

# 2D Surface-Only Liquid-Solid Coupling

by

Clara Kim

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, 2024

© Clara Kim 2024

### **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

Liquid simulations typically involve solving the fluid equations over the simulated liquid domain via some volumetric discretization of the domain. Consequently, the majority of schemes which aim to simulate the interaction between liquids and freely moving solids are built assuming a volumetrically discretized liquid model. However, storing and manipulating a surface mesh rather than a volumetric discretization, on top of allowing us to avoid storing interior volume data, has the potential to reduce the number of unknowns in the systems necessary to resolve fluid flow. Motivated by this potential, we present a method for simulating the two-way coupled interactions between solid rigid bodies and an inviscid liquid, where the liquid domain is represented and simulated entirely by its surface. Our work builds off of the surface-only liquids method first proposed by [Da et al. \[2016\]](#). We are concerned with the 2D version of the liquid-solid coupling problem. As such, the liquid surface is represented as a series of point vertices connected by line segment edges, with the velocity data stored at the vertices.

The surface-only liquid simulation method integrates outside forces, such as forces caused by scripted solids in contact with the liquid, by performing a boundary element method (BEM) solve for fluid pressures using surface tension and solid velocities to set boundary conditions. We perform liquid-solid coupling in a single unified solve by modifying this force integration step to account for solids with dynamic velocities by modeling the momentum exchange that occurs at the solid-liquid interface. We show several examples demonstrating our method’s ability to handle liquid-rigid body dynamics, as well as validate our method against analytical solutions derived using the fluid mechanics concept of *added mass*. We also demonstrate our method’s ability to support multiple solid rigid bodies interacting with each other through the liquid domain without need for direct contact between the solids. We hope that our work encourages further investigations into the surface-only liquids framework in the future, allowing for the simulation of an even wider range of interesting liquid phenomena using only a surface discretization of the simulated domain.

## Acknowledgements

First and foremost, to my supervisor Christopher Batty, thank you for taking me on as your student and guiding me through my studies. I have grown so much throughout this degree both personally and as a researcher, and much of that credit goes to you and your continued faith in me, understanding, and flexibility, not to mention, of course, your deep knowledge of the field of computer graphics and physics-based simulation. Thank you for your encouragement and your trust in me and my work even in my moments of doubt.

To Toshiya Hachisuka, thank you for taking the time to attend my research meetings and provide me with additional insight and guidance.

To my readers, Toshiya Hachisuka and Justin Wan, thank you for reviewing this work. Your questions and comments gave me much to consider and have elevated the quality of my analysis.

To all my friends and colleagues in the Computational Motion Group: Brooke, Rikin, Nathan, Ryusuke (隆介), and Joel; thank you for our discussions, both formal and informal, about research, coursework, and everything and anything else. All of you are so inspiring, and I learned so much about computer science and physically-based animation, not to mention non-work related topics, from even the briefest of conversations with any of you. In particular, thank you Ryusuke for your mathematical knowledge and your generosity in sharing it, whether that be by providing your notes, attending my weekly meetings, or by voluntarily taking the time to give me clear explanations.

To my friends and members of Plato's Cave: Jakob, Jonathan, Rose, Rikin, Xiangye (祥烨), and Chichi (赤炽); thank you for our hours of conversation and fun. Your knowledge and perspectives of mathematics and fluid dynamics informed my research and gave me the courage to hold my work to a higher standard. Above all, thank you all so much for filling my time here with joy.

To Luther Tychonievich, my undergraduate research advisor, thank you for my start in computer graphics research and for encouraging me to pursue graduate studies in the first place.

Finally, to my parents Sunhak and Soyeon, thank you for raising me with the belief that I could pursue whatever I put my mind to, and always doing all that you could to provide me with opportunities to do so. To my brother Chris, for always being there for me. To my family, for your unconditional love and support which provides me with a steady place to land even in the most turbulent of times, I am always grateful.

## **Dedication**

To my parents, Sunhak and Soyeon.

# Table of Contents

|                                              |            |
|----------------------------------------------|------------|
| <b>Author's Declaration</b>                  | <b>ii</b>  |
| <b>Abstract</b>                              | <b>iii</b> |
| <b>Acknowledgements</b>                      | <b>iv</b>  |
| <b>Dedication</b>                            | <b>v</b>   |
| <b>List of Figures</b>                       | <b>ix</b>  |
| <b>List of Tables</b>                        | <b>xi</b>  |
| <b>1 Introduction</b>                        | <b>1</b>   |
| <b>2 Related Work</b>                        | <b>4</b>   |
| <b>3 Mathematical Background</b>             | <b>8</b>   |
| 3.1 Governing Fluid Equations . . . . .      | 8          |
| 3.2 Potential Flow and Harmonicity . . . . . | 9          |
| 3.3 Rigid Body Dynamics . . . . .            | 10         |
| 3.4 Two-way coupling . . . . .               | 11         |

|          |                                                      |           |
|----------|------------------------------------------------------|-----------|
| <b>4</b> | <b>Method of Surface-Only Liquids</b>                | <b>13</b> |
| 4.1      | Surface-only Representation of Fluid Flow . . . . .  | 13        |
| 4.2      | Domain Representation and Discretization . . . . .   | 14        |
| 4.3      | Time Integration . . . . .                           | 16        |
| 4.3.1    | Advection . . . . .                                  | 17        |
| 4.3.2    | Incompressible and Irrotational Projection . . . . . | 18        |
| 4.3.3    | External force integration . . . . .                 | 21        |
| 4.4      | Spatial Discretization . . . . .                     | 23        |
| 4.4.1    | Advection . . . . .                                  | 24        |
| 4.4.2    | Incompressible and Irrotational Projection . . . . . | 24        |
| 4.4.3    | External force integration . . . . .                 | 29        |
| <b>5</b> | <b>Surface-Only Rigid Body Coupling</b>              | <b>39</b> |
| 5.1      | Solid to Fluid Coupling . . . . .                    | 41        |
| 5.2      | Fluid to Solid Coupling . . . . .                    | 43        |
| 5.3      | Two-way Coupled System . . . . .                     | 45        |
| 5.4      | Triple Points . . . . .                              | 46        |
| 5.4.1    | Solid to Fluid Coupling with Triple Points . . . . . | 46        |
| 5.4.2    | Fluid to Solid Coupling with Triple Points . . . . . | 48        |
| 5.4.3    | Two-way Coupled System with Triple Points . . . . .  | 48        |
| <b>6</b> | <b>Results and Discussion</b>                        | <b>50</b> |
| <b>7</b> | <b>Conclusion</b>                                    | <b>68</b> |
| 7.1      | Limitations . . . . .                                | 68        |
| 7.2      | Future Work . . . . .                                | 70        |
|          | <b>References</b>                                    | <b>73</b> |
|          | <b>APPENDICES</b>                                    | <b>82</b> |

|          |                                          |           |
|----------|------------------------------------------|-----------|
| <b>A</b> | <b>Quadrature schemes</b>                | <b>83</b> |
| A.1      | Tanh-Sinh Quadrature . . . . .           | 83        |
| A.2      | Four-point Gaussian Quadrature . . . . . | 84        |

# List of Figures

|      |                                                                                                              |    |
|------|--------------------------------------------------------------------------------------------------------------|----|
| 4.1  | Oriented mesh . . . . .                                                                                      | 15 |
| 4.2  | Global and local numbering for BEM boundary discretization . . . . .                                         | 31 |
| 4.3  | Surface tension forces at triple points/triple junctions . . . . .                                           | 36 |
| 5.1  | Example setup of an immersed solid . . . . .                                                                 | 41 |
| 6.1  | One period of an oscillating droplet . . . . .                                                               | 51 |
| 6.2  | Error in period of oscillation versus mesh resolution . . . . .                                              | 52 |
| 6.3  | Submerged rigid bodies with varying density . . . . .                                                        | 53 |
| 6.4  | Error in velocity versus mesh resolution for a submerged rigid body with density $\rho_s = \rho_f$ . . . . . | 54 |
| 6.5  | Irregularly shaped submerged rigid body of lower density than surrounding fluid . . . . .                    | 55 |
| 6.6  | Irregularly shaped submerged rigid body of higher density than surrounding fluid . . . . .                   | 56 |
| 6.7  | Magnitude of a disk-shaped rigid body's velocity versus the solid density . . . . .                          | 57 |
| 6.8  | Error in the magnitude of a disk-shaped rigid body's velocity versus mesh resolution . . . . .               | 58 |
| 6.9  | Floating rigid bodies with varying density . . . . .                                                         | 59 |
| 6.10 | Two rigid bodies of unequal density cause oscillations when placed at the ends of a pipe of liquid . . . . . | 60 |
| 6.11 | A system with two rigid bodies and a free surface oscillates . . . . .                                       | 62 |

|      |                                                                                                                                       |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------|----|
| 6.12 | Two floating disk-shaped rigid bodies translate and rotate due to a third rigid body accelerating in the liquid below them . . . . .  | 63 |
| 6.13 | Two submerged disk-shaped rigid bodies translate and rotate due to a third rigid body accelerating in the liquid below them . . . . . | 64 |
| 6.14 | Uneven liquid heights on each side of a pipe cause oscillations . . . . .                                                             | 66 |
| 6.15 | Liquid oscillating in a cup due to unequal surface tension forces at two mesh resolutions . . . . .                                   | 67 |

# List of Tables

|     |                                                                                                                              |    |
|-----|------------------------------------------------------------------------------------------------------------------------------|----|
| 6.1 | Period of oscillation and errors after one period of an initially perturbed 2D droplet for various mesh resolutions. . . . . | 51 |
| 6.2 | Run time and mesh size data for a sampling of example simulations . . . .                                                    | 65 |

# Chapter 1

## Introduction

Water is everywhere. It is ubiquitous in our daily lives, and as a result we witness a countless number of liquid-solid interactions throughout our day, from ice cubes floating in glasses of water to dishes submersed in the sink. Physically-based simulations of such interactions are of interest in graphics settings, as it is difficult for an artist to plausibly design fluid and fluid interactions by hand over several frames of animation; the complexities of the liquid and solid exerting forces on one another cause these dynamic systems to not always behave intuitively.

While water is technically neither incompressible nor inviscid, it is often idealized as both in graphical simulation settings. Its compressibility is negligible enough under typical conditions such that the engineering and scientific literature frequently ignore it as well. While this is not also the case for water’s viscosity, for visual purposes water can be idealized as inviscid to great effect, as it is often the case that numerical methods for simulating fluids in graphics introduce errors that appear visually similar to the viscous effects that appear in low-viscosity fluids [Bridson, 2015].

Inviscid and incompressible flows are mathematically described using the incompressible Euler equations. These equations are widely studied both in and out of graphics contexts. However, while accurately modeling internal fluid dynamics and associated internal quantities can be critical in some science and engineering applications, these dynamics often do not take center stage in computer graphics applications, where the focus is usually shifted to the movement and visual aesthetics of the liquid’s exterior. Moreover, the interaction between liquids and solids consists of a momentum exchange at the interface between the two domains, which means modeling the interplay of the two also does not necessarily require the internal dynamics of the liquid. We are thus motivated to work with a liquid

model that focuses on the liquid’s exterior in our construction of a coupled liquid-solid scheme.

However, the majority of fluid models in computer graphics involve solving fluid motion over a volumetric discretization of the fluid domain. *Lagrangian*, or particle-based, techniques such as smoothed particle hydrodynamics (SPH) [Müller et al., 2003] simulate fluid by tracking the movement of particles which represent small fluid parcels. *Eulerian*, or grid-based, methods, such as the foundational work by Stam [1999], conceptualize the entire domain as divided into a grid, with each cell containing some amount of fluid. The fluid equations can then be solved using information stored in cells or at cell boundaries. Other methods that have gained popularity in graphics for fluid simulation, such as Particle-in-Cell (PIC) [Foster and Metaxas, 1996] and Fluid-Implicit-Particle (FLIP) [Zhu and Bridson, 2005], and related work [Jiang et al., 2015; Stomakhin et al., 2013], as the names may suggest, use both particles and background grid cells to represent the domain, passing information back and forth between the particles and grid cells. Still other methods, such as the those of Feldman et al. [2005] and Clausen et al. [2013] involve discretizing the fluid volume with a mesh and solving the fluid motion using well-established numerical techniques such as the Finite Element Method (FEM), while others, such as the work by Guo et al. [2017] use Lattice-Boltzmann Methods, bypassing solving the fluid equations directly by simulating fluid density on a lattice. While this is far from a comprehensive list, in short volumetric discretizations have historically been the go-to for graphical fluid simulation. As such, liquid-solid coupling models generally also assumed such volumetrically discretized liquids.

Contrary to this general approach, Da et al. [2016] recognized that in graphics applications many interesting liquid simulation setups only require focus on the free surface of the liquid, as this liquid surface is often the most, if not only, important *visual* component of the simulated liquid. They refer to the fact that the visual phenomena that dominate when considering graphically rendered liquid simulations, such as reflection and refraction, rely most heavily on the free surface of the liquid rather than any of the internal dynamics of the liquid. Da et al. [2016] proposed a liquid model for idealized inviscid flow that circumvents a volumetric discretization of the liquid domain, only tracking information at the liquid surface. Their method is built on the assumption that the liquid domain is irrotational, in addition to incompressible, which also brings their model into the well-studied space of *potential flow* models.

We would like to extend the scheme by Da et al. [2016] to support two-way coupling between surface-only liquids and solids, specifically rigid bodies. We begin by considering how prior volumetric schemes have incorporated interactions with rigid bodies. Some prior work on two-way coupling achieve the coupled dynamics by including the solid velocities as

part of the boundary conditions of the fluid motion, with the fluid pressures in turn providing forces and torques on the solid [Klingner et al., 2006; Batty et al., 2007]. When a rigid body is introduced into a liquid domain that is modeled by the surface-only discretization of Da et al. [2016], the interface between the rigid body and the liquid, that is, the interface of the submerged part of the rigid body, is included as part of the liquid surface mesh. Clearly, the liquid surface mesh representation is well suited to carry information between the liquid domain and the rigid body.

Based on this idea, our key contribution is to extend the work of Da et al. [2016] and develop a technique to couple their surface-only inviscid liquid model with rigid bodies. We show that our model can accurately simulate coupled rigid body-liquid dynamics for rigid bodies by evaluating both qualitatively and against quantitative solutions for solids of varying density relative to the surrounding liquid. Our model is able to simulate multi-body systems, allowing rigid bodies to interact with each other through the liquid domain without any direct contact. In addition, in this thesis we also provide a comprehensive description of a 2D implementation of the work by Da et al. [2016], as subtleties in the mathematical formulation cause the 2D and 3D models to require some distinct handling. Ultimately, we hope that this thesis provides insight into the method of surface-only liquid simulation and ways in which to extend this method to simulate a wider variety of liquid phenomena, and inspires more investigations into this relatively unexplored model of fluid flow.

# Chapter 2

## Related Work

Fluids and fluid-solid coupling are well-studied problems in the computer graphics literature. We provide an outline of work relating to the surface-only liquid approach as well as our work on surface-only liquid-solid coupling.

**Free Surface Flow** Oftentimes, graphics applications are concerned with the simulation of a liquid’s free surface, or the liquid surface that is open to the atmosphere. [Foster and Fedkiw \[2001\]](#) and [Enright et al. \[2002\]](#) use grid-based approaches that use a combination of level sets and Lagrangian particles to track the free surface. [Enright et al. \[2003\]](#) also use a grid-based approach with a particle level set method, adding surface tension effects by defining ghost pressure values just past the interface to solve for pressures at the interface. The work by [Wojtan et al. \[2009\]](#) and [Brochu et al. \[2010\]](#) are notable to us in that they explicitly track the free surface with a surface mesh to capture fine surface details, though both also use a background grid for the liquid volume. While much work has been done on capturing surface tension-dominated free surface flow, we limit our review of these; though the surface-only liquids approach we build on is well suited to modeling surface tension and free surface flow due to its explicit surface mesh representation, modeling these effects is not the focus of our work.

**Non-volumetric Vortex Methods** As alluded to in the previous chapter, the majority of fluid simulation work in graphics is done via a volumetric discretization of the fluid domain. However, noticing that volumetric discretizations grow cubically when refined, whereas surface meshes grow quadratically, some researchers have turned to exploring techniques that only require a surface discretization. Several of these prior non-volumetric

approaches use vortex-based methods, which, unlike velocity-based methods, evolve the fluid’s dynamics using vorticity, or the curl of velocity, rather than the velocity itself. [Park and Kim \[2005\]](#) simulate swirling gaseous phenomena by advecting Lagrangian *vortex particles* which represent circulating basis velocity fields. [Vines et al. \[2014\]](#) extend their work to simulate two-way coupled fluid-solid interactions by enforcing a boundary condition on the fluid at solid boundaries at the same time as computing pressures from the fluid on the solid using boundary integrals. [Brochu et al. \[2012\]](#) and [Pfaff et al. \[2012\]](#) developed similar approaches to simulate smoke in linear time, observing that vorticity is concentrated in the surface separating smoky and non-smoky regions, called the *vortex sheet*. Prior to their work, [Weißmann and Pinkall \[2010\]](#) recognized that vortex sheets curl into one-dimensional structures; their work uses Lagrangian vortex *filaments* rather than sheets. [Da et al. \[2015\]](#) extend the use of vortex methods to simulating the dynamics and geometric deformations of soap bubbles. [Tao et al. \[2024\]](#) also use vortex methods for soap film simulation, but focus on interfacial flow. However, while vortex methods produce compelling results for smoke and thin film simulation, [Da et al. \[2016\]](#) point out that the current methods are not well suited for modeling free surface liquid flows due to the large density ratio that occurs at liquid-air interfaces.

**Potential Flow for Liquid Simulation** Potential flow is an idealized description of fluid flow with no vorticity, in other words, *irrotational* flow. Fluid flows that follow potential flow can be concisely described as a gradient of some scalar function, known as the potential function. The potential flow model has been found to be a useful tool in modeling a variety of liquid effects in situations where vorticity is negligible. [Davidson \[2000\]](#) models droplet impact by using the Bernoulli equation to evolve the potential function. Noting that it is common to assume that wave motion is irrotational when simulating waves, [Keeler and Bridson \[2014\]](#) also use the Bernoulli equation to simulate deep ocean waves, although they drop the non-linear term in the equation in exchange for stability. Another interesting application of potential flow can be found in the work by [Nielsen and Bridson \[2011\]](#), whose aim is to simulate liquids based on artist-provided guide shapes. Since the guide shapes do not always have well-defined interior velocities, they utilize potential flow to interpolate interior velocities from a field defined on the guide shape’s surface mesh.

**BEM in Computer Graphics** The boundary element method (BEM) is a technique to solve linear partial differential equations (PDEs) on a domain by only using a surface discretization of the domain. BEMs were first introduced to graphics by [James and Pai \[1999\]](#) to simulate static deformable solids at interactive speeds using only the solids’ surface meshes. [Sugimoto et al. \[2022\]](#) also use BEM to simulate deformable solids, but model

dynamic deformables by solving an elliptic, rather than hyperbolic, PDE. BEM has been applied to a variety of problems in graphics such as solid fracture [Hahn and Wojtan, 2016], geometry processing [Solomon et al., 2017; Wang et al., 2013], and acoustic simulation [James et al., 2006]. While BEM has an advantage over volumetric solvers in terms of the size of the system to be solved due to the dimensional reduction of the domain representation, it has a disadvantage in that BEM systems tend to be dense, whereas the systems associated with volumetric discretizations are generally sparse despite their larger size. The recent work in graphics by Chen et al. [2024] introduces a method to counter the bottleneck in BEM performance by applying a preconditioner to the solve.

BEM has also found use in fluid simulation. Notably, the work by Da et al. [2016] to simulate liquids using only a surface discretization of the liquid domain greatly informs our work in this thesis; we provide a more detailed summary of their contributions in Chapter 4. Huang and Michels [2020] extend their work, incorporating magnetic effects into the surface-only liquid approach to simulate surface-only ferrofluids. Keeler and Bridson [2014] use BEM with their potential flow model to simulate deep ocean waves, recognizing that using volumetric approaches for the ocean can quickly become expensive due to the need to represent the large domain while also capturing small surface features. However, their model is formulated such that it cannot capture overturning and breaking waves. Huang et al. [2021] also approach modeling the ocean with BEM, but do so by dividing the simulated domain into two, using a volumetric FLIP solver on areas where splashes should be found (such as the ocean surface and near obstacles) and a BEM solver based off the work of Da et al. [2016] elsewhere. By coupling the two methods they are able to keep the cost of simulating a large ocean down while capturing splashing details that previous models could not.

**Fluid-solid Coupling** Two-way fluid-solid coupling takes its name from the two interactions that it is composed of: the fluid acts on the solid via fluid pressures, and the solid acts on the fluid by enforcing some boundary condition (as fluid should not be allowed to flow into the solid). Takahashi et al. [2002] introduced a *weak coupling* approach to couple fluids and rigid bodies, where the fluid and solid are solved for in two interleaved steps; constraints imposed by each “side” of the coupling are held fixed while the other is being simulated. Guendelman et al. [2005] adopt this idea to achieve coupling of incompressible fluids to thin shells. While weakly coupled schemes are simple to implement and can be used to great effect, their alternating nature can cause unwanted errors if the time step size is not small enough. The stability of these alternating schemes also becomes an issue if one or both of the coupled materials are stiff, making them unideal for our purposes, as incompressible liquids are quite stiff.

*Strong coupling* schemes, on the other hand, compute the fluid and solid forces simultaneously. Batty et al. [2007] achieve strong two-way fluid-rigid body coupling, using a grid-based volumetric fluid scheme, by simultaneously imposing the solid velocity as a boundary condition on the fluid velocity while integrating fluid pressures acting on the rigid body’s surface to find the net forces and torques acting on the rigid body. They define an operator that converts between the pressures on the rigid body and generalized forces on the rigid body. Klingner et al. [2006] and Chentanez et al. [2006] use similar discretizations, albeit on tetrahedral meshes, to achieve fluid-solid coupling, with the method by Chentanez et al. [2006] also supporting the coupling of deformable objects. The variety of methods for simulating fluids, as well as the variety of ways solid bodies are represented, have given rise to a wide range of methods to address fluid-solid coupling. For example, just varying the fluid model to be grid-based [Robinson-Mosher et al., 2008; Losasso et al., 2008; Azevedo et al., 2016] or particle-based [Misztal et al., 2010; Clausen et al., 2013; Li and Desbrun, 2023] causes a need for different measures to be taken when considering coupling schemes. The surface-only liquid discretization that we use, however, is particularly well suited for the task of taking integrals over the surface of solid bodies, as we have a well-defined liquid mesh over each submerged surface of each body. This fact naturally suggests what is the principle objective and approach taken in this thesis: to adapt the strong coupling idea from the work of Batty et al. [2007] to the liquid model of Da et al. [2016].

Weißmann and Pinkall [2012] simulate the motion of rigid bodies immersed in fluid without needing to simulate the surrounding fluid at all by computing and using the Kirchhoff tensor of the rigid body, which depends on the shape as well as the inertia tensor and mass of the body. They make the simplifying assumptions that the surrounding fluid domain is infinite and nearly still. Ozgen et al. [2010] achieve a similar result but for cloth simulation, capturing the dynamics of immersed cloth in fluid without simulating the surrounding fluid by using fractional derivatives to describe cloth dynamics. Similar to Weißmann and Pinkall [2012], Soliman et al. [2024] and Gross et al. [2023] also simulate immersed solid bodies without an explicit fluid simulation, extending the idea to allow for scripted shape changes in the solid body to affect the motion of the body. While our current implementation does not allow for the simulation of rigid bodies embedded in an infinite domain as is done by Weißmann and Pinkall [2012], we are curious to see if our surface-only coupling approach could be extended to support such cases.

# Chapter 3

## Mathematical Background

We introduce the mathematical descriptions of the physical systems relevant to this thesis, that is, fluid and rigid body dynamics.

### 3.1 Governing Fluid Equations

In many common settings in which humans interact with liquids, the incompressible Navier-Stokes equations provide an accurate model for fluid flow. These two partial differential equations, referred to as the momentum and incompressibility equations respectively, are

$$\frac{D\mathbf{u}}{Dt} + \frac{1}{\rho_f} \nabla p = \mathbf{f} + \nu \nabla \cdot \nabla \mathbf{u}, \quad (3.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (3.2)$$

where  $\mathbf{u}$  is the fluid's velocity,  $\rho_f$  is fluid density,  $p$  is pressure,  $\mathbf{f}$  are body forces (forces that act throughout the entire fluid domain), and  $\nu$  is the kinematic viscosity. The material derivative of the velocity  $\frac{D\mathbf{u}}{Dt}$  describes how the velocity of a fluid particle changes over time, and is expanded as

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}.$$

In short, Equation (3.1) can be thought of as a differential form of Newton's second law of motion  $\mathbf{F} = m\mathbf{a}$ , which is also known as the law of conservation of momentum. Equation (3.2) simply states the condition that the fluid velocity remains divergence-free, enforcing incompressibility in the fluid. Incompressibility, of course, is an idealization; even water

can be compressed at high enough pressures. However, under normal conditions this effect is not observable. In fact, in most cases volume preservation, which is upheld by the incompressibility condition, is imperative to the visual integrity of liquid simulations.

