.178E+00
128 / 2				7.963E+00	6.137E+00

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	1.748E+00	3.799E-01	4.906E-02	5.839E-03	1.103E-03
32 / 2		4.433E+00	6.541E-01	7.708E-02	7.944E-03
64 / 2			6.394E+00	9.112E-01	1.026E-01
128 / 2				7.413E+00	1.176E+00

(b) Quadratic

volume weights in Equation 4.7 becoming identities. We see that the fully uniform case has almost exactly second-order convergence, with an overall convergence fit of 2.01. The quadratic model has better than second-order convergence, having an overall convergence fit of 2.66. This is because of the error reduction resulting from both the increase in resolution as well as the decrease in physical reduced region sizes. In effect, the method would be second order, but has a high amount of reconstruction error at low resolutions due to the large physical reduced region sizes that also gets reduced on refinement.

In comparison, the affine model has less than second-order convergence, with a convergence rate of 1.11 at the coarsest resolution, which gradually increases to 1.78 at the highest resolution. We see that the affine model's dynamics actively hampers convergence. At low resolutions, errors from the affine dynamics are significant, resulting in very poor convergence. As resolution increases, these errors gradually reduce as the local flow represented by each region becomes closer to being affine. At the highest resolutions, despite each tile having incorrect dynamics, the large number of tiles produce enough degrees of freedom to approach second-order convergence. In effect, they behave as a very coarse version of a uniform grid.

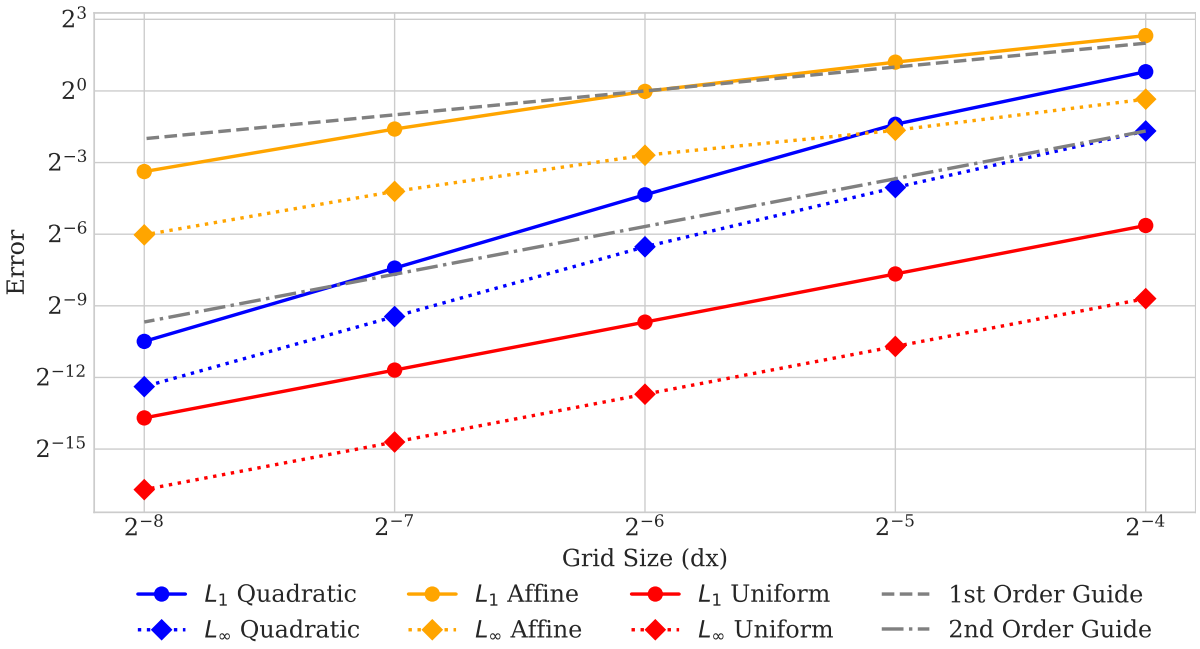


Figure 5.7: Log-log plot of error against grid size for the viscosity-only analytical test case given in Section 5.3. 16^2 tiles are used with 2-cell padding size. Reduced model L_1 and L_∞ errors are as given in the first rows of Tables 5.1 and E.1 respectively.

5.4 Analytical Free Surface Stokes Test Problem

We use a 2D analytical test case provided in Batty and Bridson [2010], consisting of a fluid disk of radius $r = 0.75$, $\rho = 1$, $\mu = 0.1$ evolved over $\Delta t = 1$. The final velocity field is defined using its streamfunction,

$$\Psi = \frac{128}{81} r^4 \cos(2\theta) \cos(\sqrt{3} \ln r)(15 - 30r + 16r^2) \tag{5.5}$$

with r as the distance from the disk's center and θ is the angle as measured from the $(1, 0)$ vector. Velocity is given by the curl of the streamfunction, $\mathbf{u} = \nabla \times \Psi$. The initial condition can be found by analytically evolving the final velocity backwards. Notice that the varying velocity on the fluid surface results in a viscous stress which necessarily must