For the purposes of this thesis we only consider simulating water-like fluids, that is, liquids with low viscosity. While most fluids of simulation interest are not truly inviscid, the effects that stem from viscosity in low-viscosity fluid simulations are usually negligible in comparison to effects that stem from numerical dissipation. As a result, ignoring the contributions of viscosity generally does not result in a visually completely inviscid flow, as the numerical dissipation often materializes similarly to how a small amount of viscosity would, at least in a visual sense. We therefore drop the viscosity term from Equation (3.1), which results in the *incompressible Euler equations* [Bridson, 2015]:

$$\frac{D\mathbf{u}}{Dt} + \frac{1}{\rho_f} \nabla p = \mathbf{f}, \quad (3.3)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (3.4)$$

The incompressible Euler equations model purely inviscid flow, and in graphics settings are often used to approximate fluid flow for low-viscosity fluids such as water. We will see later on in Section 4.3 that we will employ a technique known as operator splitting to further simplify and solve the incompressible Euler equations.

## 3.2 Potential Flow and Harmonicity

Underpinning the work by Da et al. [2016], and subsequently the work of this thesis, is the concept of potential flow. We provide a brief introduction to the principles of potential flow models and their implications for this work.

The velocity field of a fluid described by potential flow can be written as the gradient of some scalar potential function:

$$\mathbf{u} = \nabla \phi. \quad (3.5)$$

The curl of a velocity field  $\mathbf{u}$  defined in this way is always zero, since the curl of a gradient of a scalar is always equal to zero from basic vector calculus identities. Potential flow models therefore describe fluid flows that have no vorticity. That is, potential flow models implicitly satisfy the constraint

$$\nabla \times \mathbf{u} = 0. \quad (3.6)$$

By substituting the scalar field representation of the velocity field  $\mathbf{u}$  into the incompressibility condition from Equation (3.4), we can see that the scalar potential field must satisfy Laplace’s equation:

$$\nabla \cdot \mathbf{u} = \nabla \cdot (\nabla \phi) = \nabla \cdot \nabla \phi = 0. \quad (3.7)$$

That is,  $\phi$  is a harmonic function. Flows with velocity fields  $\mathbf{u}$  that are both incompressible (i.e., divergence-free) and irrotational (i.e., curl-free) can therefore be exactly represented by the gradient of a harmonic scalar potential function. We refer to such flows as *potential flow fields* for the sake of conciseness. We will see that the incompressible and irrotational nature of the velocity field  $\mathbf{u}$  will play a major role in the fluid model by Da et al. [2016] described further in Chapter 4.

### 3.3 Rigid Body Dynamics

As can be inferred from the name, rigid bodies refer to stiff objects that do not deform. While in the real world, even very stiff objects may undergo drastic deformation at very small timescales or under great forces (a classic example being a slow motion view of a golf ball being hit), these deformations are usually not the visual focus in graphics simulations. In graphics, for rigid bodies, we therefore remove unnecessary degrees of freedom that would allow for deformations, modeling them as purely undeformable objects. Rigid body dynamics are consequently described by a minimal set of degrees of freedom; a rigid body’s motion can be entirely represented with just its translation and its rotation about its center of mass. Notably, this view of rigid body dynamics implicitly enforces the rigidity of the solid(s) in question.

The evolution of the positional quantity of a rigid body is described following Newton’s second law:

$$\frac{d\mathbf{x}_{COM}}{dt} = \mathbf{v}, \quad (3.8)$$

$$\frac{d\mathbf{v}}{dt} = \frac{\mathbf{F}}{m}, \quad (3.9)$$

where  $\mathbf{x}_{COM}$  is the position of the rigid body’s center of mass,  $\mathbf{v}$  is the rigid body’s translational velocity,  $\mathbf{F}$  is net force acting on the rigid body, and  $m$  is mass of the rigid body.

The rotation, or the orientational quantity, of a rigid body is similarly described by

$$\frac{dR}{dt} = \omega^* R, \quad (3.10)$$

$$\frac{d\boldsymbol{\omega}}{dt} = I^{-1} \mathbf{T}, \quad (3.11)$$

where  $\boldsymbol{\omega}$  is the angular velocity and  $\mathbf{T}$  is torque. In 2D,  $R$  is the angle of rotation,  $I$  is the moment of inertia, and  $\omega^*$  is simply equal to  $\omega$ . In 3D,  $R$  is a 3-by-3 rotation matrix,  $I$  is the inertia tensor, and  $\omega^*$  is a skew-symmetric matrix that relates the angular velocity vector  $\boldsymbol{\omega}$  to the rate of change of the rotation matrix given by

$$\omega^* = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \quad (3.12)$$

For a force  $\mathbf{F}$  applied at a point  $\mathbf{x}$  on the surface of the rigid body, the torque is given by

$$\mathbf{T} = (\mathbf{x} - \mathbf{x}_{COM}) \times \mathbf{F}. \quad (3.13)$$

For further background on rigid body dynamics, we refer the reader to the course notes by [Baraff \[1997\]](#).

## 3.4 Two-way coupling

A two-way coupled fluid-rigid body system involves, as might be determined from the name, two interactions between the solid and the fluid in the system:

1. The solid exerts some force on the fluid, and as a result affects the flow of the fluid.
2. The fluid simultaneously exerts some pressure force on the the solid, and as a result the solid takes some momentum from the fluid.

If only one of these is true in a fluid-solid coupled system, we find ourselves in a one-way coupled system. The case in which only the fluid takes motion from solid can be useful for any setup in which we desire scripted solid motion. The opposite case, wherein the fluid alone forces the solid, can be a useful setup when the solid is significantly lighter than the fluid or if the solid is much smaller than the features of the flow that we are interested in, as is the case for marker particles.

We expand on our formulation for coupling rigid bodies with surface-only liquids in Chapter 5. For now, we summarize two-way coupling succinctly like so: solid to fluid coupling is achieved by enforcing a boundary condition on the normal component of the fluid velocities at the fluid-solid contact points, and fluid to solid coupling is achieved by calculating net force and torque on the solid from fluid pressures.

# Chapter 4

## Method of Surface-Only Liquids

For the comprehensiveness of this thesis, we include a summary of the contributions by Da et al. [2016], upon which this thesis builds. While we include the relevant details for the 3D simulation case described by Da et al. [2016], we mostly focus on the method of surface-only liquids for 2D fluid simulation as this and the extension to this 2D model to include rigid body coupling are the main contributions of this thesis. For further details regarding the 3D simulation case, we refer the reader back to the original work by Da et al. [2016].

### 4.1 Surface-only Representation of Fluid Flow

The surface-only fluid simulation technique builds on two assumptions placed on the velocity field that describes the fluid’s motion. First, as is common both in and outside of graphics applications, we assume that the simulated liquid is incompressible and therefore the velocity field must be *divergence-free*. Second, we assume that the liquid is irrotational and therefore the velocity field is also *curl-free*. Da et al. [2016] justify this second assumption by recognizing that for idealized inviscid liquids with uniform internal density, the primary sources of vorticity, that is, the curl of velocity, have negligible effects within the interior of the liquid volume. As introduced in Section 3.2, the lack of vorticity also means that this model is consistent with classic *potential flow* approximations, as zero vorticity implies that the velocity can be expressed as the gradient of a scalar field, called the velocity potential, since the curl of a gradient is always 0. We thus refer to a velocity field  $\mathbf{u}$  that is both incompressible and irrotational as a *potential flow field*:

**Definition 4.1.1** (Potential flow field). A velocity field  $\mathbf{u} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  or  $\mathbf{u} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  is a potential flow field if:

- $\nabla \cdot \mathbf{u} = 0$ ,
- $\nabla \times \mathbf{u} = 0$ ,

that is,  $\mathbf{u}$  corresponds to a flow that is both incompressible and irrotational. If  $\mathbf{u}$  is a 2D velocity field,  $\nabla \times \mathbf{u}$  is defined as the 2D curl and thus yields a scalar. If  $\mathbf{u}$  is a 3D velocity field,  $\nabla \times \mathbf{u}$  yields a vector.

We note that our terminology differs from that of [Da et al. \[2016\]](#), who refer to these incompressible and irrotational velocity fields as “harmonic velocity fields”. We find their terminology may be mildly misleading, as a “harmonic vector field” can refer to a field  $\mathbf{u}$  that itself satisfies Laplace’s equation, that is  $\nabla \cdot \nabla \mathbf{u} = 0$ , rather than an incompressible and irrotational field. These concepts are not unrelated, as it follows from vector calculus identities that an incompressible and irrotational field will satisfy Laplace’s equation, but since not every field that satisfies Laplace’s equation is necessarily both incompressible and irrotational, the “harmonic vector field” terminology remains slightly inaccurate.

In accordance with the definition of potential flow, a potential flow field  $\mathbf{u}$  over a contractible domain can be expressed as the gradient of a harmonic scalar function  $\phi$ . That is,  $\phi$  can be shown to satisfy Laplace’s equation  $\nabla \cdot \nabla \phi = 0$ . We will see this come into play later in the final step of our velocity update scheme, when we search for a harmonic scalar pressure field  $p$  that respects surface tension forces and other outside forces while keeping the velocity field harmonic. By using the the surface tension force as a Dirichlet-type boundary condition and the normal component of the surface velocity as a Neumann-type boundary condition, we will see that we establish a boundary value problem (BVP) for pressure  $p$  that has a unique solution. The key takeaway from this is that a potential flow field over a contractible domain can be fully described by its normal component over the domain’s surface.

## 4.2 Domain Representation and Discretization

The liquid domain and quantities related to the liquid domain are denoted by  $\Omega$ . The liquid domain’s surface,  $\partial\Omega$ , and quantities related to the surface are denoted by  $\Gamma$ . For both the 2D and 3D case,  $\Gamma$  is discretized into a mesh  $(V, F)$  where  $V$  is a set of vertices

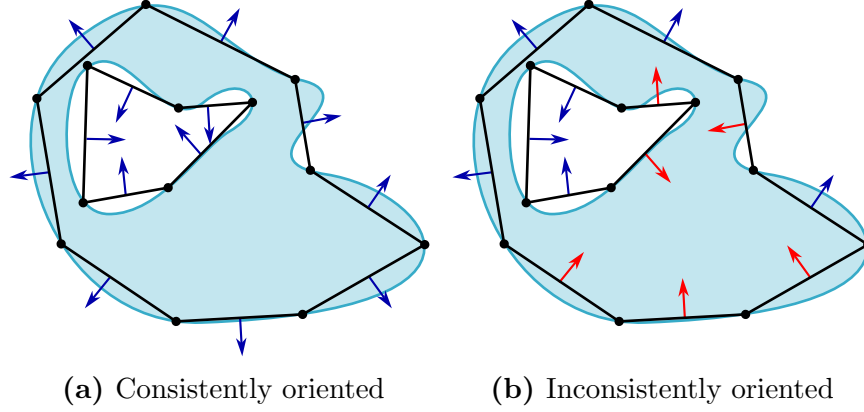


**Figure 4.1: Oriented mesh.** A sketch of a discretized 2D liquid domain with face normals shown with arrows on associated faces. In a consistently oriented mesh, all face normals point in a consistent direction relative to the interior of the liquid domain, whereas in an inconsistently oriented mesh, face normals can point both in and out of the domain.

and  $F$  is the set of faces. Let  $N_V$  denote the total number of mesh vertices, and  $N_F$  denote the total number of mesh faces.

In 2D, the boundary of each component of liquid volume of the 2D liquid domain may be represented by a one-dimensional curve, or curves in the case of a component with holes in it, embedded in  $\mathbb{R}^2$ . These curves can then be discretized into polygons. Vertices are located at positions  $\mathbf{x}_i \in \mathbb{R}^2$ . Faces can be visualized as line segments connecting pairs of vertices, and are identified by oriented pairs of vertex indices. “Oriented” simply means that normal vectors for each face, in this case line segment edges, point towards a consistent side of the mesh; see Figure 4.1 for a visual summary. In our implementation, we define normal vectors to point outward relative to the liquid mesh. Note that  $N_V = N_F$  holds in the 2D case, but this property is somewhat unique to the 2D case.

In 3D, the boundary of each component of liquid volume is a two-dimensional surface embedded in  $\mathbb{R}^3$ . We discretize each surface with a manifold triangle mesh. Vertices are located at positions  $\mathbf{x}_i \in \mathbb{R}^3$ , and each triangular face is identified by an oriented triad of vertex indices. Once again, we define normal vectors to point outwards relative to the liquid mesh.

When discretizing integrals over the surface, as is done in Sections 4.4.2 and 4.4.3, it becomes necessary to have some definition of a vertex’s area. We define a vertex’s area as the summation of area contributions from each of its adjacent faces, where in 2D each face contributes half of its length to each of its vertices’ areas, and in 3D each face contributes

a third of its area to each of its vertices' areas. That is, if  $F(i)$  is the set of faces directly adjacent to vertex  $\mathbf{x}_i$  and  $a_j$  is the length of face  $j$ , then the vertex area  $\hat{a}_i$  of  $\mathbf{x}_i$  is given in 2D as

$$\hat{a}_i = \frac{1}{2} \sum_{j \in F(i)} a_j, \quad (4.1)$$

and in 3D as

$$\hat{a}_i = \frac{1}{3} \sum_{j \in F(i)} a_j, \quad (4.2)$$

where in the 3D case,  $a_j$  refers to the area of face  $j$ .

### 4.3 Time Integration

We now present an overview of the time integration procedure. [Da et al. \[2016\]](#) utilize the well-established operator splitting technique, first introduced to graphics by [Stam \[1999\]](#), to simplify and solve the Euler equations (see Equations (3.3) and (3.4)) over the liquid domain. The Euler equations are separated into three main steps: advection, incompressible and irrotational projection, and external force integration. Mathematically, these steps are described as follow:

$$\frac{D\mathbf{u}}{Dt} = 0 \quad (\text{advection}), \quad (4.3)$$

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\rho_f} \nabla p &= 0 \\ \text{such that } \nabla \cdot \mathbf{u} &= 0 \end{aligned} \quad (\text{incompressible and irrotational projection}), \quad (4.4)$$

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{f} \\ \text{such that } \nabla \cdot \mathbf{u} &= 0. \end{aligned} \quad (\text{external forces and incompressibility}) \quad (4.5)$$

The reader more familiar with fluid simulation in a graphics context may recognize that the order in which these steps is performed is somewhat unconventional. Typically, fluid simulations will integrate external forces into the velocity field prior to projecting it down to the space of divergence-free (i.e., incompressible) velocity fields [[Bridson, 2015](#)]. [Da et al. \[2016\]](#) swap this order, performing a projection step to obtain an incompressible and irrotational field before integrating external forces. As a result it becomes imperative that

the external force integration is done in such a way that it preserves the incompressible and irrotational quality from the previous step, and in this way the typical projection step can be thought of as being spread over these two post-advection steps.

The projection step needs to be spread over two steps in this order rather than solved for at once as a result of the surface-only approach. In short, the force integration step is performed by setting up a Laplace-type BVP on pressures and solving this BVP with a BEM solve (see Section 4.3.3 for more details). For reference, the Laplace equation is given by

$$\nabla \cdot \nabla p = 0. \quad (4.6)$$

The BVP being Laplace-type is dependent on the input field being divergence-free; if the input field contained divergence, the BVP would have yielded a Poisson equation instead:

$$\nabla \cdot \nabla p = \nabla \cdot \mathbf{u}. \quad (4.7)$$

It is clear that the right-hand side of the equation above is only zero if there is no divergence in the input field  $\mathbf{u}$ . A Poisson equation with a non-zero right-hand side in the BVP would cause us to require volume integrals in the BEM solve, violating the surface-only approach. On the other hand, it is also not possible to simply perform the projection in one unified step, as it is unclear how to perform an incompressible and irrotational projection that also respects the boundary conditions at the free surface and solid boundaries using only information available at the surface. Thus, we have justified both the need for the projection to be split into two steps and the need for the incompressible and irrotational projection to precede the divergence-free external force integration step.

For notational clarity, as well as to summarize the steps of the surface-only liquid approach, we provide the overall algorithm prior to detailing each step in Algorithm 1. The remainder of this section outlines the time integration procedure; the spatial discretization is described in depth later on in Section 4.4.

### 4.3.1 Advection

At the beginning of each time step, we have as input the fluid domain's position decorated with a velocity field. This input velocity field is incompressible and irrotational, making it a potential flow field (Definition 4.1.1). The velocity field that the fluid moves through must be incompressible and irrotational throughout the simulation, and so each time step starts and ends with a potential flow field.

In the advection step, the fluid surface position is updated in accordance with the velocity field decorating it. This update is performed ignoring any forces, internal or

---

**Algorithm 1** Surface-only liquids

---

- 1: Initialize surface mesh with vertices  $\mathbf{x}^-$ . (Section 4.2)
  - 2: Initialize potential flow field  $\mathbf{u}^-$  per vertex.
  - 3: **loop**
  - 4:   Advect surface mesh:  $\bar{\mathbf{x}} = \mathbf{x}^- + \mathbf{u}^- \Delta t$ , implicitly updating  $\mathbf{u}^-$  to  $\bar{\mathbf{u}}$ . (Section 4.3.1)
  - 5:   Perform incompressible and irrotational projection (HHD) on  $\bar{\mathbf{u}}$  to get  $\mathbf{u}$ . (Section 4.3.2)
  - 6:   Apply body forces (gravity):  $\tilde{\mathbf{u}} = \bar{\mathbf{u}} + \Delta t \mathbf{g}$ . (Section 4.3.3)
  - 7:   Perform projection with external force integration on  $\tilde{\mathbf{u}}$  to get  $\mathbf{u}^+$ . (Section 4.3.3)
  - 8: **end loop**
- 

external, in the velocity field. Both we and Da et al. [2016] use the Forward Euler scheme in our implementations:

$$\bar{\mathbf{x}} = \mathbf{x}^- + \mathbf{u}^- \Delta t, \quad (4.8)$$

where  $\mathbf{x}$  is position,  $\mathbf{u}$  is the velocity, and  $\Delta t$  is the time step size. The overbar notation denotes post-advection quantities and the superscripted subtraction symbol denotes beginning of time step quantities. While only the fluid surface positions are explicitly updated in this step, with the velocity associated at each surface point unchanged, the overall velocity field represented by the velocities decorating the surface is altered by the advection step due to the positional update. As a result, although the input field to the step  $\mathbf{u}^-$  is incompressible and irrotational, the field  $\bar{\mathbf{u}}$  following the advection step is not necessarily incompressible and irrotational. That is, velocity divergence and/or curl may have been introduced by the update to the surface position. We thus follow the advection step with an incompressible and irrotational velocity projection.

Forward Euler is a simple explicit method for numerical integration. It is possible that a different time integration scheme could yield improved behaviour. We return to this consideration in Section 7.2 when discussing future work.

### 4.3.2 Incompressible and Irrotational Projection

The velocity field represented by the positions  $\bar{\mathbf{x}}$  and velocities  $\bar{\mathbf{u}}$  of the fluid surface after the advection step may no longer be a potential flow field. To return the velocity field to the space of potential flow fields we must remove any divergence or curl introduced by advection by projecting  $\bar{\mathbf{u}}$  to the subspace of incompressible and irrotational velocity fields.

We note here that Da et al. [2016] take a slightly different view of this step, as they

regard it as solely projecting out divergence; however, because of the way the projection is constructed, the net effect is the same in both our view and that of [Da et al. \[2016\]](#). [Da et al. \[2016\]](#) conceptualize the projection step as searching for a velocity field  $\mathbf{u}$  that is both divergence-free (so  $\nabla \cdot \mathbf{u} = 0$ ) and whose difference with the post-advection field  $\bar{\mathbf{u}}$  is exactly the divergence present in  $\bar{\mathbf{u}}$  (that is,  $\mathbf{u} \perp (\bar{\mathbf{u}} - \mathbf{u})$ ). We instead conceptualize the projection step as a projection to the nearest potential flow field.

Several works in graphics literature perform the divergence-free projection step by a Poisson solve to find the pressure whose gradient exactly counteracts the divergence in the velocity field. We and [Da et al. \[2016\]](#) proceed with a slightly different approach. Instead of a Poisson solve, we begin with a constructive Helmholtz decomposition (HHD) [[Bhatia et al., 2013](#)] on the output field from the advection step  $\bar{\mathbf{u}}$ :

$$\bar{\mathbf{u}} = -\nabla\Phi + \nabla \times \mathbf{A}, \quad (4.9)$$

where  $\Phi$  and  $\mathbf{A}$  are the scalar and vector potential, respectively. This allows us to conceptualize  $\bar{\mathbf{u}}$  as being composed of a curl-free component  $\nabla\Phi$  (the curl of a gradient is always 0) and a divergence-free component  $\nabla \times \mathbf{A}$  (the divergence of a curl is always 0). Both  $\Phi$  and  $\mathbf{A}$  are given as being composed of two integrals, one over the surface and one over the volume:

$$\Phi(\mathbf{x}) = \underbrace{\int_{\Gamma} \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}}}_{\Phi_{\Gamma}} - \underbrace{\int_{\Omega} \nabla_{\mathbf{y}} \cdot \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dV_{\mathbf{y}}}_{\Phi_{\Omega}}, \quad (4.10)$$

$$\mathbf{A}(\mathbf{x}) = \underbrace{\int_{\Gamma} \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}}}_{\mathbf{A}_{\Gamma}} - \underbrace{\int_{\Omega} \nabla_{\mathbf{y}} \times \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dV_{\mathbf{y}}}_{\mathbf{A}_{\Omega}}, \quad (4.11)$$

where  $\mathbf{n}$  is the outward unit surface normal and  $G(\mathbf{x}, \mathbf{y})$  is the free-space Green's function of Laplace's equation, also known as the fundamental solution of Laplace's equation. The equations for  $G(\mathbf{x}, \mathbf{y})$  are included in Sections [4.4.2.1](#) and [4.4.2.2](#), when we discuss the formulation of the spatial discretization of these steps.

We note here that in the 2D case  $\mathbf{A}$  is a scalar potential, not a vector potential, since the 2D cross products in the integrals in  $\mathbf{A}$  yield scalars; from now on, we will use  $A$  to distinguish this from the vector potential  $\mathbf{A}$  that appears in the 3D case. The 2D cross product of two 2D vectors is derived from conceptually padding the 2D vectors with zeros in their  $z$ -components so that they are technically 3D vectors, and then taking the cross product of the 3D vectors. The scalar it yields thus represents the length of the vector that

points in the  $z$ -direction that is the result of this 3D cross product. The 2D cross product of two 2D vectors  $\mathbf{a}$  and  $\mathbf{b}$  is given by

$$\mathbf{a} \times \mathbf{b} = a_x b_y - a_y b_x. \quad (4.12)$$

As such, in the 2D case, the term in Equation (4.9) that takes the curl of the vector potential (the  $\nabla \times \mathbf{A}$  term) should instead be thought of as taking the curl of a scalar field  $A$ , which can be interpreted as rotating the gradient of the field by  $\frac{\pi}{2}$ . That is, we can rewrite Equation (4.9) as

$$\bar{\mathbf{u}} = -\nabla\Phi + J\nabla A, \quad (4.13)$$

where  $J$  is the 2D rotation matrix that rotates clockwise by  $\frac{\pi}{2}$  radians:

$$J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad (4.14)$$

and scalar potential  $A$  is given by

$$A(\mathbf{x}) = \underbrace{\int_{\Gamma} \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}}}_{A_{\Gamma}} - \underbrace{\int_{\Omega} \nabla_{\mathbf{y}} \times \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dV_{\mathbf{y}}}_{A_{\Omega}}, \quad (4.15)$$

where the difference between the scalar potential  $A$  and vector potential  $\mathbf{A}$  is that the cross products in  $A$  should be interpreted as 2D cross products. This difference in the interpretation of the curl of  $\mathbf{A}$  or  $A$  does not really come into play until discussing the 2D spatial discretization in detail in Section 4.4.2.1. Up until that point we will use the  $\nabla \times \mathbf{A}$  notation to denote the curl of this field, but when this appears it should be interpreted as  $J\nabla A$  when considering the 2D scenario.

As another brief aside, by definition the free-space Green's function of Laplace's equation satisfies:

$$\nabla \cdot \nabla G = \delta_D, \quad (4.16)$$

where  $\delta_D$  is the Dirac delta. That is,  $\nabla \cdot \nabla G$  has value of 0 everywhere except when  $G = 0$  which occurs when  $\mathbf{x} = \mathbf{y}$  ( $G$  is given later by Equations (4.38) and (4.43)), where its value is undefined, and the integral of  $\nabla \cdot \nabla G$  over the entire real line is equal to 1.  $G$  is useful in that it is simple to construct solutions of variations of the original equation using convolutions (such as the integrals in Equations (4.10) and (4.11)) of it. That is,  $G$  can be used to build solutions of the Poisson equation:

$$\nabla \cdot \nabla u = f,$$

as long as  $f$  is reasonably behaved. In any case, the property laid out in Equations (4.16) gives us the basis for the proofs for Equations (4.10) and (4.11).