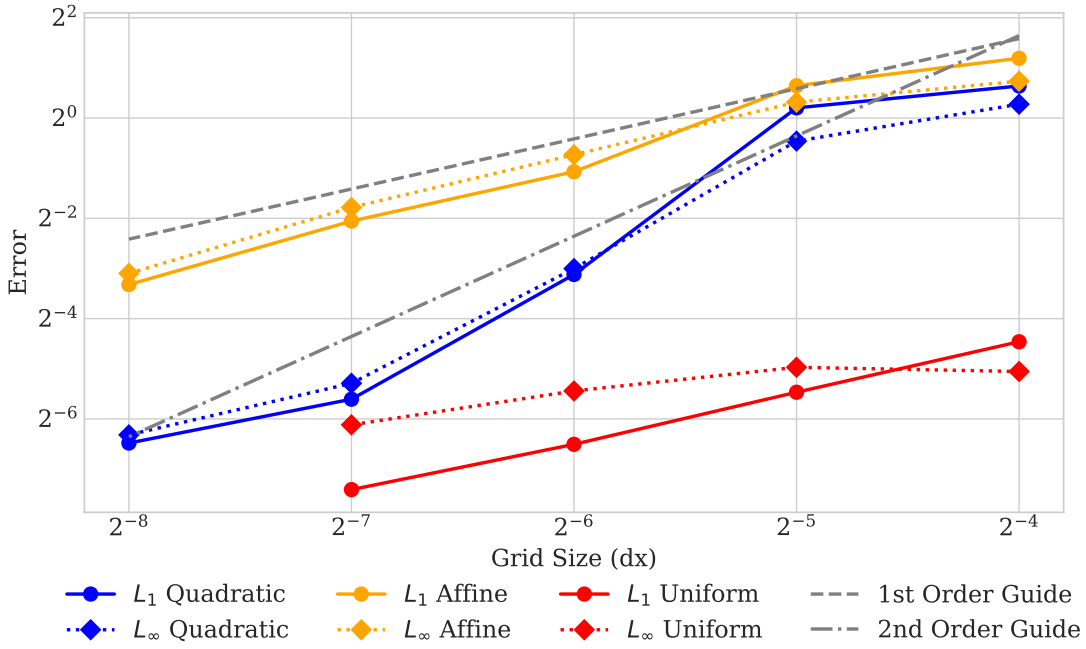


Figure 5.8: Log-log plot of error against grid size on an analytical test case solved using unified Stokes. Solid lines indicate L_1 errors and dotted lines indicate L_∞ errors. Red, yellow, and blue lines indicate uniform grids with no interior regions, affine interior regions, and quadratic interior regions respectively. 16^2 tiles are used with 3-cell padding size.

be balanced by the surface pressure. Pressure throughout the fluid is given by:

$$p = \frac{512\sqrt{3}}{81} r^2 \mu \sin(2\theta) \sin(\sqrt{3} \ln r)(15 - 30r + 16r^2) \tag{5.6}$$

Error plots for our Stokes method using affine interior regions, quadratic interior regions, and a reference uniform solver are shown on Figures 5.8 and 5.9, with the former plotting error against grid size and the latter plotting against runtime. Exact error values are given on the table in Appendix F. Timing results of each method are given on Table 5.4.

The quadratic model consistently outperforms the affine model, achieving lower errors on the same grid resolution. While the affine method expectedly has less computational cost than the quadratic method at the same grid resolution, the affine method's error tends

Table 5.4: Timing comparison for a fluid disk with known analytical solution. All numbers are wallclock times given in seconds.

	Affine	Quadratic	Uniform
1/16	0.696009	0.867416	0.685828
1/32	3.53121	4.40174	7.44412
1/64	18.8396	33.6367	101.695
1/128	184.232	241.191	2740.18
1/256	1703.46	4469.21	

to be so large such that the quadratic method is more accurate for similar runtimes. Both methods are shown to be considerably faster than a uniform grid of the same resolution. While the uniform grid does have lower error, even with similar runtimes, the resolution difference between it and the reduced models is quite stark. At high resolutions, both reduced models achieve approximately double the resolution at the same runtimes.

Interestingly, the quadratic model has an apparent second-order convergence. Because this problem has an irregular free surface boundary condition, we expect the variational finite difference scheme to have first-order spatial convergence [Larionov et al., 2017], which is what we see with the uniform method. Once again, we associate the increased convergence rate of the quadratic model to error reduction due to the physical shrinking of the reduced regions. As the reduced regions shrink, the local flow becomes sufficiently quadratic such that shrinking them further does not provide much increase in representation accuracy. In this regime, the secondary error source disappears and errors from the variational handling of the free surface dominates, causing the quadratic model to behave more like the uniform model with first-order accuracy. We see this at the highest resolution plotted in Figure 5.8.

We show spatial error plots of pressures and velocities on Figures 5.10 and 5.11 respectively. Errors in pressure are prevalent in the 2-cell padding result, but seem to be dissipated for larger padding sizes. These errors naturally occur on the surface of each interior region, and is indicative of the attempt to match the reduced model with the original fluid. The increased error for smaller padding sizes reinforces our claim that these setups are dominated by the "compatibility" between neighbouring reduced regions. When padding sizes are too small, there is not enough room to resolve conflicting errors from neighbouring regions.

We should point out that although we speak of pressure errors here, the pressure results from our method should not be taken to represent physical pressure in practice, even on the usual Cartesian regions. Pressure is entirely meaningless in the reduced model setting,

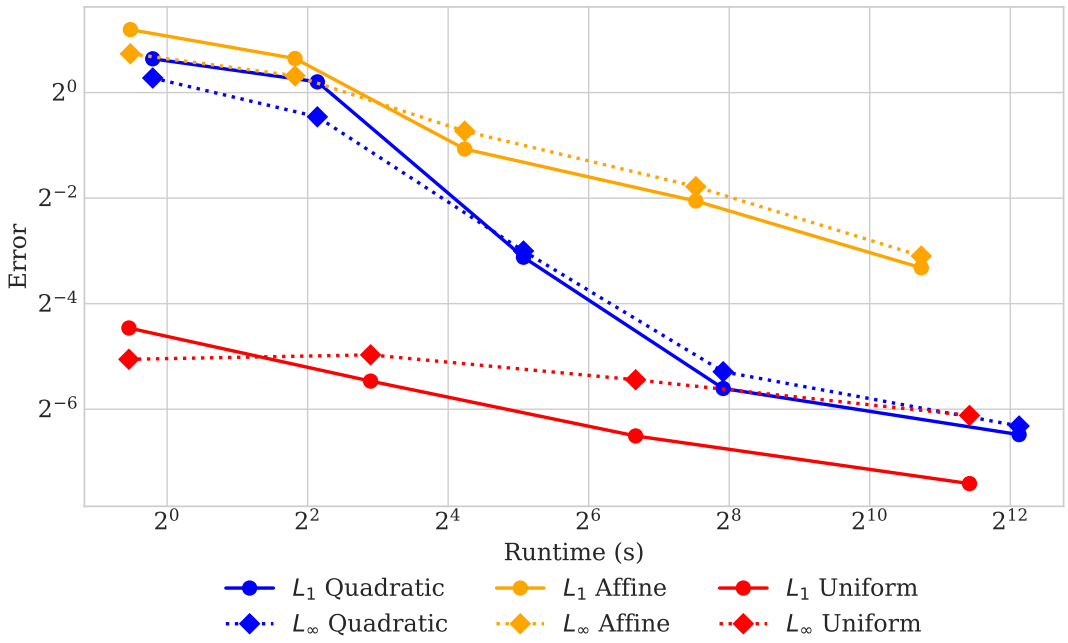
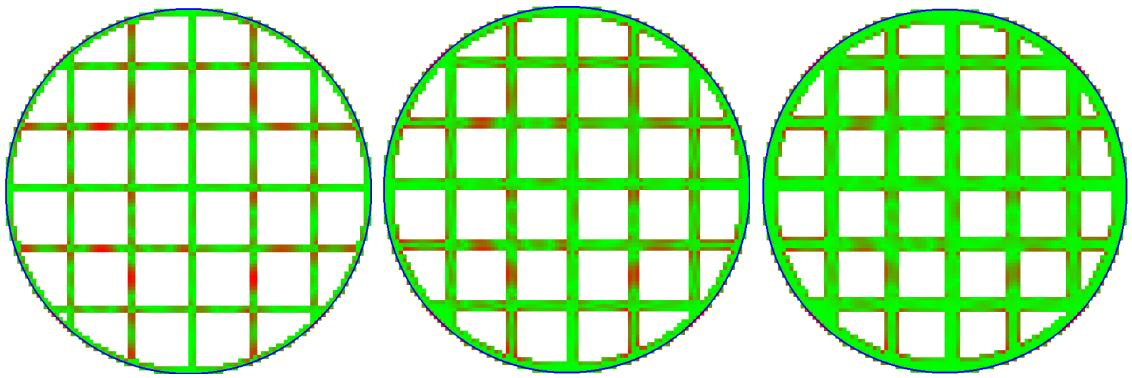


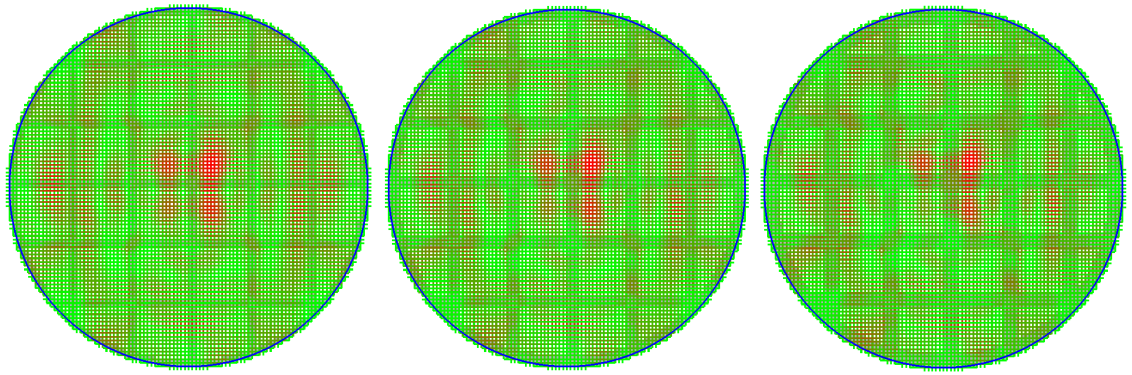
Figure 5.9: Log-log plot of error against runtime on an analytical test case solved using unified Stokes. Solid lines indicate L_1 errors and dotted lines indicate L_∞ errors. Red, yellow, and blue lines indicate uniform grids with no interior regions, affine interior regions, and quadratic interior regions respectively. 16^2 tiles are used with 3-cell padding size.

once again noting that incompressibility is enforced by the definition of the reduced model. Since pressure in the Cartesian setting acts as the Lagrange multiplier for enforcing the divergence-free constraint, by creating regions without defined pressures, we effectively defer errors to the boundary regions where the reduced model meets the original fluid. This is also why warm starting using pressures from the last timestep may not necessarily be useful, especially if the boundary regions are not consistent between timesteps.



(a) (b) (c)

Figure 5.10: Pressure spatial error plots for the analytical test case given in Section 5.4. Normalized error for each subfigure is plotted from green (lowest) to red (highest), and is only shown for uniform grid cells. The simulation resolution is $dx = 1/64$ in all three cases, with a tile size of 16^3. (a), (b), and (c) show results for 2-, 3-, and 4-cell padding sizes respectively. The black circular outline represents the free surface boundary condition.