Returning to  $\bar{\mathbf{u}}$ , we observe we may now write Equation (4.9) as:

$$\bar{\mathbf{u}} = \underbrace{\nabla \times \mathbf{A}_\Gamma - \nabla \Phi_\Gamma}_{\text{div-free and curl-free}} + \underbrace{\nabla \Phi_\Omega}_{\text{curl-free}} - \underbrace{\nabla \times \mathbf{A}_\Omega}_{\text{div-free}}. \quad (4.17)$$

The three orthogonal components are: an incompressible and irrotational (div-free and curl-free) component, a curl-free (but not div-free) component, and a div-free (but not curl-free) component. The orthogonality of these components is shown in the Appendix of the original paper by Da et al. [2016]. Since the  $\nabla \times \mathbf{A}_\Gamma - \nabla \Phi_\Gamma$  component is in fact a potential flow field, we identify it as the incompressible and irrotational post-advection, post-projection velocity field that we were looking for:

$$\mathbf{u} = \nabla \times \mathbf{A}_\Gamma - \nabla \Phi_\Gamma. \quad (4.18)$$

As both  $\mathbf{A}_\Gamma$  and  $\nabla \Phi_\Gamma$  are surface integrals, we see that  $\mathbf{u}$  can be computed at any point in the domain  $\Omega$  using only information available on the surface of the liquid.

Da et al. [2016] further simplify Equation (4.19) by noting that the pressure gradient in the Euler equations (see Equations (3.3) and (3.4)) is always normal to the liquid surface under the assumption of free-surface flow. In other words,  $\nabla p$  does not have a tangential component. As a result there should be no change in the tangential velocity between  $\bar{\mathbf{u}}$  and  $\mathbf{u}$ , and rather than using the entirety of Equation (4.19) to find  $\mathbf{u}$ , we can instead copy over the tangential component of  $\bar{\mathbf{u}}$  and take just the normal component of  $\mathbf{u}$  from Equation (4.19), denoted as  $u^n$ . The post-projection velocity field  $\mathbf{u}$  is then given by

$$\mathbf{u} = P\bar{\mathbf{u}} + u^n \mathbf{n}, \quad (4.19)$$

$$u^n = \mathbf{n} \cdot (\nabla \times \mathbf{A}_\Gamma - \nabla \Phi_\Gamma), \quad (4.20)$$

where

$$P = I - \mathbf{n}\mathbf{n}^T \quad (4.21)$$

is the tangential projection operator.  $I$  denotes the identity matrix. We will see that this treatment of  $\mathbf{u}$  will allow us to make simplifications in our discretizations in Section 4.4.2.

### 4.3.3 External force integration

After recovering the incompressible and irrotational post-projection velocity field  $\mathbf{u}$ , the final step in the time integration is to integrate external forces into  $\mathbf{u}$  and obtain the end

of step velocity  $\mathbf{u}^+$ . This needs to be done somewhat carefully, as we must ensure that the velocity field remains incompressible throughout this step.

We begin by updating  $\mathbf{u}$  to  $\tilde{\mathbf{u}}$  with an intermediate step that incorporates body forces using a Forward Euler step:

$$\tilde{\mathbf{u}} = \mathbf{u} + \mathbf{g}\Delta t, \quad (4.22)$$

where the body force  $\mathbf{g}$  is typically just gravitational acceleration, but can include any constant force that we would like to apply throughout the volume of the liquid. While it is possible to construct a non-uniform body force, such a force could potentially violate incompressibility and is thus not viable here. Since we assume a constant body force, this is a purely translational update; that is, no curl or divergence is introduced by this addition, so  $\tilde{\mathbf{u}}$  is still a potential flow field.

Since we assume our fluid flow is a potential flow field, any change in the velocity field caused by external forces must also be representable as a potential flow field and as a result can be represented by a gradient of a scalar harmonic field. That is, the change in  $\tilde{\mathbf{u}}$  from the external forces is given by some  $\nabla p$ , where  $p$  is a harmonic pressure field. As  $p$  is a harmonic scalar field, it must satisfy Laplace's equation. Given appropriate boundary conditions, we can therefore set up a Laplace-type boundary value problem (BVP) to solve for  $\nabla p$ .

The boundary  $\Gamma$  of the liquid domain comprises of two parts: a liquid-air boundary  $\Gamma_A$  and a solid-liquid boundary  $\Gamma_S$ . Together, the two interfaces satisfy  $\Gamma_A \cup \Gamma_S = \Gamma$ . Points that fall into both categories, that is, points in  $\Gamma_A \cap \Gamma_S$ , are known as triple points. For a 2D simulation, triple points are simply a point on the surface of the mesh. For a 3D simulation, triple points fall along a curve called the triple junction.

Apart from body forces, [Da et al. \[2016\]](#) model forces that act normal to the surface of the liquid. Along the liquid-air interface, they consider the surface tension force induced by the curvature of the liquid surface and the liquid's surface tension coefficient. Considering this in terms of  $p$ , we can see that this establishes a Dirichlet boundary condition on  $\Gamma_A$ . Along the solid-liquid interface, they consider the normal force caused by the interaction of the solid and the liquid. This establishes a Neumann boundary condition on  $\Gamma_S$ , as the solid and its velocity places a restriction on the normal component of the pressure gradient. With these observations, [Da et al. \[2016\]](#) set up the BVP below:

$$\nabla \cdot \nabla p = 0, \quad \text{subject to} \quad (4.23)$$

$$p = \sigma H, \quad \text{on } \Gamma_A \quad (4.24)$$

$$\frac{\partial p}{\partial \mathbf{n}} = \frac{\rho_f}{\Delta t}(\tilde{\mathbf{u}} - \mathbf{u}_S) \cdot \mathbf{n}, \quad \text{on } \Gamma_S \quad (4.25)$$

where  $H$  is the signed curvature (in 2D) or the signed mean curvature (in 3D),  $\rho_f$  is fluid density, and  $\mathbf{u}_S$  is the velocity of the solid evaluated at the solid-liquid interface. In our 2D implementation, we use the following formula for discrete signed curvature at a point  $\mathbf{x}_i$ :

$$H_i = \frac{2\psi}{\|d_{i-1/2}\| + \|d_{i+1/2}\|},$$

where  $\psi$  is the signed turning angle and  $d_{i-1/2}$  and  $d_{i+1/2}$  are the lengths of the adjacent previous and next edges, respectively. Additional formulas for discrete signed curvature in 2D and some of their properties can be found in the notes by Vouga [2014]. Da et al. [2016] refer to Mantic [1993] for a discretization of signed mean curvature in 3D.

Solving BVP (4.23) yields the  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values for each point on the liquid surface. With these values, it is possible to construct  $\nabla p$ , the gradient of a harmonic scalar function that will update  $\tilde{\mathbf{u}}$  to the end-of-step velocity. The normal component of  $\nabla p$  is given by  $\nabla p \cdot \mathbf{n}$ ; we notice that this is simply the directional derivative of pressure in the normal direction  $\frac{\partial p}{\partial \mathbf{n}}$  that we calculated from boundary condition (4.25) or found from the solve. We approximate the tangential component of  $\nabla p$ , which we refer to as  $\frac{\partial p}{\partial \mathbf{t}}$ , using adjacent  $p$  values, which are also either known from boundary condition (4.24) or found from the solve.

We then construct the pressure gradient out of these two components, multiplied by the unit normal  $\mathbf{n}$  or unit tangent  $\mathbf{t}$  as appropriate:

$$\nabla p = \frac{\partial p}{\partial \mathbf{n}} \mathbf{n} + \frac{\partial p}{\partial \mathbf{t}} \mathbf{t}, \quad (4.26)$$

and use the pressure gradient to obtain the end-of-step velocity  $\mathbf{u}^+$ :

$$\mathbf{u}^+ = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho_f} \nabla p, \quad (4.27)$$

which, as desired, is a potential flow field.

## 4.4 Spatial Discretization

We now discuss how we spatially discretize each of the time integration steps discussed in the previous section. Due to differences in the Green's functions between the 2D and 3D case, discretization details diverge slightly for these two cases in both the incompressible

and irrotational projection and external force integration steps. These details will be discussed as they appear.

Recall from Section 4.2 that the liquid domain is represented by a discrete surface mesh, with each mesh point carrying information about its position and velocity.

#### 4.4.1 Advection

Step 4 in Algorithm 1 is the advection step. Advection is straightforward. We simply apply the advection scheme, in this case Equation (4.8), to each of the discrete surface mesh points  $\mathbf{x}_i$ :

$$\bar{\mathbf{x}}_i = \mathbf{x}_i^- + \mathbf{u}_i^- \Delta t, \quad (4.28)$$

where  $\mathbf{u}_i$  denotes the liquid velocity associated with and stored at  $\mathbf{x}_i$ .

#### 4.4.2 Incompressible and Irrotational Projection

In Step 5 of Algorithm 1, we perform the incompressible and irrotational projection. The evaluation required in the projection step is that of  $u^n$  from Equation (4.20), reproduced below:

$$u^n = \mathbf{n} \cdot (\nabla \times \mathbf{A}_\Gamma - \nabla \Phi_\Gamma), \quad (4.20)$$

where the BIEs  $\Phi_\Gamma$  and  $\mathbf{A}_\Gamma$  from Equations (4.10) and (4.11) are given:

$$\Phi_\Gamma = \int_\Gamma \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}}, \quad (4.29)$$

$$\mathbf{A}_\Gamma = \int_\Gamma \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}}. \quad (4.30)$$

Due to differences between the 2D and 3D Green's function  $G(\mathbf{x}, \mathbf{y})$  (see Equations (4.38) and (4.43)) as well as the differences in the singularities that arise related to them, the techniques to spatially discretize the projection differ slightly for each case. Before discussing these details, we discuss the discretization steps up to the point of this divergence.

Da et al. [2016] begin the spatial discretization for Equation (4.20) by integrating it over the surface using the basis functions  $\theta_i$ :

$$\hat{a}_i u_i^n = \int_\Gamma \theta_i u^n dS = \int_\Gamma \theta_i (\mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma - \mathbf{n} \cdot \nabla \Phi_\Gamma) dS, \quad (4.31)$$

where  $u_i^n$  is the discretization of  $u_n$  at vertex  $\mathbf{x}_i$  and  $\theta_i$  is the basis function for vertex  $\mathbf{x}_i$ . We elect to use linear basis functions, or hat functions, for  $\theta_i$ . As the basis function for vertex  $\mathbf{x}_i$  evaluates to 0 at all faces that are not adjacent to  $\mathbf{x}_i$  (that is, all faces that are not in  $F(i)$ ), the outer surface integral in Equation (4.31) is equivalent to a summation of the integral evaluated over just the neighboring faces:

$$\int_{\Gamma} \theta_i \mathbf{n}_{\mathbf{x}} \cdot \nabla \Phi_{\Gamma} dS_{\mathbf{x}} = \sum_{j \in F(i)} \int_j \theta_i \mathbf{n}_{\mathbf{x}} \cdot \nabla \Phi_{\Gamma} dS_{\mathbf{x}}, \quad (4.32)$$

$$\int_{\Gamma} \theta_i \mathbf{n}_{\mathbf{x}} \cdot \nabla \times \mathbf{A}_{\Gamma} dS_{\mathbf{x}} = \sum_{j \in F(i)} \int_j \theta_i \mathbf{n}_{\mathbf{x}} \cdot \nabla \times \mathbf{A}_{\Gamma} dS_{\mathbf{x}}. \quad (4.33)$$

We have introduced  $\mathbf{x}$  into the above notation for clarity in the following steps, as the inner  $\Phi_{\Gamma}$  and  $\mathbf{A}_{\Gamma}$  are also both integrals over the surface.

We make one more critical observation regarding Equation (4.32). We recognize the  $\mathbf{n}_{\mathbf{x}} \cdot \nabla \Phi_{\Gamma}$  term as a normal derivative with respect to the outer integral. That is,  $\mathbf{n}_{\mathbf{x}} \cdot \nabla \Phi_{\Gamma} = \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} \Phi_{\Gamma}$ . We cannot, however, simply move the partial derivative into the integral  $\Phi_{\Gamma}$  in our evaluation of this term. While this is valid at points *within* the domain  $\Omega$ , it is invalid for points exactly *on* the boundary  $\Gamma$  due to the existence of a jump discontinuity in the normal derivative across the boundary [Sugimoto et al., 2023]. Since the outer integral in Equation (4.32) integrates over the boundary, it is this latter case that is relevant to our evaluation. When taking the directional derivative  $\frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}}$  of  $\Phi_{\Gamma}$ , we therefore need to take the limit as  $\mathbf{x}$  approaches the boundary, which results in the integral equation

$$\frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} \Phi_{\Gamma}(\mathbf{x}) = \lim_{\mathbf{x} \rightarrow \Gamma} \int_{\Gamma} \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}} \quad (4.34)$$

$$= \Theta(\mathbf{x}) (\mathbf{n}(\mathbf{x}) \cdot \bar{\mathbf{u}}(\mathbf{x})) + \int_{\Gamma} \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}} \quad \text{for } \mathbf{x} \in \Gamma, \quad (4.35)$$

where  $\Theta(\mathbf{x})$  is the internal solid angle. Internal solid angle is trivial in 2D, as it is simply the interior angle that the edges stemming from vertex  $\mathbf{x}$  create. For the 3D case, Da et al. [2016] refer to Mantic [1993] for the discretization of the internal solid angle. This  $\Theta(\mathbf{x}) (\mathbf{n}(\mathbf{x}) \cdot \bar{\mathbf{u}}(\mathbf{x}))$  term is referred to as the *charge term*. With the above, we expand and rewrite the right hand side of Equation (4.32) as

$$\begin{aligned} \int_{\Gamma} \theta_i \mathbf{n} \cdot \nabla \Phi_{\Gamma} dS &= \sum_{j \in F(i)} \int_j \theta_i \left( \Theta(\mathbf{x}) (\mathbf{n}(\mathbf{x}) \cdot \bar{\mathbf{u}}(\mathbf{x})) \right. \\ &\quad \left. + \int_{\Gamma} \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}} \right) dS_{\mathbf{x}}. \end{aligned} \quad (4.36)$$

Rewriting the inner integral into a sum over face integrals then gives

$$\int_{\Gamma} \theta_i \mathbf{n} \cdot \nabla \Phi_{\Gamma} dS = \sum_{j \in F(i)} \int_j \theta_i \left( \Theta(\mathbf{x}) (\mathbf{n}(\mathbf{x}) \cdot \bar{\mathbf{u}}(\mathbf{x})) + \left( \sum_{k \in F} \int_k \mathbf{n}(\mathbf{y}) \cdot \bar{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} G(\mathbf{x}, \mathbf{y}) dS_{\mathbf{y}} \right) \right) dS_{\mathbf{x}}. \quad (4.37)$$

#### 4.4.2.1 2D discretization

The 2D Green's function is given by

$$G(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi} \ln \|\mathbf{x} - \mathbf{y}\|. \quad (4.38)$$

Its gradient  $\nabla G(\mathbf{x}, \mathbf{y})$  can be found to be

$$\nabla G(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi} \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|^2}. \quad (4.39)$$

**Scalar potential  $\Phi_{\Gamma}$  BIE** We begin by considering Equation (4.37). Consider the discretization of the outer integral in Equation (4.37), which integrates over a face (in 2D, a line segment edge). All the values to compute the charge term are simple to obtain. The normal derivative of  $G(\mathbf{x}, \mathbf{y})$  can be written as  $\mathbf{n}_{\mathbf{x}} \cdot \nabla G(\mathbf{x}, \mathbf{y})$ , where  $\nabla G(\mathbf{x}, \mathbf{y})$  is given above. However, the outer integral is not straightforward to approximate due to the fact that the inner integral, which is a component of the outer integral's integrand, diverges to infinity in a weakly singular manner near vertex  $\mathbf{x}_i$ . To handle this divergence of the integrand we use the tanh-sinh quadrature, also known as the double-exponential (DE) formula, to approximate the outer integral [Aimi et al., 1997]. Tanh-sinh quadrature is a numerical approximation scheme that removes endpoint singularities by mapping the endpoints to positive and negative infinity on the real line. We provide the quadrature and the step size we used in the 2D implementation in Appendix A.1.

While  $G(\mathbf{x}, \mathbf{y})$  has a singularity when  $\mathbf{x}$  and  $\mathbf{y}$  coincide, when  $\mathbf{x}$  and  $\mathbf{y}$  are on the same face the vector  $\mathbf{x} - \mathbf{y}$  is exactly orthogonal to the normal vector  $\mathbf{n}_{\mathbf{x}}$ . As a result the  $\mathbf{n}_{\mathbf{x}} \cdot \nabla G(\mathbf{x}, \mathbf{y})$  term evaluates to 0 on the face where the singularity would be found, and this face does not contribute to the integral. We are thus able to discretize the inner integral in a fairly straightforward manner with a Gaussian quadrature. We find a one-point per face Gaussian quadrature to be sufficient in our 2D implementation.

**Scalar potential  $A_\Gamma$  BIE** Next, we consider Equation (4.33), the BIE involving the scalar potential  $A_\Gamma$ . As was discussed in Section 4.3.2, the curl of this potential should be interpreted as the gradient of  $A_\Gamma$  rotated  $\frac{\pi}{2}$  clockwise. That is, in the 2D case Equation (4.33) should be thought of as

$$\int_{\Gamma} \theta_i \mathbf{n}_x \cdot J \nabla A_\Gamma dS_x = \sum_{j \in F(i)} \int_j \theta_i \mathbf{n}_x \cdot J \nabla A_\Gamma dS_x, \quad (4.40)$$

where  $J$  is the rotation matrix

$$J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \quad (4.41)$$

We move the rotation by  $J$  onto the unit normal  $\mathbf{n}_x$ , and, recognizing that a  $\frac{\pi}{2}$  rotation of the normal vector is simply the tangent vector, we replace  $\mathbf{n}_x \cdot J \nabla A_\Gamma$  with  $\mathbf{t}_x \cdot \nabla A_\Gamma$ . We can then rewrite and expand Equation (4.40) as

$$\begin{aligned} \int_{\Gamma} \theta_i \mathbf{n}_x \cdot J \nabla A_\Gamma dS_x &= \sum_{j \in F(i)} \int_j \theta_i \mathbf{t}_x \cdot \nabla A_\Gamma dS_x \\ &= \sum_{j \in F(i)} \int_j \theta_i \frac{\partial}{\partial \mathbf{t}_x} \nabla A_\Gamma dS_x \\ &= \sum_{j \in F(i)} \int_j \theta_i \int_{\Gamma} \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{t}_x} G(\mathbf{x}, \mathbf{y}) dS_y dS_x \\ &= \sum_{j \in F(i)} \int_j \theta_i \int_{\Gamma} \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) \mathbf{t}_x \cdot \nabla G(\mathbf{x}, \mathbf{y}) dS_y dS_x. \end{aligned}$$

Rewriting the inner integral as a summation of the face integrals gives

$$\int_{\Gamma} \theta_i \mathbf{n}_x \cdot J \nabla A_\Gamma dS_x = \sum_{j \in F(i)} \int_j \theta_i \left( \sum_{k \in F} \int_k \mathbf{n}(\mathbf{y}) \times \bar{\mathbf{u}}(\mathbf{y}) \mathbf{t}_x \cdot \nabla G(\mathbf{x}, \mathbf{y}) dS_y \right) dS_x. \quad (4.42)$$

To discretize the outer integral we once again use tanh-sinh quadrature.

As for numerically approximating the inner integral, we must be careful of the singularity in the Green's function when  $\mathbf{x}$  and  $\mathbf{y}$  coincide. To handle this singularity in our implementation, when  $\mathbf{x}$  and  $\mathbf{y}$  are on the same face, we “split” the face around  $\mathbf{x}$  and perform two integrals over each side of the face around  $\mathbf{x}$  so that each of these sub-integrals has the singularity at their endpoint. For these we again use tanh-sinh quadrature, eliminating the singularity. When  $\mathbf{x}$  and  $\mathbf{y}$  are on different faces and we do not need to consider the possibility of a singularity, we simply use a four-point Gaussian quadrature. The four-point Gaussian quadrature points and weights are provided in Appendix A.2.

#### 4.4.2.2 3D discretization

The 3D Green's function is given by

$$G(\mathbf{x}, \mathbf{y}) = -\frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|}. \quad (4.43)$$

Its gradient is then

$$\nabla G(\mathbf{x}, \mathbf{y}) = -\frac{1}{4\pi} \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|^3}. \quad (4.44)$$

**Scalar potential  $\Phi_\Gamma$  BIE** The discretization of Equation (4.37), the BIE containing the scalar potential  $\Phi_\Gamma$ , in the 3D scenario is fairly similar to its 2D counterpart. Again, the contributions from the charge term are simple to obtain. As for the normal derivative of  $G(\mathbf{x}, \mathbf{y})$ , we can once again write it as  $\mathbf{n}_\mathbf{x} \cdot \nabla G(\mathbf{x}, \mathbf{y})$  and recognize that the inner integral is trivially discretized as this term evaluates to 0 on faces where singularities would occur. Da et al. [2016] approximate the inner integral directly with a one-point per face Gaussian quadrature in their implementation.

As was the case in the 2D scenario, since the inner integral itself diverges near vertex  $\mathbf{x}_i$  in a weakly singular fashion, the discretization of the outer integral is again not quite as trivial. To accurately perform the numerical approximation, Da et al. [2016] use a Duffy quadrature technique, wherein they first use a Duffy transformation to remove the vertex singularity on the face prior to performing a Gaussian quadrature on the transformed face. The Duffy transformation removes the singularity by a simple variable transformation that essentially maps the triangular face with a vertex singularity to a square with no singularity. A straightforward summary of the Duffy transformation can be found in the work of Keeler and Bridson [2014]. Da et al. [2016] use a four-point Gaussian quadrature following the Duffy transformation.

**Vector potential  $\mathbf{A}_\Gamma$  BIE** Some treatment is required to accurately evaluate Equation (4.33) in 3D as taking the curl of the  $\mathbf{A}_\Gamma$  term introduces a strongly singular integrand. Da et al. [2016] address this by first transforming the integral using integration by parts to move the derivative, and then eliminating one of the resultant terms through observing cancellations that occur due to the mesh structure. They arrive at the reformulation

$$\int_j \theta_i \mathbf{n}_\mathbf{x} \cdot \nabla \times \mathbf{A}_\Gamma dS_\mathbf{x} = - \int_j \mathbf{n}_\mathbf{x} \cdot (\nabla \theta_i \times \mathbf{A}_\Gamma) dS_\mathbf{x}. \quad (4.45)$$

For details we refer the reader to Equations (17) through (20) in the paper by Da et al. [2016]. The above reformulation now only has a weakly singular integrand, which can therefore be accurately approximated using the same Duffy quadrature technique established for the scalar potential BIE. Da et al. [2016] again use a four-point Gaussian quadrature.

### 4.4.3 External force integration

In Steps 6 and 7 of Algorithm 1, we integrate external forces while keeping the velocity field incompressible and irrotational. Applying body forces is straightforward; we simply apply Equation (4.22) to each of the velocities  $\mathbf{u}_i$  stored on the discrete surface mesh points  $\mathbf{x}_i$ :

$$\tilde{\mathbf{u}}_i = \mathbf{u}_i + \mathbf{g}\Delta t. \quad (4.46)$$

To construct the pressure gradient that updates the post-incompressible and irrotational projection velocity to the end-of-step velocity in Equation (4.27), we approximate the pressure gradient's tangential and normal components along the boundary of the liquid domain. This requires that we find both the pressure and the pressure normal derivative at each discrete point of the surface mesh. As can be seen from (4.24) and (4.25), at a point  $\mathbf{y} \in \Gamma$  either  $p(\mathbf{y})$  or  $\frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}}$  is given by the boundary condition depending on whether  $\mathbf{y} \in \Gamma_A$  or  $\mathbf{y} \in \Gamma_S$ ; the other quantity is unknown. Triple points, where both  $\mathbf{y} \in \Gamma_A$  and  $\mathbf{y} \in \Gamma_S$  are true, are a somewhat special case and require some additional treatment; we discuss these later on in Subsection 4.4.3.3. To find the unknown quantities and construct the pressure gradient we solve the Laplace BVP set up in Section 4.3.3.

We use the Boundary Element Method (BEM) technique to solve the Laplace BVP using only a surface discretization. BEM relies on formulating the partial differential equation as an integral equation. We thus begin with the direct boundary integral equation (BIE) for the Laplace BVP for pressure [Atkinson, 1997; Da et al., 2016] described by Equation (4.23), given by

$$\Theta(\mathbf{x})p(\mathbf{x}) = \int_{\Gamma} \left( p(\mathbf{y}) \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} G(\mathbf{x}, \mathbf{y}) \right) dS_\mathbf{y}, \quad (4.47)$$

where  $\Theta(\mathbf{x})$ , we recall, is the internal solid angle of the liquid domain  $\Omega$  at the point  $\mathbf{x} \in \Gamma$ .

The unknown pressure values and normal components of the pressure gradient are found by discretizing BIE (4.47) using a vertex-located linear isoparametric BEM formulation. *Vertex-located* simply indicates that the pressure values and normal components of the pressure gradient are associated with and solved for at the vertices (rather than, for example, the faces). The formulation is *isoparametric* because both the boundary and the

unknown values have the same linear form in terms of their parameterizations; the unknown values are linearly interpolated between the vertex points. The BEM applied to our problem can be summarized in a few steps:

1. The boundary  $\Gamma$  is discretized into  $N$  non-overlapping boundary elements; in our case, the boundary elements correspond exactly with the mesh faces. The pressures and normal components of the pressure gradient are thought of as linearly interpolated between their nodal values (i.e., their values at mesh vertices).
2. We apply BIE (4.47) to each vertex and discretize it such that it is expressed as a sum over face integrals like so:

$$\Theta(\mathbf{x})p(\mathbf{x}) = \sum_{j=1}^N \int_{F_j} p(\mathbf{y}) \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_y} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_y} G(\mathbf{x}, \mathbf{y}) dS_y, \quad (4.48)$$

where  $\mathbf{x}$  is a vertex on the mesh and  $F_j$  denotes the  $j$ -th face in the discretized mesh boundary. In our formulation, each face integral can be thought of as “contributing” some amount to each of the vertices it is adjacent to; we make note of this here to reiterate that we formulate a system where the unknowns are associated with the vertices. This yields an underdetermined system of  $N_V$  equations, with  $N_V$   $p$  values and  $N_V$   $\frac{\partial p}{\partial \mathbf{n}}$  values.

3. Applying our boundary conditions allows for the system to become determined, as we introduce one known  $p$  or  $\frac{\partial p}{\partial \mathbf{n}}$  for each vertex (so  $N_V$  of the previously unknown values in the system from the previous step are now known). We then solve the system. This yields the unknown  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values at each vertex.

Triple points are unique in that the boundary conditions at these points must be formulated and handled slightly differently. We discuss triple point handling in greater detail later in Section 4.4.3.3. For now, for clarity of discussion, we will continue this description under the assumption that all points are strictly either on the liquid-air interface or the solid-liquid interface.

We use the nodal  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values from the boundary conditions and the BEM solve to approximate the pressure gradient needed in Equation (4.27). Some finer points of this step will be detailed later in Section 4.4.3.4.

We now include a fairly comprehensive description of the linear isoparametric BEM for the 2D case. We follow it with a summary of the 3D case, which has differences in subtleties but overall follows the same general structure. We describe the 2D formulation

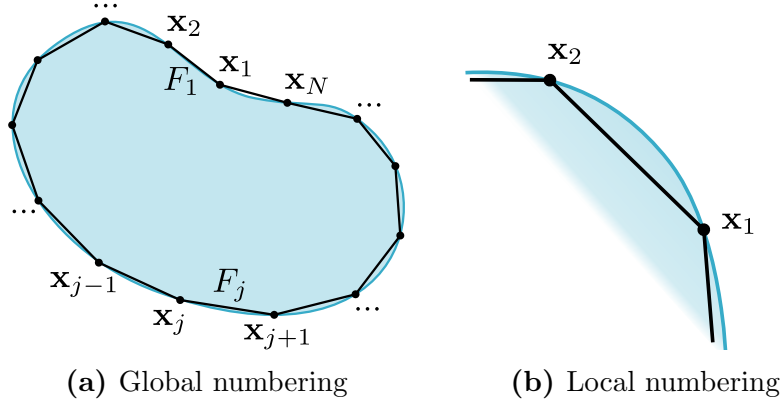


**Figure 4.2: Global and local numbering for BEM boundary discretization.** Sketches of a linearly discretized 2D boundary where (a) is the full boundary with a global numbering of nodes and (b) is a linear element with local numbering of nodes. Figure adapted from [Hall \[1994\]](#).

in detail in the hopes that it will provide the reader with clarity regarding BEM as well as a solid foundation to build upon when we discuss the coupling extension in Chapter 5. We follow these discussions with the aforementioned subsections on triple point handling and pressure gradient construction from the BEM output.

#### 4.4.3.1 2D linear isoparametric BEM

For this description we adopt some notation from the book by [Hall \[1994\]](#). We also refer the reader to the book by [Hall \[1994\]](#) for more in-depth discussion of BEM.

We begin with the linear discretization of the boundary into the boundary elements. Since the boundary elements correspond with the faces, we denote the  $N$  elements as  $F_j$ ,  $j = 1, 2, \dots, N$  in some order (see Figure 4.2a). Each element is also given a local numbering (see Figure 4.2b). We parameterize element  $F_j$ , which has endpoints at  $\mathbf{x}_j$  and  $\mathbf{x}_{j+1}$ , by  $t \in [-1, 1]$  as

$$\mathbf{x} = \mathbf{x}_j(t) = M_1(t)\mathbf{x}_j + M_2(t)\mathbf{x}_{j+1},$$

where  $M_1(t)$  and  $M_2(t)$  are the linear shape functions given by

$$M_1(t) = \frac{1-t}{2}, \quad M_2(t) = \frac{1+t}{2}.$$

The Jacobian of this parameterization of  $\mathbf{x}_j(t)$  is denoted by  $J_j(t)$  and is given by

$$J_j(t) = \frac{1}{2} \|\mathbf{x}_{j+1} - \mathbf{x}_j\| = \frac{1}{2} a_j,$$

where  $a_j$  is the length of element  $F_j$ .

For a linear isoparametric BEM, we assume that the unknown functions  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  are also approximated linearly over each linear boundary element. Setting up a local numbering system for each element as shown in Figure 4.2b, we can write  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  as

$$\begin{aligned} p(\mathbf{y}) &= M_1(t)p(\mathbf{y}_1) + M_2(t)p(\mathbf{y}_2) = \sum_{\alpha=1}^2 M_\alpha(t)p(\mathbf{y}_\alpha), \\ \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}} &= M_1(t)\frac{\partial p(\mathbf{y}_1)}{\partial \mathbf{n}} + M_2(t)\frac{\partial p(\mathbf{y}_2)}{\partial \mathbf{n}} = \sum_{\alpha=1}^2 M_\alpha(t)\frac{\partial p(\mathbf{y}_\alpha)}{\partial \mathbf{n}}, \end{aligned}$$

where  $p(\mathbf{y}_\alpha)$  and  $\frac{\partial p(\mathbf{y}_\alpha)}{\partial \mathbf{n}}$  denote the values of  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  at the endpoints of each element; that is, the vertices of the boundary discretization.

We can substitute the linear approximation into each term of the discretized BIE in Equation (4.48). The first term, involving the  $\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_\mathbf{y}}$  kernel, then becomes

$$\begin{aligned} \sum_{j=1}^N \int_{F_j} p(\mathbf{y}) \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} dS_\mathbf{y} &= \sum_{j=1}^N \int_{-1}^1 p(\mathbf{y}_j(t)) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_\mathbf{y}} J_j(t) dt \\ &= \sum_{j=1}^N \int_{-1}^1 \left( \sum_{\alpha=1}^2 M_\alpha(t)p_\alpha \right) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_\mathbf{y}} J_j(t) dt \\ &= \sum_{j=1}^N \sum_{\alpha=1}^2 p_\alpha J_j(t) \int_{-1}^1 M_\alpha(t) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_\mathbf{y}} dt \\ &= \sum_{j=1}^N \sum_{\alpha=1}^2 p_\alpha \frac{1}{2} a_j \int_{-1}^1 M_\alpha(t) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_\mathbf{y}} dt. \end{aligned}$$

The second term, involving the logarithmic  $G(\mathbf{x}, \mathbf{y})$  kernel, can similarly be manipulated to come to the conclusion:

$$\sum_{j=1}^N \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} G(\mathbf{x}, \mathbf{y}) dS_\mathbf{y} = \sum_{j=1}^N \sum_{\alpha=1}^2 \left( \frac{\partial p}{\partial \mathbf{n}_\mathbf{y}} \right) \frac{1}{2} a_j \int_{-1}^1 M_\alpha(t) G(\mathbf{x}, \mathbf{y}_j(t)) dt.$$

Equation (4.48) can then be written as

$$\begin{aligned} \Theta(\mathbf{x})p(\mathbf{x}) = & \sum_{j=1}^N \sum_{\alpha=1}^2 p_{\alpha} \frac{1}{2} a_j \int_{-1}^1 M_{\alpha}(t) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_{\mathbf{y}}} dt \\ & - \sum_{j=1}^N \sum_{\alpha=1}^2 \left( \frac{\partial p}{\partial \mathbf{n}_{\mathbf{y}}} \right) \frac{1}{2} a_j \int_{-1}^1 M_{\alpha}(t) G(\mathbf{x}, \mathbf{y}_j(t)) dt. \end{aligned} \quad (4.49)$$

Now, if we consider applying Equation (4.49) to each vertex on the mesh, it is clear that this collection of equations can be rearranged and written out in the form of a linear system

$$\Theta \mathbf{p} = D \mathbf{p} - C \mathbf{q}, \quad (4.50)$$

or alternatively,

$$0 = (D - \Theta) \mathbf{p} - C \mathbf{q}, \quad (4.51)$$

where  $\mathbf{p}$  and  $\mathbf{q}$  are vectors of size  $N_V$  respectively containing the nodal  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values,  $\Theta$  is a diagonal  $N_V \times N_V$  matrix whose diagonal entries are the internal solid angles at the vertices, and  $C$  and  $D$  are  $N_V \times N_V$  matrices containing the appropriate sums of the integrals containing  $G(\mathbf{x}, \mathbf{y})$  and  $\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}}$  from Equation (4.49), respectively.

We turn our attention to evaluating these integrals. When  $\mathbf{x}$  is not on (or at the endpoint of) the element being integrated over which is parameterized by  $\mathbf{y}_j(t)$ , there is no singularity in either integrand, and these integrals can be simply approximated with a direct Gaussian quadrature. In our implementation we use a four-point Gaussian quadrature when there is no singularity. However, when  $\mathbf{x}$  is on the element being integrated over, a singularity occurs in both integrals. We are able to avoid using numerical approximation on the singular integrand involving the logarithmic  $G(\mathbf{x}, \mathbf{y})$  integrand by exact integration using a principal value integration approach; we refer the reader to the relevant chapter in the book by Hall [1994] for a full derivation. This yields the following results when  $\mathbf{x}$  is at the endpoint at local node 1 of the element parameterized by  $\mathbf{y}_j(t)$ :

$$\begin{aligned} \int_{-1}^1 G(\mathbf{x}, \mathbf{y}_j(t)) M_1(t) dt &= \ln a_j - \frac{3}{2}, \\ \int_{-1}^1 G(\mathbf{x}, \mathbf{y}_j(t)) M_2(t) dt &= \ln a_j - \frac{1}{2}. \end{aligned}$$

When  $\mathbf{x}$  is at the other endpoint, at local node 2, we have the result:

$$\begin{aligned}\int_{-1}^1 G(\mathbf{x}, \mathbf{y}_j(t)) M_1(t) dt &= \ln a_j - \frac{1}{2}, \\ \int_{-1}^1 G(\mathbf{x}, \mathbf{y}_j(t)) M_2(t) dt &= \ln a_j - \frac{3}{2}.\end{aligned}$$

The singularity involving  $\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_\mathbf{y}}$  is stronger than the singularity in the logarithmic integrand; instead of approaching the evaluation of this singularity directly, we eliminate it using the *row-sum elimination* technique.

Row-sum elimination is as follows. We know a simple solution to Laplace's equation is

$$p(\mathbf{x}) = 1, \quad \frac{\partial p(\mathbf{x})}{\partial \mathbf{n}} = 0.$$

By using these values in Equation (4.49), we find a simplified expression:

$$\Theta(\mathbf{x}) = \sum_{j=1}^N \sum_{\alpha=1}^2 \frac{1}{2} a_j \int_{-1}^1 M_\alpha(t) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(t))}{\partial \mathbf{n}_\mathbf{y}} dt.$$

The singularity that occurs on the face can then be found by isolating and solving for the coefficient.

Once we have constructed the  $C$  and  $D$  matrices in the system described by Equation (4.51) we apply the boundary conditions and solve the system. Either  $p$  or  $\frac{\partial p}{\partial \mathbf{n}}$  is known at each point in the system. We can construct a vector  $\mathbf{w}$  which takes the value  $p$  for points where  $p$  is known and  $\frac{\partial p}{\partial \mathbf{n}}$  where  $\frac{\partial p}{\partial \mathbf{n}}$  is known so that  $\mathbf{w}$  is known fully from these boundary conditions. We can also construct a vector  $\mathbf{y}$  that is its counterpart, such that  $\mathbf{y}$  is wholly unknown. Swapping around columns of  $C$  and  $D$ , we can then rearrange Equation (4.51) into a form such that unknown values are to the left of the equality and known values are to the right:

$$A\mathbf{y} = B\mathbf{w},$$

where  $A$  and  $B$  are constructed out of the columns of  $C$  and  $D$  that correspond to the unknown and known coefficients. Since  $B$  and  $\mathbf{w}$  are fully known, we can simply perform the multiplication and further summarize this system as:

$$A\mathbf{y} = \mathbf{z}. \tag{4.52}$$

Solving for  $\mathbf{y}$  then yields all the unknown  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values of each vertex, and our BEM solve is complete.

#### 4.4.3.2 3D linear isoparametric BEM

We provide only a brief summary of the linear isoparametric BEM in the 3D case here, and refer the interested reader to the work of [Da et al. \[2016\]](#) and [Keeler and Bridson \[2014\]](#) for a more detailed discussion regarding the specific techniques used by [Da et al. \[2016\]](#).

The BEM in the 3D case proceeds, at a high level, just as it did in the 2D case. We once again have mesh faces  $F_j$  corresponding to boundary elements, except that now each face is a triangle rather than a line segment. We can follow similar steps for parameterizing and discretizing the inner integral in Equation (4.48) into a form similar to Equation (4.49):

$$\begin{aligned} \Theta(\mathbf{x})p(\mathbf{x}) = & \sum_{j=1}^N \sum_{\alpha=1}^3 p_{\alpha} J_j(u, v) \int_{-1}^1 \int_{-1}^1 M_{\alpha}(u, v) \frac{\partial G(\mathbf{x}, \mathbf{y}_j(u, v))}{\partial \mathbf{n}_{\mathbf{y}}} du dv \\ & - \sum_{j=1}^{N_F} \sum_{\alpha=1}^3 \left( \frac{\partial p}{\partial \mathbf{n}_{\mathbf{y}}} \right) J_j(u, v) \int_{-1}^1 \int_{-1}^1 M_{\alpha}(u, v) G(\mathbf{x}, \mathbf{y}_j(t)) du dv, \quad (4.53) \end{aligned}$$

where  $u$  and  $v$  are the variables used to linearly parameterize each triangular face and  $M_{\alpha}$  are the linear shape functions. The above formulation assumes that the  $M_{\alpha}$  functions are chosen such that  $u, v \in [-1, 1]$  parameterize each element. We direct the reader to relevant chapters in books on BEM such as the one by [Hall \[1994\]](#) for more details regarding this formulation, the parameterization, and 3D BEM.

We can see how the above can be written into the form of Equation (4.51); the only task that remains prior to rearranging this into the familiar  $\mathbf{A}\mathbf{y} = \mathbf{z}$  form and solving the BEM is evaluating the integrals present in entries of the  $C$  and  $D$  matrices. Once again, some care must be taken when approximating these integrals due to singularities in the both the Green's function and its normal derivative. For triangles that would not include the singularity, that is triangles that do not contain vertex  $\mathbf{x}$ , [Da et al. \[2016\]](#) and [Keeler and Bridson \[2014\]](#) use a simple direct 3-point Gaussian quadrature for both the integrals involving  $G(\mathbf{x}, \mathbf{y})$  and the integrals involving  $\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}}$ . Equations (10) and (11) in the work by [Keeler and Bridson \[2014\]](#) are relevant here. For triangles that do contain vertex  $\mathbf{x}$ , the singularity is treated differently between the two types of integrals. For integrals involving  $\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}}$ , the weight of the singularity point is set to 0 and its contribution is ignored; [Keeler and Bridson \[2014\]](#) justify this by recognizing that in the limit of approaching the boundary, the  $\Theta(\mathbf{x})p(\mathbf{x})$  term dominates contributions that would come from this singularity point. For integrals involving  $G(\mathbf{x}, \mathbf{y})$ , they use a Duffy transformation to eliminate the weak singularity prior to taking a nine-point Gaussian quadrature.

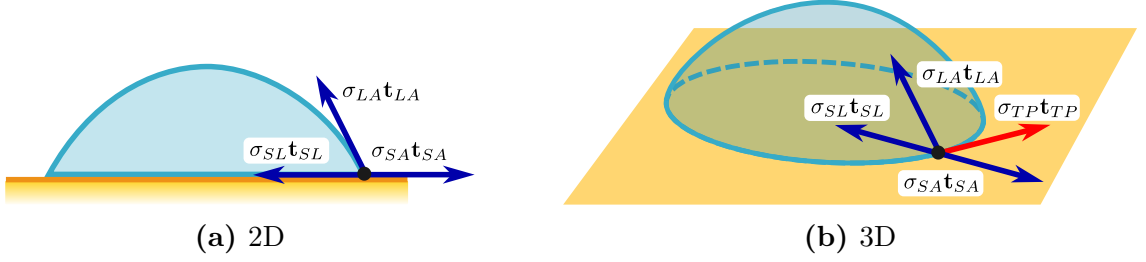


**Figure 4.3: Surface tension forces at triple points/triple junctions.** Surface tension forces and directions at a triple point/triple junction in the 2D and 3D scenarios.  $\mathbf{t}_{TP}$  points along the triple junction curve and is only meaningful in the 3D case. 3D figure adapted from [Da et al. \[2016\]](#).

Once the  $C$  and  $D$  matrices from (4.51) are formed, we follow the same concept as was outlined in the previous section to rearrange the system such that we solve for the unknown  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values at each vertex and complete the BEM solve.

#### 4.4.3.3 Triple junctions

As alluded to previously, triple points require some special handling. First, the formulation for both boundary conditions differs slightly from how they are stated in BVP (4.23) at triple points. Second, since triple points are by definition situated in both the liquid-air interface and the solid-liquid interface, we have information for both  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  at these points. However, while  $p$  is continuous, the sharp crease that occurs at triple junctions due to the solid contact causes a discontinuity in  $\frac{\partial p}{\partial \mathbf{n}}$ . We address how the BEM needs to be modified to handle these differences below.

At purely liquid-air points, the pressure  $p$  boundary condition is purely dominated by the surface tension and is calculated using the mean curvature at the vertex in BVP (4.23). At triple points, mean curvature is no longer well-defined due to the sharp crease that occurs due to the solid contact. Moreover, at this contact point (in 2D) or contact line (in 3D), surface tension at the liquid air interface is no longer the only relevant tension force, as there are also tension forces along the solid-liquid and solid-air interfaces as well. [Da et al. \[2016\]](#) therefore calculate the  $p$  boundary condition at triple points by integrating pressure over a thin strip that lies over both sides of the triple junction:

$$\int p_{phases} ds = (\sigma_{LA} \mathbf{t}_{LA} + \sigma_{SA} \mathbf{t}_{SA} + \sigma_{SL} \mathbf{t}_{SL}) \cdot \mathbf{t}_{SA}, \quad (4.54)$$

where  $LA$ ,  $SL$ , and  $SA$  denote quantities for the liquid-air, solid-liquid, and solid-air interfaces, respectively,  $\sigma$  are tension coefficients, and  $\mathbf{t}$  denotes tangent vectors in the interfaces that point outward from the triple junction (see Figure 4.3). In theory, the above integral would be taken over a thin strip of infinitesimal width. In practice, both we and Da et al. [2016] use a width of 0.25 times the average edge length; dividing the calculated right-hand side of Equation (4.54) by this value yields  $p$ .

The normal component of the gradient  $\frac{\partial p}{\partial \mathbf{n}}$  is not continuous at triple points as the sharp crease in the geometry at these points causes  $\mathbf{n}$  to take on different values on either side of the crease. To deal with this, Da et al. [2016] use a simplification of the *double nodes technique* wherein they view triple points as having two  $\frac{\partial p}{\partial \mathbf{n}}$  values: one for each of the two interfaces on each side of the triple junction. The  $\frac{\partial p}{\partial \mathbf{n}}$  value on the solid-liquid interface, which we denote as  $\frac{\partial p}{\partial \mathbf{n}_S}$ , is known from the boundary condition:

$$\frac{\partial p}{\partial \mathbf{n}_S} = \frac{\rho_f}{\Delta t} (\tilde{\mathbf{u}} - \mathbf{u}_S) \cdot \mathbf{n}_S, \quad (4.55)$$

which is essentially the same as the usual Neumann boundary condition, except that we take care to use only the solid normal  $\mathbf{n}_S$  rather than the full normal. On the other hand, the  $\frac{\partial p}{\partial \mathbf{n}}$  value on the liquid-air interface  $\frac{\partial p}{\partial \mathbf{n}_A}$  is an unknown quantity. The unknown  $\frac{\partial p}{\partial \mathbf{n}_A}$  keeps the resulting BEM system from being overdetermined, as we still have one unknown quantity and one collocation equation at each point on the triple junction.

The BEM system is only slightly modified from what was previously described to account for these differences. The coefficients summarized in the  $C$  and  $D$  matrices in, for example, Equation (4.51) are mildly less cleanly separated when the boundary conditions are applied and the system is organized into unknowns and knowns as in Equation (4.52). This is because  $C$  is essentially split up into contributions coming from the liquid-air interface and contributions coming from the solid-liquid interface. The vector of unknowns  $\mathbf{y}$  in Equation (4.52) then holds  $\frac{\partial p}{\partial \mathbf{n}_A}$  for the collocation equations corresponding to triple points, and the known  $p$  and  $\frac{\partial p}{\partial \mathbf{n}_S}$  are incorporated into the right-hand side  $\mathbf{z}$  of the system.

#### 4.4.3.4 Constructing the pressure gradient

After the BEM solve is complete, we construct the full pressure gradient out of the nodal  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$  values, which we have from either the boundary conditions or the output of the BEM solve. For vertices that are not on the triple-junction, that is vertices that are on smooth parts of the mesh, we proceed as follows. We approximate the tangential component of the pressure gradient  $\frac{\partial p}{\partial \mathbf{t}}$  at vertices by linearly interpolating the  $p$  values across each face,

and then averaging those values back from the faces to the vertices.  $\nabla p$  is obtained by combining the normal and tangential components as in Equation (4.26).

As for the non-smooth vertices on the triple junction, Da et al. [2016] essentially perform a least-square approximation for each of the directional derivatives that are known. We construct the tangential directional derivative similar to how they are constructed for non-triple-point vertices, except that they are split up into their liquid-air and solid-liquid components. In 2D this looks like solving the following overdetermined system for  $\nabla p$ :

$$\begin{bmatrix} \mathbf{n}_S^T \\ \mathbf{n}_A^T \\ \mathbf{t}_{SL}^T \\ \mathbf{t}_{LA}^T \end{bmatrix} \nabla p = \begin{bmatrix} \frac{\partial p}{\partial \mathbf{n}_S} \\ \frac{\partial p}{\partial \mathbf{n}_A} \\ \frac{\partial p}{\partial \mathbf{t}_{SL}} \\ \frac{\partial p}{\partial \mathbf{t}_{LA}} \end{bmatrix}.$$

In 3D, we have an additional equation in the system:

$$\begin{bmatrix} \mathbf{n}_S^T \\ \mathbf{n}_A^T \\ \mathbf{t}_{SL}^T \\ \mathbf{t}_{LA}^T \\ \mathbf{t}_{TP}^T \end{bmatrix} \nabla p = \begin{bmatrix} \frac{\partial p}{\partial \mathbf{n}_S} \\ \frac{\partial p}{\partial \mathbf{n}_A} \\ \frac{\partial p}{\partial \mathbf{t}_{SL}} \\ \frac{\partial p}{\partial \mathbf{t}_{LA}} \\ \frac{\partial p}{\partial \mathbf{t}_{TP}} \end{bmatrix},$$

where  $\mathbf{t}_{TP}$  is the unit tangent of the triple junction curve. As can be seen from Figure 4.3, while  $\mathbf{t}_{TP}$  can be defined as pointing out of the plane in 2D, the directional derivative of pressure in this direction  $\frac{\partial p}{\partial \mathbf{t}_{TP}}$  is not a valid quantity in 2D, hence why this equation is omitted in the 2D system.

With the pressure gradient  $\nabla p$  constructed, we follow Equation (4.27) for each vertex in the mesh to update to the end-of-step potential flow field. This end-of-step potential flow field then acts as the required potential flow field input to the next time step.

Having described the steps required to solve the incompressible Euler equations for a liquid domain using only a surface mesh discretization, we now move on to discussing how this method is modified to allow for two-way rigid body coupling.

# Chapter 5

## Surface-Only Rigid Body Coupling

The surface-only liquids method as originally described by Da et al. [2016] handles a one-way coupled fluid-solid system. Specifically, it describes the case in which only the solid exerts influence over the fluid; solids are essentially assumed to have infinite mass and their movement is not affected by the fluid. In other words, all solids are *scripted* solids.

This one-way coupling is achieved by the BEM solve during the external force integration step by the boundary condition imposed on liquid points in contact with solids. We reproduce boundary condition (4.25) below for the reader’s convenience:

$$\frac{\partial p}{\partial \mathbf{n}} = \frac{\rho_f}{\Delta t} (\tilde{\mathbf{u}} - \mathbf{u}_S) \cdot \mathbf{n}, \quad \text{on } \Gamma_S. \quad (4.25)$$

We see that the normal gradient of pressure for points on the interface between the fluid and a solid is set by assuming we have knowledge of the normal component of the solid velocity at these points. This is a valid assumption in the case of scripted solids, as by definition we know the velocities of the solids in the system and we can simply include them in the calculation of this Neumann boundary condition.

However, in the case of an unscripted solid we no longer have the solid velocity information readily available. The solid velocity instead becomes a function of the solid’s degrees of freedom. Clearly, it is necessary to modify the BEM system sketched in Section 4.4.3.