(a) (b) (c)

Figure 5.11: Velocity spatial error plots for the analytical test case given in Section 5.4. Normalized error for each subfigure is plotted from green (lowest) to red (highest). A light grey background shows the positioning of uniform grid cells. The simulation resolution is $dx = 1/64$ in all three cases, with a tile size of 16^3. (a), (b), and (c) show results for 2-, 3-, and 4-cell padding sizes respectively. The black circular outline represents the free surface boundary condition.

Chapter 6

Conclusions and Future Work

We have demonstrated that the reduced model provides qualitatively similar results with faster runtimes. While the quadratic model is more computationally expensive than the affine model, it is better able to capture viscous forces, with errors that reliably reduce under refinement. The affine model is, however, shown to be sufficient in simple flows represented by small enough interior regions, as demonstrated by the armadillo drop test. Both methods thus allow for physically plausible simulation of Stokes and Navier-Stokes flow at higher resolutions than fully uniform solvers.

We note that runtime improvement highly differs between problems; simpler velocity fields with more bulk fluid are amenable to larger tiles, providing greater runtime improvements. Simulations with small feature sizes and large changes in the velocity field, such as the liquid rope coil, only result in marginal performance benefits which may be offset by the increased overhead in constructing the interior regions.

We also claim, however, that a significant issue in attempting to achieve greater performance is the difficulty in choosing parameters—interior region size, padding size, and error tolerance—for any given problem. The system is highly sensitive to the size of the padding between interior regions; smaller padding sizes seem to result in systems dominated by compatibility error between neighbouring regions, but larger padding sizes introduce more degrees of freedom and hence higher computational cost. Likewise, larger interior regions couple more boundary samples, increasing stiffness in exchange for the reduction in internal degrees of freedom. Physically larger reduced regions also result in larger representation error, with larger domains being more likely to contain higher order modes not captured by the reduced models.

Despite these issues, we believe that the quadratic reduced model for the Stokes problem

holds promise, particularly the SPD form given in Equation 4.20. While an SPD form is not a panacea, we nonetheless note the desirable property of being able to use the more efficient conjugate gradient method for the linear solve. In addition, a strictly pressure-stress form has the added benefit of providing a consistent metric for error tolerance in choosing when to halt the iterative linear solver. Because the variables we chose to solve for on Equation 4.19 include $\mathbf{v}_R$, whose entries wholly depend on the reduced model being used, error tolerances are not comparable between models. For example, the quadratic model contains velocity entries modulated by $\tilde{x}^2$, meaning that adopting a tolerance of 0.1 from the uniform case could produce velocities that deviate by as much as $0.1\tilde{x}^2$. Using a pressure-stress form mitigates this issue, setting a problem domain defined using the same variables as the uniform grid method, allowing for more fair comparisons to be drawn.

Our results have also shown the potential value of pursuing an adaptivity method for the interior regions. While our reduced method does have spatial adaptivity in the sense of using a uniform grid on the surface and the reduced model on a tiled interior, further adaptivity that controls the size of interior regions could improve runtime and accuracy. A single global tile size was shown to be problematic for the armadillo drop test, which had a simple bulk fluid but transient smaller scale flows. Tile-level adaptivity modulated by the complexity of the velocity field would avoid such issues. Because we use a least squares fit, we already have a practical metric for measuring how well the local flow is approximated. A straightforward method for achieving adaptivity would simply be to start coarse and refine any interior region with large least squares residuals. This also provides a potential framework for p-adaptivity, in which different tiles could use different degree polynomials. Before refining spatially, one could first check if fitting a higher order model is sufficient. Of course, much of this is contingent on whether the increased cost of overhead is outweighed by the performance improvement, which we suspect may be true only for very high resolution problems.

Temporally consistent tiles may also provide accuracy improvements, as well as allowing use of warm starting to reduce iterative solver runtimes. This also leads directly into a more monolithic solver that forgoes transferring reduced model regions back onto the regular grid. Keeping interior regions between timesteps, only updating them near surfaces where they change, may improve the overhead cost of our method. This line of research is significantly more involved, requiring new methods particularly with regards to handling advection within the reduced regions.

References

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2015. A dimension-reduced pressure solver for liquid simulations. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 473–480.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 100–es.

Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids.. In *Symposium on Computer Animation*. 219–228.

Christopher Batty and Robert Bridson. 2010. A Variational Finite Difference Method for Time-Dependent Stokes Flow on Irregular Domains. *CoRR* (2010).

Christopher Batty, Stefan Xenos, and Ben Houston. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 695–704.

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete viscous threads. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–10.

Jeremiah U Brackbill and Hans M Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational physics* 65, 2 (1986), 314–343.

Robert Bridson. 2015. *Fluid simulation for computer graphics*. CRC press.

Mark Carlson, Peter J Mucha, R Brooks Van Horn III, and Greg Turk. 2002. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 167–174.

Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 219–228.

Alexandre Joel Chorin. 1967. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Amer. Math. Soc.* 73, 6 (1967), 928–931.

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable laplacian eigenfluids. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph.* 34, 4 (2015), 50–1.

Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid simulation using Laplacian eigenfunctions. *ACM Transactions on Graphics (TOG)* 31, 1 (2012), 1–11.

Essex Edwards and Robert Bridson. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–9.

Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics* 183, 1 (2002), 83–116.