Additionally, as stated before, the BEM system in Section 4.4.3 only handles the scripted solid case. This is because in building that system, we implicitly assumed infinite mass for all solids and omitted updates to the solid velocities due to the fluid forces. For two-way coupling, it is then also necessary to update the system to allow the fluid to have an effect on solid velocities.

For fluid to solid coupling, the fluid pressures acting on the surface of the solid causes forces and torques on the solid. Recalling the rigid body model of Section 3.3, the updates to the solid's translational and rotational velocities are given by discretizing Equations (3.9) and (3.11), respectively:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \frac{\Delta t}{m} \mathbf{F}, \quad (5.1)$$

$$\boldsymbol{\omega}^{n+1} = \boldsymbol{\omega}^n + \Delta t I^{-1} \mathbf{T}, \quad (5.2)$$

where  $\mathbf{v}$  and  $\boldsymbol{\omega}$  are the linear and rotational components of the solid's velocity,  $\mathbf{F}$  and  $\mathbf{T}$  are the net pressure force and torque of the fluid acting on the solid,  $m$  is the solid's mass, and  $I$  is the solid's inertia tensor (or its moment of inertia in the 2D case).

Net pressure force  $\mathbf{F}$  and torque  $\mathbf{T}$  are given by integrals over the solid's surface:

$$\mathbf{F} = - \iint_{\partial S} p(\mathbf{x}) \mathbf{n}_x dS_x, \quad (5.3)$$

$$\mathbf{T} = - \iint_{\partial S} (\mathbf{x} - \mathbf{x}_{COM}) \times p(\mathbf{x}) \mathbf{n}_x dS_x, \quad (5.4)$$

where  $\partial S$  is the surface of the solid,  $\mathbf{n}$  is the outward normal relative to the solid, and  $\mathbf{x}_{COM}$  is the center of mass. In a 3D simulation, the integrals above are surface double integrals as written above, while in 2D they are appropriately replaced with a single integral. To couple a rigid body, we follow in the notation of Batty et al. [2007] and use the net force and torque equations above to define an operator  $J$  that maps pressures on the solid's boundary to net force and torque on the body. The translational and rotational components of the  $J$  operator are respectively

$$J_{trans} \mathbf{p} = - \iint_{\partial S} p(\mathbf{x}) \mathbf{n}_x dS_x, \quad (5.5)$$

$$J_{rot} \mathbf{p} = - \iint_{\partial S} (\mathbf{x} - \mathbf{x}_{COM}) \times p(\mathbf{x}) \mathbf{n}_x dS_x, \quad (5.6)$$

where

$$J = \begin{bmatrix} J_{trans} \\ J_{rot} \end{bmatrix}. \quad (5.7)$$

With both coupling directions outlined above, we are ready to discuss the discretizations of these formulations and the subsequent additions to our original BEM system to achieve surface-only two-way coupling.

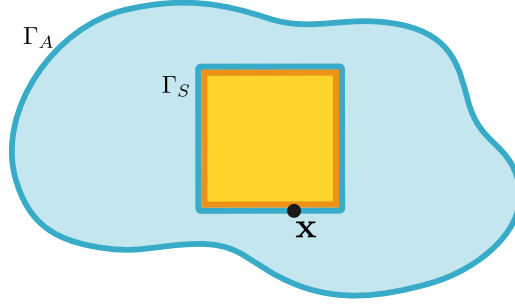


**Figure 5.1: Example setup of an immersed solid.** A visualization of a simple example setup of a solid immersed in a fluid blob, with a point  $\mathbf{x}$  on the fluid-solid boundary  $\Gamma_S$ .

## 5.1 Solid to Fluid Coupling

To aid in the illustration of the modified system, in this section we consider the discretized formulation of BIE (4.47) for an arbitrary point  $\mathbf{x}$  on the solid-liquid interface  $\Gamma_S$  in the simple setup of a rigid body fully submerged in a liquid domain with no other solids present (see Figure 5.1). This setup allows us to avoid addressing triple points at this time for clarity of explanation. We will return to the question of triple points in Section 5.4.

The liquid surface  $\Gamma$  is composed of the liquid-air interface  $\Gamma_A$  and the solid-liquid interface  $\Gamma_S$ , that is  $\Gamma = \Gamma_A \cup \Gamma_S$ . Both  $\Gamma_A$  and  $\Gamma_S$  consist of some arbitrary number of discrete points, and there is no overlap between the two sets, that is  $\Gamma_A \cap \Gamma_S = \emptyset$ .

We begin by rewriting BIE (4.47) for the point  $\mathbf{x}$  in a discretized form, similar to the work done in Section 4.4.3:

$$\Theta(\mathbf{x})p(\mathbf{x}) = \sum_{\mathbf{y} \in \Gamma} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right). \quad (5.8)$$

We use the notation  $G_{\mathbf{x},\mathbf{y}}$  and  $\frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}}$  rather than  $G(\mathbf{x},\mathbf{y})$  and  $\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}}$  to highlight that the former are values that are calculated from taking the *integral* of the latter over the face that contains point  $\mathbf{y}$ , rather than the function evaluated at the single point  $\mathbf{y}$ . As a result, we also note that  $G_{\mathbf{x},\mathbf{y}}$  and  $\frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}}$  incorporate the face area terms that were present in our derivations in Section 4.4.3. We can expand the above by splitting up the sum over the points  $\mathbf{y} \in \Gamma$  into sums over  $\mathbf{y} \in \Gamma_A$  and  $\mathbf{y} \in \Gamma_S$ :

$$\Theta(\mathbf{x})p(\mathbf{x}) = \sum_{\mathbf{y} \in \Gamma_A} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right) + \sum_{\mathbf{y} \in \Gamma_S} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right).$$

Recall that, from the boundary conditions in BVP (4.23),  $p$  is a known quantity on  $\Gamma_A$  and  $\frac{\partial p}{\partial \mathbf{n}_y}$  is a known quantity on  $\Gamma_S$ . Given this, we now rearrange the terms so that unknown quantities are gathered on the left-hand side of the equality and known quantities are gathered on the right. Note that for the case of a scripted solid,  $\frac{\partial p}{\partial \mathbf{n}_y}$  is fully known at solid boundary points from boundary condition (4.25), and we are leaving those contributions in the right-hand side for now, though we will revisit this momentarily:

$$\begin{aligned} \sum_{\mathbf{y} \in \Gamma_A} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_y} G_{\mathbf{x}, \mathbf{y}} - \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y} = \\ - \Theta(\mathbf{x})p(\mathbf{x}) - \sum_{\mathbf{y} \in \Gamma_A} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_y} G_{\mathbf{x}, \mathbf{y}} + \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y}. \end{aligned}$$

Substituting in the expression from boundary condition (4.25) we have

$$\begin{aligned} \sum_{\mathbf{y} \in \Gamma_A} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_y} G_{\mathbf{x}, \mathbf{y}} - \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y} = \\ - \Theta(\mathbf{x})p(\mathbf{x}) - \sum_{\mathbf{y} \in \Gamma_A} \left( \frac{\rho_f}{\Delta t} (\tilde{\mathbf{u}}_y - (\mathbf{u}_S)_y) \cdot \mathbf{n}_y \right) G_{\mathbf{x}, \mathbf{y}} + \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y}. \end{aligned}$$

As mentioned, the above is appropriate in the case of a scripted solid. That is, the unknowns and knowns have been sorted out appropriately on either side of the equality for a solid for which  $\mathbf{u}_S$  is given. However, for an unscripted solid,  $\mathbf{u}_S$  at any point in the liquid mesh is no longer a known value. It is yet another unknown in our system and should be moved over to the left side:

$$\begin{aligned} \sum_{\mathbf{y} \in \Gamma_A} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_y} G_{\mathbf{x}, \mathbf{y}} - \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y} + \sum_{\mathbf{y} \in \Gamma_A} \left( \frac{\rho_f}{\Delta t} (\mathbf{u}_S)_y \cdot \mathbf{n}_y \right) G_{\mathbf{x}, \mathbf{y}} = \\ - \Theta(\mathbf{x})p(\mathbf{x}) - \sum_{\mathbf{y} \in \Gamma_A} \left( \frac{\rho_f}{\Delta t} \tilde{\mathbf{u}}_y \cdot \mathbf{n}_y \right) G_{\mathbf{x}, \mathbf{y}} + \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x}, \mathbf{y}}}{\partial \mathbf{n}_y}. \quad (5.9) \end{aligned}$$

The velocity of a rigid body at a point  $\mathbf{x}$  on its boundary can be evaluated from the rigid body degrees of freedom as

$$\mathbf{u}_S(\mathbf{x}) = \mathbf{v} + \boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_{COM}), \quad (5.10)$$

where  $\mathbf{v}$  is the translational velocity,  $\boldsymbol{\omega}$  is the rotational velocity, and  $\mathbf{x}_{COM}$  is the center of mass. Calculating the normal component of this velocity can be summarized with the  $J$  linear operator as

$$\mathbf{u}_S(\mathbf{x}) \cdot \mathbf{n}_x = J_x^T \mathbf{V}, \quad (5.11)$$

where  $\mathbf{n}$  is the inward normal relative to the solid at point  $\mathbf{x}$ ,  $\mathbf{V}$  is the generalized velocity of the solid (a three or six-dimensional vector for a single rigid body, depending on whether the simulation is in 2D or 3D, respectively), and the linear operator  $J_{\mathbf{x}}^T$  is the transpose of  $J_{\mathbf{x}}$  where the  $J_{\mathbf{x}}$  is the column corresponding to coordinate  $\mathbf{x}$  in  $J$  (see Equation (5.7)).  $J^T$  maps from the rigid body velocities to the normal velocities on the boundary. We will find in Section 5.2 that  $J$  maps fluid pressures on the boundary to net force and torque on the solid.

Equation (5.9) can then be rewritten using the generalized velocity of the solid  $\mathbf{V}$ :

$$\begin{aligned} \sum_{\mathbf{y} \in \Gamma_A} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} - \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} + \sum_{\mathbf{y} \in \Gamma_A} \frac{\rho_f}{\Delta t} G_{\mathbf{x},\mathbf{y}} J_{\mathbf{y}}^T \mathbf{V} = \\ - \Theta(\mathbf{x}) p(\mathbf{x}) - \sum_{\mathbf{y} \in \Gamma_A} \left( \frac{\rho_f}{\Delta t} \tilde{\mathbf{u}}_{\mathbf{y}} \cdot \mathbf{n}_{\mathbf{y}} \right) G_{\mathbf{x},\mathbf{y}} + \sum_{\mathbf{y} \in \Gamma_S} p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}}. \end{aligned} \quad (5.12)$$

Thus it is clear that moving the solid velocity term from the right-hand to the left-hand side only increases the number of unknowns by the size of  $\mathbf{V}$ ; that is, by three or six degrees of freedom depending on the dimension of the simulation.

Rearranging terms in this way allows us to continue to use the BEM technique to keep the fluid incompressible and handle the solid to fluid coupling for the case of unknown solid velocities. However, if we consider building a system by writing an equation similar to Equation (5.12) for all  $N_V$  liquid mesh points, we would be left with an underdetermined system due to the addition of  $\mathbf{V}$  to the list of unknowns. It is evident that additional modifications are necessary to build a system with a unique solution.

## 5.2 Fluid to Solid Coupling

We now present the additions to the system for fluid to solid coupling. As discussed earlier in the chapter, in this coupling direction, we calculate the forces and torques on the rigid body caused by the fluid pressures by integrating over the rigid body's surface, and use these to update the rigid body velocity.

We begin by discretizing the  $J$  operator as described in Equations (5.5) through (5.7). To discretize  $J$  we discretize the surface integrals in Equations (5.5) and (5.6) into sums

over the discrete points  $\mathbf{x}$  on the surface of the solid  $\partial S$ :

$$J_{trans}\mathbf{p} = - \sum_{\mathbf{x} \in \partial S} \hat{a}_{\mathbf{x}} p(\mathbf{x}) \mathbf{n}_{\mathbf{x}}, \quad (5.13)$$

$$J_{rot}\mathbf{p} = - \sum_{\mathbf{x} \in \partial S} \hat{a}_{\mathbf{x}} (\mathbf{x} - \mathbf{x}_{COM}) \times p(\mathbf{x}) \mathbf{n}_{\mathbf{x}}, \quad (5.14)$$

where  $\hat{a}_{\mathbf{x}}$  is the area of vertex  $\mathbf{x}$  (see Equations (4.1) and (4.2)). We see that  $J$  is a  $3 \times N_S$  matrix in 2D, and a  $6 \times N_S$  matrix in 3D, where  $N_S$  is the number of points on the rigid body. In 2D, for example, the column of  $J$  that corresponds to the point  $\mathbf{x}$  on the solid is

$$J_{\mathbf{x}} = \begin{bmatrix} \hat{a}_{\mathbf{x}}(\mathbf{n}_{\mathbf{x}})_x \\ \hat{a}_{\mathbf{x}}(\mathbf{n}_{\mathbf{x}})_y \\ \hat{a}_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_{COM}) \times \mathbf{n}_{\mathbf{x}} \end{bmatrix}. \quad (5.15)$$

Defined in this way, it is clear that  $J\mathbf{p}$  appropriately approximates the  $J$  operator as defined by Equations (5.5) and (5.6) when multiplied by a vector  $\mathbf{p}$  containing the pressure at the points on the surface of the solid.

With  $J$  discretized, we now move on to considering the known and unknown values in our system, similarly to the derivations done in Section 5.1. For notational convenience, we define the generalized velocity  $\mathbf{V}$  as

$$\mathbf{V} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}. \quad (5.16)$$

We can now write the velocity update in Equations (5.1) and (5.2) succinctly as

$$\mathbf{V}^{n+1} = \mathbf{V}^n - \Delta t M^{-1} J \mathbf{p}, \quad (5.17)$$

where  $M$  is the mass matrix, and is constructed from the mass and the inertia tensor. In 2D  $M$  is a  $3 \times 3$  matrix, and in 3D it is a  $6 \times 6$  matrix:

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \quad M = \begin{bmatrix} m & 0 & 0 & \mathbf{0}^T \\ 0 & m & 0 & \mathbf{0}^T \\ 0 & 0 & m & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}. \quad (5.18)$$

We recall that  $\mathbf{p}$  is a vector that contains the fluid pressure values at points on the solid's surface. We also recall that  $\mathbf{p}$  is not known on  $\Gamma_S$  in BVP (4.23). Equation (5.17) can be

rearranged and rewritten with unknown values on the left-hand of the equality and known values on the right-hand side as:

$$J\mathbf{p} + \frac{1}{\Delta t}M\mathbf{V}^{n+1} = \frac{1}{\Delta t}M\mathbf{V}^n. \quad (5.19)$$

For each rigid body in the system we thus have an additional set of equations in our system, the number of which is determined by the size of  $\mathbf{V}$ . Together with the modifications described in the previous sections, the system once again has an equal number of unknowns and equations.

### 5.3 Two-way Coupled System

We can now build the system of equations for the two-way coupled system. The full system follows from Equations (5.12) and (5.17) and can be summarized as a linear system:

$$\begin{bmatrix} A & K \\ J & \frac{1}{\Delta t}M \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{V}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ \frac{1}{\Delta t}M\mathbf{V}^n \end{bmatrix}. \quad (5.20)$$

$A$  and  $\mathbf{y}$  are constructed as they were described in Section 4.4.3. The vector containing known values  $\mathbf{z}$  is also similar to the original BEM system in 4.4.3, but omits contributions from unscripted solid velocities, as they are no longer known values. Instead, the  $K$  component incorporates the coefficients for these contributions. For each unscripted rigid body in the system,  $K$  is an  $N_V \times 3$  or an  $N_V \times 6$  matrix (for 2D or 3D, respectively), where each row in  $K$  is calculated by

$$\sum_{\mathbf{y} \in \Gamma_A} \frac{\rho_f}{\Delta t} G_{\mathbf{x}, \mathbf{y}} J_{\mathbf{y}}^T,$$

from the left-hand side of Equation (5.12).  $J$  is computed for each unscripted solid as described in Section 5.2, and the columns are arranged to correspond with the entries of  $\mathbf{y}$  that correspond with the appropriate  $p$  values. Columns of  $J$  that correspond with  $\frac{\partial p}{\partial \mathbf{n}}$  in  $\mathbf{y}$ , or  $p$  values that are not on the relevant solid, are simply  $\mathbf{0}$  vectors.

Solving Equation (5.20) yields  $\mathbf{y}$ , which contains the unknown pressure and pressure normal values, and  $\mathbf{V}^{n+1}$ , the updated velocity of the unscripted rigid body, while simultaneously enforcing both coupling directions.

## 5.4 Triple Points

To finish our discussion of the coupling scheme we now return to the handling of triple points.

### 5.4.1 Solid to Fluid Coupling with Triple Points

For the solid to fluid coupling direction, the treatment of triple points is similar to their treatment in Section 4.4.3.3. At points that are purely on the liquid-air or solid-liquid boundary, information is only available for one of  $p$  and  $\frac{\partial p}{\partial \mathbf{n}}$ . By contrast, at triple points we have some amount of information for both quantities. Once again,  $p$  is continuous across the triple point, and is given by the balance of tension forces described by Equation (4.54). On the other hand,  $\frac{\partial p}{\partial \mathbf{n}}$  is not continuous across the triple point, and while the full  $\frac{\partial p}{\partial \mathbf{n}}$  is not known, the solid component  $\frac{\partial p}{\partial \mathbf{n}_S}$  is known and given by Equation (4.25). That is, we still have one unknown quantity  $\frac{\partial p}{\partial \mathbf{n}_A}$ . To describe how the triple points affect the coupling setup, we now proceed similarly to Section 5.1, beginning with Equation (5.8), reproduced below:

$$\Theta(\mathbf{x})p(\mathbf{x}) = \sum_{\mathbf{y} \in \Gamma} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right). \quad (5.8)$$

We split the sum over the points  $\mathbf{y} \in \Gamma$  into sums over the purely liquid-air points ( $\mathbf{y} \in \Gamma_A, \mathbf{y} \notin \Gamma_S$ ), the purely solid-liquid points ( $\mathbf{y} \in \Gamma_S, \mathbf{y} \notin \Gamma_A$ ), and the triple points ( $\mathbf{y} \in \Gamma_A, \mathbf{y} \in \Gamma_S$ ):

$$\begin{aligned} \Theta(\mathbf{x})p(\mathbf{x}) = & \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \notin \Gamma_S}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right) + \sum_{\substack{\mathbf{y} \in \Gamma_S \\ \mathbf{y} \notin \Gamma_A}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right) \\ & + \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right). \end{aligned} \quad (5.21)$$

It is this last sum, over the triple points, that interests us in this section:

$$\sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} \right).$$

For triple points, as we did in Section 4.4.3.3, while  $p$  is still continuous across the triple point, instead of a single  $\frac{\partial p}{\partial \mathbf{n}}$ , we need to consider the point to have two  $\frac{\partial p}{\partial \mathbf{n}}$  values: one that comes from the liquid-air interface  $\frac{\partial p}{\partial \mathbf{n}_A}$  and the other that comes from the solid-liquid interface  $\frac{\partial p}{\partial \mathbf{n}_S}$ . Similarly, we can also separate  $G_{\mathbf{x},\mathbf{y}}$  into two such components. The sum above can then be rewritten in terms of these components as

$$\sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \left( \left( \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} \right)_A (G_{\mathbf{x},\mathbf{y}})_A + \left( \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} \right)_S (G_{\mathbf{x},\mathbf{y}})_S \right) \right).$$

The  $\frac{\partial p}{\partial \mathbf{n}_A}$  term is unknown. The  $\frac{\partial p}{\partial \mathbf{n}_S}$  term is given by a slightly modified version of boundary condition (4.25):

$$\frac{\partial p}{\partial \mathbf{n}_S} = \frac{\rho_f}{\Delta t} (\tilde{\mathbf{u}} - \mathbf{u}_S) \cdot \mathbf{n}_S,$$

where  $\mathbf{n}_S$  is the solid normal. Substituting this into the sum over the triple points gives

$$\sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \left( p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} - \left( \left( \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} \right)_A (G_{\mathbf{x},\mathbf{y}})_A + \left( \frac{\rho_f}{\Delta t} (\tilde{\mathbf{u}}_{\mathbf{y}} - (\mathbf{u}_S)_{\mathbf{y}}) \cdot (\mathbf{n}_S)_{\mathbf{y}} \right) (G_{\mathbf{x},\mathbf{y}})_S \right) \right).$$

Of course, for the case of unscripted solids,  $\mathbf{u}_S$  is unknown. With this, we can rewrite Equation (5.21) so that unknown and known values are separated on the left and right sides of the equality, respectively. To avoid redundancy, we refer the reader back to the work that led to Equation (5.12) in Section 5.1 for rewriting the sums over the points on the purely liquid-air and solid-liquid boundaries. We note that the sums over  $\mathbf{y} \in \Gamma_A, \mathbf{y} \notin \Gamma_S$  correspond to the sums over  $\mathbf{y} \in \Gamma_A$  in Equation (5.12), and similarly the sums over  $\mathbf{y} \in \Gamma_S, \mathbf{y} \notin \Gamma_A$  correspond to the sums over  $\mathbf{y} \in \Gamma_S$  in Equation (5.12):

$$\begin{aligned} & \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \notin \Gamma_S}} \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} G_{\mathbf{x},\mathbf{y}} - \sum_{\substack{\mathbf{y} \in \Gamma_S \\ \mathbf{y} \notin \Gamma_A}} p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} + \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \notin \Gamma_S}} \frac{\rho_f}{\Delta t} G_{\mathbf{x},\mathbf{y}} J_{\mathbf{y}}^T \mathbf{V} \\ & + \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \left( \frac{\partial p(\mathbf{y})}{\partial \mathbf{n}_{\mathbf{y}}} \right)_A (G_{\mathbf{x},\mathbf{y}})_A - \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \frac{\rho_f}{\Delta t} (G_{\mathbf{x},\mathbf{y}})_S (J_S)_{\mathbf{y}}^T \mathbf{V} = \\ & - \Theta(\mathbf{x}) p(\mathbf{x}) - \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \notin \Gamma_S}} \left( \frac{\rho_f}{\Delta t} \tilde{\mathbf{u}}_{\mathbf{y}} \cdot \mathbf{n}_{\mathbf{y}} \right) G_{\mathbf{x},\mathbf{y}} + \sum_{\substack{\mathbf{y} \in \Gamma_S \\ \mathbf{y} \notin \Gamma_A}} p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}} \\ & - \sum_{\substack{\mathbf{y} \in \Gamma_S \\ \mathbf{y} \in \Gamma_A}} \left( \frac{\rho_f}{\Delta t} \tilde{\mathbf{u}}_{\mathbf{y}} \cdot (\mathbf{n}_S)_{\mathbf{y}} \right) (G_{\mathbf{x},\mathbf{y}})_S + \sum_{\substack{\mathbf{y} \in \Gamma_S \\ \mathbf{y} \in \Gamma_A}} p(\mathbf{y}) \frac{\partial G_{\mathbf{x},\mathbf{y}}}{\partial \mathbf{n}_{\mathbf{y}}}. \quad (5.22) \end{aligned}$$

### 5.4.2 Fluid to Solid Coupling with Triple Points

The modifications needed to handle fluid to solid coupling for triple points are relatively straightforward. The integrals for pressure force and torque in Equations (5.5) and (5.6) integrate the  $p$  terms on the surface of the solid. For points purely on the solid-liquid boundary,  $p$  is an unknown quantity. As a result,  $J\mathbf{p}$  appears fully on the left-hand side of the velocity update in Equation (5.19). At triple points, however,  $p$  is a known quantity. Consequently, the contributions to the pressure force and torque integrals over the surface of the solid at these points are a known quantity, and should be on the right-hand side of our system. That is, Equation (5.19) should be modified to

$$J_S\mathbf{p} + \frac{1}{\Delta t}M\mathbf{V}^{n+1} = \frac{1}{\Delta t}M\mathbf{V}^n - J_{TP}\mathbf{p}, \quad (5.23)$$

where  $J_S$  and  $J_{TP}$  denote the parts of the  $J$  operator that correspond to points purely on the solid-liquid interface and triple points, respectively. When calculating the columns of  $J_{TP}$ , instead of using the full vertex normal  $\mathbf{n}_\mathbf{x}$  as is done in Equation (5.15), we use the solid vertex normal  $(\mathbf{n}_S)_\mathbf{x}$ , which only takes normal contributions from the faces adjacent to the vertex that are classified as solid faces.

### 5.4.3 Two-way Coupled System with Triple Points

We can summarize the full two-way coupled system as

$$\begin{bmatrix} A & K \\ J_S & \frac{1}{\Delta t}M \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{V}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ \frac{1}{\Delta t}M\mathbf{V}^n - J_{TP}\mathbf{p} \end{bmatrix}. \quad (5.24)$$

We note the apparent similarities and differences from Equation (5.20). Again,  $A$  and  $\mathbf{y}$  are constructed as they were described in Section 4.4.3. The vector containing known values  $\mathbf{z}$  is handled similarly to how it was in Section 5.3, except we are careful to ensure that contributions from triple points use the correct component of the vertex normal (i.e., solid component only, when appropriate), rather than the full normal.  $K$  differs from its counterpart in Equation (5.20) in a similar fashion. For each unscripted rigid body in the system,  $K$  is once again an  $N_V \times 3$  or an  $N_V \times 6$  matrix (for 2D or 3D, respectively), but now each row in  $K$  is calculated by:

$$\sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \notin \Gamma_S}} \frac{\rho_f}{\Delta t} G_{\mathbf{x},\mathbf{y}} J_{\mathbf{y}}^T - \sum_{\substack{\mathbf{y} \in \Gamma_A \\ \mathbf{y} \in \Gamma_S}} \frac{\rho_f}{\Delta t} (G_{\mathbf{x},\mathbf{y}})_S (J_S)_{\mathbf{y}}^T,$$

from the left-hand side of Equation (5.22). The bottom rows of the system are derived from Equation (5.23), and are otherwise constructed similarly to how  $J$  was constructed in Section 5.3.