Henrik Fält and Douglas Roble. 2003. Fluids with extreme viscosity. In *ACM SIGGRAPH 2003 Sketches & Applications*. 1–1.

Bryan E Feldman, James F O'brien, and Bryan M Klingner. 2005. Animating gases with hybrid meshes. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 904–909.

Nick Foster and Dimitri Metaxas. 1996. Realistic animation of liquids. *Graphical models and image processing* 58, 5 (1996), 471–483.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.

Frederic Gibou, Ronald P Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.* 176, 1 (2002), 205–227.

Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. 2020. Constraint bubbles and affine regions: reduced fluid models for efficient immersed bubbles and flexible spatial coarsening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 43–1.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Francis H Harlow. 1962. *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Technical Report. Los Alamos Scientific Lab., N. Mex.

Francis H Harlow and J Eddie Welch. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids* 8, 12 (1965), 2182–2189.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Bryan M Klingner, Bryan E Feldman, Nuttapong Chentanez, and James F O'brien. 2006. Fluid animation with dynamic meshes. In *ACM SIGGRAPH 2006 Papers*. 820–825.

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational stokes: a unified pressure-viscosity solver for accurate viscous liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987), 163–169.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*. 457–462.

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 812–819.

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

Neil M Ribe, Mehdi Habibi, and Daniel Bonn. 2012. Liquid rope coiling. *Annual review of fluid mechanics* 44 (2012), 249–266.

Side Effects Software. 2020. Houdini 18.0.532.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer physics communications* 87, 1-2 (1995), 236–252.

Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 826–834.

Martin Wicke, Matt Stanton, and Adrien Treuille. 2009. Modular bases for fluid dynamics. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.

G Wyvill, C McPheeters, and B Wyvill. 1986. Data Structure for Soft Objects The Visual Computer.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

APPENDICES

Appendix A

Reduced Model Viscosity Proof

Here follows a proof that the objective function given in Equation 4.2 solves the system given in Equations 4.3-4.5.

We construct the optimality condition by taking $J[\mathbf{u}_C + \epsilon\omega_C, \mathbf{u}_R + \epsilon\omega_R]$ and equating the term linear in ϵ to 0:

$$\begin{aligned} 0 = &\iiint_{\Omega_C} \rho(\mathbf{u}_C - \mathbf{u}_C^*) \cdot \omega_C + \rho(\mathbf{u}_C - \mathbf{u}_C^*) \cdot \omega_R \\ &\qquad + 2\Delta t\mu\ \varepsilon(\mathbf{u}_C) : \varepsilon(\omega_C) + 2\Delta t\mu\ \varepsilon(\mathbf{u}_C) : \varepsilon(\omega_R)\ dV \\ &+ \iiint_{\Omega_R} \rho(\mathbf{u}_R - \mathbf{u}_R^*) \cdot \omega_C + \rho(\mathbf{u}_R - \mathbf{u}_R^*) \cdot \omega_R \\ &\qquad + 2\Delta t\mu\ \varepsilon(\mathbf{u}_R) : \varepsilon(\omega_C) + 2\Delta t\mu\ \varepsilon(\mathbf{u}_R) : \varepsilon(\omega_R)\ dV \end{aligned} \tag{A.1}$$

where $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal)$ is the deformation rate tensor. Since ω_C and ω_R are arbitrary test functions, first take $\omega_R = 0$. This eliminates half of the above terms; taking the remaining terms and applying integration-by-parts to the tensor contractions, we get:

$$0 = \iiint_{\Omega_C} (\rho(\mathbf{u}_C - \mathbf{u}_C^*) - 2\Delta t\mu\nabla \cdot \varepsilon(\mathbf{u}_C)) \cdot \omega_C\ dV \tag{A.2}$$

$$+ \iint_{\partial\Omega_C} 2\Delta t\mu\varepsilon(\mathbf{u}_C)\hat{\mathbf{n}}_C \cdot \omega_C\ dA \tag{A.3}$$

$$+ \iiint_{\Omega_R} (\rho(\mathbf{u}_R - \mathbf{u}_R^*) - 2\Delta t\mu\nabla \cdot \varepsilon(\mathbf{u}_R)) \cdot \omega_C\ dV \tag{A.4}$$

$$+ \iint_{\partial\Omega_R} 2\Delta t\mu\varepsilon(\mathbf{u}_R)\hat{\mathbf{n}}_R \cdot \omega_C\ dA \tag{A.5}$$

Factoring out ω_R, the rest of the integrand of each term is thus identically zero, arriving at the conditions,

$$0 = \rho(\mathbf{u}_C - \mathbf{u}_C^*) - 2\Delta t\mu\nabla \cdot \varepsilon(\mathbf{u}_C), \qquad \text{in } \Omega_C \tag{A.6}$$

$$0 = \rho(\mathbf{u}_R - \mathbf{u}_R^*) - 2\Delta t\mu\nabla \cdot \varepsilon(\mathbf{u}_R), \qquad \text{in } \Omega_R \tag{A.7}$$

$$0 = \varepsilon(\mathbf{u}_C) - \varepsilon(\mathbf{u}_R), \qquad \text{on } \partial\Omega_R \tag{A.8}$$

with the last term resulting from the shared boundary, $\partial\Omega_R \subseteq \partial\Omega_C$, with boundary normals pointing in opposite directions, $\hat{\mathbf{n}}_C = -\hat{\mathbf{n}}_R$. It thus clearly follows that:

$$\mathbf{u}_C = \mathbf{u}_R, \qquad \text{on } \partial\Omega_R \tag{A.9}$$

Taking $\omega_C = 0$ results in the same system, just with ω_R as the arbitrary test function. The objective function thus satisfies the required viscosity system.