With these updates, we are able to support simulations involving fluid-solid two-way rigid-body coupling in a variety of settings, as appropriate triple point handling is crucial to many interesting setups.

# Chapter 6

## Results and Discussion

In this chapter we present results demonstrating our scheme’s ability to couple surface-only liquids with rigid bodies, as well as qualitative and quantitative validation of the two-way coupling. Our results were generated on a Mac M1 8-core processor, with our serialized implementation written in C++, using the Eigen library [Guennebaud et al., 2010] for vector and matrix operations. We use Eigen’s direct LU factorization-based solver to solve the linear system in the BEM step described by Equation (5.24). We re-mesh the liquid surface at each time step with a basic re-meshing algorithm that consists of an edge-collapse, edge-split, and smoothing step, with velocities on mesh vertices linearly interpolated as appropriate. Edges are collapsed or split based on a user-defined edge length tolerance. For further reference, our re-meshing technique is essentially a simplified, 2D version of that which is detailed in Section 4 of the work by Botsch and Kobbelt [2004]. Our current implementation does not support mesh topology changes, and all results shown in this chapter are designed such that the mesh topology stays consistent for the duration of each simulation. We also do not support solid-solid collisions.

Figures 6.3, 6.5, 6.6, 6.12, and 6.13 visualize marker particles within the liquid domain. These passive particles are advected by reconstructing the interior potential flow field of the liquid domain from the liquid boundary at each time step using Equation (4.9), but do not affect any part of the simulation of the liquid domain.

Collectively, our examples demonstrate that despite the relative simplicity of the potential flow model of fluids, it is able to capture a wide range of nontrivial interactions among rigid bodies and liquids.

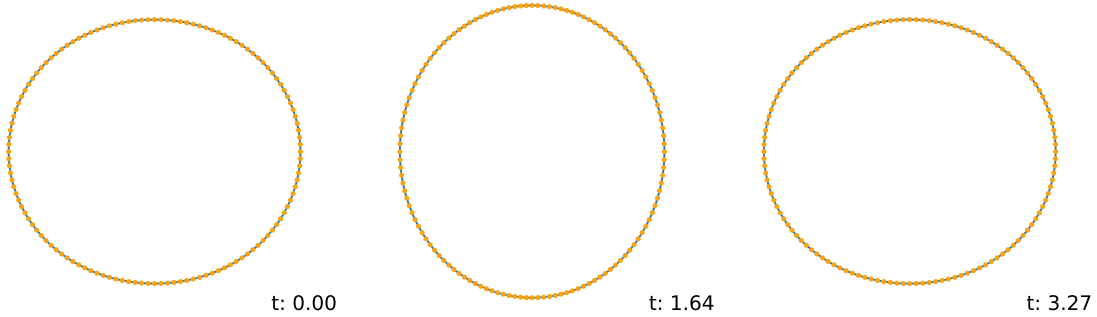


**Figure 6.1: One period of an oscillating droplet.** A droplet with mesh resolution  $N = 128$  is started from rest. Surface tension acting on the initially perturbed shape (relative to a perfect circle) causes the droplet to begin to oscillate between a horizontally and vertically oblong shape. One period of oscillation is shown.

| Mesh Resolution, $N$ | Period, $T$ | $T_{error} = \ T - \pi\ $ |
|----------------------|-------------|---------------------------|
| 16                   | 4.012       | 0.870                     |
| 32                   | 3.482       | 0.340                     |
| 64                   | 3.312       | 0.170                     |
| 128                  | 3.266       | 0.124                     |

**Table 6.1:** Period of oscillation and errors after one period of an initially perturbed 2D droplet for various mesh resolutions.

**Surface Tension-Driven Droplet Oscillation** We validated our 2D implementation of the Surface-Only Liquids approach by calculating the period of oscillation of an inviscid 2D droplet with some initial perturbation and comparing to the following analytical result by [Lamb \[1932\]](#), as is done in Section 3.2 of the work by [Enright et al. \[2003\]](#). If a 2D droplet has an initial surface given in polar coordinates by

$$r = a + \epsilon \cos(n\theta), \quad (6.1)$$

where  $a$  is the unperturbed radius,  $n$  is the mode of amplitude, and  $\epsilon$  is the amplitude of the oscillation, then the frequency  $f$  of the droplet’s oscillation is given by

$$f^2 = (n^3 - n) \frac{\sigma}{\rho_f a^3}, \quad (6.2)$$

where  $\sigma$  is the surface tension coefficient and  $\rho_f$  is the liquid density.

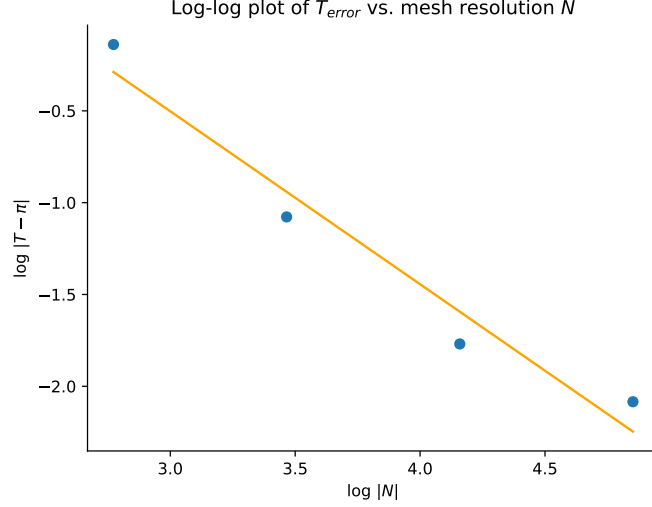
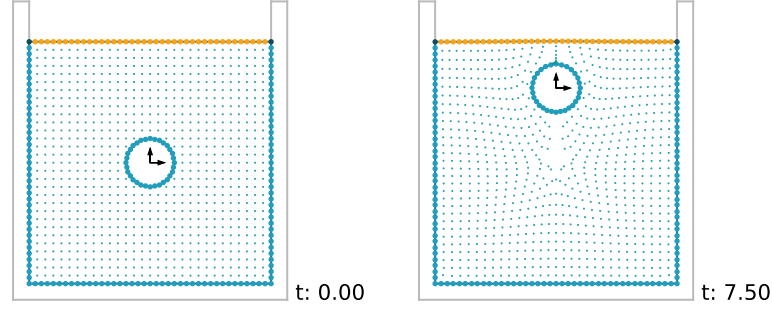


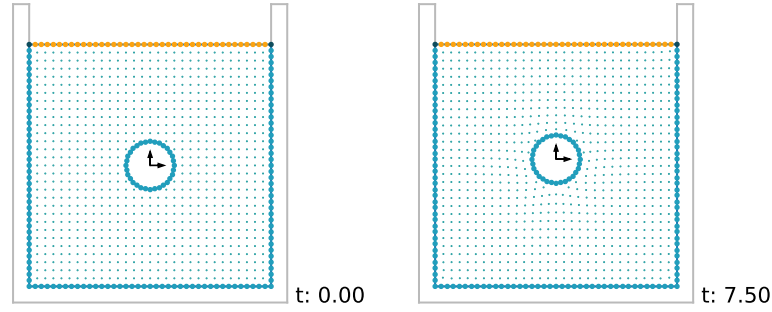
**Figure 6.2: Error in period of oscillation versus mesh resolution.** Log-log plot of error in period of oscillation after one period of an initially perturbed 2D droplet versus mesh resolution. Line of best fit (shown in orange) has slope of  $-0.942$ .

Following the test performed by [Enright et al. \[2003\]](#), we obtain a theoretical period of oscillation of  $\pi$  with the  $n = 2$  mode by setting the coefficients as follows:  $\rho_f = 27$ ,  $a = \frac{1}{3}$ ,  $\sigma = \frac{2}{3}$ , and  $\epsilon = 0.05a$ . To approximate the amplitude of oscillation, we track the value of  $r$  at the point at  $\theta = 0$  through times for different levels of mesh refinement. Figure 6.1 shows one period of oscillation of a droplet with a starting shape set by these specifications with mesh resolution of  $N = 128$ . We calculate the experimental period of oscillation by taking the difference between the two peaks in the amplitude; these can be found in Table 6.1. As seen from Table 6.1 and Figure 6.2, the period converges to  $\pi$  as we increase mesh resolution.

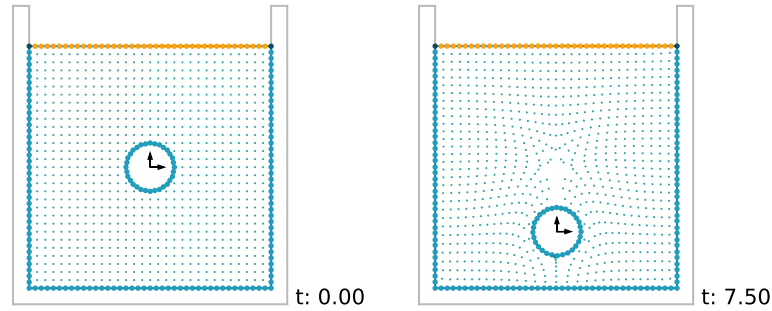
**Submerged Solids of Varying Density in a Small Domain** Figure 6.3 shows rigid bodies of varying density in a 2D cup of liquid, where the liquid density is held at  $\rho_f = 1$  and gravity points downwards. All simulations were performed with a time step size of 0.005 and an initial mesh resolution of  $N = 174$ . Surface tension coefficients for liquid-air, solid-liquid, and solid-air were set equal to one another. The solids are shown at  $t = 0$  and  $t = 7.5$ . As expected, the solid with lower density than the surrounding liquid rises towards the top of the cup, and the solid with higher density sinks towards the bottom of the cup. The solid with equal density to the fluid also appears to rise, contrary to the



(a)  $\rho_s = 0.9\rho_f$



(b)  $\rho_s = \rho_f$



(c)  $\rho_s = 1.1\rho_f$

**Figure 6.3: Submerged rigid bodies with varying density.** Submerged rigid bodies of various density relative to the surrounding liquid starting from rest at  $t = 0$  (left) until  $t = 7.5$  (right).  $\rho_s$  and  $\rho_f$  denote solid and fluid/liquid density, respectively.

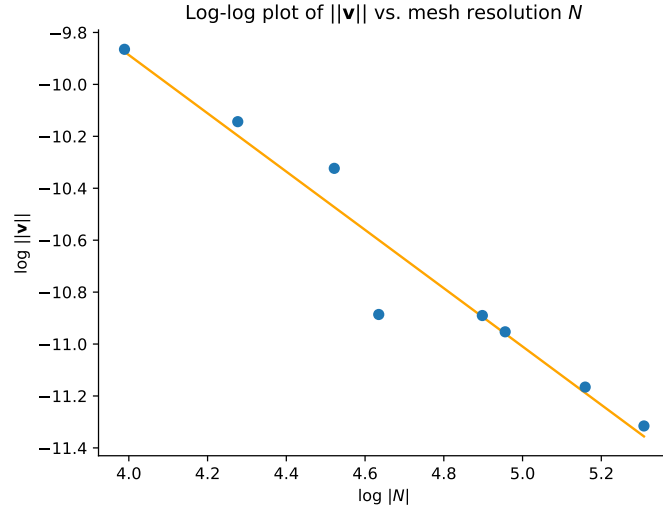


**Figure 6.4: Error in velocity versus mesh resolution for a submerged rigid body with density  $\rho_s = \rho_f$ .** Log-log plot of error in velocity after one time step of size  $\Delta t = 0.0001$  of a submerged solid with equal density to the surrounding liquid versus mesh resolution. Expected velocity is 0. Line of best fit (shown in orange) has slope of  $-1.123$ .

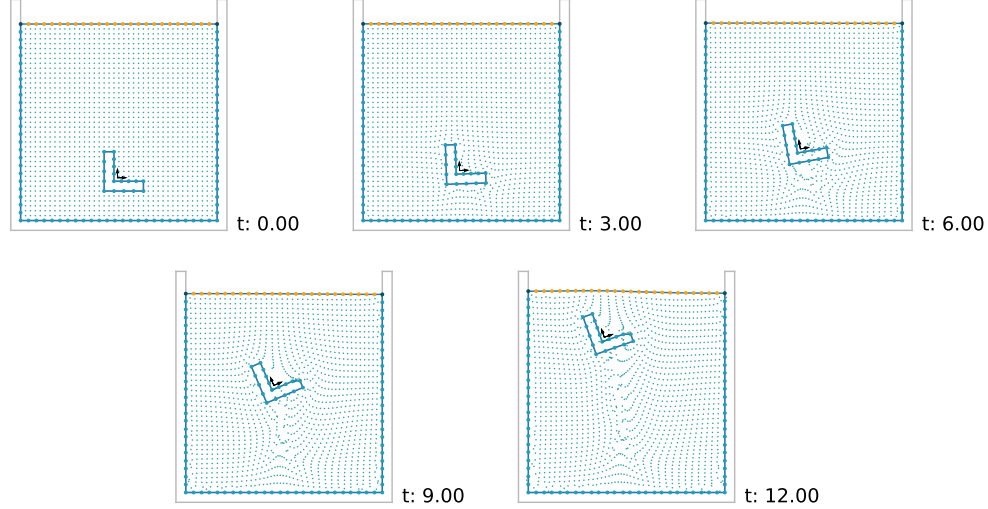


**Figure 6.5: Irregularly shaped submerged rigid body of lower density than surrounding fluid.** An L-shaped rigid body of density  $\rho_s = 0.9\rho_f$  started from rest immersed in a cup of liquid translates and rotates as it rises.

expectation that it would remain at a neutral position. However, we show that the rigid body converges towards zero vertical velocity under spatial refinement; see Figure 6.4.

#### Submerged Irregularly-Shaped Solids of Varying Density in a Small Domain

Figure 6.5 shows an L-shaped rigid body in a 2D cup of liquid, where the liquid density is held at  $\rho_f = 1$  and the solid density is  $\rho_s = 0.9$ . The simulation is started from rest. The irregular shape of the rigid body causes it to rotate and translate horizontally relative to its starting position as it rises in the fluid. Figure 6.6 shows a similar setup, except with the solid density held at  $\rho_s = 1.5$ . Once again, we see that the irregular shape of the rigid body causes it to rotate and translate horizontally as it sinks.

#### Submerged Solids of Varying Density in a Large Domain

To further validate our coupling model for a submerged rigid body, we compared against the analytical solution for a submerged 2D disk accelerating in an infinite domain of fluid. The analytical solution also assumes that the fluid is incompressible and irrotational, consistent with our model.

A naive approach to determining the analytical solution for an accelerating submerged rigid body would be to simply calculate the net forces acting on the body as the sum of

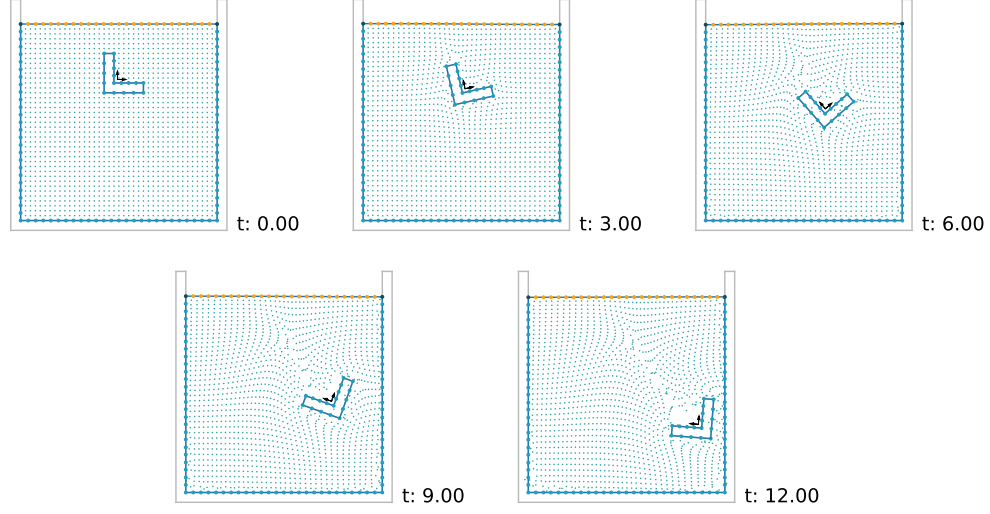


**Figure 6.6: Irregularly shaped submerged rigid body of higher density than surrounding fluid.** An L-shaped rigid body of density  $\rho_s = 1.5\rho_f$  started from rest immersed in a cup of liquid translates and rotates as it sinks.

the gravitational force and the buoyancy force, and determine the acceleration from the net forces. This results in the acceleration of the solid being given as

$$\mathbf{a} = \left( \frac{M_f - M_s}{M_s} \right) \mathbf{g} = \left( \frac{\rho_f - \rho_s}{\rho_s} \right) \mathbf{g}, \quad (6.3)$$

where  $M$  is mass,  $\rho$  is density, and  $\mathbf{g}$  is gravitational acceleration, with subscripts  $f$  and  $s$  denoting quantities for the fluid and solid, respectively. However, this model of the net forces is incomplete. It does not account for the fact that as the solid accelerates through the fluid, the surrounding fluid also must accelerate to move around the solid; this effect can be observed in the behaviour of the marker particles in Figure 6.3. The acceleration of the fluid thus adds inertia to the system that is unaccounted for by Equation (6.3). It can be shown that a submerged body accelerates as if it has *some additional mass* that is related to the mass of the fluid displaced, rather than just its own mass; in fluid mechanics, this additional mass is known as the *added mass* [Lamb, 2022; Newman, 1977]. For a 2D disk in an infinite domain, it can be shown that the added mass is equal to the mass of the displaced fluid. That is, accounting for added mass, the acceleration is given by [Lamb, 2022; Newman, 1977]

$$\mathbf{a} = \left( \frac{M_f - M_s}{M_s + M_f} \right) \mathbf{g} = \left( \frac{\rho_f - \rho_s}{\rho_s + \rho_f} \right) \mathbf{g}. \quad (6.4)$$

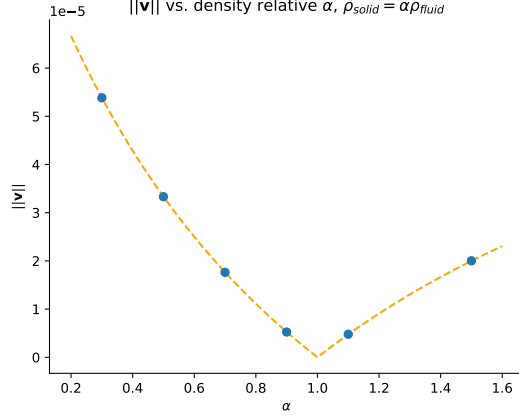


**Figure 6.7: Magnitude of a disk-shaped rigid body’s velocity versus the solid density.** Plot of the magnitude of a disk-shaped rigid body’s velocity,  $\|\mathbf{v}\|$ , after one fixed time step versus the relative density of the solid to the fluid. The disk is small (radius of 0.3) compared to the square fluid domain (length of one side of the domain is 80). The analytical/expected velocity magnitude is shown with a dashed line.

It is this analytical solution given by Equation (6.4) above that we refer to for the remainder of this analysis.

Since our simulation does not currently support an infinite domain, we performed the analysis as follows. We simulated a disk-shaped rigid body of a fixed size (in our case, radius of 0.3) submerged at the center of a relatively large square fluid domain (one side of the domain had length 80) beginning at rest for a single fixed time step (in our case,  $\Delta t = 0.001$ ). The square fluid domain was placed in a cup of the same size; see Figure 6.3 for an image of a similar setup. For an infinite domain, the expected translational velocity after one time step of size  $\Delta t$  of a disk is given by

$$\mathbf{v}_{theoretical} = \mathbf{a}\Delta t = \left( \frac{\rho_f - \rho_s}{\rho_s} \mathbf{g} \right) \Delta t. \quad (6.5)$$

We then varied the density of the rigid body disk relative to the fluid density. Figure 6.7 shows the magnitude of the velocity of the rigid body after the single time step plotted against the density of the rigid body. The dashed line in Figure 6.7 shows the analytical solution. The density of mesh points on the disk and liquid domain were held constant.

Figure 6.8 shows that our simulation approaches the analytical solution for this scenario with mesh refinement. We again simulate an immersed disk-shaped rigid body with radius

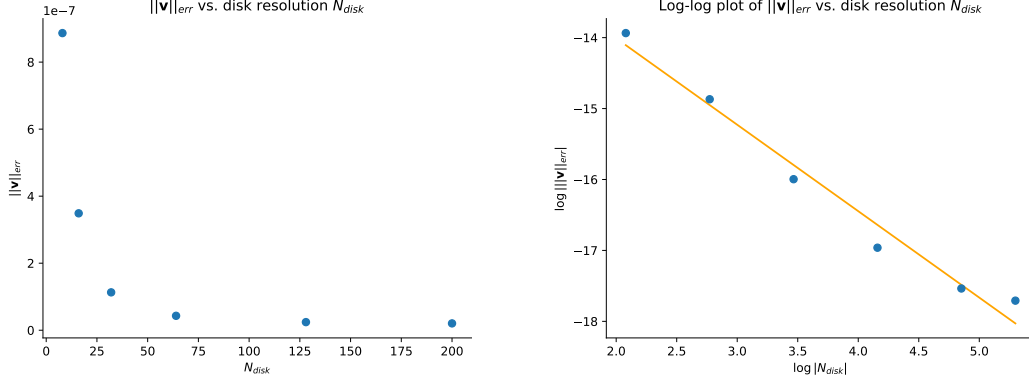
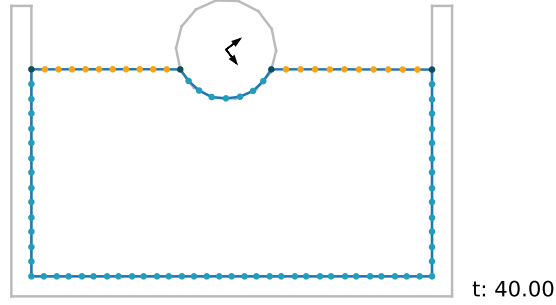


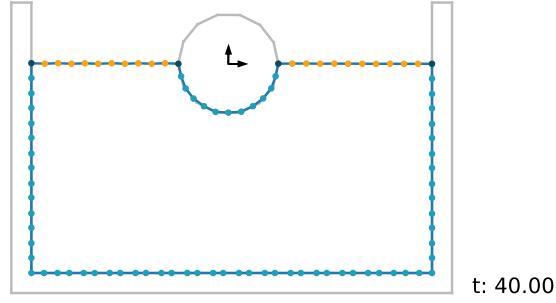
**Figure 6.8: Error in the magnitude of a disk-shaped rigid body’s velocity versus mesh resolution.** Plot and log-log plot of error in the magnitude of velocity after one timestep for a disk with varying mesh resolution ( $N_{disk}$  is the number of mesh points on the disk) and fixed size in a square domain of fixed size with side length  $d = 80$ , where the disk has density 0.9 times that of the liquid density. Line of best fit (shown in orange) for the log-log plot has slope  $-1.219$ .

0.3 started from rest in a relatively large square liquid domain with side lengths of  $d = 80$  and fixed mesh density for a single time step. We use step size  $\Delta t = 0.0001$ . However, this time we fix the density of the rigid body to 0.9 times that of the density of the surrounding fluid. We then varied the the mesh resolution of the liquid mesh on the circular disk.

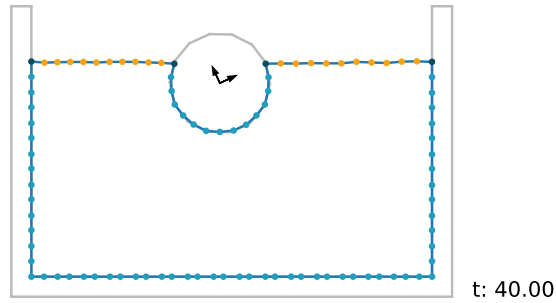
**Floating Solids of Varying Density** Figure 6.9 shows rigid bodies of varying density floating in a 2D cup of liquid, where again the liquid density is held at  $\rho_f = 1$  and gravity points downwards. Each simulation was performed with a time step size of 0.005 and an initial mesh resolution of  $N = 90$ . Surface tension coefficients for liquid-air, solid-liquid, and solid-air were all set to 0. For each scenario, the simulation was configured so that the rigid body was started close to its expected equilibrium position. The simulation was then run until the liquid level mostly stabilized, at  $t = 40$ . As seen in the figure, varying the density of the solid causes different amounts of the solid to be submerged at equilibrium. We can approximately calculate the percentage of the disk that is submerged from the position of the disk and the triple points where the liquid contacts the disk on each side. For the  $\rho_s = 0.25\rho_f$ ,  $\rho_s = 0.5\rho_f$ , and  $\rho_s = 0.75\rho_f$  cases respectively, the percentage of the disk submerged was 25.37%, 50.12%, and 73.99%, compared to the expected values of 25%, 50%, and 75%, corresponding to the density ratios of the solid to the liquid.



(a)  $\rho_s = 0.25\rho_f$

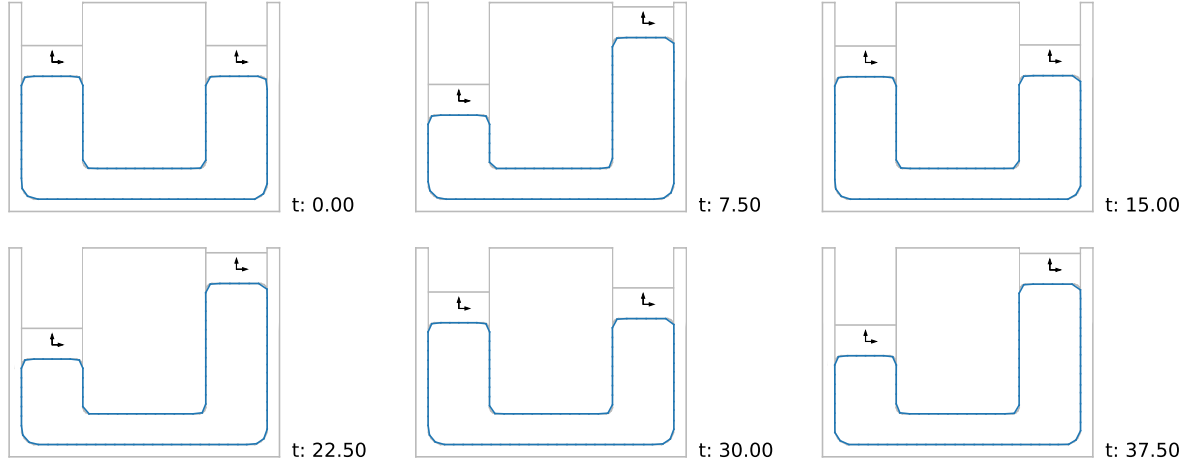


(b)  $\rho_s = 0.5\rho_f$

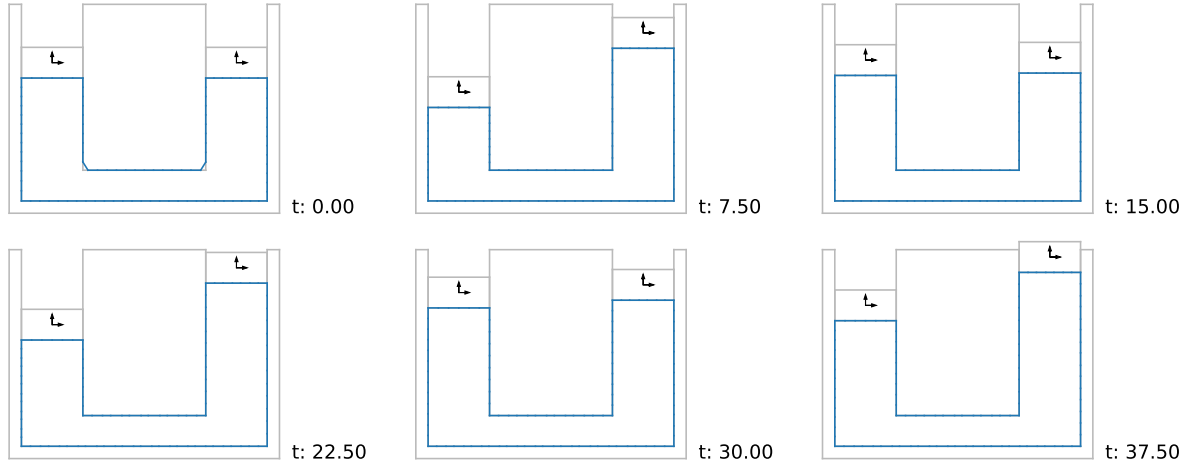


(c)  $\rho_s = 0.75\rho_f$

**Figure 6.9: Floating rigid bodies with varying density.** Floating rigid bodies of various density relative to the surrounding fluid, with simulations run until liquid mostly stabilized. The percentage of the disk that is submerged for each case is (a) 25.37%, (b) 50.12%, and (c) 73.99%. Note that the submerged areas closely agree with the corresponding density ratios, as expected.



(a) Rounded liquid domain corners



(b) Sharp liquid domain corners

**Figure 6.10: Two rigid bodies of unequal density cause oscillations when placed on the ends of a pipe of liquid.** Liquid is set up in a 2D pipe with rigid bodies set at either end of the liquid domain. The rigid body on the left end is twice the density of the liquid and the rigid body on the right end. Rigid bodies are constrained to only move in the vertical direction. Starting the simulation from rest, we observe oscillating behaviour, as expected. The cup, block, and rigid bodies in (a) are shaped such that the liquid mesh has round corners when in contact with them, while the corners in (b) are 90 degrees. Noticeable volume gain occurs in the latter simulation during each oscillation.

**Multibody Systems** Figures 6.10a, 6.11, 6.12, and 6.13 demonstrate our method’s ability to support multiple rigid bodies coupled to the fluid domain. Rigid bodies are able to affect each other’s movement through the liquid domain.

Figure 6.10a shows a simulation of a 2D pipe of liquid with rigid bodies of equal size set flush at both ends of the liquid. The rigid body on the left has double the density of both the fluid and the rigid body on the right. The simulation is started from a rest state with the liquid level set equal on both sides of the pipe. As expected, due to the imbalance in the masses of the rigid bodies, the heavier rigid body on the left pushes the liquid down, causing the liquid level and rigid body on the right to rise up, and the system begins to oscillate. Each period of oscillation returns the system to its original state, as expected in a system with no energy dissipation. These simulations are set up such that the rigid bodies are only free to move in the vertical direction, with their horizontal and rotational degrees of freedom constrained. Our implementation does not currently support solid-solid collision handling, without which small errors in the horizontal or rotational velocities in the rigid bodies build up and cause the rigid bodies to erroneously fall off the top of the liquid and through the cup walls.

We see noticeable non-physical volume gain in the liquid domain occur when the simulated liquid domain has sharp corners; see Figure 6.10b. We identified the source of this erroneous volume gain to be limitations in the way in which sharp corners are handled by the BEM solve when integrating external forces. Since we use a vertex-located BEM to solve for  $\frac{\partial p}{\partial \mathbf{n}}$ , we require a normal vector to be well-defined at each vertex on the mesh. In our implementation, we calculate a normal for each vertex by taking a weighted average of the adjacent face normals. While this is a reasonable approach for vertices on smooth parts of the liquid domain, at sharp corners this approach introduces inaccuracies. Sharp corners in the liquid domain should have two distinct normals—one on each side of the vertex—similar to how the sharp corner at triple points had distinct solid-liquid and liquid-air normals. By averaging out these normals to find the vertex normal, we end up treating what should be sharp corners in the liquid domain as angled edges. As a result, we introduce inaccurate behaviour in the velocity field at these points, leading to issues with volume preservation in our model. We discuss possible methods to alleviate this issue in Chapter 7. For now, we find that rounding the corners in contact with the liquid so that the liquid domain does not have sharp corners, such as is shown in Figure 6.10a, allows us to observe significantly less erroneous volume gain, confirming our expectation that this issue is the source of the error.

Figure 6.11 demonstrates a similar oscillation phenomena as Figure 6.10a, with the introduction of a free surface and floating rigid body. The rigid body on the right and the liquid have equal density. The circular rigid body on the left has half that density. While

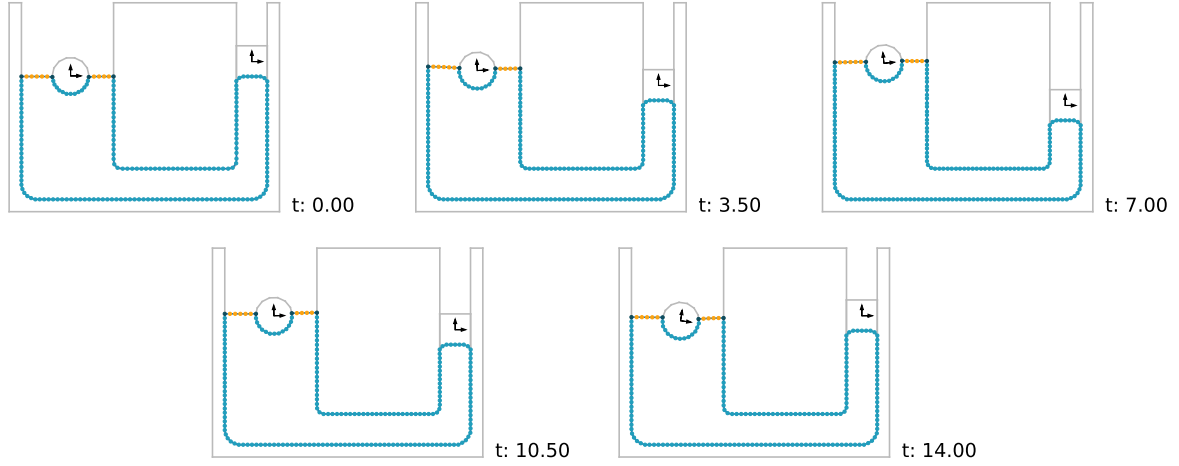


**Figure 6.11: A system with two rigid bodies and a free surface oscillates.** Similar to the previous oscillating setup, but with a disk set on a free surface on the left. The disk is half the density of the rigid body on the right, which has equal density to the liquid. The rigid body on the right is constrained to only move in the vertical direction, while the disk is free to rotate and move in any direction.

the rigid body on the right is still constrained to only move in the vertical direction, the circular rigid body's degrees of freedom are not constrained.

Figure 6.12 shows a multibody system with a liquid domain that contains a hole in its topology due to a submerged rigid body. All rigid bodies in the system have half the density of the liquid domain. The difference in density causes the larger submerged disk, started at rest, to accelerate upwards. Due to this upward movement, the top surface of the liquid peaks and the two smaller disks move apart as the larger disk approaches them. We again see that the system supports the simulation of rigid bodies interacting with the liquid domain and in turn affecting other rigid bodies.

Figure 6.13 demonstrates a similar capability as the last, now using three disks with equal size. Two disks of density equal to that of the liquid are submerged a small horizontal distance apart from one another. A third disk of density 0.3 times the liquid density is submerged underneath these two disks. As the third disk accelerates upwards due to its buoyancy, the two smaller disks move apart, allowing the third disk to pass between them.

**Preliminary Timing Results** Table 6.2a shows some preliminary wall-clock timing results for a few of our example simulations. Table 6.2b shows the associated mesh sizes.

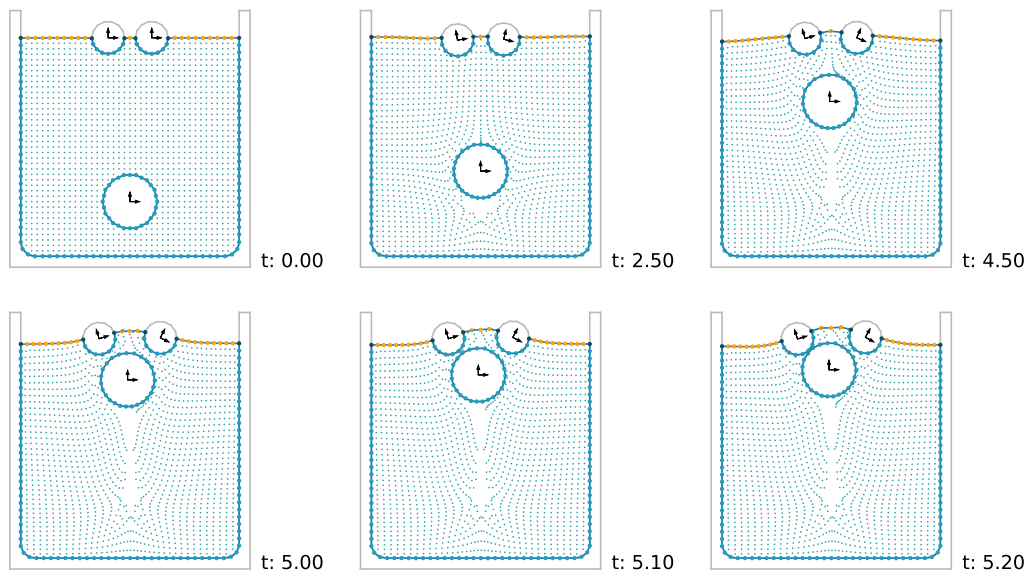


**Figure 6.12:** Two floating disk-shaped rigid bodies translate and rotate due to a third rigid body accelerating in the liquid below them. A submerged disk accelerating upwards causes two floating disks at the surface to move apart from each other before the submerged disk makes contact. All disks have half the density of the liquid.

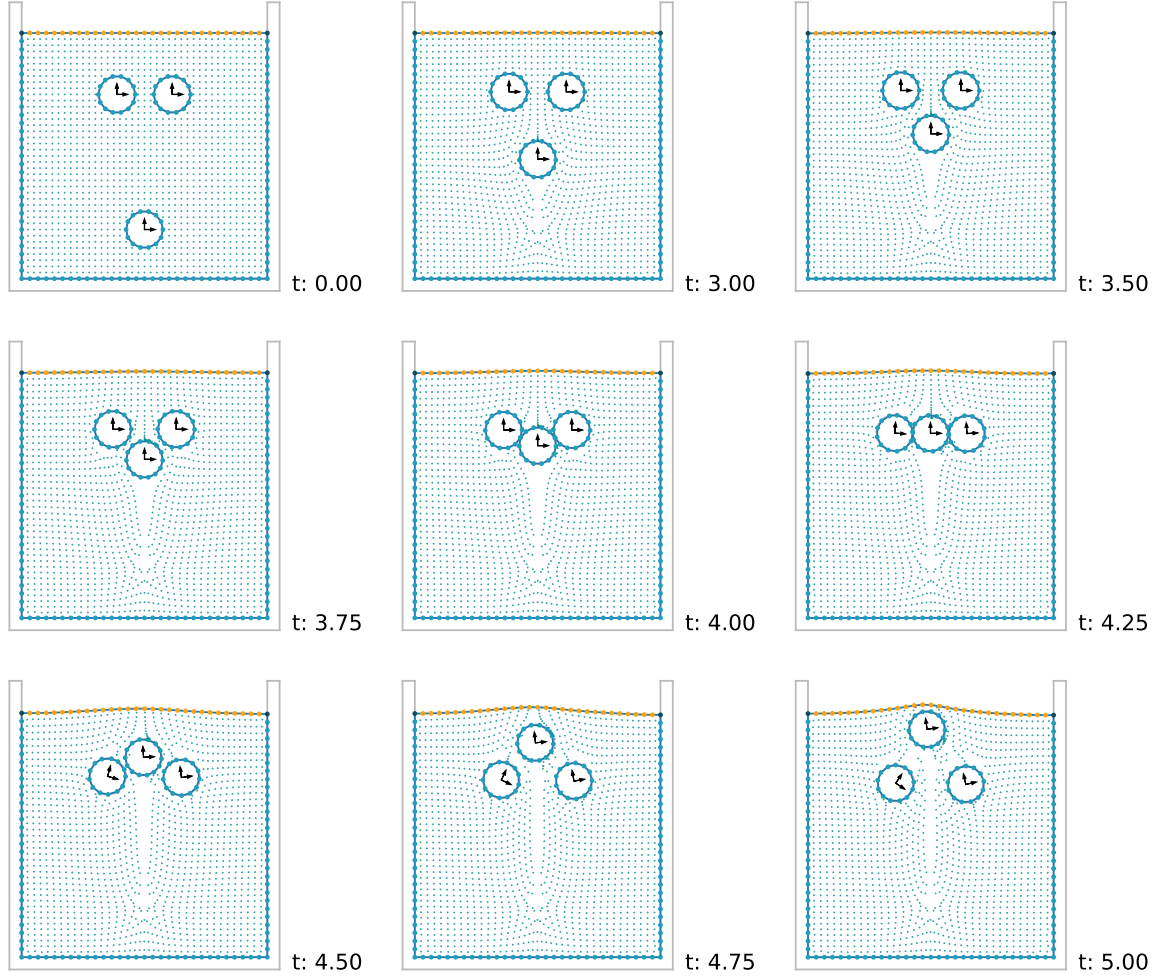


**Figure 6.13:** Two submerged disk-shaped rigid bodies translate and rotate due to a third rigid body accelerating in the liquid below them. A submerged disk accelerating upwards causes two submerged disks above it to move apart from each other and allow the accelerating disk to pass between them. The accelerating disk has 0.3 times the density of the liquid, and the two other disks have equal density to the liquid.

| Simulation                                                                                         | Mean<br>step time<br>(ms) | Max step<br>time (ms) | Total run<br>time (s) | Time step<br>size (s) |
|----------------------------------------------------------------------------------------------------|---------------------------|-----------------------|-----------------------|-----------------------|
| Disk floating in cup, Fig. 6.9b                                                                    | 17.6837                   | 151.608               | 141.823               | 0.005                 |
| Multi-body oscillation, Fig. 6.10a                                                                 | 16.6392                   | 196.96                | 66.569                | 0.01                  |
| Liquid-only oscillation, Fig. 6.14                                                                 | 16.2674                   | 235.079               | 65.0698               | 0.01                  |
| Cup of liquid ( $\sigma_{SA} = 1.1\sigma_{SL} = 1.1\sigma_{LA}$ ),<br>$N \approx 80$ , Fig. 6.15a  | 12.2909                   | 123.911               | 98.327                | 0.005                 |
| Cup of liquid ( $\sigma_{SA} = 1.1\sigma_{SL} = 1.1\sigma_{LA}$ ),<br>$N \approx 160$ , Fig. 6.15b | 44.6182                   | 186.978               | 356.946               | 0.005                 |

(a) Run time data

| Simulation                                                                                         | Mean mesh size | Max mesh size |
|----------------------------------------------------------------------------------------------------|----------------|---------------|
| Disk floating in cup, Fig. 6.9b                                                                    | 93             | 93            |
| Multi-body oscillation, Fig. 6.10a                                                                 | 94             | 94            |
| Liquid-only oscillation, Fig. 6.14                                                                 | 97             | 97            |
| Cup of liquid ( $\sigma_{SA} = 1.1\sigma_{SL} = 1.1\sigma_{LA}$ ),<br>$N \approx 80$ , Fig. 6.15a  | 80             | 80            |
| Cup of liquid ( $\sigma_{SA} = 1.1\sigma_{SL} = 1.1\sigma_{LA}$ ),<br>$N \approx 160$ , Fig. 6.15b | 160.259        | 162           |

(b) Mesh size data

**Table 6.2:** Run time and mesh size data for a sampling of example simulations. All simulations were run for a simulated time of 40 seconds.

We include both mean and maximum mesh sizes as we allow for re-meshing in our implementation, although the examples shown in the table seem to have stayed fairly consistent in terms of mesh size throughout the simulation time. Timing results for Figures 6.9b and 6.10a, both of which use our coupling scheme, are shown. We also include results for simulations that do not involve our coupling scheme, both to act as a baseline to compare our coupled simulations against, as well as to gain some preliminary insight into the behaviour of the surface-only liquids framework in terms of efficiency.

To determine whether adding a coupled rigid body to a system would cause any major increase in the run time of the simulation, we set up a simulation similar to the multi-body oscillation simulation shown in Figure 6.10a, except without any rigid bodies. That is, we once again construct a tube of liquid, but instead of using unequal density rigid bodies to cause the oscillations of the liquid, we initialize the liquid to be at uneven heights on either

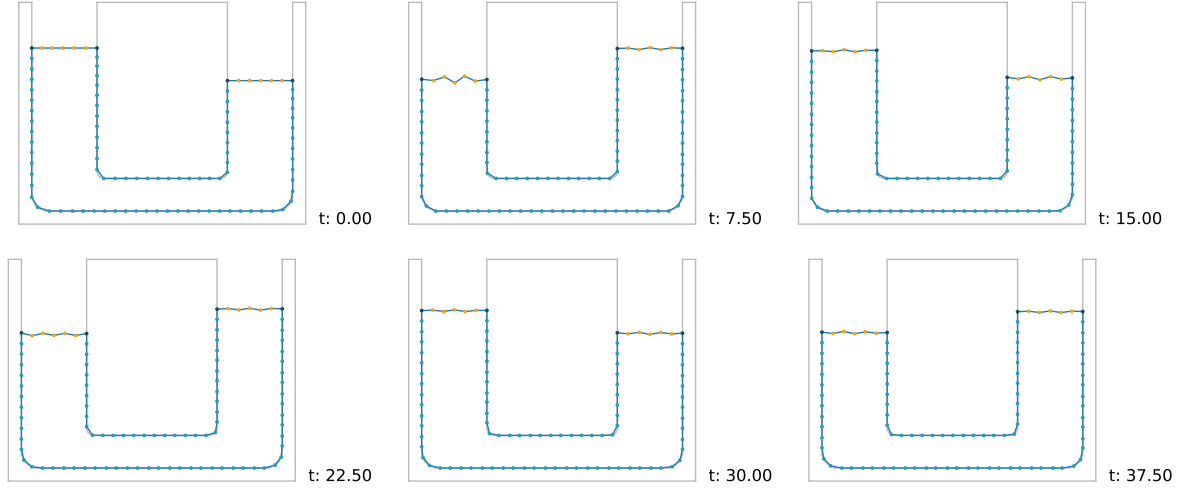
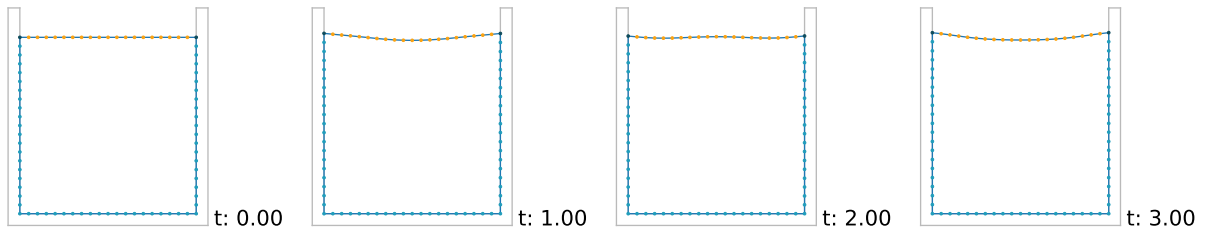


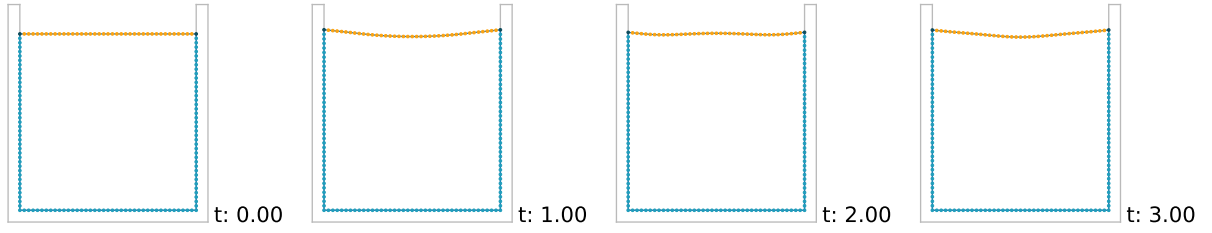
**Figure 6.14: Uneven liquid heights on each side of a pipe cause oscillations.** Liquid is set up in a 2D pipe with unequal amounts of liquid on each side of the pipe. Starting the simulation from rest, we observe oscillating behaviour, as expected. This simulation is designed so that the liquid will exhibit similar behaviour to that in Figure 6.10a, but omits rigid bodies.

side of the tube to cause the oscillations; see Figure 6.14. As seen in Table 6.2b, the two oscillation tests have similar mesh sizes throughout their duration. We observe from Table 6.2a that the two oscillation tests also have similar run times; the addition of coupled rigid bodies to the system does not appear to cause any substantial change in either overall or per-step run time. This aligns with expectations; as the coupled rigid bodies only add a few degrees of freedom to the BEM system, we would not expect the addition of rigid bodies to dominate the computational cost.

We also simulated a cup of liquid with a solid-air surface tension coefficient unequal to the solid-liquid and liquid-air coefficients at two different mesh resolutions; see Figure 6.15. We set the surface tension coefficients unequal to one another simply so that the simulation has slightly more interesting behaviour to model; when all coefficients are equal, we would expect a cup of liquid to remain static. We see in Table 6.2a that roughly doubling the mesh resolution results in about a four-fold increase in total run time.



(a)  $N \approx 80$



(b)  $N \approx 160$

**Figure 6.15: Liquid oscillating in a cup due to unequal surface tension forces at two mesh resolutions.** The solid-air surface tension coefficient  $\sigma_{SA}$  is held above the solid-liquid and liquid-air coefficients  $\sigma_{SL}$  and  $\sigma_{LA}$ , so that the liquid's free surface oscillates when started from rest at  $t = 0$ . Two mesh resolutions are shown.