□

Appendix B

Full Quadratic Model

We demonstrate the full quadratic model with 26 degrees of freedom via the definition of $\mathbf{C}(\mathbf{x})$ (given on next page):

$$\mathbf{C}^{\intercal}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \tilde{x} & 0 & -\tilde{z} \\ \tilde{y} & 0 & 0 \\ \tilde{z} & 0 & 0 \\ \tilde{x}^2 & 0 & -2\tilde{x}\tilde{z} \\ \tilde{x}\tilde{y} & 0 & -\tilde{y}\tilde{z} \\ \tilde{x}\tilde{z} & 0 & -\frac{1}{2}\tilde{z}^2 \\ \tilde{y}^2 & 0 & 0 \\ \tilde{y}\tilde{z} & 0 & 0 \\ \tilde{z}^2 & 0 & 0 \\ 0 & \tilde{x} & 0 \\ 0 & \tilde{y} & -\tilde{z} \\ 0 & \tilde{z} & 0 \\ 0 & \tilde{x}^2 & 0 \\ 0 & \tilde{x}\tilde{y} & -\tilde{x}\tilde{z} \\ 0 & \tilde{x}\tilde{z} & 0 \\ 0 & \tilde{y}^2 & -2\tilde{y}\tilde{z} \\ 0 & \tilde{y}\tilde{z} & -\frac{1}{2}\tilde{z}^2 \\ 0 & \tilde{z}^2 & 0 \\ 0 & 0 & \tilde{x} \\ 0 & 0 & \tilde{y} \\ 0 & 0 & \tilde{x}^2 \\ 0 & 0 & \tilde{x}\tilde{y} \\ 0 & 0 & \tilde{y}^2 \end{bmatrix} \tag{B.1}$$

where $\tilde{x} = x - x_{COM}$ and likewise for $\tilde{y}$ and $\tilde{z}$.

Appendix C

Reduced Model Stokes Proof for Free Surfaces

Here follows a proof that the objective function given in Equation 4.11 solves the system given in Equations 4.12-4.16.

We can construct the optimality conditions by equating the linear term of $J[\mathbf{u}_C + \epsilon\omega_C, \mathbf{u}_R + \epsilon\omega_R, p + \epsilon q, \tau + \epsilon\sigma]$ to zero:

$$\begin{aligned} 0 = \iiint_{\Omega_C} & \frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*) \cdot \omega_C + \frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*) \cdot \omega_R - q\nabla \cdot \mathbf{u}_C + \sigma : \varepsilon(\mathbf{u}_C) \\ & - p\nabla \cdot \omega_C - p\nabla \cdot \omega_R + \tau : \varepsilon(\omega_C) + \tau : \varepsilon(\omega_R) - \frac{1}{2\mu}\tau : \sigma \; dV \\ + \iiint_{\Omega_R} & \frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*) \cdot \omega_C + \frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*) \cdot \omega_R - q\nabla \cdot \mathbf{u}_R + \sigma : \varepsilon(\mathbf{u}_R) \\ & - p\nabla \cdot \omega_C - p\nabla \cdot \omega_R + \tau : \varepsilon(\omega_C) + \tau : \varepsilon(\omega_R) \\ & + \mu\varepsilon(\mathbf{u}_R) : \varepsilon(\omega_C) + \mu\varepsilon(\mathbf{u}_R) : \varepsilon(\omega_R) \; dV \end{aligned} \tag{C.1}$$

Applying integration-by-parts to the pressure and viscous stress terms, we get:

$$
\begin{aligned}
0 = &\iiint_{\Omega_C} \frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*)\cdot\omega_C + \frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*)\cdot\omega_R - q\nabla\cdot\mathbf{u}_C + \sigma : \varepsilon(\mathbf{u}_C) \\
&\qquad + \nabla p\cdot\omega_C + \nabla p\cdot\omega_R - \nabla\cdot\tau\cdot\omega_C - \nabla\cdot\tau\cdot\omega_R - \frac{1}{2\mu}\tau : \sigma \; dV \\
&+ \iint_{\partial\Omega_C} -(p\mathbf{I}\hat{\mathbf{n}}_C)\cdot\omega_C - (p\mathbf{I}\hat{\mathbf{n}}_C)\cdot\omega_R + \tau\hat{\mathbf{n}}_C\cdot\omega_C + \tau\hat{\mathbf{n}}_C\cdot\omega_R \; dA \\
&+ \iiint_{\Omega_R} \frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*)\cdot\omega_C + \frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*)\cdot\omega_R - q\nabla\cdot\mathbf{u}_R + \sigma : \varepsilon(\mathbf{u}_R) \\
&\qquad + \nabla p\cdot\omega_C + \nabla p\cdot\omega_R - \nabla\cdot\tau\cdot\omega_C - \nabla\cdot\tau\cdot\omega_R - \mu\nabla\cdot\varepsilon(\mathbf{u}_R)\cdot\omega_C - \mu\nabla\cdot\varepsilon(\mathbf{u}_R)\cdot\omega_R \; dV \\
&+ \iint_{\partial\Omega_R} -(p\mathbf{I}\hat{\mathbf{n}}_R)\cdot\omega_C - (p\mathbf{I}\hat{\mathbf{n}}_R)\cdot\omega_R + \tau\hat{\mathbf{n}}_R\cdot\omega_C + \tau\hat{\mathbf{n}}_R\cdot\omega_R \\
&\qquad + \mu\varepsilon(\mathbf{u}_R)\hat{\mathbf{n}}_R\cdot\omega_C + \mu\varepsilon(\mathbf{u}_R)\hat{\mathbf{n}}_R\cdot\omega_R \; dA
\end{aligned}
$$