# Chapter 7

## Conclusion

We have demonstrated that the surface-only liquid simulation method originally developed by Da et al. [2016] can be successfully extended to handle two-way coupling with solid rigid bodies. In Chapter 6 we showed that our surface-only coupled model behaves as we would expect in a variety of simulation scenarios. Collectively, our examples demonstrate that despite the relative simplicity of the potential flow model of fluids, it is able to capture a wide range of nontrivial interactions among rigid bodies and liquids. We consider below some shortcomings of our method and highlight some potential directions for future investigation.

### 7.1 Limitations

As our coupling model builds off of the surface-only liquid scheme proposed by Da et al. [2016], our method shares in the limitations of their method. Since both parts of the projection step in our formulation depend on the assumption that the interior of the liquid domain is irrotational as well as incompressible, we are unable to simulate liquid flows with internal vorticity. Da et al. [2016] note that this is not an issue for flows with irrotational initial states, as vorticity cannot spontaneously appear inside of a uniform density fluid with no rotation. However, physically, vorticity can be generated at the interface between liquids and solids because of viscosity at these interfaces that can be summarized by *no-slip* boundary conditions. That is, viscosity at solid-liquid interfaces generates internal vorticity which our model is unable to capture. This perhaps results in our model not behaving completely consistently with physical liquid-solid coupled flows, where vorticity generated from solid-liquid interfaces can permeate into the interior of the liquid where it

may also affect other solids in the liquid. However, in the idealized inviscid flow described by the Euler equations, the absence of viscosity means that vorticity generated at these surfaces does not diffuse into the interior of the liquid domain, suggesting that at the very least our model should behave in accordance with the idealized inviscid flow model.

As shown in the oscillating multibody simulation setups in Chapter 6, our current model experiences noticeable volume gain in certain simulations setups. This can be attributed to the fact that our vertex-located BEM formulation is unable to properly handle sharp features in the liquid surface mesh, as vertex normals are ill-defined on such corners. Instead of treating sharp corners as having two distinct normals on each side of the crease, we currently approximate the vertex normal by averaging out the face normals onto the vertex. Because the sharp corners are therefore treated as if they were angled edges, our BEM pressure solve ends up finding pressures that are inconsistent with the true surface mesh. As a result, the update to the end-of-step velocity from the BEM pressure solve is also inconsistent. Coupled with the simple liquid-solid collision handling scheme we have implemented in our code, wherein liquid points that cross into the solid are simply projected onto the closest point on the solid surface, this method introduces conspicuous issues with volume conservation in our model.

There are several approaches with which we might alleviate the issues with volume conservation. Since we identify the BEM step as one of the main sources of error in our end-of-step velocities, it is natural to consider different BEM formulations to improve the model. We suspect that a BEM formulation that is vertex-located for pressure values but face-located for the pressure derivative values may provide improved results, as it would likely not have the issues with handling sharp features in the liquid mesh that appear when using the purely vertex-located method. For example, in their Galerkin (as opposed to a collocated) BEM solve for pressures, [Huang and Michels \[2020\]](#) use piecewise constant boundary elements on faces for the normal derivatives of pressure while using linear bases defined on vertices for pressure values; we refer the reader to their supplementary material for further details. In fact, this mixed representation appears to be fairly conventional for symmetric Galerkin BEM schemes [[Rjasanow and Steinbach, 2007](#)]. Alternatively, an approach such as that by [Shen and Sterz \[1998\]](#), where a Galerkin method is used on corners and edges, while collocation is used elsewhere on the mesh in the interest of efficiency, could prove effective as well. Such changes in BEM formulation would likely also necessitate an update to the treatment of triple points from the method described in this thesis. Alternatively, we could consider adding a volume correction step to enforce volume preservation during or after the advection step in our method, since any volume loss or gain issues occur in the advection step as it is the only step that directly manipulates the surface mesh position. However, adding volume correction is likely not a trivial process,

especially when considering the degrees of freedom at the free surface as well as the fact that unscripted rigid bodies could be moved by volume corrections steps. Additionally, considering that we have identified the source of the issues with volume preservation to be part of the model itself, it would be preferable to first improve the BEM step prior to resorting to incorporating a step explicitly for volume correction. Finally, since the volume conservation issues stem from both the BEM as well as our liquid-solid collision model, it is also possible that incorporating an improved scheme for handling the solid-liquid interface would also be beneficial in alleviating these problems.

## 7.2 Future Work

Our implementation currently does not allow for mesh topology changes (i.e., merging and splitting of the mesh), which is why we do not recreate several interesting phenomena that require mesh topology changes such as liquid splashes, droplet collisions, and rigid bodies breaking through surfaces. We suspect incorporating a surface-tracking library such as *El Topo* [Brochu and Bridson, 2009] into our implementation would allow for the simulation of many more interesting phenomena. As this is a limitation of our implementation rather than of the liquid simulation or liquid-solid coupling model, we include it in our future work rather than in our limitations.

While we are able to simulate several interesting scenarios with our current implementation, and would undoubtedly expand the range of scenarios further by incorporating a robust 2D surface-tracking library, we recognize that our work here is primarily a prototype for a 3D simulation. As such, the natural immediate next step is a 3D implementation of the strong coupling method presented in Chapter 5. We do not foresee the transition from 2D to 3D requiring any major modifications to our coupling scheme. As described, we achieve two-way coupling by making modifications to the right- and left-hand sides of the BEM step as well as including equations for determining forces on rigid bodies from the fluid. Making similar modifications to the BEM step in a 3D implementation of surface-only liquids, such as the source code provided by Da et al. [2016], should not present any new complexities past the differences that are already present between 2D and 3D BEMs.

Apart from transferring this method to 3D, there are several additions that could further improve the model presented here. The inclusion of handling for solid-solid collisions would certainly allow for the simulation of a wider variety of scenes. We briefly alluded to the possibility of exploring improved time stepping schemes for advection in Section 4.3.1, as the Forward Euler approach that we currently employ is a scheme known to be numerically unstable. However, we recognize that employing different time stepping schemes may be

non-trivial. As an example, employing a method such as Runge-Kutta for this step could require an intermediate velocity of some sort, but it is not immediately obvious how one would evaluate such a value in the surface-only liquids framework.

An important, but outstanding, question that neither we nor [Da et al. \[2016\]](#) have directly investigated is the efficiency of the surface-only liquids framework compared to “standard” fluid solvers. While we show some preliminary timing data in Chapter 6, these are far from rigorous analyses that are imperative to determining whether the surface-only liquids framework, as well as modifications to it like the one presented here, are viable alternatives to volumetric methods in terms of practical usage. Comparing our method against a standard grid-based coupling model, such as that of [Batty et al. \[2007\]](#), would allow for much insight into how the surface-only framework performs in terms of efficiency and scalability.

A direct comparison to an existing liquid-solid coupling model would also serve a secondary purpose as further validation of the scheme presented here. It is difficult to set up real-world tests to determine if our model exactly replicates physical coupled behaviours, which is even more so the case considering that the work presented here deals with 2D examples, as opposed to the 3D physical world. While we do validate our model to a degree both qualitatively and quantitatively in Chapter 6, barring such real-world tests, validating our model against existing schemes is a viable alternative that could provide further insight into the accuracy of our scheme.

As an additional note on computational efficiency, we confirm in Chapter 6 that the addition of rigid body coupling does not appear to dominate overall computation costs. It is therefore likely the case that the efficiency of our model, similar to the original implementation and findings by [Da et al. \[2016\]](#), is mainly bottlenecked by the HHD projection and BEM steps. As a result, a future direction of work to improve the surface-only liquids framework may be to pursue methods that would improve on the efficiency of these individual steps, such as incorporating the fast method of fundamental solutions proposed by [Chen et al. \[2024\]](#) for speeding up the BEM solve.

As for other modifications to the surface-only liquids framework itself, we are curious to see what can be done to support vortical effects in the liquid. As mentioned, because of the irrotational assumption placed on the liquid velocity, the surface-only liquid method is not able to capture the diffusion of vorticity generated at the liquid interface into the interior of the liquid. While [Da et al. \[2016\]](#) mention that several purely vortex-based methods are unsuitable for simulating liquids due to the sharp density gradient that occurs between at the liquid-air interfaces, we believe that there is potential in augmenting the surface-only liquids scheme with a vortex-based method. For example, one could consider generating

and tracking vortex particles at the liquid surface, which could then flow into the interior of the liquid, allowing for the simulation of interesting vortical flows in the interior of the liquid domain.

In this work, we have shown that the surface-only liquids framework can be extended to support liquid-rigid body coupling by making the appropriate modifications to the step in the scheme in which we integrate external forces. In a similar vein, we believe the surface-only liquids framework should be extensible to simulate several other physical phenomena commonly seen in the graphical fluid simulation literature by updating this step. For instance, extending the surface-only liquid framework to support two-way coupling with solid deformables is a natural next step from the work presented here. The work by [Chentanez et al. \[2006\]](#) suggests that the main difference between such an extension and our work would be that deformable solids have an elastic force that would act on the fluid as well as the fact that deformable velocities follow their own constitutive equations which of course allow for more degrees of freedom than rigid body dynamics. Another phenomena that we suspect the surface-only liquid framework may be extended to handle is multiphase flow of liquids with differing densities; the *Los Topos* surface tracker already incorporated in the original surface-only liquid implementation was, after all, in part designed to robustly handle multi-material interactions [[Da et al., 2014](#)]. Extending surface-only liquids to handle multiphase flow would also likely mainly consist of updating the external force integration step, as a previously unconsidered liquid-liquid boundary defined by distinct fluid pressures stemming from each distinct fluid would likely require attention.

# References

- Alessandra Aimi, Mauro Diligenti, and Giovanni Monegato. New numerical integration schemes for applications of galerkin bem to 2-d problems. *International Journal for Numerical Methods in Engineering*, 40(11):1977–1999, 1997. doi: [https://doi.org/10.1002/\(SICI\)1097-0207\(19970615\)40:11<1977::AID-NME150>3.0.CO;2-O](https://doi.org/10.1002/(SICI)1097-0207(19970615)40:11<1977::AID-NME150>3.0.CO;2-O). URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0207%2819970615%2940%3A11%3C1977%3A%3AAID-NME150%3E3.0.CO%3B2-O>.
- Kendall E. Atkinson. *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1997.
- Vinicius C. Azevedo, Christopher Batty, and Manuel M. Oliveira. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. Graph.*, 35(4), July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925919. URL <https://doi.org/10.1145/2897824.2925919>.
- David Baraff. An introduction to physically based modeling: Rigid body simulation i—unconstrained rigid body dynamics. <https://www.cs.cmu.edu/~baraff/sigcourse/notesd1.pdf>, August 1997. Accessed: 2024-10-28.
- Christopher Batty, Florence Bertails, and Rook Bridson. A fast variational framework for accurate solid-fluid coupling. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 100–es, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781450378369. doi: 10.1145/1275808.1276502. URL <https://doi.org/10.1145/1275808.1276502>.
- Harsh Bhatia, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer. The helmholtz-hodge decomposition—a survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1386–1404, 2013. doi: 10.1109/TVCG.2012.316.

- Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, page 185–192, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 3905673134. doi: 10.1145/1057432.1057457. URL <https://doi.org/10.1145/1057432.1057457>.
- Rook Bridson. *Fluid simulation for computer graphics, Second Edition*. A K Peters/CRC Press, 01 2015.
- Rook Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 46–es, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781450378369. doi: 10.1145/1275808.1276435. URL <https://doi.org/10.1145/1275808.1276435>.
- Tyson Brochu and Rook Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493, 2009. doi: 10.1137/080737617. URL <https://doi.org/10.1137/080737617>.
- Tyson Brochu, Christopher Batty, and Rook Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4), July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778784. URL <https://doi.org/10.1145/1778765.1778784>.
- Tyson Brochu, Todd Keeler, and Rook Bridson. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, page 87–95, Goslar, DEU, 2012. Eurographics Association. ISBN 9783905674378.
- Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, 23(3):377–384, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015733. URL <https://doi.org/10.1145/1015706.1015733>.
- Jiong Chen, Florian Schaefer, and Mathieu Desbrun. Lightning-fast method of fundamental solutions. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658199. URL <https://doi.org/10.1145/3658199>.
- Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, page 65–es, New York, NY, USA, 2006. Association for

- Computing Machinery. ISBN 1595933646. doi: 10.1145/1179849.1179930. URL <https://doi.org/10.1145/1179849.1179930>.
- Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. Schrödinger’s smoke. *ACM Trans. Graph.*, 35(4), July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925868. URL <https://doi.org/10.1145/2897824.2925868>.
- Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O’Brien. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Trans. Graph.*, 32(2), April 2013. ISSN 0730-0301. doi: 10.1145/2451236.2451243. URL <https://doi.org/10.1145/2451236.2451243>.
- Fang Da, Christopher Batty, and Eitan Grinspun. Multimaterial mesh-based surface tracking. *ACM Trans. on Graphics (SIGGRAPH North America 2014)*, 2014.
- Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Double bubbles sans toil and trouble: discrete circulation-preserving vortex sheets for soap films and foams. *ACM Trans. Graph.*, 34(4), July 2015. ISSN 0730-0301. doi: 10.1145/2767003. URL <https://doi.org/10.1145/2767003>.
- Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Surface-only liquids. *ACM Trans. on Graphics (SIGGRAPH North America 2016)*, 2016.
- Malcolm R. Davidson. Boundary integral prediction of the spreading of an inviscid drop impacting on a solid surface. *Chemical Engineering Science*, 55(6):1159–1170, 2000. ISSN 0009-2509. doi: [https://doi.org/10.1016/S0009-2509\(99\)00307-3](https://doi.org/10.1016/S0009-2509(99)00307-3). URL <https://www.sciencedirect.com/science/article/pii/S0009250999003073>.
- Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free Surface Flows. volume 2: Symposia, Parts A, B, and C of *Fluids Engineering Division Summer Meeting*, pages 337–342, 07 2003. doi: 10.1115/FEDSM2003-45144. URL <https://doi.org/10.1115/FEDSM2003-45144>.
- Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83–116, November 2002. ISSN 0021-9991. doi: 10.1006/jcph.2002.7166. URL <https://doi.org/10.1006/jcph.2002.7166>.

- Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph.*, 24(3):904–909, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073281. URL <https://doi.org/10.1145/1073204.1073281>.
- Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 23–30, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113374X. doi: 10.1145/383259.383261. URL <https://doi.org/10.1145/383259.383261>.
- Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996. ISSN 1077-3169. doi: <https://doi.org/10.1006/gmip.1996.0039>. URL <https://www.sciencedirect.com/science/article/pii/S1077316996900398>.
- Oliver Gross, Yousuf Soliman, Marcel Padilla, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. Motion from shape change. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592417. URL <https://doi.org/10.1145/3592417>.
- Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981, jul 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073299. URL <https://doi.org/10.1145/1073204.1073299>.
- Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. <http://eigen.tuxfamily.org>, 2010.
- Yulong Guo, Xiaopei Liu, and Xuemiao Xu. A unified detail-preserving liquid simulation by two-phase lattice boltzmann modeling. *IEEE Transactions on Visualization and Computer Graphics*, 23(5):1479–1491, May 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2532335. URL <https://doi.org/10.1109/TVCG.2016.2532335>.
- David Hahn and Chris Wojtan. Fast approximations for boundary element based brittle fracture simulation. *ACM Trans. Graph.*, 35(4), July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925902. URL <https://doi.org/10.1145/2897824.2925902>.
- W.S. Hall. *The Boundary Element Method*, chapter 4. Linear Isoparametric Solution, pages 85–119. Springer Dordrecht, Dordrecht, 1994.
- Libo Huang and Dominik L. Michels. Surface-only ferrofluids. *ACM Trans. Graph.*, 39(6), November 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417799. URL <https://doi.org/10.1145/3414685.3417799>.

- Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.*, 40(6), December 2021. ISSN 0730-0301. doi: 10.1145/3478513.3480495. URL <https://doi.org/10.1145/3478513.3480495>.
- Doug L. James and Dinesh K. Pai. Artdefo: accurate real time deformable objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 65–72, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311542. URL <https://doi.org/10.1145/311535.311542>.
- Doug L. James, Jernej Barbič, and Dinesh K. Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph.*, 25(3):987–995, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141983. URL <https://doi.org/10.1145/1141911.1141983>.
- Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4), July 2015. ISSN 0730-0301. doi: 10.1145/2766996. URL <https://doi.org/10.1145/2766996>.
- Todd Keeler and Rook Bridson. Ocean waves animation using boundary integral equations and explicit mesh tracking. In *ACM SIGGRAPH 2014 Posters*, SIGGRAPH '14, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329583. doi: 10.1145/2614217.2614245. URL <https://doi.org/10.1145/2614217.2614245>.
- Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141961. URL <https://doi.org/10.1145/1141911.1141961>.
- Horace Lamb. *Hydrodynamics*. The University Press, New York, 1932. ISBN 9780521055154. URL <https://books.google.ca/books?id=BfZQAAAAAAAJ>.
- Kevin Lamb. Amath 463/663: Added mass and the acceleration reaction force, November 2022.
- Wei Li and Mathieu Desbrun. Fluid-solid coupling in kinetic two-phase flow simulation. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592138. URL <https://doi.org/10.1145/3592138>.

- Frank Losasso, Jerry Talton, Nipun Kwatra, and Ronald Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008. doi: 10.1109/TVCG.2008.37.
- Vladislav Mantic. A new formula for the c-matrix in the somigliana identity. *Journal of Elasticity*, 33(3):191–201, Dec 1993. ISSN 1573-2681. doi: 10.1007/BF00043247. URL <https://doi.org/10.1007/BF00043247>.
- Marek Krzysztof Misztal, Rook Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. Optimization-based Fluid Simulation on Unstructured Meshes. In Kenny Erleben, Jan Bender, and Matthias Teschner, editors, *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)*. The Eurographics Association, 2010. ISBN 978-3-905673-78-4. doi: /10.2312/PE/vriphys/vriphys10/011-020.
- Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, page 154–159, Goslar, DEU, 2003. Eurographics Association. ISBN 1581136595.
- J. N. Newman. *Marine Hydrodynamics*, pages 141–153. The MIT Press, 08 1977. ISBN 9780262280617. doi: 10.7551/mitpress/4443.001.0001. URL <https://doi.org/10.7551/mitpress/4443.001.0001>.
- Michael B. Nielsen and Rook Bridson. Guide shapes for high resolution naturalistic liquid simulation. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450309431. doi: 10.1145/1964921.1964978. URL <https://doi.org/10.1145/1964921.1964978>.
- Stanley Osher and R. Fedkiw. *The Level Set Methods and Dynamic Implicit Surfaces*, volume 57, pages xiv+273. Springer, 05 2004. ISBN 0-387-95482-1. doi: 10.1115/1.1760520.
- Oktar Ozgen, Marcelo Kallmann, Lynnette Es Ramirez, and Carlos Fm Coimbra. Underwater cloth simulation with fractional derivatives. *ACM Trans. Graph.*, 29(3), July 2010. ISSN 0730-0301. doi: 10.1145/1805964.1805967. URL <https://doi.org/10.1145/1805964.1805967>.
- Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, page 261–270, New York, NY, USA, 2005. Association for Computing Machinery.

- ISBN 1595931988. doi: 10.1145/1073368.1073406. URL <https://doi.org/10.1145/1073368.1073406>.
- Tobias Pfaff, Nils Thuerey, and Markus Gross. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph.*, 31(4), July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185608. URL <https://doi.org/10.1145/2185520.2185608>.
- Sergej Rjasanow and Olaf Steinbach. *The Fast Solution of Boundary Integral Equations*, pages 77–78. Mathematical and Analytical Techniques with Applications to Engineering. Springer New York, NY, 04 2007. ISBN 9780387340425. doi: <https://doi.org/10.1007/0-387-34042-4>. URL <https://link.springer.com/book/10.1007/0-387-34042-4>.
- Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27(3):1–9, August 2008. ISSN 0730-0301. doi: 10.1145/1360612.1360645. URL <https://doi.org/10.1145/1360612.1360645>.
- J. Shen and O. Sterz. A mixed galerkin and collocation approach for treating edge and corner problems in the boundary element method. *IEEE Transactions on Magnetics*, 34(5):3296–3299, 1998. doi: 10.1109/20.717774.
- Yousuf Soliman, Marcel Padilla, Oliver Gross, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. Going with the flow. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658164. URL <https://doi.org/10.1145/3658164>.
- Justin Solomon, Amir Vaxman, and David Bommes. Boundary element octahedral fields in volumes. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3065254. URL <https://doi.org/10.1145/3072959.3065254>.
- Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’99, page 121–128, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311548. URL <https://doi.org/10.1145/311535.311548>.
- Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4), July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461948. URL <https://doi.org/10.1145/2461912.2461948>.
- Ryusuke Sugimoto, Christopher Batty, and Toshiya Hachisuka. Surface-only dynamic deformables using a boundary element method. *Computer Graphics Forum*, 41(8):75–86,

2022. doi: <https://doi.org/10.1111/cgf.14625>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14625>.
- Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. A practical walk-on-boundary method for boundary value problems. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592109. URL <https://doi.org/10.1145/3592109>.
- Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunitatsu, and Hiroko Fujii. The simulation of fluid-rigid body interaction. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH '02, page 266, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135254. doi: 10.1145/1242073.1242279. URL <https://doi.org/10.1145/1242073.1242279>.
- Ningxiao Tao, Liangwang Ruan, Yitong Deng, Bo Zhu, Bin Wang, and Baoquan Chen. A vortex particle-on-mesh method for soap film simulation. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658165. URL <https://doi.org/10.1145/3658165>.
- Mauricio Vines, Ben Houston, Jochen Lang, and Won-Sook Lee. Vortical inviscid flows with two-way solid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 20(2):303–315, 2014. doi: 10.1109/TVCG.2013.95.
- Etienne Vouga. Lectures in discrete differential geometry 1 – plane curves. <https://www.cs.utexas.edu/~evouga/uploads/4/5/6/8/45689883/notes1.pdf>, February 2014. Accessed: 2024-10-24.
- He Wang, Kirill A. Sidorov, Peter Sandilands, and Taku Komura. Harmonic parameterization by electrostatics. *ACM Trans. Graph.*, 32(5), October 2013. ISSN 0730-0301. doi: 10.1145/2503177. URL <https://doi.org/10.1145/2503177>.
- Steffen Weißmann and Ulrich Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph.*, 29(4), July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778852. URL <https://doi.org/10.1145/1778765.1778852>.
- Steffen Weißmann and Ulrich Pinkall. Underwater rigid body dynamics. *ACM Trans. Graph.*, 31(4), July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185600. URL <https://doi.org/10.1145/2185520.2185600>.
- Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009.

Association for Computing Machinery. ISBN 9781605587264. doi: 10.1145/1576246.1531382. URL <https://doi.org/10.1145/1576246.1531382>.

Shiying Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. A clebsch method for free-surface vortical flow simulation. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530150. URL <https://doi.org/10.1145/3528223.3530150>.

Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. Clebsch gauge fluid. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459866. URL <https://doi.org/10.1145/3450626.3459866>.

Yongning Zhu and Rook Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073298. URL <https://doi.org/10.1145/1073204.1073298>.

# APPENDICES

# Appendix A

## Quadrature schemes for line elements

We provide the quadratures for line elements used in this work for comprehensiveness.

### A.1 Tanh-Sinh Quadrature

Also known as the Double Exponential (DE) formula, tanh-sinh quadrature removes endpoint singularities by using a change of variable to send the endpoints of the original integral to positive and negative infinity. An integral over the  $(-1, 1)$  interval can be approximated by:

$$\int_{-1}^1 f(x)dx \approx \sum_{k=-N}^N w_k f(x_k), \quad (\text{A.1})$$

where:

$$x_k = \tanh\left(\frac{\pi}{2} \sinh(kh)\right), \quad (\text{A.2})$$

$$w_k = \frac{\frac{1}{2}\pi h \cosh(kh)}{\cosh^2\left(\frac{1}{2} \sinh(kh)\right)}, \quad (\text{A.3})$$

where  $h$  is some step size. For our use of tanh-sinh quadrature for treating the singularities in the BIE evaluations for the HHD projection step, we used  $N = 9$  and  $h = 0.3$ .

## A.2 Four-point Gaussian Quadrature

An  $n$ -point Gaussian quadrature rule gives an exact result for polynomials of degree up to  $2n - 1$ . An integral over the  $(-1, 1)$  interval can be approximated by:

$$\int_{-1}^1 f(x)dx \approx \sum_{k=-N}^N w_k f(x_k), \quad (\text{A.4})$$

where  $x_k$  is the  $k$ -th root of  $P_n(x)$ .  $P_n(x)$  is  $n$ -th Legendre polynomial normalized such that  $P_n(1) = 1$ . The weights are then given by:

$$w_k = \frac{2}{(1 - x_k^2)(P'_n(x_k))^2}. \quad (\text{A.5})$$

The points and associated weights for a four-point Gaussian quadrature are provided below:

$$\begin{aligned} x_0 &= -\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}, & w_0 &= \frac{18 - \sqrt{30}}{36}, \\ x_1 &= -\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, & w_1 &= \frac{18 + \sqrt{30}}{36}, \\ x_2 &= \sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, & w_2 &= \frac{18 + \sqrt{30}}{36}, \\ x_3 &= \sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}, & w_3 &= \frac{18 - \sqrt{30}}{36}. \end{aligned}$$