Since ω_C, ω_R, q, and σ are arbitrary test functions, first take all except ω_C to be zero. This gives:

$$
\begin{aligned}
0 = &\iiint_{\Omega_C} \Big(\frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*) + \nabla p - \nabla\cdot\tau\Big)\cdot\omega_C \\
&+ \iint_{\partial\Omega_C} (-p\mathbf{I} + \tau)\,\hat{\mathbf{n}}_C\cdot\omega_C \; dA \\
&+ \iiint_{\Omega_R} \Big(\frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*) + \nabla p - \nabla\cdot\tau - \mu\nabla\cdot\varepsilon(\mathbf{u}_R)\Big)\cdot\omega_C \; dV \\
&+ \iint_{\partial\Omega_R} (-p\mathbf{I} + \tau)\hat{\mathbf{n}}_R\cdot\omega_C \; dA
\end{aligned}
$$

The integrands each translate directly to the system:

$$
\begin{aligned}
\frac{\rho}{\Delta t}(\mathbf{u}_C - \mathbf{u}_C^*) + \nabla p + \nabla\cdot\tau = 0, &\qquad \text{in } \Omega_C &\qquad \text{(C.2)}\\
\frac{\rho}{\Delta t}(\mathbf{u}_R - \mathbf{u}_R^*) + \nabla p + \nabla\cdot\tau + \mu\nabla\cdot\varepsilon(\mathbf{u}_R) = 0, &\qquad \text{in } \Omega_R &\qquad \text{(C.3)}\\
(-p\mathbf{I} + \tau)\hat{\mathbf{n}}_C = 0, &\qquad \text{on } \partial\Omega_C &\qquad \text{(C.4)}\\
(-p\mathbf{I} + \tau)\hat{\mathbf{n}}_R = 0, &\qquad \text{on } \partial\Omega_R &\qquad \text{(C.5)}
\end{aligned}
$$

Remembering that $\partial\Omega_L = \partial\Omega_C \setminus \partial\Omega_R$, and that the normals of the shared boundary point in opposite directions, $\hat{\mathbf{n}}_C = \hat{\mathbf{n}}_R$, then the last two equations cancel on the shared boundary,

leaving the free surface boundary condition:

$$(-p\mathbf{I} + \tau)\hat{\mathbf{n}} = 0, \qquad \text{on } \partial\Omega_L \tag{C.6}$$

The same system is given when taking all test functions except ω_R to be zero. Setting all except q as zero gives:

$$0 = \iiint_{\Omega_C} q\nabla \cdot \mathbf{u}_C \; dV + \iiint_{\Omega_R} q\nabla \cdot \mathbf{u}_R \; dV \tag{C.7}$$

We independently constrain $\nabla \cdot \mathbf{u}_R = 0$ on the interior Ω_R° by the definition of $\mathbf{u}_R$, therefore it follows that,

$$0 = \iiint_{\Omega_C} q\nabla \cdot \mathbf{u}_C + q\nabla \cdot \mathbf{u}_R \; dV \tag{C.8}$$

where domains of integration are enforced by where $\mathbf{u}_R$ is defined. That is, the second term applies only on the shared boundary $\partial\Omega_R$. This recovers the divergence-free condition from the integrands after factoring out q. Setting all except σ as zero gives:

$$0 = \iiint_{\Omega_C} \sigma : \varepsilon(\mathbf{u}_C) - \frac{1}{2\mu}\tau : \sigma \; dV + \iiint_{\Omega_R} \sigma : \varepsilon(\mathbf{u}_R) \; dV \tag{C.9}$$

Once again, combining the integrations where the last term applies only on the shared boundary gives,

$$0 = \iiint_{\Omega_C} \sigma : \varepsilon(\mathbf{u}_C) - \frac{1}{2\mu}\tau : \sigma \; dV + \sigma : \varepsilon(\mathbf{u}_R) \; dV \tag{C.10}$$

which recovers the viscous equation after factoring out σ. We thus have demonstrated that optimizing the objective function recovers the required system.

□

Appendix D

Least Squares Error Minimization

We derive the least squares system, Equation 4.28, from the minimization of the sum of the squared residuals, Equation 4.27. The vector residual for a single face j is:

$$e_j = \mathbf{u}_j - \mathbf{C}(\mathbf{x}_j)\mathbf{v} \tag{D.1}$$

The sum of the squared residuals over all faces is thus:

$$\sum_j e_j^\intercal e_j = \sum_j \left(\mathbf{u}_j - \mathbf{C}(\mathbf{x}_j)\mathbf{v}\right)^\intercal \left(\mathbf{u}_j - \mathbf{C}(\mathbf{x}_j)\mathbf{v}\right) \tag{D.2}$$

$$= \sum_j \left(\mathbf{u}_j^\intercal \mathbf{u}_j - 2\mathbf{v}^\intercal \mathbf{C}_j^\intercal \mathbf{u}_j + \mathbf{v}^\intercal \mathbf{C}_j^\intercal \mathbf{C}_j \mathbf{v}\right) \tag{D.3}$$

Minimizing this error with respect to $\mathbf{v}$, the first derivative must be zero:

$$\frac{\partial}{\partial \mathbf{v}} \left(\sum_j e_j^\intercal e_j \right) = \sum_j \left(-2\mathbf{C}_j^\intercal \mathbf{u} + 2\mathbf{C}_j^\intercal \mathbf{C}_j \mathbf{v}\right) = 0 \tag{D.4}$$

This gives our least squares system:

$$\sum_j \mathbf{C}_j^\intercal \mathbf{u}_j = \sum_j \mathbf{C}_j^\intercal \mathbf{C}_j \mathbf{v} \tag{D.5}$$

Appendix E

Error Tables for Analytical Viscosity Problem

Table E.1: L_∞ errors for a viscosity-only step applied to the analytical test case given on Section 5.3. Both tables use tiled interior regions with (a) using the affine model and (b) using the quadratic model. Tile and padding sizes were chosen such that each diagonal has interior interior regions at exactly the same spatial locations. Grey cells indicate tile and padding sizes too coarse for the given resolution. The main diagonal has exactly one full-size tile, and the upper diagonals have 4, 16, 64, and 256 tiles respectively.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	7.878E-01	3.187E-01	1.547E-01	5.417E-02	1.537E-02
32 / 4		8.074E-01	3.345E-01	1.645E-01	5.782E-02
64 / 8			8.156E-01	3.432E-01	1.702E-01
128 / 16				8.194E-01	3.477E-01

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	3.129E-01	6.054E-02	1.084E-02	1.432E-03	1.870E-04
32 / 4		3.415E-01	6.445E-02	1.139E-02	1.598E-03
64 / 8			3.557E-01	6.656E-02	1.213E-02
128 / 16				3.627E-01	6.784E-02

(b) Quadratic

Table E.2: L_∞ reconstruction errors for the analytical test case given on Section 5.3. The reconstruction error compares solely the (a) affine and (b) quadratic least squares fits to the initial condition, before dynamics are applied. Tile and padding sizes were chosen such that each diagonal has interior interior regions at exactly the same spatial locations. Grey cells indicate tile and padding sizes too coarse for the given resolution. The main diagonal has exactly one full-size tile, and the upper diagonals have 4, 16, 64, and 256 tiles respectively.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	2.616E+00	1.086E+00	3.658E-01	1.115E-01	2.923E-02
32 / 4		2.712E+00	1.121E+00	3.803E-01	1.158E-01
64 / 8			2.757E+00	1.138E+00	3.876E-01
128 / 16				2.778E+00	1.146E+00

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	1.293E+00	3.407E-01	4.766E-02	6.107E-03	7.695E-04
32 / 4		1.406E+00	3.611E-01	5.021E-02	6.441E-03
64 / 8			1.462E+00	3.708E-01	5.144E-02
128 / 16				1.490E+00	3.756E-01

(b) Quadratic

Table E.3: L_∞ errors for a viscosity-only step applied to the analytical test case given on Section 5.3 with constant padding size. Both tables use tiled interior regions with (a) using the affine model and (b) using the quadratic model. Grey cells indicate tile and padding sizes too coarse for the given resolution. The main diagonal has exactly one full-size tile, and the upper diagonals have 4, 16, 64, and 256 tiles respectively.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	7.878E-01	3.187E-01	1.547E-01	5.417E-02	1.537E-02
32 / 2		9.454E-01	4.708E-01	2.667E-01	1.059E-01
64 / 2			9.878E-01	6.259E-01	4.153E-01
128 / 2				9.979E-01	7.617E-01

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	3.129E-01	6.054E-02	1.084E-02	1.432E-03	2.172E-04
32 / 2		6.366E-01	9.054E-02	1.599E-02	2.325E-03
64 / 2			8.422E-01	1.170E-01	2.127E-02
128 / 2				9.428E-01	1.423E-01

(b) Quadratic

Table E.4: L_∞ reconstruction errors for a viscosity-only step applied to the analytical test case given on Section 5.3 with constant padding size. The reconstruction error compares solely the (a) affine and (b) quadratic least squares fits to the initial condition, before dynamics are applied. Grey cells indicate tile and padding sizes too coarse for the given resolution. The main diagonal has exactly one full-size tile, and the upper diagonals have 4, 16, 64, and 256 tiles respectively.

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	2.616E+00	1.086E+00	3.658E-01	1.115E-01	2.923E-02
32 / 2		3.362E+00	1.344E+00	4.723E-01	1.342E-01
64 / 2			3.699E+00	1.470E+00	5.295E-01
128 / 2				3.854E+00	1.532E+00

(a) Affine

Tile / Padding \ dx	1/16	1/32	1/64	1/128	1/256
16 / 2	1.293E+00	3.407E-01	4.766E-02	6.107E-03	7.695E-04
32 / 2		2.429E+00	4.485E-01	6.210E-02	7.941E-03
64 / 2			3.162E+00	5.072E-01	7.011E-02
128 / 2				3.568E+00	5.376E-01

(b) Quadratic

Appendix F

Error Tables for Analytical Free Surface Stokes Problem

Table F.1: Error values for the analytical test case given in Section 5.4.

	Affine L_1	Affine L_∞	Quadratic L_1	Quadratic L_∞	Uniform L_1	Uniform L_∞
1/16	2.2858E+00	1.6646E+00	1.5582E+00	1.2112E+00	4.5395E-02	3.0139E-02
1/32	1.5643E+00	1.2452E+00	1.1511E+00	7.2727E-01	2.2576E-02	3.1938E-02
1/64	4.7645E-01	6.0261E-01	1.1502E-01	1.2482E-01	1.0988E-02	2.3042E-02
1/128	2.4068E-01	2.9071E-01	2.0523E-02	2.5571E-02	5.8821E-03	1.4434E-02
1/256	1.0007E-01	1.1680E-01	1.1193E-02	1.2536E-02		